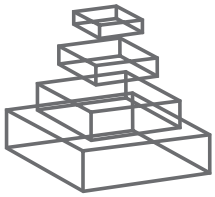
frontiers
RESEARCH TOPICS

# NEUROMORPHIC ENGINEERING SYSTEMS AND APPLICATIONS

Topic Editors
André van Schaik, Tobi Delbruck and Jennifer Hasler

frontiers in NEUROSCIENCE



# frontiers




**ISSN** 1664-8714
**ISBN** 978-2-88919-454-4
**DOI** 10.3389/978-2-88919-454-4


## ABOUT FRONTIERS

Frontiers is more than just an open-access publisher of scholarly articles: it is a pioneering approach to the world of academia, radically improving the way scholarly research is managed. The grand vision of Frontiers is a world where all people have an equal opportunity to seek, share and generate knowledge. Frontiers provides immediate and permanent online open access to all its publications, but this alone is not enough to realize our grand goals.

## FRONTIERS JOURNAL SERIES

The Frontiers Journal Series is a multi-tier and interdisciplinary set of open-access, online journals, promising a paradigm shift from the current review, selection and dissemination processes in academic publishing.
All Frontiers journals are driven by researchers for researchers; therefore, they constitute a service to the scholarly community. At the same time, the Frontiers Journal Series operates on a revolutionary invention, the tiered publishing system, initially addressing specific communities of scholars, and gradually climbing up to broader public understanding, thus serving the interests of the lay society, too.

## DEDICATION TO QUALITY

Each Frontiers article is a landmark of the highest quality, thanks to genuinely collaborative interactions between authors and review editors, who include some of the world's best academicians. Research must be certified by peers before entering a stream of knowledge that may eventually reach the public - and shape society; therefore, Frontiers only applies the most rigorous and unbiased reviews.
Frontiers revolutionizes research publishing by freely delivering the most outstanding research, evaluated with no bias from both the academic and social point of view.
By applying the most advanced information technologies, Frontiers is catapulting scholarly publishing into a new generation.

## WHAT ARE FRONTIERS RESEARCH TOPICS?

Frontiers Research Topics are very popular trademarks of the Frontiers Journals Series: they are collections of at least ten articles, all centered on a particular subject. With their unique mix of varied contributions from Original Research to Review Articles, Frontiers Research Topics unify the most influential researchers, the latest key findings and historical advances in a hot research area!
Find out more on how to host your own Frontiers Research Topic or contribute to one as an author by contacting the Frontiers Editorial Office: researchtopics@frontiersin.org

# NEUROMORPHIC ENGINEERING SYSTEMS AND APPLICATIONS

Topic Editors:
**André van Schaik,** University of Western Sydney, Australia
**Tobi Delbruck,** University of Zurich and ETH Zurich, Switzerland
**Jennifer Hasler,** Georgia Institute of Technology, USA


Malcolm Slaney (Google) leads the Neuromorphs team down the July 4th parade route in Telluride Colorado in 2012. Copyright owner is S.C. Liu.

Neuromorphic engineering has just reached its 25th year as a discipline. In the first two decades neuromorphic engineers focused on building models of sensors, such as silicon cochleas and retinas, and building blocks such as silicon neurons and synapses. These designs have honed our skills in implementing sensors and neural networks in VLSI using analog and mixed mode circuits.

Over the last decade the address event representation has been used to interface devices and computers from different designers and even different groups. This facility has been essential for our ability to combine sensors, neural networks, and actuators into neuromorphic systems. More recently, several big projects have emerged to build very large scale neuromorphic systems.

The Telluride Neuromorphic Engineering Workshop (since 1994) and the CapoCaccia Cognitive Neuromorphic Engineering Workshop (since 2009) have been instrumental not only in creating a strongly connected research community, but also in introducing different groups to each other's hardware. Many neuromorphic systems are first created at one of these workshops. With this special research topic, we showcase the state-of-the-art in neuromorphic systems.

# Table of Contents


**05** ***Research Topic: Neuromorphic Engineering Systems and Applications a Snapshot of Neuromorphic Systems Engineering***
André van Schaik, Tobi Delbruck and Jennifer Hasler

**07** ***Adaptive Pulsed Laser Line Extraction for Terrain Reconstruction Using a Dynamic Vision Sensor***
Christian Brandli, Thomas A. Mantel, Marco Hutter, Markus A. Höpflinger, Raphael Berner, Roland Siegwart and Tobi Delbruck

**16** ***Robotic Goalie with 3ms Reaction Time at 4% CPU Load using Event-Based Dynamic Vision Sensor***
Tobi Delbruck and Manuel Lang

**23** ***Event-Driven Visual Attention for the Humanoid Robot Icub***
Francesco Rea, Giorgio Metta and Chiara Bartolozzi

**34** ***On the use of Orientation Filters for 3D Reconstruction in Event-Driven Stereo Vision***
Luis A. Camunas-Mesa, Teresa Serrano-Gotarredona, Sio H. Ieng, Ryad B. Benosman and Bernabe Linares-Barranco

**51** ***Asynchronous Visual Event-Based Time-to-Contact***
Xavier Clady, Charles Clercq, Sio-Hoi Ieng, Fouzhan Houseini, Marco Randazzo, Lorenzo Natale, Chiara Bartolozzi and Ryad Benosman

**61** ***Real-Time Classification and Sensor Fusion with a Spiking Deep Belief Network***
Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck and Michael Pfeiffer

**74** ***Event-Driven Contrastive Divergence for Spiking Neuromorphic Systems***
Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado and Gert Cauwenberghs

**88** ***Compiling Probabilistic, Bio-Inspired Circuits on a Field Programmable Analog Array***
Bo Marr and Jennifer Hasler

**97** ***An Adaptable Neuromorphic Model of Orientation Selectivity Based on Floating Gate Dynamics***
Priti Gupta and C.M. Markan

**118** ***A Mixed-Signal Implementation of a Polychronous Spiking Neural Network with Delay Adaptation***
Runchun M. Wang, Tara J. Hamilton, Jonathan C. Tapson and André van Schaik

**134** ***Real-Time Biomimetic Central Pattern Generators in an FPGA for Hybrid Experimentsf***
Matthieu Ambroise, Timothée Levi, Sébastien Joucla, Blaise Yvert and Sylvain Saïghi

**145** ***Dynamic Neural Fields as a Step Toward Cognitive Neuromorphic Architectures***
Yulia Sandamirskaya

**158** ***A Robust Sound Perception Model Suitable for Neuromorphic Implementation***
Martin Coath, Sadique Sheik, Elisabetta Chicca, Giacomo Indiveri, Susan L. Denham and Thomas Wennekers

**168** ***An Efficient Automated Parameter Tuning Framework for Spiking Neural Networks***
Kristofor D. Carlson, Jayram Moorkanikara Nageswaran, Nikil Dutt and Jeffrey L. Krichmar

frontiers in
NEUROSCIENCE

EDITORIAL
published: 19 December 2014
doi: 10.3389/fnins.2014.00424

# Research topic: neuromorphic engineering systems and applications. A snapshot of neuromorphic systems engineering

***Tobi Delbruck[1], André van Schaik[2]* and Jennifer Hasler[3]***

[1] *Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland*
[2] *Bioelectronics and Neuroscience, The MARCS Institute, University of Western Sydney, Sydney, NSW, Australia*
[3] *School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA*
**Correspondence: a.vanschaik@uws.edu.au*

***Edited and reviewed by:***
*Giacomo Indiveri, University of Zurich and ETH Zurich, Switzerland*

**Keywords: neuromorphic engineering, neural networks, event-based, spiking neural networks, dynamic vision sensor, floating gate, neural simulation, synaptic plasticity**

The 14 papers in this research topic were solicited primarily from attendees to the two most important hands-on workshops in neuromorphic engineering: the *Telluride Neuromorphic Cognition Engineering Workshop* (www.ine-web.org) and the *Capo Caccia Cognitive Neuromorphic Engineering Workshop* (capocaccia.ethz.ch). The papers show the results of feasibility studies of new concepts, as well as neuromorphic systems that have been constructed from more established neuromorphic technologies. Five papers exploit neuromorphic dynamic vision sensor (DVS) events that mimic the asynchronous and sparse spikes on biology's optic nerve fiber (Delbruck and Lang, 2013; O'Connor et al., 2013; Rea et al., 2013; Brandli et al., 2014; Camunas-Mesa et al., 2014; Clady et al., 2014). Two papers are on the hot topic (based on largest number of views) of event-driven computation in deep belief networks (DBNs) (O'Connor et al., 2013; Neftci et al., 2014). Two papers use floating gate technology for neuromorphic analog circuits (Gupta and Markan, 2014; Marr and Hasler, 2014). The collection is rounded out by papers on central pattern generators (Ambroise et al., 2013), neural fields for cognitive architectures (Sandamirskaya, 2014), sound perception (Coath et al., 2014), polychronous spiking networks (Wang et al., 2014), and automatic parameter tuning for large network simulations (Carlson et al., 2014).

## REGARDING THE EVENT-BASED VISION PAPERS

Brandli et al. (2014) report on a novel method for rapidly and cheaply tracking a flashing laser line using a DVS, which is aimed at building a fast obstacle detector for small ground-based robots, such as vacuum cleaners or toy cars. Particular novelties in this paper are the adaptive temporal filter and the efficient algorithmic update of the laser line position.

Delbruck and Lang (2013) report on the detailed implementation of a fun robotic goalie, which uses a DVS to help track the balls and robot arm. The paper includes measurements of USB latency. The novelty in this paper is the self-calibration of the goalie arm so that it can be rapidly placed in a particular place in visual space. Both of the preceding two papers include YouTube citations to videos and both have open-source implementations.

Rea et al. (2013) report on the integration of a stereo pair of DVS sensors with the iCub robot, and how they are used for quick, low power saliency detection for the iCub. In particular, the Koch-Ittie visual saliency model was adapted to event-driven sensors and many experiments were done to characterize its effectiveness and efficiency.

Camunas-Mesa et al. (2014) reports on a stereo vision system that uses a pair of their DVS cameras together with an FPGA that computes low level oriented features. This paper has a wealth of characterization results.

Clady et al. (2014) report the first results of computing the old problem of time to contact (TTC) for a moving entity from DVS events. They take a geometrical approach in order to extract low level motion features from the DVS events to obtain the TTC information. This paper includes robotic experiments.

## REGARDING EVENT-DRIVEN DEEP NETWORKS

O'Connor et al. (2013) present the first of the papers to focus on event-based learning and networks. Their system also uses a DVS, as well as an AEREAR2 binaural silicon cochlea, to build a spike-based DBN for recognizing MNIST digits presented in conjunction with pure tones. They demonstrate that a DBN constructed from stacks of restricted Boltzmann machines (RBMs) is valuable for learning and computing sensor fusion. They also show that a DBN's recurrent persistent activity is useful particularly with sparse event-driven sensor input. This network was trained off-line, and then the weights were transferred onto the spiking network.

Neftci et al. (2014) report on the same target application of MNIST digit recognition, but their paper takes a further step by proposing how a network of integrate and fire neurons can implement a RBM, and can be trained with an event-driven version of the well-known contrastive divergence training algorithm for RBMs.

## REGARDING FLOATING GATE TECHNOLOGY

Two papers show the versatility of floating-gate (FG) circuit approaches. Marr and Hasler (2014) describe a collaborative project started and effectively completed during the Telluride 2008 workshop as a representative of the possible opportunity at any of these workshops. In this case, the opportunity was enabled through the use of large-scale field programmable analog arrays (FPAA) as a mixed mode processor for which functions can be compiled enabling a range of circuit, system,

and application design. The focus was on stochastic computations that are dynamically controllable via voltage-controlled amplifiers and comparator thresholds. From Bernoulli variables it is shown that exponentially distributed random variables, and random variables of an arbitrary distribution can be computed. The trajectory of a biological system computed stochastically with this probabilistic hardware results in a 127X performance improvement over current software approaches.

Gupta and Markan (2014), report on a FG adaptive system for investigating self-organization of image patterns. They describe adaptive feature selectivity as a mechanism by which nature optimizes resources so as to have greater acuity for more abundant features. The authors look to exploit hardware dynamics to build adaptive systems utilizing time-staggered winner-take-all circuits, exploiting the adaptation dynamics of FG transistors, to model an adaptive cortical cell.

## REGARDING OTHER TOPICS IN NETWORK ARCHITECTURES

Wang et al. (2014) report results from a polychronous multineuron chip. Polychronization is the process in which spikes travel down axons with specific delays to arrive at a common target neuron simultaneously and cause it to fire, despite the source neurons firing asynchronously. This paper shows digital and analog tradeoffs and offers advice for scaling to future technologies.

Ambroise et al. (2013) describe a neuromorphic implementation of a network of 240 Central Pattern Generator modules modeling the leech heartbeat neural network on a field programmable gate array. It uses the Izhikevich neuron model, implemented as a single computational core, time multiplexed to update all the neurons in the network. In order to fit the digital implementation to the data from the biological system without implementing all the detailed synaptic dynamics, which would take up too many resources, they propose a new synaptic adaptation model: an activity-dependent depression synapse.

Sandamirskaya (2014) leverages the relationship between dynamic field theory networks and neuromorphic circuits using soft winner take all circuits (WTA) to formally describe the equivalence between the two and establish a common ground. It sets a possible roadmap for the development of cognitive neuromorphic systems using WTA implementations.

Coath et al. (2014) describes a pattern recognition network implemented using a column of three neurons in which the columns are connected via axons with delays that explicitly depend on the distance between the columns. The networked is trained using spike-timing dependent plasticity and it is shown that the performance of the network is robust to natural variations in the input stimuli.

Carlson et al. (2014) address the significant problem of finding solutions in the enormous parameter space found in implementations of spiking neural networks by proposing an automated tuning framework. Their approach uses evolutionary algorithms implemented on graphics processing units for speed. They use an objective function based on the Efficient Coding Hypothesis to tune these networks. In their example, they demonstrate the evolution of V1 simple cell responses. Using GPU parallelization, they report 65x speedups over CPU implementations.

## SUMMARY

Amidst the promises offered by projects with major chunks of funding in neuromorphic engineering like HBP, BrainScaleS, SpiNNaker, and TrueNorth, this research topic offers a refreshing glimpse into some of the current actual accomplishments in neuromorphic systems engineering and applications.

## REFERENCES


Ambroise, M., Levi, T., Joucla, S., Yvert, B., and Saïghi, S. (2013). Real-time biomimetic central pattern generators in an FPGA for hybrid experiments. *Front. Neurosci.* 7:215. doi: 10.3389/fnins.2013.00215

Brandli, C., Mantel, T. A., Hutter, M., and Delbruck, T. (2014). Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor. *Front. Neurosci.* 7:275. doi: 10.3389/fnins.2013.00275

Camunas-Mesa, L. A., Serrano-Gotarredona, T., Ieng, S. H., Benosman, R. B., and Linares-Barranco, B. (2014). On the use of orientation filters for 3D reconstruction in event-driven stereo vision. *Front. Neurosci.* 8:48. doi: 10.3389/fnins.2014.00048

Carlson, K. D., Nageswaran, J. M., Dutt, N., and Krichmar, J. L. (2014). An efficient automated parameter tuning framework for spiking neural networks. *Front. Neurosci.* 8:10. doi: 10.3389/fnins.2014.00010

Clady, X., Clercq, C., Ieng, S.-H., Houseini, F., Randazzo, M., Natale, L., et al. (2014). Asynchronous visual event-based time-to-contact. *Front. Neurosci.* 8:9. doi: 10.3389/fnins.2014.00009

Coath, M., Sheik, S., Chicca, E., Indiveri, G., Denham, S., and Wennekers, T. (2014). A robust sound perception model suitable for neuromorphic implementation. *Front. Neurosci.* 7:278. doi: 10.3389/fnins.2013.00278

Delbruck, T., and Lang, M. (2013). Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Front. Neurosci.* 7:223. doi: 10.3389/fnins.2013.00223

Gupta, P., and Markan, C. M. (2014). An adaptable neuromorphic model of orientation selectivity based on floating gate dynamics. *Front. Neurosci.* 8:54. doi: 10.3389/fnins.2014.00054

Marr, B., and Hasler, J. (2014). Compiling probabilistic, bio-inspired circuits on a field programmable analog array. *Front. Neurosci.* 8:86. doi: 10.3389/fnins.2014.00086

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Front. Neurosci.* 7:272. doi: 10.3389/fnins.2013.00272

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Rea, F., Metta, G., and Bartolozzi, C. (2013). Event-driven visual attention for the humanoid robot iCub. *Front. Neurosci.* 7:234. doi: 10.3389/fnins.2013.00234

Sandamirskaya, Y. (2014). Dynamic neural fields as a step toward cognitive neuromorphic architectures. *Front. Neurosci.* 7:276. doi: 10.3389/fnins.2013.00276

Wang, R. M., Hamilton, T. J., Tapson, J., and van Schaik, A. (2014). A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 8:51. doi: 10.3389/fnins.2014.00051


**Conflict of Interest Statement:** The Associate Editor Giacomo Indiveri declares that, despite being affiliated to the same institution as author Tobi Delbruck, the review process was handled objectively and no conflict of interest exists. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 24 November 2014; accepted: 03 December 2014; published online: 19 December 2014.*

*Citation: Delbruck T, van Schaik A and Hasler J (2014) Research topic: neuromorphic engineering systems and applications. A snapshot of neuromorphic systems engineering. Front. Neurosci.* ***8****:424. doi: 10.3389/fnins.2014.00424*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 17 January 2014
doi: 10.3389/fnins.2013.00275

# Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor

**Christian Brandli[1]*, Thomas A. Mantel[2], Marco Hutter[2], Markus A. Höpflinger[2], Raphael Berner[1], Roland Siegwart[2] and Tobi Delbruck[1]**

[1] Department of Information Technology and Electrical Engineering, Institute of Neuroinformatics, ETH Zurich and University of Zurich, Zurich, Switzerland
[2] Autonomous Systems Lab, Department of Mechanical and Process Engineering, ETH Zurich, Zurich, Switzerland

**Edited by:**
*André Van Schaik, The University of Western Sydney, Australia*

**Reviewed by:**
*Christoph Posch, Universite Pierre et Marie Curie, France*
*Viktor Gruev, Washington University in St. Louis, USA*
*Garrick Orchard, National University of Singapore, Singapore*

***Correspondence:**
*Christian Brandli, Department of Information Technology and Electrical Engineering, Universität Zürich, Winterthurerstr. 190, 8057, Zurich, Switzerland*
*e-mail: braendch@ethz.ch*

Mobile robots need to know the terrain in which they are moving for path planning and obstacle avoidance. This paper proposes the combination of a bio-inspired, redundancy-suppressing dynamic vision sensor (DVS) with a pulsed line laser to allow fast terrain reconstruction. A stable laser stripe extraction is achieved by exploiting the sensor's ability to capture the temporal dynamics in a scene. An adaptive temporal filter for the sensor output allows a reliable reconstruction of 3D terrain surfaces. Laser stripe extractions up to pulsing frequencies of 500 Hz were achieved using a line laser of 3 mW at a distance of 45 cm using an event-based algorithm that exploits the sparseness of the sensor output. As a proof of concept, unstructured rapid prototype terrain samples have been successfully reconstructed with an accuracy of 2 mm.

**Keywords: neuromorphic, robotics, event-based, address-event representation (AER), dynamic vision sensor (DVS), silicon retina**

## INTRODUCTION

Motion planning in mobile robots requires knowledge of the terrain structure in front of and underneath the robot; possible obstacles have to be detected and their size has to be evaluated. Especially legged robots need to know the terrain on which they are moving so that they can plan their steps accordingly. A variety of 3D scanners such as the Microsoft Kinect© (Palaniappa et al., 2011) or LIDAR (Yoshitaka et al., 2006; Raibert et al., 2008) devices can be used for this task but these sensors and their computational overhead typically consume on the order of several watts of power while having a sample rate limited to tens of Hertz. Passive vision systems partially overcome these limitations but they exhibit a limited spatial resolution because their terrain reconstruction is restricted to a small set of feature points (Weiss et al., 2010).

Many of the drawbacks in existing sensor setups (active as well as passive) arise from the fact that investigating visual scenes as a stroboscopic series of (depth) frames leads to redundant data that occupies communication and processing bandwidth and limits its sample rates to the frame rate. If the redundant information is already suppressed at the sensor level and the sensor asynchronously reports its output, the output can be evaluated faster and at a lower computational cost. In this paper such a vision sensor, the so called dynamic vision sensor (DVS; Lichtsteiner et al., 2008) is combined with a pulsed line laser, forming an active sensor to reconstruct the terrain in front of the system while it is moved. This terrain reconstruction is based on a series of surface profiles based on the line laser pulses. The proposed algorithm allows extracting the laser stripe from the asynchronous temporal contrast events generated by the DVS using only the event timing so that the laser can be pulsed at arbitrary frequencies from below 1 Hz up to 500 Hz. The flexibility in choosing the pulsing frequencies allows fast and detailed surface reconstructions for fast robot motions as well as saving laser power for slow motions.

### THE DYNAMIC VISION SENSOR (DVS)

The DVS used in this setup is inspired by the functionality of the retina and senses only changes in brightness (Lichtsteiner et al., 2008). Each pixel reports a change in log-illuminance larger than a given threshold by sending out an asynchronous address-event: if it becomes brighter it generates a so called "ON event," and if darker, it generates an "OFF event." The asynchronously generated address-events are communicated to a synchronous processing device by a complex programmable logic device (CPLD) which also transmits the time in microseconds at which the event occurred. Each event contains the pixel horizontal and vertical address $(u,v)$, its polarity (ON/OFF) and the timestamp. After the event is registered, it is written into a FIFO buffer which is transferred through a high-speed USB 2.0 interface to the processing platform. Real-time computations on the processing platform operate on the basis of so called event packets which can contain a variable number of events but are delivered at a minimum frequency of 1 kHz. This approach of sensing a visual scene has the following advantages:

1. The absence of a global exposure time lets each pixel settle to its own operating point which leads to a dynamic range of more than 120 dB.
2. Because the pixels only respond to brightness changes, the output of the sensor is non-redundant. This leads to a decrease in processor load and therefore to a reduction in power consumption of the system.

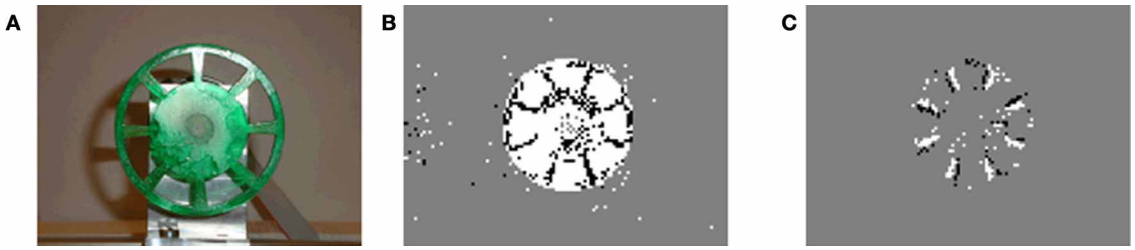



**FIGURE 1 | Wheel spinning at 3000 rpm. (A)** Still image. **(B)** Events generated in 30 ms: ON events rendered white, OFF events in black. **(C)** Events generated in 200 us.

3. The asynchronous readout allows a low latency of as little as 15 us. This latency allows to close control loops very quickly as demonstrated in Delbruck and Lichtsteiner (2007); Conradt et al. (2009); Ni et al. (2012). **Figure 1** shows the speed of the DVS, which is capable of resolving fast movements such as a wheel spinning at 3000 rpm.
4. Since the events are timestamped as they occur (with a temporal resolution of 1 us), the output allows a detailed analysis of the dynamics in a scene or to process its output using temporal filters.

In the following, the output of the DVS is described as a set of events and each event *Ev* carries its *u*- and *v*-address, a timestamp and its polarity as a value of +1 if it is an ON event and a −1 for OFF events [with notation adapted from Ni et al. (2012)].

$$Ev(u, v, t) = \begin{cases} +1, & \text{if } \Delta \ln(I_{u,v}) > \Theta_{\text{ON}} \\ -1, & \text{if } \Delta \ln(I_{u,v}) < \Theta_{\text{OFF}} \end{cases} \quad (1)$$

where $\Delta \ln(I_{u,v})$ denotes the change in illumination at the pixel with coordinates *u,v* since the last event. $\Theta_{\text{ON}}$ and $\Theta_{\text{OFF}}$ denote the event thresholds that must be crossed to trigger an event. These thresholds can be set independently which allows balancing the number of ON and OFF events.

In addition to these visually triggered events, the DVS allows the injection of special, timestamped trigger events to the output stream by applying a pulse to a pin on the back of the sensor. These *Et* events are numbered in software so that they carry a pulse number and a timestamp:

$$Et_n = t. \quad (2)$$

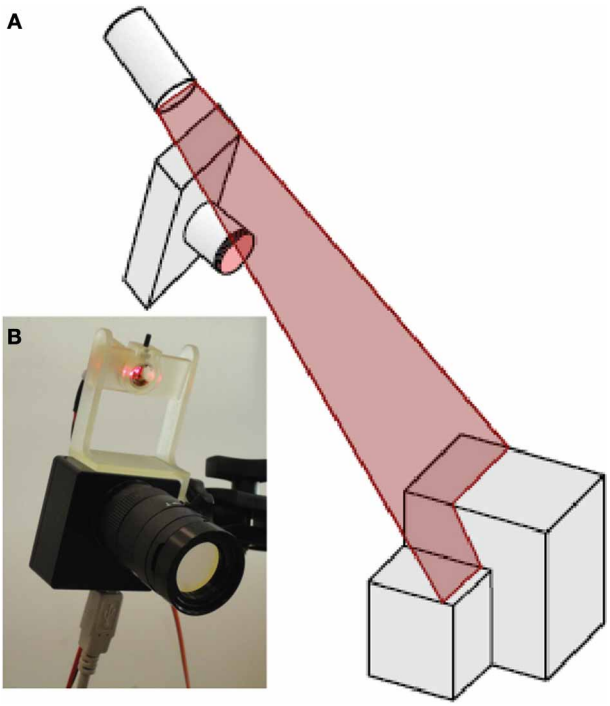



**FIGURE 2 | Setup of the DVS together with the line laser. (A)** Schematic view of the setup. **(B)** Photo of the DVS128 camera with line laser: the rigid laser mount allows a constant distance and inclination angle of the laser with respect to the camera. The optical filter is mounted on the lens.

## MATERIALS AND METHODS

### HARDWARE SETUP

As reviewed in Forest and Salvi (2002), there are several variations of combining a line laser and a camera to build a 3D scanner. Since it is intended to apply this scanner setup on a mobile robot that already has a motion model for the purpose of navigation, a mirror free, fixed geometry setup was chosen. As shown in **Figure 2**, a red line laser (Laser Components GmbH LC-LML-635) with a wavelength of 635 nm and an optical power of about 3 mW was mounted at a fixed distance above the DVS. (The laser power consumption was 135 mW.) The relative angle of the laser plane and the DVS was fixed. To run the terrain reconstruction, the system is moved over the terrain while the laser is pulsed at a frequency $f_p$. Each pulse of the laser initiated the acquisition of a set of events for further analysis and laser stripe extraction. The background illumination level was a brightly-lit laboratory at approximately 500 lx.

For the measurements described in the results section, the system was fixed and the terrain to scan was moved on an actuated sled on rails underneath it. This led to a straight-forward camera motion model controlled by the speed of the DC motor that pulled the sled toward the sensor system. The sled was fixed to rails which locked the system in one dimension and led to highly repeatable measurements. The DVS was equipped with a lens having a focal length of 10 mm and it was aimed at the terrain from

a distance of 0.45 m. The laser module was placed at a distance of 55 mm from the sensor at an inclination angle $\alpha_L$ of 8° with respect to the principal axis of the DVS. The system observed the scene at an inclination angle $\alpha_C$ of 39°.

To enhance the signal to noise ratio, i.e., the percentage of events originating from the pulsed laser line, the sensor was equipped with an optical band pass filter (Edmund Optics NT65-167) centered at 636 nm. The filter has full width at half maximum (FWHM) of 10 nm and a transmittance of 85% in the pass band and less than 0.01% in the stop band (optical density 4.0).

To mark the laser pulses within the event stream, the event trigger pin on the back of the DVS was connected to the function generator triggering the laser.

## CALIBRATION

To extract the laser stripe, i.e., the pixels whose events originate from the laser line, the sensor is calibrated based on the approach described in Siegwart (2011). The model was simplified by the following assumptions:

1. For the intrinsic camera model, rectangular pixels with orthogonal coordinates $u$,$v$ are assumed. This leads to the following transformation from pixel coordinates to camera coordinates $x_C$, $y_C$, $z_C$:

$$u = \frac{kf_l}{z_C} x_C + u_0 \tag{3}$$

$$v = \frac{kf_l}{z_C} y_C + v_0 \tag{4}$$

   where $k$ denotes the inverse of the pixel size, $f_l$ the focal length in pixels, and $u_0$, $v_0$ the center pixel coordinates.
2. For the extrinsic camera model it was assumed that the rail restricts the origin of the camera $x_{C0}$, $y_{C0}$, $z_{C0}$ to a planar translation (by $t_y$ and $t_z$) within a plane spanned by the y- and z-axis of the world reference frame $x_R$, $y_R$, and $z_R$ as depicted in **Figure 3**. In the setup used for the measurement, the rotational degrees of freedom of the system were constrained so that the that the camera could only rotate (by $\alpha_C$) around its x-axis which leads to following transformation from camera to world coordinates:

$$\begin{pmatrix} x_R \\ y_R \\ z_R \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_C + \frac{\pi}{2}) & \sin(\alpha_C + \frac{\pi}{2}) \\ 0 & -\sin(\alpha_C + \frac{\pi}{2}) & \cos(\alpha_C + \frac{\pi}{2}) \end{pmatrix} \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} + \begin{pmatrix} 0 \\ t_y \\ t_z \end{pmatrix} \tag{5}$$

The fact that the DVS does not produce any output for static scenes makes it difficult to find and align correspondences and therefore the typical checkerboard pattern could not be used for calibration. As an alternative, the laser was pulsed onto two striped blocks of different heights as depicted in **Figure 4**. The black stripes on the blocks absorb sufficient laser light to not excite any events in the DVS. This setup allows finding sufficient correspondence points between the real world coordinates and the pixel coordinates to solve the set of calibration equations (Equations 3–5). This procedure is done manually in Matlab but needs only to be done once.

## LASER STRIPE EXTRACTION

The stripe extraction method is summarized in **Figure 5**. Most laser stripe extraction algorithms perform a simple column-wise maximum computation to find the peak in light intensity e.g., Robinson et al. (2003); Orghidan et al. (2006). Accordingly for the DVS the simplest approach to extract the laser stripe would be to accumulate all events after a laser pulse and find the column-wise maximum in activity. This approach performs poorly due to background activity: Even with the optical filter in place, contrast edges that move relative to the sensor also induce events which corrupt the signal to noise ratio. For a more robust laser strip extraction, spatial constraints could be introduced but this would restrict the generality of the approach (Usamentiaga et al., 2010). Instead the proposed approach exploits the highly resolved temporal information of the output of the DVS.

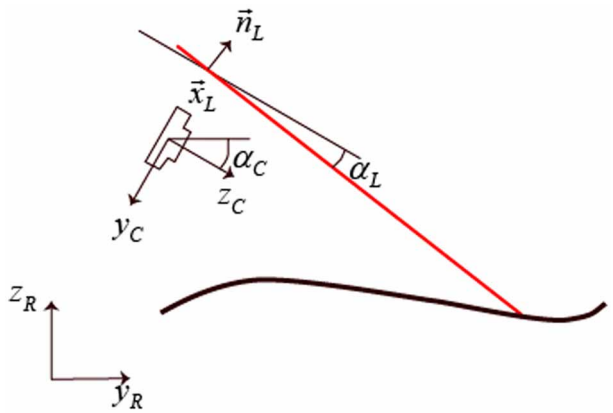



**FIGURE 3 | The coordinate systems used along the scanning direction.** $y_R$, $z_R$ are the real world coordinates, $y_C$, $z_C$ the ones of the camera. $x_L$ is the distance of the laser line plane perpendicular to $n_L$ from the camera origin. $\alpha_C$is the inclination angle of the sensor with respect to the horizontal plane and $\alpha_L$ the laser inclination angle with respect to the camera.

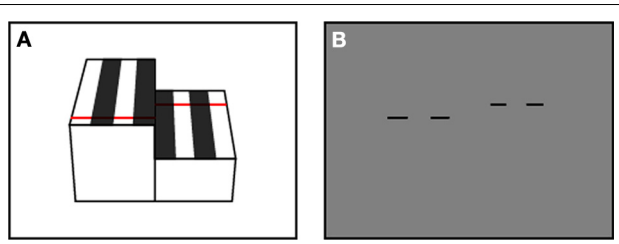



**FIGURE 4 | The calibration setup.** The pulsed laser shines onto two striped blocks of different height. **(A)** Schematic view. **(B)** Schematic of the DVS output: the laser is absorbed by the black stripes and only the white stripes generate events.

With the help of the laser trigger events $Et_n$, the event stream can be sliced into a set of time windows $W_n$ each containing a set of events $S_n$ where $n$ denotes the $n$'th trigger event. ON and OFF events are placed into separate sets (for simplicity only the formulas for the ON events are shown):

$$W_n = \{t : t > Et_n \wedge t < Et_{n+1}\} \quad (6)$$

$$S_n^{\mathrm{ON}} = \{Ev(u, v, t) : t \in W_n \wedge Ev > 0\} \quad (7)$$

The timing of the events is jittered by the asynchronous communication and is also dependent on the sensor's bias settings and light conditions. Our preliminary experiments showed that it is not sufficient to only accumulate the events in a fixed time window after the pulse. Instead a stable laser stripe extraction algorithm must adaptively collect relevant events. This adaptation is achieved by using of a temporal scoring function $P$ which is continually updated as illustrated in **Figure 6**.

The scoring function is used as follows: Each event obtains a score $s = P(Ev)$ depending only on its time relative to the last trigger. From these $s$ a score map $M_n$ (**Figure 5**) is established where each pixel $(u,v)$ of $M_n$ contains the sum of the scores of all the events with address $(u,v)$ within the set $S_n$ [these subsets of $S_n$ are denoted as $C_n(u, v)$]. In other words, $M_n$ is a 2D histogram of event scores. This score map tells us for each pixel how well-timed the events were with respect to the $n$'th trigger event, and it is computed by Equations 8–9:

$$C_n^{\mathrm{ON}}(u, v) = \{Ev(u', v', t) : Ev \in S_n^{\mathrm{ON}} \wedge u' = u \wedge v' = v\} \quad (8)$$

$$M_n(u, v) = \sum_{C^{\mathrm{ON}}(u,v)} P_n^{\mathrm{ON}}(Ev) + \sum_{C^{\mathrm{OFF}}(u,v)} P_n^{\mathrm{OFF}}(Ev) \quad (9)$$

The scoring function $P$ that assigns each event a score indicating how probable it is that it was caused by the laser pulse $Et_n$ is obtained by using another histogram-based approach. The rationale behind this approach is the following: All events that are caused by the laser pulse should be temporally correlated with it while noise events should show a uniform temporal distribution. In a histogram with binned relative times the events triggered by the laser pulse should form peaks. In the proposed algorithm, the histogram $H_n$ consists of $k$ bins $B_n$ of width $fk$. For stability, $H_n$ is an average over $m$ laser pulses. $H_n$ is constructed by Equations 10–12:

$$D_n^{\mathrm{ON}}(l) = \left\{ Ev(u, v, t) : Ev \in S_n^{\mathrm{ON}} \wedge t - Et_n \geq \frac{l}{fk} \wedge t - Et_n < \frac{l+1}{fk} \right\} \quad (10)$$

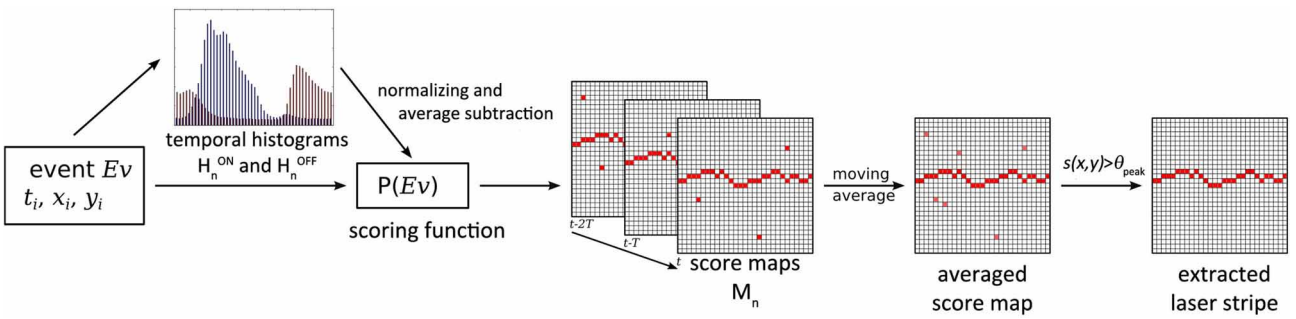



**FIGURE 5 | Schematic overview of the laser stripe extraction filter.** At the arrival of each laser pulse the temporal histograms are used to adapt the scoring function P, and each event's score is calculated and mapped on the score maps. The maps are averaged and the laser stripe is extracted by selecting the maximum scoring pixel for each column, if it is above the threshold $\theta_{peak}$.

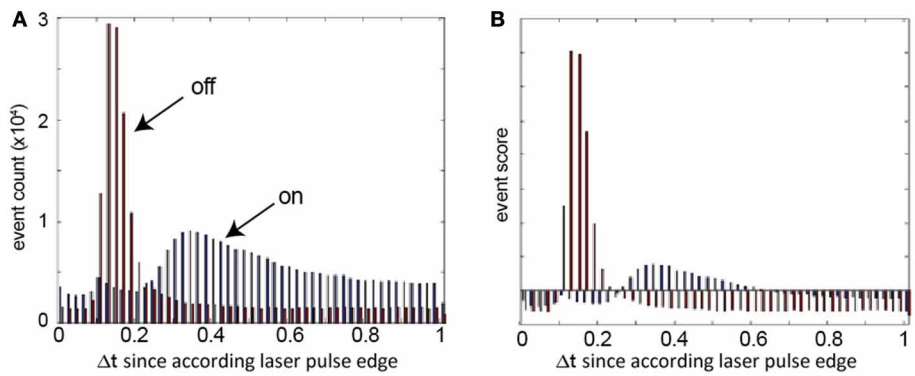



**FIGURE 6 | Scoring function: examples of event histograms of the laser pulsed at 1 kHz at the relief used for the reconstruction. (A)** Measured histograms of ON and OFF events following laser pulse ON and OFF edges. **(B)** Resulting OFF and ON scoring functions after normalization and mean subtraction.

$$B_n^{\mathrm{ON}}(l) = \sum_{i=n-m}^{n-1} \sum_{D_i^{\mathrm{ON}}(l)} \|Ev\| \tag{11}$$

$$H_n^{\mathrm{ON}} = \left\{ B_n^{\mathrm{ON}}(l) : l \in [0, k-1] \right\} \tag{12}$$

where $f$ is the laser frequency, $l$ is the bin index, $k$ is the number of bins, $D_n(l)$ is a temporal bin of the set $S_n$, $B_n(l)$ is a bin of the averaged histogram over the $m$ and the histogram $H_n$ is the set of all bins $B_n$. It is illustrated in **Figure 6A.**

To obtain the scoring function $P$, the $H_n^{\mathrm{ON}}$ and $H_n^{\mathrm{OFF}}$ histograms are normalized by the total number $T$ of events in them. To penalize bins that have a count below the average i.e., bins that are dominated by the uniformly distributed noise, the average bin count $T/k$ is subtracted from each bin. An event can have a negative score. This is the case if it is more probable that it is noise than signal. $T_n$ is computed from Equation 13:

$$T_n^{\mathrm{ON}} = \sum \left\{ B_n^{\mathrm{ON}} : B_n^{\mathrm{ON}} \in H_n^{\mathrm{ON}} \right\} \tag{13}$$

The $n$'th scoring function $P_n$ (illustrated in **Figure 6B**) is computed from Equation 14:

$$P_n^{\mathrm{ON}}(Ev) = \frac{\sum \left\{ B_n^{\mathrm{ON}} : Ev \in B_n^{\mathrm{ON}} \right\} - \left( \frac{T_n^{\mathrm{ON}}}{k} \right)}{T_n^{\mathrm{ON}}} \tag{14}$$

To extract the laser stripe, the last $o$ score maps are averaged and the maximum score $s(u,v)$ and its $y$ value are determined for each column. If the maximum value is above a threshold $\vartheta_{\mathrm{peak}}$ it is considered to be a laser stripe pixel. If the neighboring pixels are also above the threshold, a weighted average is applied among them to determine the center of the laser stripe. The positions of the laser stripe are then transformed into real world coordinates using Equations 3–5 and thus mapped as surface points.

The pseudo-code shown in Algorithm 1 illustrates how the algorithm is executed: Only on the arrival of a new laser trigger event, the histograms are averaged, the score maps are averaged to an average score map and the laser stripe is extracted. Otherwise, for each DVS event only its contribution to the current score map is computed, using the current scoring function. The laser stripe extraction and computation of the scoring function operate on different time scales. While the length $o$ of the moving average for the scoring function is chosen as small as possible to ensure a low latency, the number of histograms $m$ to be averaged is chosen as large as possible to obtain higher stability and dampen the effect of variable background activity.

**Algorithm 1 | Pseudo code for the laser stripe extraction.**

```
//iterate over all events in a packet
for event:packet
  //the laser stripe extraction is only done at
  //the arrival of a new pulse
  if(event.isTrigger)
    lastTrigger = event.timestamp
    histogramAverage.removeOldest()
    histogramAverage.add(histogram)
    histogram.clear()
    //update done according to Equation (14)
    scoreFunction.update(histogramAverage)
    averageMap.removeOldest()
    averageMap.add(scoreMap)
    laserLine = averageMap.findColumPeaks()
else
  //update of histogram
  deltaT = lastTrigger-event.timestamp
  binIndex = deltaT*k/period
  histogram.bin[binIndex]++
  //update of score map
  score = scoreFunction.get(binIndex)
  scoreMap[event.u][event.v]+=score
end if
```

### *Algorithm optimization*

To reduce the memory consumption and the computational cost of this "frame-based" algorithm, the computations of the scoring function, the accumulation of evidence into a score map, and the search for the laser line columns were optimized to be event-based.

The average histogram changes only on a long time scale (depending on lighting conditions and sensor biasing) and this fact is exploited by only updating the averaged histogram every $m$'th pulse. The $m$ histograms do not have to be memorized and each event only increases the bin count. The new score function is computed from the accumulated histogram by normalizing it only after the $m$'th pulse.

The score map computation is optimized by accumulating event scores for $o$ laser pulses. Each event requires a lookup of its score and a sum into the score map. After each sum, if the new score value is higher than the previous maximum score for that column, then the new maximum score value and its location are stored for that column. This accumulation increases the latency by a factor of $o$, but is necessary in any case when the DVS events are not reliably generated by each pulse edge.

After the $o$ laser pulses are accumulated, the search of the column wise maxima laser line pixels is based on the maximum values and their locations stored during accumulation. For each column, the weighted mean location of the peak is computed starting at the stored peak value and iterating over pixels up and down from the peak location until the score drops below the threshold value. This way, only a few pixels of the score map are inspected for each column.

The final step is to reset the accumulated score map and peak values to zero. This low-level memory reset is done by microprocessor logic hardware and is very fast.

Results of these optimizations are reported in Results.

## PARAMETER SETTINGS

Because the DVS does analog computation at the pixel level, the behavior of the sensor depends on the sensor bias settings. These settings can be used to control parameters such as the temporal contrast cutoff frequency and the threshold levels. For the experiments described in the following, the bias settings were optimized to report small as well as fast changes. These settings

lead to an increase in noise events which does not affect the performance because they are filtered out successfully with the algorithm described previously. Furthermore, the biases are set to produce a clear peak in the temporal histogram of the OFF events (**Figure 6**). The variation in the peak form for ON and OFF events is caused by the different detection circuits for the two polarities in the pixel (Lichtsteiner et al., 2008) and different starting illumination conditions before the pulse edges.

The parameters for the algorithm are chosen heuristically: The bin size is fixed to 50 us, the scoring function average is taken over a sliding window size $m = 1000$ histograms, the stripe detection is set to average $o = 3$ probability maps, and the peak threshold for the line detection is chosen to be $\Theta_{\text{peak}} = 1.5$.

Firstly, the performance of the stripe extraction algorithm was measured. Because the performance of the system is limited by the strength of the laser used, the capabilities of the DVS using a stronger laser were characterized to investigate the limits of the approach. Finally, a complex 3D terrain was used to assess the performance under more realistic conditions.

## RESULTS

The laser stripe extraction results presented in the following were run in real-time as the open-source jAER-filter *FilterLaserLine* (jAER, 2007) on an Intel Core i7 975 @ 3.33 GHz Windows 7 × 64 platform using Java 1.7u45. The 3D reconstruction was run off-line in Matlab on the same platform.

Comparing the computational cost to process an event (measured in CPU time) between the frame-based and the event-based algorithm with $o = 10$ pulses showed an 1800% improvement from 900 to 50 ns per event. This improvement is a direct result of the sparse sensor output: For each laser line point update, only a few active pixels around the peak value in the score map column are considered, rather than the entire column. At the typical event rate of 500 keps observed in the terrain reconstruction example, using a laser pulse frequency of 500 Hz, a single core of this (powerful) PC is occupied 2.5% of its available processor time using the event-based algorithm. Turning off the scoring function histogram update further decreases compute time to an average of 30 ns/event, only 25 ns more than processing event packets with a "no operation" jAER filter that iterates over packets of DVS events without doing anything else.

### EXTRACTION PERFORMANCE

To assess the line-detection performance of the stripe extraction algorithm, a ground truth was manually established for a scenario in which a plain block of uniform height was passed under the setup. The block was moved at about 2 cm/s to investigate the performance of the laser stripe extraction algorithm at different frequencies. In **Table 1**, the results of these measurements are displayed: "False positives" designates the ratio of events wrongly associated to the line over the total number of events. The performance of the algorithm drops at a frequency of 500 Hz and because the DVS should be capable of detecting temporal contrasts in the kHz regime, this was further investigated. For optimal algorithm performance, each pulse should at least excite one event per column. This is not the case for the line laser pulsed at 500 Hz because the pixel bandwidth at the laser intensity used is limited to about this frequency. Therefore, not every pulse results in a DVS event, and so the laser stripe can only be found in a few columns which leads to a degradation of the reconstruction quality.

**Table 1 | Performance of the line extraction algorithm.**

| Frequency (Hz) | False positives (%) |
|---|---|
| 50 | 0.14 |
| 100 | <0.01 |
| 200 | 0.03 |
| 500 | 5.75 |

*The line laser is not strong enough to perform well at frequencies above 200 Hz.*


**FIGURE 7 | Number of events at a pixel per laser pulse of a 4.75 mW point laser.** Although the event count drop with higher frequencies, the average does not drop below 1 event per cycle even at 2 kHz.

To explore how fast the system could go, another laser setup was used: A stronger point laser (4.75 mW, Class C) was pulsed using a mechanical shutter to avoid artifacts from the rise and fall time of the electronic driver. This point was recorded with the DVS to investigate whether it can elicit more at least one event per polarity and pulse at high frequencies. The measurements in **Figure 7** show that even at frequencies exceeding 2 kHz sufficient events are triggered by the pulse. The mechanical shutter did not allow pulsing the laser faster than 2.1 kHz so the DVS might even go faster. The increase of events per pulse above 1.8 kHz is probably caused by resonances in the DVS photoreceptor circuits which facilitate the event generation. These findings indicate that a system using a sufficiently strong line laser should be capable of running at up to 2 kHz.

### TERRAIN RECONSTRUCTION

As a proof of concept and as well for studying possible applications and shortcomings of the approach, an artificial terrain was designed with a CAD program and it was fabricated on a 3D printer (**Figure 8**). The sensor setup of **Figure 2** was used together with the sled to capture data at a speed of 1.94 cm/s over this terrain using a laser pulse frequency of 200 Hz, translating in the $t_y$ direction (Equation 5). (This slow speed was a limitation of

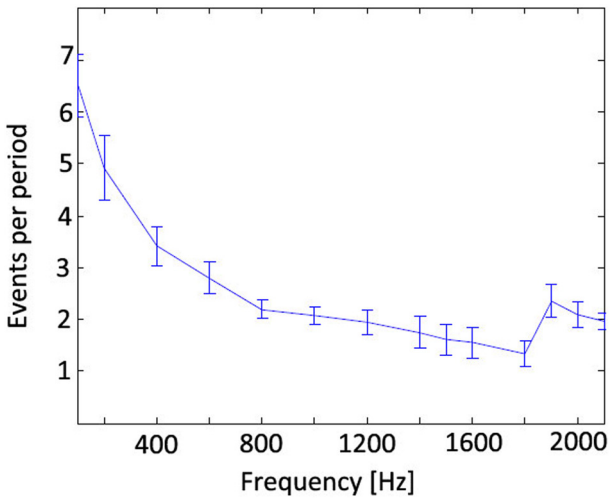



**FIGURE 8 | Artificial 3D rapid prototype terrain used for proof of concept reconstruction.** Blue: area depicted in **Figure 9**, Red: laser line, Black: scan direction.

the DC motor driving the sled.) **Figure 9** shows results of these measurements: **Figure 9A** shows the CAD model and **Figure 9B** shows the raw extracted line data after transformation through Equation 5 using the calibration parameters and the measured sled speed. The blind spots where the laser did not reach the surface and the higher sampling density on front surfaces are evident. These blind spots were filled by applying the MATLAB© function *TriScatteredInterp* on the sample points as shown in **Figure 9C**. Finally, **Figure 9D** shows the error between the reconstruction and model as explained in the next paragraph.

To quantify the error, the data was compared to the ground truth of the CAD model. However, the model and data lack alignment marks and therefore they were first aligned by hand using a global translation. Next, the alignment was refined using the iterative closest point algorithm (ICP; Besl and McKay, 1992), which slightly adjusted the global translation and rotation to minimize the summed absolute distance errors. Thirdly the closest 3D point of the model was determined for each point of the non-interpolated **Figure 9B** raw data and fourthly the distance to this model point was measured. The resulting accuracy i.e., the mean 3D distance between these two points in the 3D data is $1.7 \pm 1.1$ mm, i.e., the mean absolute distance between the sample and data points is 1.7 mm but the errors vary with a standard deviation of 1.1 mm. This accuracy represents $\pm 0.25$ pixel precision of measurement of the laser line given the geometry of the measurement setup. In the resampled, linearly interpolated data shown in **Figure 9D**, most of the error originates from the parts of the surface where the line laser is occluded by the surface, which are interpolated as flat surfaces, and in particular the bottoms of the valleys show the worst error, as could be expected.

An online movie showing the stripe extraction for the terrain reconstruction using a higher laser pulse frequency of 500 Hz is available (Adaptive filtering of DVS pulsed laser line response for terrain surface reconstruction, 2013). This video also shows various stages of the sensor output and laser line extraction.

This recording is done at a sled speed of about 1 m/s using a free-falling sled on an incline, which was not limited by the DC motor speed. In this movie it is also clear that some parts of the terrain where the laser hits the surface at a glancing angle do not generate line data. The movie also shows that background DVS activity caused by image contrast is also effectively filtered out by the algorithm although at this high frequency many pixels do not generate events on each laser pulse.

## DISCUSSION

In this paper the first application of a DVS as a sensing device for terrain reconstruction was demonstrated. An adaptive event-based filtering algorithm for efficiently extracting the laser line position was proposed. The proposed application of DVSs in active sensor setups such as 3D scanners allows terrain reconstruction with high temporal resolution without the necessity of using a power-consuming high-speed camera and subsequent high frame rate processing or any moving parts. The event-based output of DVSs has the potential to reduce the computational load and thereby decreasing the latency and power consumption of such systems. The system benefits from the high dynamic range and the sparse output of the sensor as well as the highly resolved time information on the dynamics in a scene. With the proposed algorithm, temporal correlations between the pulsed stimulus and the recorded signal can be extracted as well as used as filtering criterion for the stripe extraction.

Further improvements to the system are necessary to realize the targeted integration to mobile robots. The Java and jAER overhead would have to be removed and the algorithm would have to be implemented on a lower level programming language (such as C) using the optimized event-based algorithm. A camera motion model and surface reconstruction would have to be integrated into the software and for portability of the system it would need to be embedded in a camera such as the eDVS (Conradt et al., 2009). Motion models could be obtained from 3D surface SLAM algorithms (Newcombe et al., 2011) and/or inertial measurement units (IMUs). The use of DVSs with a higher sensitivity (Serrano-Gotarredona and Linares-Barranco, 2013) would allow using weaker lasers to save power. Higher resolution sensors that include a static readout (Posch et al., 2011; Berner et al., 2013) would facilitate the calibration and increase the resolution. The use of a brighter line laser would allow higher laser pulsing frequencies, a wider sensing range as well as possible outdoor applications.

But despite its immature state, the proposed approach compares well to existing commercial depth sensing systems like the Microsoft Kinect© and a LIDAR optimized for mobile robots such as the SOKUIKI (comparison shown in **Table 2**). The system has a higher maximal sampling rate than the other sensors, a much lower average latency of 5 ms at a 200 Hz pulse rate, and it is more accurate at short distances. These features are crucial for motion planning and obstacle avoidance in fast moving robots. The latency of the proposed approach is, however, dependent on the reliability of the DVS pixel responses, so there is a tradeoff between latency and noise that has not yet been fully

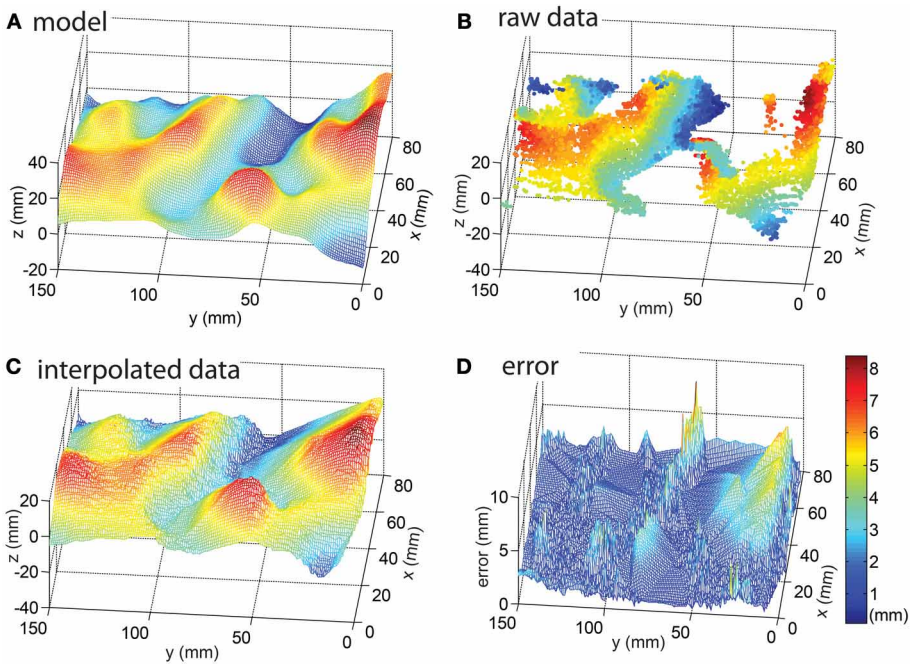



**FIGURE 9 | The reconstructed surface. (A)** CAD model of the surface. **(B)** Measured data points. **(C)** Interpolated reconstruction of the surface using Matlab's *TriScatteredInterp* function. **(D)** Distance between closest reconstruction point and model aligned using ICP (Besl and McKay, 1992). This section of the reconstruction was chosen for display because in the surrounding area border effects were observed caused by the Gaussian profile of the laser line that reduced the DVS event rate to be too low to result in acceptable reconstruction.

**Table 2 | Performance comparison of the proposed approach with existing depth sensors.**

| | This work | Microsoft Kinect for Xbox 360 | LIDAR (SOKUIKI) |
|---|---|---|---|
| Spatial resolution (pixels) | 128 | ∼320 × 240[b] | 680[e] |
| Field of view (degree) | 28° | 58° × 44°[b] | 240°[e] |
| Output data | Surface profile | Depth map | Range profile |
| Accuracy | 2 mm @ 0.45 m (0.45%) | ∼1.5 cm @ 3 m[c] (0.5%) | 3 cm @ 1 m[e] (3%) |
| Power consumption | USB camera + laser: ∼535 mW[a] | 2.25–4.7 W (active)[b] | 2.5 W[e] |
| Max sample rate (Hz) | 500 | 30[d] | 10[e] |
| Average latency (ms) | 5[f] | 45[g]–120[h] | 100[e] |

[a] *DVS:400 mW + Laser: 135 mW.*

[b] *Nominally* 640 × 480 *(Viager, 2011) but spatial pattern used reduces to* ∼ ±*1 pixel in each direction (Andersen et al., 2012).*

[c] *Khoshelham and Elberink, 2012.*

[d] *Kinect for Windows Sensor Components and Specifications.*

[e] *URG-04LX-UG01.*

[f] *200 Hz laser pulse rate.*

[g] *VGA depth map output with Core2 E6600 CPU @ 2.4 GHz (Specs about OpenNI compliant 3D sensor Carmine 1.08 | OpenNI, 2012).*

[h] *Skeleton model w/1 skeleton tracked (Livingston et al., 2012).*

studied, and this tradeoff will also depend on other conditions such as background lighting and surface reflectance. On the downside, the system's spatial resolution is limited by the use of the first-generation DVS128 camera and the field of view for the proposed system is narrow. But these drawbacks are not fundamental and they can easily be improved (e.g., by using newer sensors, shorter lenses and stronger lasers). The limitation that the system does not deliver depth maps but surface profiles could be overcome by projecting sparse 2D light patterns instead of a laser line. The power consumption of 500 mW for the USB camera and laser does not include the power to process the events nor to reconstruct the surface but because the sensor system power consumption is comparably lower, the data processing will probably fit into the power budget of the other two approaches when embedded into a 32-bit ARM-based microcontroller, e.g., as in Conradt et al. (2009). In summary, this paper demonstrates the applicability of DVSs combined with pulsed line lasers to provide surface profile measurement with low latency and low computational cost, but integration onto mobile platforms will require further work.

## ACKNOWLEDGMENTS

This research was supported by the European Union funded project SeeBetter (FP7-ICT-2009-6), the Swiss National Science Foundation through the NCCR Robotics, ETH Zurich, and the University of Zurich. The authors thank the reviewers for their helpful critique which had a big impact on the final form of this paper.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2013.00275/abstract

## REFERENCES

Adaptive filtering of DVS pulsed laser line response for terrain surface reconstruction. (2013). Available online at: http://youtu.be/20OGD5Wwe9Q. (Accessed: December 23, 2013).

Andersen, M. R., Jensen, T., Lisouski, P., Mortensen, A. K., Hansen, M. K., Gregersen, T., et al. (2012). *Kinect Depth Sensor Evaluation for Computer Vision Applications.* Aarhus: Aarhus University, Department of Engineering. Available online at: http://eng.au.dk/fileadmin/DJF/ENG/PDF-filer/Tekniske rapporter/TechnicalReportECE-TR-6-samlet.pdf. (Accessed: December 11, 2013).

Berner, R., Brandli, C., Yang, M., Liu, S.-C., and Delbruck, T. (2013). "A 240 × 180 10 mW 12 us latency sparse-output vision sensor for mobile applications," in *Symposium on VLSI Circuits* (Kyoto), C186–C187.

Besl, P. J., and McKay, N. D. (1992). A Method for Registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 239–256. doi: 10.1109/34.121791

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R., and Delbruck, T. (2009). "A pencil balancing robot using a pair of AER dynamic vision sensors," in *IEEE International Symposium on Circuits and Systems (ISCAS) 2009* (Taipei), 781–784. Available online at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5117867. (Accessed: August 13, 2013). doi: 10.1109/ISCAS.2009.5117867

Delbruck, T., and Lichtsteiner, P. (2007). "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *International Symposium on Circuits and Systems (ISCAS) 2007* (New Orleans, LA), 845–848. Available online at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4252767. (Accessed: August 13, 2013). doi: 10.1109/ISCAS.2007.378038

Forest, J., and Salvi, J. (2002). "A review of laser scanning three-dimensional digitisers," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Lausanne: IEEE), 73–78. Available online at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1041365. (Accessed: August 13, 2013).

jAER. (2007). JAER Open Source Proj. Available online at: http://jaerproject.net. (Accessed: September 17, 2013).

Khoshelham, K., and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors* 12, 1437–1454. doi: 10.3390/s120201437

Kinect for Windows Sensor Components and Specifications. Available online at: http://msdn.microsoft.com/en-us/library/jj131033.aspx. (Accessed: October 23, 2013).

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128 × 128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Livingston, M. A., Sebastian, J., Ai, Z., and Decker, J. W. (2012). "Performance measurements for the Microsoft Kinect skeleton," in *2012 IEEE Virtual Reality Short Papers and Posters (VRW)* (Costa Mesa, CA), 119–120. doi: 10.1109/VR.2012.6180911

Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., et al. (2011). "KinectFusion: real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (Basel), 127–136.

Ni, Z., Pacoret, C., Benosman, R., Ieng, S.-H., and Régnier, S. (2012). Asynchronous event-based high speed vision for microparticle tracking. *J. Microsc.* 245, 236–244. doi: 10.1111/j.1365-2818.2011.03565.x

Orghidan, R., Salvi, J., and Mouaddib, E. M. (2006). Modelling and accuracy estimation of a new omnidirectional depth computation sensor. *Pattern Recognit. Lett.* 27, 843–853. doi: 10.1016/j.patrec.2005.12.015

Palaniappa, R., Mirowski, P., Ho, T. K., Steck, H., Whiting, P., and MacDonald, M. (2011). *Autonomous RF Surveying Robot for Indoor Localization and Tracking.* Gumaraes. Available online at: http://ipin2011.dsi.uminho.pt/PDFs/Shortpaper/49_Short_Paper.pdf

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid-State Circuits* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Raibert, M., Blankespoor, K., Nelson, G., Playter, R., and Big Dog Team. (2008). *BigDog, the Rough-Terrain Quadruped Robot.* Seoul. Available online at: http://web.unair.ac.id/admin/file/f_7773_bigdog.pdf

Robinson, A., Alboul, L., and Rodrigues, M. (2003). "Methods for indexing stripes in uncoded structured light scanning systems," in *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (Plzen; Bory). Available online at: http://wscg.zcu.cz/WSCG2004/Papers_2004_Full/I11.pdf

Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). A 128 × 128 1.5% contrast sensitivity 0.9% FPN 3 μs latency 4 mW asynchronous frame-free dynamic vision sensor using transimpedance preamplifiers. *IEEE J. Solid-State Circuits* 48, 827–838. doi: 10.1109/JSSC.2012.2230553

Siegwart, R. (2011). *Introduction to Autonomous Mobile Robots*. 2nd Edn. Cambridge, MA: MIT Press.

Specs about OpenNI compliant 3D sensor Carmine 1.08 |OpenNI. (2012). Available online at: http://www.openni.org/rd1-08-specifications/. (Accessed: November 12, 2013).

URG-04LX-UG01. Scanning Range Finder URG-04LX-UG01. Available online at: http://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html. (Accessed: October 24, 2013).

Usamentiaga, R., Molleda, J., and García, D. F. (2010). Fast and robust laser stripe extraction for 3D reconstruction in industrial environments. *Mach. Vis. Appl.* 23, 179–196. doi: 10.1007/s00138-010-0288-6

Viager, M. (2011). *Analysis of Kinect for Mobile Robots*. Lyngby: Technical University of Denmark.

Weiss, S., Achtelik, M., Kneip, L., Scaramuzza, D., and Siegwart, R. (2010). Intuitive 3D maps for MAV terrain exploration and obstacle avoidance. *J. Intell. Robot. Syst.* 61, 473–493. doi: 10.1007/s10846-010-9491-y

Yoshitaka, H., Hirohiko, K., Akihisa, O., and Shin'ichi, Y. (2006). "Mobile robot localization and mapping by scan matching using laser reflection intensity of the SOKUIKI sensor," in *IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics* (Paris), 3018–3023.

**Conflict of Interest Statement:** One of the Authors (Tobi Delbruck) is one of the research topic editors. One of the Authors (Tobi Delbruck) has a financial participation in iniLabs, the start-up which commercially distributes the DVS camera prototypes. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 23 August 2013; accepted: 23 December 2013; published online: 17 January 2014.*

*Citation: Brandli C, Mantel TA, Hutter M, Höpflinger MA, Berner R, Siegwart R and Delbruck T (2014) Adaptive pulsed laser line extraction for terrain reconstruction using a dynamic vision sensor. Front. Neurosci. 7:275. doi: 10.3389/fnins.2013.00275*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 21 November 2013
doi: 10.3389/fnins.2013.00223

# Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor

***Tobi Delbruck* and Manuel Lang***

*Department of Information Technology and Electrical Engineering, Institute of Neuroinformatics, UNI-ETH Zurich, Zurich, Switzerland*

***Edited by:***
*André van Schaik, The University of Western Sydney, Australia*

***Reviewed by:***
*Jorg Conradt, Technische Universität München, Germany*
*Gregory K. Cohen, Bioelectronics and Neuroscience Research Group at the MARCS Institute, Australia*

****Correspondence:***
*Tobi Delbruck, Department of Information Technology and Electrical Engineering, Institute of Neuroinformatics, Winterhurerstr. 190, UNI-ETH Zurich, CH-8057 Zurich, Switzerland*
*e-mail: tobi@ini.phys.ethz.ch*

Conventional vision-based robotic systems that must operate quickly require high video frame rates and consequently high computational costs. Visual response latencies are lower-bound by the frame period, e.g., 20 ms for 50 Hz frame rate. This paper shows how an asynchronous neuromorphic dynamic vision sensor (DVS) silicon retina is used to build a fast self-calibrating robotic goalie, which offers high update rates and low latency at low CPU load. Independent and asynchronous per pixel illumination change events from the DVS signify moving objects and are used in software to track multiple balls. Motor actions to block the most "threatening" ball are based on measured ball positions and velocities. The goalie also sees its single-axis goalie arm and calibrates the motor output map during idle periods so that it can plan open-loop arm movements to desired visual locations. Blocking capability is about 80% for balls shot from 1 m from the goal even with the fastest-shots, and approaches 100% accuracy when the ball does not beat the limits of the servo motor to move the arm to the necessary position in time. Running with standard USB buses under a standard preemptive multitasking operating system (Windows), the goalie robot achieves median update rates of 550 Hz, with latencies of $2.2 \pm 2$ ms from ball movement to motor command at a peak CPU load of less than 4%. Practical observations and measurements of USB device latency are provided[1].

**Keywords: asynchronous vision sensor, address-event representation, AER, high-speed visually guided robotics, high frame rate, neuromorphic system, soccer**

## INTRODUCTION

The notion of a "frame" of video data is embedded in machine vision. High speed frame-based vision is expensive because it is based on a series of pictures taken at a constant rate. The pixels are sampled repetitively even if their values are unchanged. Short-latency vision problems require high frame rate and produce massive amount of input data. At high frame rate, few CPU instructions are available for processing each pixel. For example, a VGA 640 × 480 pixel image sensor at 1 kHz frame rate delivers data at a rate of 307 M pixels/s, or a pixel every 3.3 ns. At usable instruction rates of 1 GHz a computer would only be able to dedicate 3 instructions per pixel to processing this information. This high data rate, besides requiring specialized computer interfaces and cabling (Wilson, 2007), makes it expensive in terms of power to deal with the data, especially in real time or embedded devices. Specialized high-frame-rate machine vision cameras with region of interest (ROI) or binning (sub-sampling) capabilities can reduce the amount of data significantly, but the ROI and binning must be controlled by software and the ROI is limited to a single region, reducing its usefulness for tracking multiple objects. Tracking a single object requires steering the ROI to follow the object. The latency of this ROI control must be kept short to avoid losing the object and ROI control can become quite complex to implement. Ref. (Graetzel et al., 2006), for example, describes a fruit-fly wing-beat analyzer that uses Kalman filtering to move the ROI in anticipation of where it should be according to the Kalman filter parameters, and even to time-multiplex the ROI between different parts of the scene. The computer must process all the pixels for each ROI or binned frame of data and ROI control latencies must be kept short if the object motion is not predictable.

By contrast, in the camera used for this paper, data are generated and transmitted asynchronously only from pixels with changing brightness. In a situation where the camera is fixed and the illumination is not varying only moving objects generate events. This situation reduces the delay compared to waiting for and processing an entire frame. Also, processor power consumption is related to the scene activity and can be reduced by shorter processing time and longer processor sleep phases between processing cycles.

This paper describes the results of experiments in low-latency visual robotics using an asynchronous dynamic vision sensor (DVS) (Lichtsteiner et al., 2006, 2007) as the input sensor, a standard PC as the processor, standard USB interfaces, and a standard hobby servo motor as the output.

Specifically, this paper demonstrates that independent pixel event data of a DVS are well-suited for object tracking and real-time visual feedback control. The simple but highly efficient object-tracking algorithm is implemented on a general purpose CPU. The experiments show that such a robot, although based on traditional, cheap, ubiquitous PC components like USB and a standard preemptive operating system (Windows) a simple

[1]During this work the authors were with the Inst. of Neuroinformatics, Winterhurerstr. 190, UNI-ETH Zurich, CH-8057 Zurich, Switzerland., e-mail: tobi@ini.uzh.ch, phone: +41(44) 635-3038.

programmable Java control application achieves reaction times on par with high speed conventional machine vision hardware running on dedicated real-time operating systems consuming the resources of an entire computer.

This paper expands on a brief conference report (Delbruck and Lichtsteiner, 2007) by including the new feature of self-calibration, more detailed descriptions of the algorithms, and new measurements of performance and latency particularly relating to USB interfaces. Other related work that has integrated an event-based neuromorphic vision sensor in a robot includes CAVIAR, a completely spike-hardware based visual tracking system (Serrano-Gotarredona et al., 2009), a pencil balancing robot using a pair of embedded-processor DVS cameras (Conradt et al., 2009a), which was first prototyped using two DVS cameras interfaced by USB (Conradt et al., 2009b), a demonstration of real-time stereo distance estimation computed on an FPGA with 2 DVS cameras (Domínguez-Morales et al., 2012), an embedded FPGA-based visual feedback system using a DVS (Linares-Barranco et al., 2007), and a micro gripper haptic feedback system (Ni et al., 2013) which uses a DVS as one of the two input sensors.

## MATERIALS AND METHODS: GOALIE ARCHITECTURE

The application presented here is a self-calibrating soccer goalie robot (**Figure 1**). The robotic goalie blocks balls shot at a goal using a single-axis arm with only a single degree of freedom. **Figure 1** shows our goalie robot hardware architecture. Players attempt to score by shooting balls at the goal (either by rolling or flicking with their fingernails) and the goalie robot tries to block all balls from entering the goal. Only balls that roll or slide along or near the table surface can be blocked and this limitation is what enables the solution to the blocking problem without stereo vision or some other means of determining the height of the ball over the table. The fact that the balls move along the surface of the table means that their 3D position can (implicitly in this application) be determined from the ball's 2D image position. The goalie is self-calibrating i.e., by visual observation it learns the motor control to arm position relationship. When turned on the goalie is one of 4 distinct states. In the *active* state, the goalie has determined that a ball is approaching the goal that can be blocked and tries to block it. Between balls, the goalie is *relaxed* to the middle position. When no definite balls have been seen for a few seconds, the goalie enters *sleeping* state where it does not respond to every movement in the scene. This state reduces apparently spastic movements in response to people walking by, hands, etc. After several minutes in sleeping state the goalie enters the *learning* in which it recalibrates itself. The goalie wakes up from *sleeping* to become *active* when it again sees a definite ball.

The rest of this section will describe the individual components of the system.

### DYNAMIC VISION SENSOR

Conventional image sensors see the world as a sequence of frames, each consisting of many pixels. In contrast, the DVS is an example of a sensor that outputs digital address events (spikes) in response of temporal contrast at the moments that pixels see changing intensity (Lichtsteiner et al., 2006, 2007; Delbruck et al., 2010) (**Figure 2**). Like an abstraction of some classes of retinal ganglion

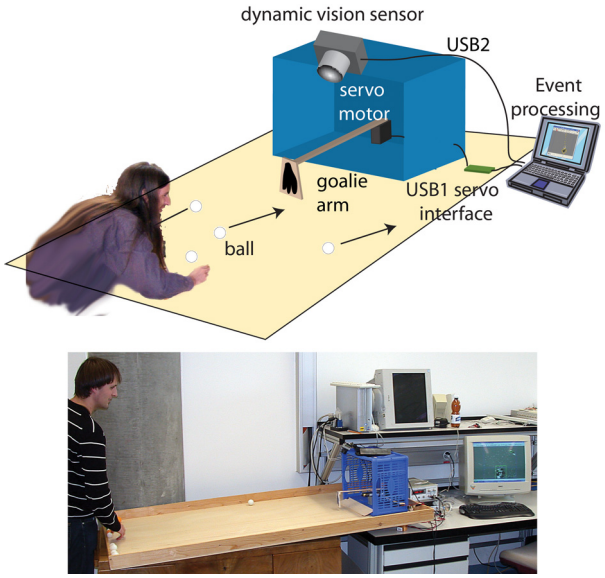



**FIGURE 1 | Goalie robot illustration and a photo of the setup, showing the placement of vision sensor, goalie arm, and goal.** The white or orange balls have a diameter of 3 or 4 cm and are viewed against the light brown wood table. The reflectance ratio between balls and table is about 1.3. The retina view extends out to 1 m from the goal line. The goalie hand is 5 cm wide and the goal is 45 cm wide.

cell spikes seen in biology, each event that is output from the DVS indicates that the log intensity at a pixel has changed by an amount $T$ since the last event. $T$ is a global event threshold which is typically set to about 15% contrast in this goalie robot application. In contrast to biology, the serial data path used requires the events to carry address information of what pixels number has changed. The address encodes the positive or negative brightness changes (ON or OFF) with one bit and the rest of the bits encode the row and column addresses of the triggering pixel. This representation of "change in log intensity" encodes scene reflectance change, as long as the illumination is constant over time, but not necessarily over space. Because this computation is based on a compressive logarithmic transformation in each pixel, it also allows for wide dynamic range operation (120 dB, compared with e.g., 60 dB for a high quality traditional image sensor).

This neuromorphic abstraction of the transient pathway seen in biology turns out to be useful for a number of reasons. The wide dynamic range means that the sensor can be used with uncontrolled natural lighting, even when the scene illumination is non-uniform and includes strong shadows, as long as they are not moving. The asynchronous response property also means that the events have the timing precision of the pixel response rather than being quantized to the traditional frame rate. Thus, the "effective frame rate" is typically several kHz and is set by the available illumination which determines the pixel bandwidth. The temporal redundancy reduction reduces the output data rate for scenes in which most pixels are not changing. The design of the pixel also allows for uniformity of response: the mismatch between pixel contrast thresholds is 2.1% contrast and the event threshold can be set down to 10% contrast, allowing the device

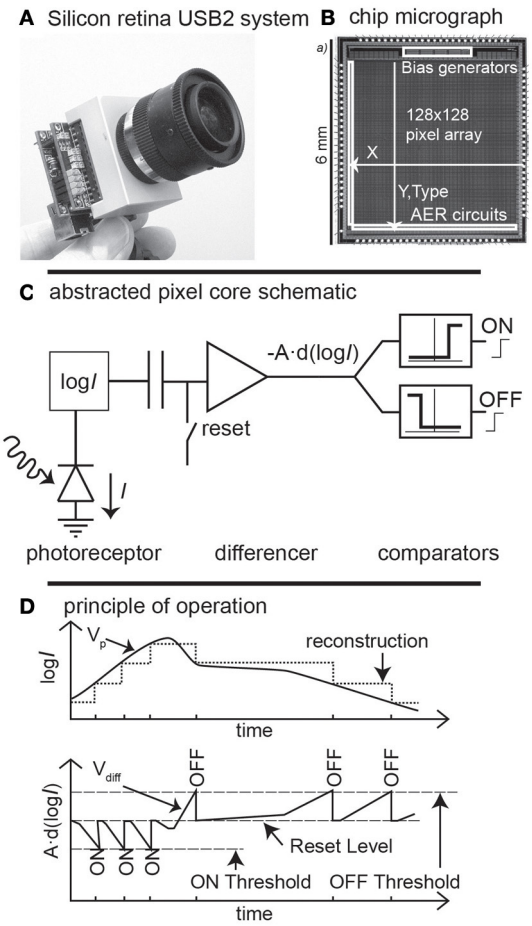



**FIGURE 2 | Characteristics of the dynamic vision sensor (Tmpdiff128).** **(A)** the dynamic vision sensor with its lens and USB2.0 interface. **(B)** A die photograph. Pixels generate address-events, with the address formed from the *x, y*, location and ON or OFF type **(C)** an abstracted schematic of the pixel, which responds with events to fixed-size changes of log intensity. **(D)** How the ON and OFF events are internally represented and output in response to an input signal. Figure adapted from Lichtsteiner et al. (2006).

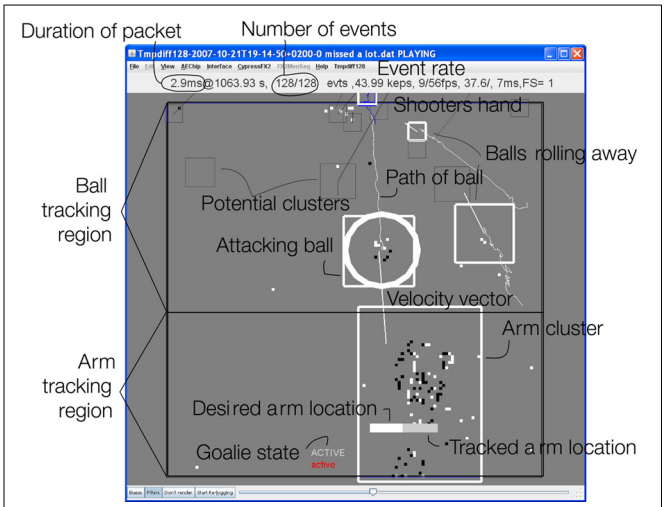



**FIGURE 3 | Snapshot of action showing 128 events (all events within 2.9 ms) from the vision sensor.** It shows 5 tracked objects (the middle 3 are real balls, the top one is the shooter's hand, and the bottom object is the goalie arm). The Attacking ball rolling toward the goal (and being blocked) is marked with a circle; other balls are tracked but ignored. The thin squares represent potential clusters that have not received sufficient support. The velocity vectors of each ball are also shown as a slightly thicker line and have been computed by least squares linear regression over the past 10 packets of events. The goalie arm is being moved to the left bar and the presently tracked location of the arm is shown as a light bar inside the arm cluster. The state of the goalie is indicated as "active" meaning a tracked ball is being blocked. The balls generate average event rates of 3–30 keps (kilo events per second). The mean event rate for this packet was 44 keps.

to sense real-world contrast signals rather than only artificial high contrast stimuli. The vision sensor has integrated digitally controlled biases that minimize chip-to-chip variation in parameters and temperature sensitivity. Equipped with an USB2.0 high-speed interface, the DVS camera delivers the time-stamped address-event representation (**AER**) address-events to a host PC with timestamp resolution of 1 us.

## EVENT-DRIVEN TRACKING ALGORITHM

Events from the DVS are processed inside jAER, an open-source Java software infrastructure for processing event-based sensor outputs (2007). The goalie implementation consists of about 3 k non-comment lines of code. The goalie software implementation is open-sourced in jAER.

The ball and arm tracker is an event-driven cluster tracker described briefly in (Lichtsteiner et al., 2006; Litzenberger et al., 2006) (**Figure 3**) and further enhanced in this work. This algorithm is inspired by the mean-shift approach used in frame-based vision (Cheng, 1995; Comaniciu and Ramesh, 2000). Each "cluster" models a moving object as a source of events. Visible clusters are indicated by the boxes in **Figure 3**. Events that fall within the cluster move the cluster position, and a cluster is only considered supported ("visible") when it has received a threshold number of events. Clusters that lose support for a threshold period are pruned. Overlapping clusters are merged periodically at 1 ms intervals. Cluster positions are updated by using a mixing factor that mixes the old position with the new observations using fixed factors. Thus, the time constant governing cluster position is inversely proportional to the evidence (event rate).

The advantages of the cluster tracker are:

(1) There is no frame correspondence problem because the events continuously update the cluster locations during the movement of the objects, and the faster the objects move, the more events they generate.
(2) Only pixels that generate events need to be processed. The cost of this processing is dominated by the search for the nearest existing cluster, which is a cheap operation because there are only a few clusters.
(3) Memory cost is low because there is no full frame memory, only cluster memory, and each cluster requires only a few hundred bytes of memory.

In the goalie application the objects have a known size and roll on a flat surface so tracked clusters have an image space radius determined by their perspective location in the scene.

The algorithm runs on each packet of combined events received from USB transmission, typically 128 (or fewer):

(1) Pruning: Iterate over all existing clusters, pruning out those clusters that have not received sufficient support. A cluster is pruned if it has not received an event to support it within a given time, typically 10 ms in this application.
(2) Merging: Iterate over all clusters to merge overlapping clusters. This merging operation is necessary because new clusters can be formed when an object grows larger as it approaches the vision sensor. For each cluster rectangle that overlaps the rectangle of another cluster, merge the two clusters into a new cluster and discard the previous clusters. The new cluster takes on the history of the older two clusters and its position is the weighted average of the locations of the source clusters. The averaging is weighted by the number of events in each source cluster. This weighting reduces the jitter in the cluster location caused by merging. This iteration continues as long as there are overlapping clusters.
(3) Positioning: For each event, find the nearest cluster that contains the event. The predicted location of each cluster that is considered in this step is computed using its present cluster location combined with the present cluster velocity estimate and the time between this event and the last one that updated the cluster. This way, an event can be in a cluster's predicted location even if it is not inside the last location of the cluster.
    (a) If the event is within the cluster, add the event to the cluster by pushing the cluster a bit toward the event and updating the last event time of the cluster. The new cluster location $\vec{x}_{n+1}$ is given by mixing the predicted value of the old location $(\vec{x}_n + \vec{v}\Delta t)$, where $\vec{v}$ is the cluster velocity and $\Delta t$ is the time between this event and the last one that updated this cluster, with the event location $\vec{e}$ using an adjustable mixing factor $\alpha \approx 0.01$:

$$\vec{x}_{n+1} = (1 - \alpha)(\vec{x}_n + v_n \Delta t) + \alpha \vec{e}$$

    This step implements a predictive tracker by giving clusters a kind of momentum that helps keep clusters attached to rapidly moving objects even if they emit few events. If the present event appears at the predicted location of the cluster, the clusters location is only modified to the predicted location. Events from the leading edge of the object pull the cluster forward and speed it up, while events at the cluster's trailing edge pull the cluster back and slow it down.
    (b) If the event is not in any cluster, seed a new cluster if there are spare unused clusters to allocate. The goalie typically uses 20 potential clusters.

A cluster is not marked as "visible" until it receives a certain number of events (typically 10 in the goalie) and is moving at a minimum speed (typically 20 pixels/s in the goalie).

The goalie robot determines the ball object as the cluster that will next hit the goal line, based on the cluster positions and velocities. The ball cluster's location and velocity measurement are used to position the servo to intercept the ball. If there is no threatening ball, the goalie relaxes.

Accurate and rapid measurement of cluster velocity is important in the goalie application because it allows forward prediction of the proper position of the arm. A number of algorithms for estimating cluster velocity were tried. Low-pass filtering the instantaneous cluster velocity estimates that come from the cluster movements caused by each event is cheap to compute, but was not optimal because the lowpass filter takes too long to settle to an accurate estimate. The method presently used is a "rolling" least squares linear regression on the cluster locations at the ends of the last N packets of events. This method is almost as cheap to compute because it only updates least-squares summary statistics by adding in the new location and removing the oldest location, and it instantaneously settles to an "optimal" estimate. A value of $N = 10$ computes velocity estimates over about the last 10–30 ms of ball location.

## GOALIE SELF-CALIBRATION

In an earlier version of the goalie (Delbruck and Lichtsteiner, 2007) the arm position was specified by adjustable "offset" and "gain" parameters that mapped a motor command to a certain arm position. It was difficult to calibrate this goalie accurately and every time the aim of the vision sensor was adjusted or moved accidentally (the goal bounces around quite a bit due to the arm movements) laborious manual calibration had to be done again. The goalie arm was also not visible to the goalie and so there was no straightforward way for the goalie to calibrate itself. In the present goalie, the orientation of the arm was changed so that it swings on a horizontal plane rather than hanging like a pendulum and used a wide angle lens (3.6 mm) that allows the vision sensor to see both incoming balls and the goalie's hand. The horizontal arm orientation has the additional advantage that it allows the goalie to block corner shots much better.

Goalie calibration occurs in the learning state. When active, the arm position is tracked by using a motion tracker like the ball tracker but with a single cluster sized to the size and aspect ratio of the arm (**Figure 3**). The $x$ position of the arm tracker is the arm coordinate in image space. The motor is controlled in coordinates chosen in software to span [0-1] units. The calibration algorithm has the following steps demonstrated by the data shown in **Figure 4**:

(1) The present calibration is checked by randomly placing the arm in 5 pixel positions (using the current calibration parameters to determine the mapping) and measuring the actual arm position in pixel coordinates. If the average absolute error is smaller than a threshold (typically 5 pixels) calibration is finished. In the situation shown in **Figure 4A**, the calibration is initially very incorrect, and learning is initiated.
(2) If calibration is needed, the algorithm places the arm in randomly chosen motor positions within a range specified in a GUI interface to be in roughly in the center of the field of view. (The GUI allows interactive determination of the

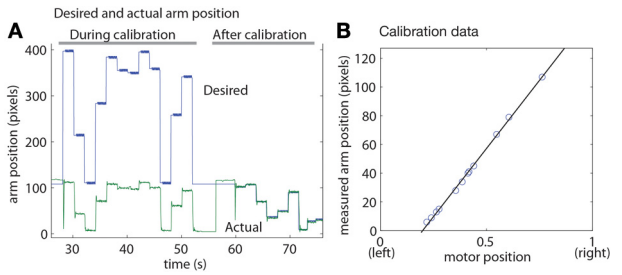



**FIGURE 4 | Goalie self-calibration. (A)** Desired and actual arm position before and after calibration (the desired positions are the inverse mapping from motor commands in the allowed physical range to the pixel space). **(B)** Example measured arm positions vs. servo command used for least squares linear regression calibration.

servo motor limits). For each placement position, the actual pixel position is measured from the arm tracker. Typically 20 points are collected. A least-squares linear regression then determines the linear mapping from desired pixel position to motor position (**Figure 4B**). The algorithm then goes back to step 1. In **Figure 4A**, the calibration is checked after fitting and is satisfactory, so the calibration algorithms is terminated.

Calibration typically achieves accuracy within 2–5 pixels over the entire range. The linear approximation $\sin(x) = x$near $x = 0$ was sufficiently accurate that it was not necessary to account for the sinusoidal relation between servo command and location of the arm across the goal.

### USB INTERFACES AND SERVO CONTROL

Both the vision sensor and the servo controller use the Java interface provided by the Thesycon Windows USB driver development kit for Windows (www.thesycon.de). The servo commands are sent to the microcontroller in a separate writer thread that takes commands placed in a queue by the retina event processing thread. This decoupling allows for full speed USB 2.0 event processing although servo controller commands are transmitted using USB full-speed protocol at 12 Mbps (Axelson, 2001). The servo motor control command rate is 500 Hz, because each command requires 2 polls from the host controller and the minimal possible USB2.0 full-speed polling interval of 1 ms. The command queue length is set to one to minimize latency. New commands replace old ones if they have not yet been transmitted. Likewise, the incoming DVS events are transmitted in 128-event (or smaller) blocks and processed in a high priority thread that runs independently from the GUI or rendering threads. The DVS uses a USB2.0 high-speed interface with a data rate of 480 Mbps and a polling interval of 128 us. The USB interface threads were set to high priority, with highest priority given to the servo writing thread. Java's maximum priority is equivalent to Windows TIME_CRITICAL priority (Oaks and Wong, 2004).

A HiTec HS-6965 MG digital servo moves the goalie arm. This $120 hobby digital servo accepts pulse-width modulation (PWM) input up to at least the183 Hz frequency that we used and is rated to rotate 60° with no load in 100 ms. It can move the 40 cm long 20 g mass arm across the goal in about 100 ms and is slightly (∼10%) underdamped with the goalie arm as only load. Other fast servos can be severely underdamped and actually oscillate (e.g., the Futaba S9253). The remaining overshoot with the HiTec servo is enough that the servo occasionally overshoots its intended location enough that the ball is not blocked.

A custom board based on the Silicon Labs C8051F320 USB2.0 full-speed microcontroller (www.silabs.com) interfaces between the PC and the servo motor. The microcontroller accepts commands over a USB bulk endpoint (Axelson, 2001) that program the PWM output width. The servo motor is powered directly by the 5V USB VBUS and 0.5F of ultracapacitor on the controller board helps to ballast the 5V USB VBUS voltage. The servo controller design is open-sourced in jAER (2007).

The servo arm is constructed from a paint stirrer stick with a balsa wood "hand" glued to its end. A goal of this project was to make this hand as small as possible to demonstrate the precision of tracking. The hand width used in this study was about 1.5 times the ball width (**Figure 1**).

## RESULTS

Ordinarily a good shooter can aim most of the shots within the goal; thus a good shooter can potentially score on most shots. In a trial with several experienced shooters who were told they could take as much time as they needed to shoot, it required an average of 40 shots to score 10 goals. This means that each ball had to be shot about 4 times to score once, representing a shot success rate of 25%. A post experiment analysis of the data showed that the shooters could potentially have scored on 75% of their shots, with the rest of the shots representing misses wide of the goal (the shooters were intentionally aiming at the corners of the goal). Therefore, they had 30 shots on the goal and the goalie blocked 20 of these shots. The missed blocks consisted of a mixture of shots were not blocked for three reasons, ranked from highest to lowest occurrence: (1) they were so hard that they exceeded the ability of the servo to move the arm to the correct position in time; (2) tracking noise so that the arm position was not correctly computed well-enough; (3) servo overshoot, where the servo tries to move the arm to the correct position but because of the underdamped dynamics, the arm momentarily overshoots the correct position, allowing the ball to pass by.

The cluster tracker algorithm is effective for ignoring distracters In **Figure 3** four balls are simultaneously tracked. The topmost "ball" is probably the shooter's hand. Two balls are rolling away from the goal and are thus ignored. One is approaching the goal and the arm is moving to block it, based on the ball's position and velocity. Ignoring the many distracters would be impossible using a simpler method of ball tracking, such as median event location. **Figure 5** shows the dynamics of a single blocking event for a ball that was shot quite fast, so that that it covers the distance from the top of the scene to the goal in about 100 ms. During the ball's 100 ms approach, about 50 packets of events, and thus samples of the ball position ("*ballx*" and "*bally*"), are captured by the tracker. The bounce off the arm is visible as the inflection in *bally*. The "desired arm position" is shown also as a function of time and is computed from *ballx*, *bally*, and the

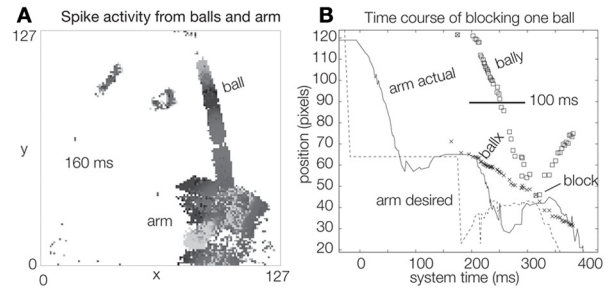



**FIGURE 5 | Single shot dynamics. (A)** 2D histogram of spike activity caused by balls and goalie arm over 160 ms. **(B)** Time course of blocking one ball.

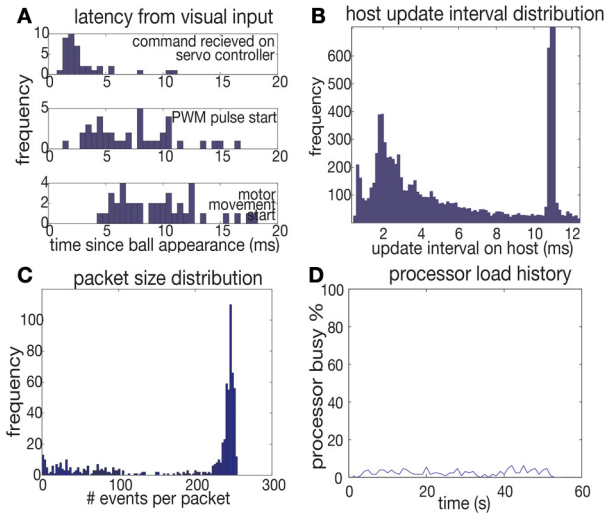



**FIGURE 6 | Statistics. (A)** Latency measurements. Stimulus was flashing LED turned on at time 0. **(B)** Host packet processing interval distribution during normal operation while goalie is under attack. **(C)** Histogram of number of events per packet during normal operation. **(D)** Processor load during normal operation (2 balls/second attack).

ball $x$ and $y$ velocities (not shown). The ball velocities are estimated by rolling linear regressions over the past 10 ball position samples for *ballx* and *bally* vs. time. The "actual arm position" is the position of the arm as measured by the arm tracker and it can be seen that the arm requires about 80 ms to move to the correct blocking position and also exhibits about 10% overshoot which is due to slight under-damping in the servo's controller. The response latency is dominated by the arm movement and the delay between knowing the desired arm position and the initiation of arm movement.

Events are processed by the goalie software at a rate of 2 Meps (million events per second) on a 2.1 GHz Pentium M laptop running Windows XP, Java JVM version 1.6. During typical goalie operation, the average event rate is 20 keps, varying between <1 keps when idle to a maximum of 100 keps during active 10 ms windows of time. For buffers of 128 events processing the goalie code requires about 60 us. **Figure 6B** shows a histogram of processing intervals as recorded on the host PC using Java's *System.nanoTime()*. The median interval is 1.8 ms (the peak in the histogram at 10 ms is caused by forced transfers of data from the vision sensor at 100 Hz rate even when the USB FIFOs have not filled). During processing the computer's CPU load never rises over 4% (**Figure 6D**).

In this system sensor-to-computer latency is dominated by the USB FIFO filling time. The vision sensor pixel latency is inversely proportional to illumination (Lichtsteiner et al., 2007) and is about 100 us at normal indoor office illumination levels of 500 lux. A single ball that produces events at a peak rate of 100 keps causes a device-side 128-event USB packet about every 1 ms, although bursts of events can cause USB transfers that are received as often as every 128 us, the minimum USB2.0 high-speed polling interval. Increased retina activity (caused, say, by the arm movement) actually reduces this latency, but only because the USB device FIFO buffers are filled more rapidly. We used host side USB packet sizes of 256 events to match the maximum 500 Hz rate of writing commands to the servo motor, and the distribution of packet sizes reflects this (**Figure 6C**).

To measure latency, an artificial stimulus consisting of a flashing LED was set up so that it could be activated in bursts to mimic an instantaneously appearing ball. The servo controller was programmed to toggle an output pin when it received a servo motor command. The start of PWM output from the servo controller and the actual start of motor movement were measured. (The motor movement was measured from the power supply drop on the servo power supply). The measured median latency of 2.2 ms between the beginning of the LED flashing and the microcontroller output is the response latency leaving out the latency of the random PWM phase and the servo motor (**Figure 6A**). This latency was achieved by setting the servo controller USB2.0 full speed interrupt polling interval to 1 ms in the device's USB descriptor (Axelson, 2001); using the default polling interval of 10 ms resulted in substantially higher median latency of 5.5 ms that varied approximately bi-modally between 3 and 10 ms. The total latency for actuating the motor (5–15 ms) is dominated by the variable delay of PWM phase. The 183 Hz servo pulse frequency used in the robot has a period of 5.5 ms. A custom servo which directly accepted USB commands could reduce servo latency to about 1–2 ms, the delay to send a single USB1.1 command.

## CONCLUSION

The main achievement of this work is the concrete demonstration of a spike-event driven hybrid of a neuromorphic-sensor coupled to conventional procedural processing for low latency object tracking, sensory motor processing, and self-calibration. Secondary achievements are developments of robust and high speed event-based object tracking and velocity estimation algorithms. This paper also reports practical observations on the use of USB interfaces for sensors and actuators.

The goalie robot can successfully block balls even when these are low contrast white-on-gray objects and there are many background distracters. Running with standard USB buses for vision

sensor input and servo-motor output under a standard preemptive multitasking operating system, this system achieves median update rates of 550 Hz, with latencies of 2.2 ± 2 ms from ball movement to motor command at a peak CPU load of less than 4%.

A comparable system based on using a standard image sensor would require a frame rate of at least 500 Hz. At the same spatial resolution (16 k pixels), a computer would need to continuously process 16 MBps of raw pixel information (with an 8-bit sensor output) to extract the basic visual information about changing pixels. Although this computation is certainly possible, the scaling to higher resolution is very unfavorable to the frame-based approach. Increasing the resolution to VGA resolution (640 × 480) at 1 kHz, for instance, would require processing 307 MBps, about 3 times the effective capacity of a high speed USB 2.0 interface and would allow only 3.3 ns per pixel of processing time. A VGA-sized DVS would generate about 18 times more data than the 128 × 128 sensor used for this paper if the objects filled a proportionally larger number of pixels, but even then the processing of the estimated 400 keps from the sensor would barely load a present-day's microprocessor CPU load and would be within the capabilities of modestly-powered embedded processors. As demonstrated by this work and other implementations (Linares-Barranco et al., 2007; Conradt et al., 2009a; Domínguez-Morales et al., 2012; Ni et al., 2013), the use of event-driven sensors can enable faster and lower-power robots of the future.

## ACKNOWLEDGMENTS

This work was supported by the University of Zurich and ETH Zurich, the Swiss NCCR Robotics, and the EU project SEEBETTER. The authors gratefully acknowledge the opportunity to prototype this system at the Telluride Neuromorphic Engineering Workshop.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2013.00223/abstract

## REFERENCES

(2007). *jAER Open Source Project: Real Time Sensory-Motor Processing for Event-Based Sensors and Systems.* Available online at: http://www.jaerproject.org

Axelson, J. (2001). *USB Complete.* Madison, WI: Lakeview Research.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Analysis Mach. Intell.* 17, 790–799. doi: 10.1109/34.400568

Comaniciu, D., and Ramesh, V. (2000). "Mean shift and optimal prediction for efficient object tracking," in *IEEE International Conference on Image Processing (ICIP)* (Vancouver, BC), 70–73.

Conradt, J., Berner, R., Cook, M., Delbruck, T. (2009a). "An embedded AER dynamic vision sensor for low-latency pole balancing," in *5th IEEE Workshop on Embedded Computer Vision (in Conjunction with ICCV 2009)* (Kyoto), 1–6.

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R. J., and Delbruck, T. (2009b). "Live demonstration: a pencil balancing robot using a pair of AER dynamic vision sensors," in *ISCAS 2009,* (Taipei), 781–785.

Delbruck, T., and Lichtsteiner, P. (2007). "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *ISCAS 2007* (New Orleans, LA), 845–848.

Delbruck, T., Linares-Barranco, B., Culurciello, E., and Posch, C. (2010). "Activity-driven, event-based vision sensors," *Presented at the IEEE International Symposium on Circuits and Systems* (Paris), 2426–2429.

Domínguez-Morales, M., Jimenez-Fernandez, A., Paz-Vicente, R., Jimenez, G., and Linares-Barranco, A. (2012). "Live demonstration: on the distance estimation of moving targets with a stereo-vision AER system," in *IEEE International Symposium on Circuits and Systems (ISCAS) 2012,* (Seoul), 721–725.

Graetzel, C. G., Fry, S. N., and Nelson, B. J. (2006). A 6000 Hz computer vision system for real-time wing beat analysis of Drosophila. *Biorob* 2006, 278–284. doi: 10.1109/BIOROB.2006.1639099

Lichtsteiner, P., Posch, C., and Delbruck, T. (2006). "A 128×128 120dB 30mW asynchronous vision sensor that responds to relative intensity change," in *Visuals Supplement to ISSCC Digest of Technical Papers* (San Francisco, CA), 508–509 (27.9).

Lichtsteiner, P., Posch, C., and Delbruck, T. (2007). A 128×128 120dB 15us latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Linares-Barranco, A., Gómez-Rodríguez, F., Jiménez, A., Delbruck, T., and Lichtsteiner, P. (2007). "Using FPGA for visuo-motor control with a silicon retina and a humanoid robot," in *ISCAS 2007,* (New Orleans, LA), 1192–1195.

Litzenberger, M., Posch, C., Bauer, D., Schön, P., Kohn, B., Garn, H., et al. (2006). "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *IEEE Digital Signal Processing Workshop 2006* (Grand Teton, WY), 173–178.

Ni, Z., Bolopion, A., Agnus, J., Benosman, R., and Regnier, S. (2013). Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Transactions on Robotics* 28, 1081–1089. doi: 10.1109/TRO.2012.2198930

Oaks, S., and Wong, H. (2004). *Java Threads.* O'Reilly.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R. Gomez-Rodriguez, F., et al. (2009). CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER hardware sensory–processing– learning–actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Wilson, A. (2007). Beyond camera link: looking forward to a new camera/frame grabber interface standard. *Vis. Syst. Design.* 12, 79–83. Available online at: http://www.vision-systems.com/articles/print/volume-12/issue-10/features/product-focus/beyond-camera-link.html

**Conflict of Interest Statement:** The spinoff inilabs GmbH of the Inst. of Neuroinformatics is actively marketing dynamic vision sensor technology, selling vision sensor prototypes, and supporting users of the technology. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 07 October 2013; paper pending published: 02 November 2013; accepted: 05 November 2013; published online: 21 November 2013.*

*Citation: Delbruck T and Lang M (2013) Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. Front. Neurosci. 7:223. doi: 10.3389/fnins.2013.00223*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 13 December 2013
doi: 10.3389/fnins.2013.00234

# Event-driven visual attention for the humanoid robot iCub

***Francesco Rea[1]*, Giorgio Metta[2] and Chiara Bartolozzi[2]***

[1] *Robotics, Brain and Cognitive Science, Istituto Italiano di Tecnologia, Genova, Italy*
[2] *iCub Facility, Istituto Italiano di Tecnologia, Genova, Italy*

***Edited by:***
*Tobi Delbruck, University of Zurich and ETH Zurich, Switzerland*

***Reviewed by:***
*Theodore Yu, Texas Instruments Inc., USA*
*Nabil Imam, Cornell University, USA*

****Correspondence:***
*Francesco Rea, Robotics, Brain and Cognitive Science, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova, Italy*
*e-mail: francesco.rea@iit.it*

Fast reaction to sudden and potentially interesting stimuli is a crucial feature for safe and reliable interaction with the environment. Here we present a biologically inspired attention system developed for the humanoid robot iCub. It is based on input from unconventional event-driven vision sensors and an efficient computational method. The resulting system shows low-latency and fast determination of the location of the focus of attention. The performance is benchmarked against an instance of the state of the art in robotics artificial attention system used in robotics. Results show that the proposed system is two orders of magnitude faster that the benchmark in selecting a new stimulus to attend.

**Keywords: visual attention, neuromorphic, humanoid robotics, event-driven, saliency map**

## 1. INTRODUCTION

For successfully interacting with the environment in daily tasks, it is crucial to quickly react to ubiquitous dynamic stimuli. However, reaction time of state-of-the-art robotic platforms is limited by the low temporal resolution of sensory data acquisition and by the time required to process the corresponding sensory data.

In conventional robotic systems, sensory information is available in a sequence of "snapshots" taken at regular intervals. Highly redundant data are received at fixed frame-rate. High dynamics can be sensed only by increasing the sampling rate, at the cost of increasing the quantity of data that needs to be transmitted, stored and processed.

Additionally, the available bandwidth limits the amount of information that can be transmitted, and the available computing platforms limit the speed at which data can be processed, forcing a compromise between resolution and speed. As a result, current robotic systems are less efficient in reacting appropriately to unexpected, dynamic events (Delbruck, 2008). For example, in robotic soccer competitions (e.g., Robocup, 2011), the performance strongly depends on the latencies in the perception loop, where the robot has to detect, track and predict the trajectory of the ball, to plan where and when it should be catched. For the same reason, in manipulation tasks, unexpected failures of the grasping are difficult to correct online, resulting in the fall of the object to be grasped. On the contrary, robotic systems equipped with vision neuromorphic chips show remarkable performance in tracking (Serrano-Gotarredona et al., 2009), ball goalkeeping and pencil balancing (Conradt et al., 2009).

Differently from main-stream state-of-the art vision systems that repeatedly sample the visual input, event-driven vision (Camunas-Mesa et al., 2012; Wiesmann et al., 2012) samples changes in the visual input, being driven by the stimuli, rather than by an external clock. As such, event-driven systems are inherently more efficient because they acquire, transmit and perform computation only when and where a change in the input has been detected, removing redundancies at the lowest possible level.

Selective attention is a key component of artificial sensory systems; in robotics, it is the basis for object segmentation (Qiaorong et al., 2009), recognition (Miau et al., 2001; Walther et al., 2005) and tracking (Ouerhani et al., 2005), for scene understanding and action selection for visual tracking and object manipulation. It is also used in navigation, for self-localization and simultaneous localization and mapping (SLAM) (Frintrop and Jensfelt, 2008). Moreover, the implementation of biologically inspired models of attention is helpful in robots that interact with human beings. Engaging attention on similar objects can be the basis for a common understanding of the environment, of shared goals and hence of successful cooperation. State-of-the art artificial attention systems, based on traditional video acquisition, suffer from the high computational load needed to process each frame. Extreme computational demand limits the speed of the selection of new salient stimuli and therefore the dynamics of the attention scan path. Specific implementations of such models have been explicitly developed for real-time applications, exploiting either parallelization on several CPUs (Itti, 2002; Siagian et al., 2011) or dedicated hardware (Ouerhani and Hügli, 2003), or the optimization and simplification of the algorithms (Itti et al., 1998; Frintrop et al., 2007) for the extraction of features from images, or combination of them (Bumhwi et al., 2011).

An alternative approach is the implementation of simplified models of attention systems based on frame-less event-driven neuromorphic vision sensors, so far realized with the design of *ad hoc* dedicated hardware devices (Bartolozzi and Indiveri, 2009; Sonnleithner and Indiveri, 2012).

Along this line of research, we developed an event-driven, attention system capable of selecting interesting regions of the visual input with a very short latency. The proposed system exploits low latency, high temporal resolution and data compression given by event-driven dynamic vision sensors, as well as an efficient strategy for the computation of the attention model that directly uses the output spikes from the sensors. The proposed implementation is therefore fully "event-driven", exploiting the advantages offered by neuromorphic sensors at its maximum.

Intermediate hybrid approaches can be implemented by reconstructing frames from the events and applying the vast collection of available standard machine vision algorithms. However, this approach would suffer from errors in the frame reconstruction due to drifts in the gray level calculation, it would increase the latency of the response and loose the temporal resolution gained by the use of event-driven sensors, hindering the full exploitation of the neuromorphic approach advantages.

The output of the *Event-Driven Visual Attention (EVA)* system has been implemented for the humanoid robot iCub which will therefore be able to quickly orient its gaze, scrutinize and act on the selected region and react to unexpected, dynamical events. Additionally, it can be of generic interest for robotics systems with fast actuation.

In the following, we will describe EVA, show the improved latency in the selection of salient stimulus and compare its performance with the well-known state-of-the art frame-based selective attention system from the iLab Neuromorphic Vision C++ Toolkit (iNVT),[1] developed at the University of Southern California.

## 2. METHODS

The selective attention system described in this work has been developed on the iCub humanoid robot (www.icub.org) and is entirely based on the input from non-standard sensors. Such sensors use a new way of encoding information based on a custom asynchronous communication protocol Address Event Representation (AER). In the following we shortly describe the custom hardware and software modules developed for the attention system implementation.

### 2.1. HARDWARE

The robot is equipped with two asynchronous bio-inspired *Dynamic Vision Sensors (DVS)* (Lichtsteiner et al., 2008). It features three degrees of freedom in the eyes to realize the tilt, vergence and version movements required for the implementation of active vision. As opposed to the traditional "frame-based" approach, in the DVS each pixel responds to local variations of contrast. It emits an asynchronous digital pulse ("spike" or "event") when the change of the logarithm of light intensity exceeds a pre-defined threshold. This bio-inspired sensory transduction method is inherently efficient, as it discards redundancies at the lowest level, reducing the data acquisition, transfer, storage and processing needs. This technique preserves the high dynamic content of the visual scene with a temporal granularity of few hundreds of nanoseconds.

The visual system is entirely based on the AER protocol (Deiss et al., 1998). The sensors asynchronously send digital spikes or "events" that signal a relative contrast change in the pixel. The address transmitted with the event corresponds to the identity of the active pixel. Information is self encoded in the timings of the spikes.

A dedicated printed circuit board located in the head of the robot hosts a Field Programmable Gate Array (FPGA) and an embedded processor specialized for asynchronous data, the General Address Event Processor (GAEP) (Hofstaetter et al., 2010). The FPGA merges the data streams from left and right camera sensors and interfaces them with the GAEP. The GAEP provides effective data processing, protocol verification and accurate time-stamping of the events, with a temporal resolution of **160 ns**. Processed events are connected to the rest of the system thanks to an USB connection to a PC104 embedded CPU. The PC104 gathers the data and passes them to the processing infrastructure of the iCub (Metta et al., 2006).

### 2.2. SOFTWARE

An application running on the embedded PC104 configures the sensors in the preferred operating state. The same software module reads the data through the USB port, checking for protocol errors and formatting the stream of asynchronous events. Each address event (AE) is composed of the emitting pixel address and the corresponding time-stamp. The application sends the received collection of events on the gigabit network where distributed processing takes advantage from middleware YARP[2] library. From this point, any process connected to the network can acquire data and perform computation. There is no limit in the number of nodes that can be recruited for processing events.

Finally, specific classes are used to efficiently transmit and (un)mask the AER stream into a dedicated format. The AE format consists in: *address event*, *polarity*, *timestamp* and *type*. The structure transparently manages events from the DVS sensor, as well as generic events such as complex objects deriving from clustering and feature extraction (Wiesmann et al., 2012).

Buffers of asynchronous data are handled with a two-threads method. N-buffering is used to guarantee concurrent access to data in process, thus avoiding conflicts and allowing each module to run at the desired rate irrespective of the incoming flow of event. Examples of developed modules are used to: display DVS activity, generate feature maps, perform weighted combination of multiple feature maps.

### 2.3. EVENT DRIVEN VISUAL ATTENTION—EVA

EVA is an event-driven reduced implementation of the saliency-map based attention model proposed by Koch and Ullman (1985) and Itti and Koch (2001). In this foundational work, the authors propose a biophysically plausible model of bottom–up attention where multiple feature maps concur to form a unique saliency map used to compute the location of the focus of attention. Each feature map encodes for a characteristic of the visual input such as color opponency, orientation, contrast, flicker, motion, etc. computed at different spatial scales. These maps are then normalized and summed together to form the final saliency map. The saliency map topologically encodes for local scene conspicuity, irrespective of the feature dimension that has contributed to its salience. That is, an active location in the saliency map encodes the fact that this location is salient, no matter whether it corresponds to a 45° oriented object in a field of prevalent orientation of 90°, or to a stimulus moving in a static background. Eventually, a winner-take-all (WTA) network selects regions in the map in order of decreasing saliency, and guides the deployment of the focus of

[1] http://ilab.usc.edu/toolkit/home.shtml

[2] Yet Another Robotic platform.

attention and gaze. In EVA, events from the visual sensor concur to generate a number of feature maps. **Figure 1** shows the model and the distribution of the diverse computing modules on the hardware platform described in paragraph 2.1. Once collected by the dedicated hardware, the sensor's events are sent to software modules that extract diverse visual features. The corresponding feature maps are then normalized and summed. The resulting saliency map is then transmitted to a WTA network that generates the attentional shifts.

### 2.3.1. Feature extraction

In EVA a number of features are extracted from the DVS output to populate diverse feature maps. As the DVS does not convey information about color or absolute intensity, we implemented a subset of feature maps from Itti and Koch (2001): contrast, orientation (0°, 45°, 90°, −45°) and flicker map. Specifically, the flicker map encodes for the scene temporal changes and in EVA it is implemented by directly using the sensor's output. Contrast and orientation feature maps are generated by the output of filters inspired by receptive fields of center-surround retinal ganglion cells and simple cells of primary visual cortex (Hubel and Wiesel, 1962; Movshon et al., 1978; De Valois et al., 1982; Kandel et al., 2000), respectively. Receptive field activation is usually obtained by convolving the image with DOG (Difference of Gaussians) and Gabor filters, respectively. On the contrary, EVA uses a much simpler and efficient implementation: the *mapping*. In the mapping, a RF is defined as a look-up table. The level of activation of the RF increases when it receives ON-spikes from the DVS pixel located in the ON-region of the RF and OFF-spikes in the OFF-region. If the neuron does not receive any spike over time, the activation decreases. When the neuron activation crosses a threshold, it generates a new event in the corresponding location of the feature map. **Figures 2A,B** show two center-surround RFs. Each RF has a defined location in the visual space and a specific size. The algorithms below explain the procedure that generates the response of the RF.

The visual field is covered with RFs following a multiscale approach. In the current implementation, we use two different scales with $4 \times 4$ and $8 \times 8$ pixels receptive fields. **Figures 2C,D** show the RFs of oriented cells with different sizes. Sub-regions contribute to facilitation for aligned RFs: spikes from the visual field contributing to the activation of the RF at the border of the elongated central region (in green) contribute to the activation of neighboring RFs aligned along the same orientation. This feature enhances the representation of long oriented edges (Ferster and Koch, 1987) by reinforcing the activity of RFs responding to the same oriented edge.

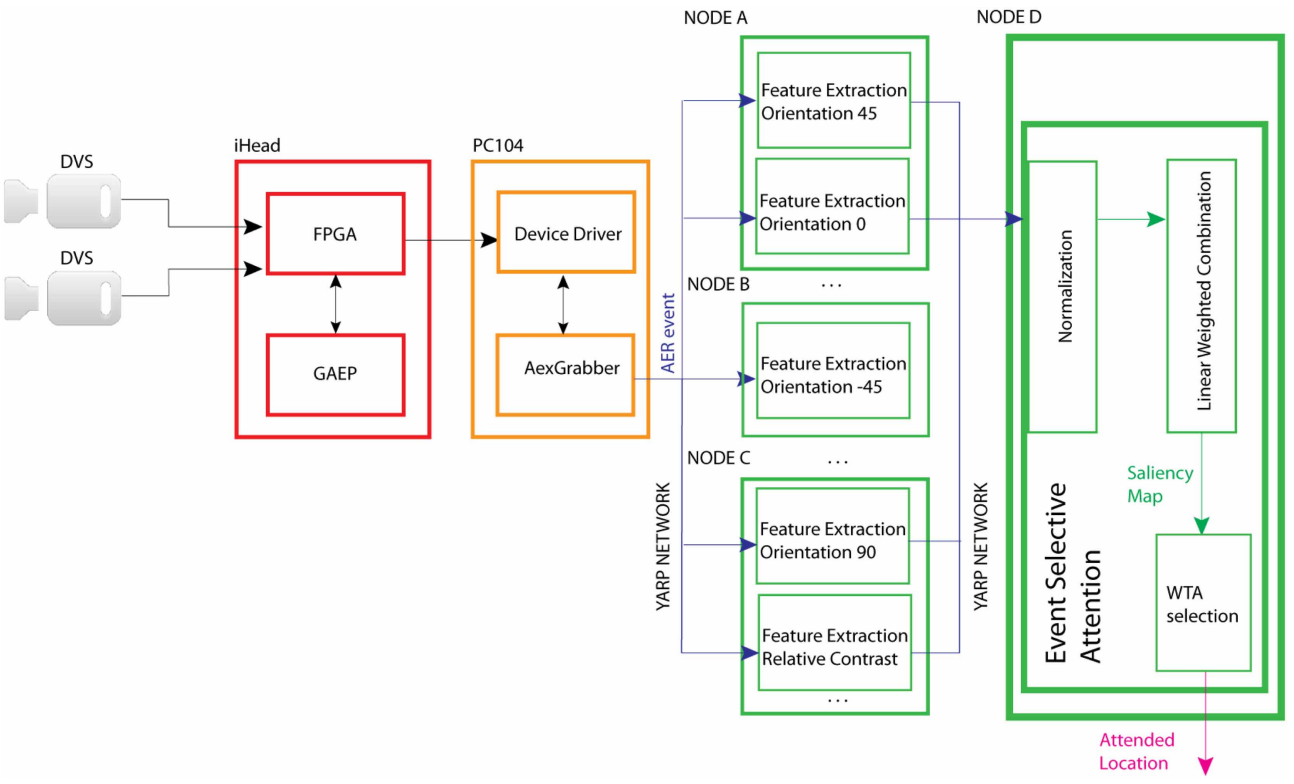



**FIGURE 1 | Structure of EVA and its implementation on the diverse HW modules: the DVS cameras send asynchronous events, the FPGA merges left and right DVS events, the GAEP assigns a timestamp to each event.** The resulting list of addresses and timestamps is sent to the PC104 that makes them available to the YARP network through the AexGrabber module. From then on, any SW module can acquire the events buffer and perform a specific computation (feature extraction, normalization, linear weighted sum, and WTA selection). A dedicated connection between the feature extracting modules and the Event-Selective-Attention module avoids interferences between feature maps and other trains of events.

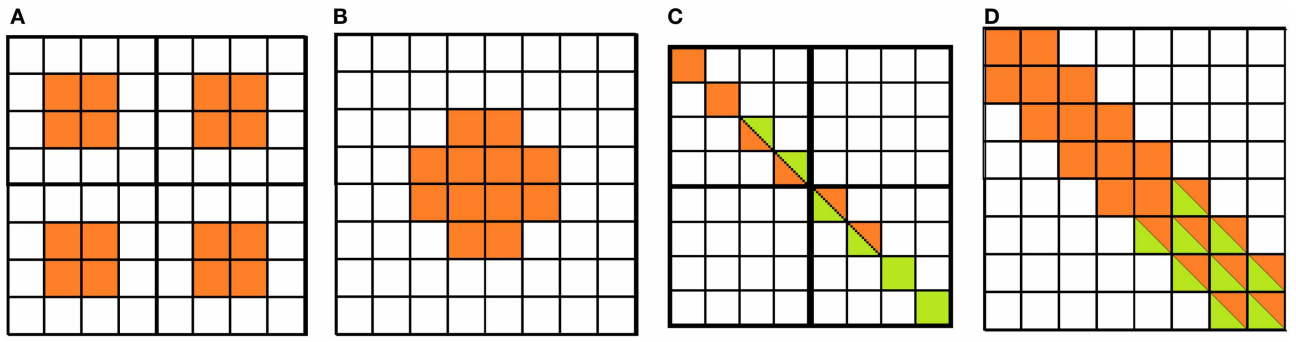



**FIGURE 2 | Receptive fields of cells used for the *mapping* procedure: center-surround cells. (A)** Four $4 \times 4$ cells, **(B)** one $8 \times 8$ cell. Simple oriented cells: **(C)** two $4 \times 4$ cells, **(D)** one $8 \times 8$ cell. ON- and OFF-regions in orange and white, respectively. In green, the pixels that contribute to the activity of neighboring cells.

```
Data: bRE: buffer of retina events, featMap: mapping related
      to the feature map
Result: bFE: buffer of feature-maps events
c = constant;
foreach event ∈ bRE do
    mapEvent = map(event, featMap);
    RF = belong(mapEvent);
    updateActivation(RF);
    if affectNeighbor(mapEvent) then
        RFNeighbor = lateralConnection(mapEvent);
        updateActivation(RFNeighbor);
        if RFNeighbor.activation > positiveThreshold then
            featureEvent =
            generateFeatureEvent(RFNeighbor);
            featureEvent.polarity = positive;
        end
        if RF.activation < negativeThreshold then
            featureEvent = generateFeatureEvent(RF);
            featureEvent.polarity = negative;
        end
    end
    if RF.activation > positiveThreshold then
        featureEvent = generateFeatureEvent(RF);
        featureEvent.polarity = positive;
    end
    if RF.activation < negativeThreshold then
        featureEvent = generateFeatureEvent(RF);
        featureEvent.polarity = negative;
    end
end
```

```
Data: RF: receptive field; r : event
Result: update of the activation of RF
if RF.type == ON then
    if r.polarity == ON then
        if r ∈ RF.center then
            RF.response := RF.response + c;
        end
        else
            RF.response := RF.response − c;
        end
    end
    else
        if r ∈ RF.center then
            RF.response := RF.response − c;
        end
        else
            RF.response := RF.response + c;
        end
    end
end
else
    if r.polarity == ON then
        if r ∈ RF.center then
            RF.response := RF.response + c;
        end
        else
            RF.response := RF.response − c;
        end
    end
    else
        if r ∈ RF.center then
            RF.response := RF.response − c;
        end
        else
            RF.response := RF.response + c;
        end
    end
end
```

The *mapping* is less computationally demanding than a traditional convolution operation. Additionally, with this approach, the feature maps are only updated at the arrival of a new spike, without calculation of the RF activation for the complete image at each time step. To further reduce the computational load, we implemented non-overlapping receptive fields, at the cost of reducing the output resolution. However, in EVA the final goal is to obtain short latency in relation to saliency map resolution that guarantees reliable gaze shift. As a result the selected region is focused in the sensor's fovea for detailed inspection.

#### 2.3.2. Saliency map and attention selection

The final saliency map is obtained through weighted linear combination of the computed contrast (I), orientation (O) feature maps and flicker feature map ($F$):

$$S = \text{Norm}\,(k_I \cdot I + k_O \cdot O + k_F \cdot F) \tag{1}$$

The weights $k_I$, $k_O$, and $k_F$ can be changed in real-time to bias saliency computation toward behaviorally relevant features, implementing a task-dependent bias (Itti and Koch, 2001). Finally, a WTA module selects the most conspicuous location of the saliency map, defining the current focus of attention. Feature extraction can be performed in parallel by multiple modules, however, the normalization and sum of feature maps into the saliency map is sequential and requires time. The data-driven system further improves the speed of computation, as the saliency map is updated only with the last train of events, avoiding a complete generation of the entire map.

In iNVT, as well as in most of saliency map based selective attention models, the currently selected location is deselected thanks to a self-inhibition mechanism, known as Inhibition of Return (IOR). This mechanism prevents the system from immediately re-select the current winner, and allows for a scan of many points of the saliency map in order of decreasing conspicuity. However, in our setup neither EVA nor iNVT implement IOR, rather, the shifts of the focus of attention are determined by intrinsic noise in the system.

#### 2.3.3. Ocular movements

A dedicated module implements saccades or gaze shifts toward salient regions selected by EVA. Tremor and microsaccades are used to generate motion of static visual stimuli on the DVS sensor focal plane, to elicit activity of the pixels that only respond to stimulus changes. This approach is similar to the mammals visual system, where small eye movements counteract photoreceptors bleaching adaptation (Kowler, 2011). Tremor is implemented as an omnidirectional movement of 0.45° amplitude with frequency of 500 Hz and random direction, superimposed on microsaccades of amplitude 0.75° and frequency 2.5 Hz in exclusively horizontal direction.

## 3. PERFORMANCE AND BENCHMARK

The absolute novelty of EVA is in the short latency of the attentional shifts that guarantees fast reaction times. The proposed processing of the attention system generates short latency that hardly compares with the performance of frame-based attention systems. The selected attended location can be communicated to the oculomotor controllers to direct the robot's gaze toward salient regions with a saccade command. It continuously updates the saliency map and the resulting focus of attention location, allowing for fast reaction to unexpected, dynamic events and for a more natural interaction of robots with the environment.

The improvement in computation latency is obtained thanks to many factors, among which the asynchronous low-latency and low-redundancy input, efficient sensory encoding, and efficient computing strategy (the mapping).

To assess its performances and validate our results, we tested EVA in three different experimental setups. Unfortunately, a direct quantitative comparison of the performance of EVA with literature state-of-the art artificial bottom–up attention systems cannot be performed as each has its own characteristics in terms of feature maps, methods for feature map calculation, hardware, software implementation, and stimuli (Borji and Itti, 2012). For this reason, we rather preferred to benchmark our implementation against the state-of-the art main-stream system based on the Itti and Koch (2001) model: the *iLab Neuromorphic Vision Toolkit (iNVT)* (Itti et al., 1998, 2003; Navalpakkam and Itti, 2005) sharing the same number and type of feature maps, hardware platform and stimuli. The iNVT algorithm is based on traditional frame-based cameras and convolution operation for the calculation of the feature maps.

The two systems are at the two opposite extremes, one is fully event-driven, the other fully frame-based. Other intermediate solutions might be implemented, where the output of the DVS is first translated into frames by integrating spikes over time, then iNVT is used on the resulting sensory output. However, the necessary transition from event-driven to frame-based information coding spoils some of the advantages of event-driven acquisition, such as temporal resolution and low latency and brings additional costs and relevant overhead in the computation. It is worth to further detail at which extent the performance improvement inherits from the use of DVS sensor as compared to the use of event-based algorithm implementation. As shown in the summary table 2, the latency of EVA amounts to 23 us, of which 15us can be attributed to the characteristic latency of the DVS sensor (Lichtsteiner et al., 2008) and the remaining 8 μs as result of the event-based algorithm. On the contrary, in frame-based scenario, the latency is affected by both the acquisition time (for 30 fps acquisition the acquisition time interval is 33 ms) and the frame-based algorithm for the image processing which we measured in 23 ms. The performance of such systems would be in terms of qualitative performance and computational cost in between the two extremes that are analyzed in the following.

The two systems are implemented on the iCub robot using respectively the DVS and the standard robot's Dragonfly cameras. They simultaneously run on two identical machines[3]; both of them distribute the processing over the four available CPU cores. To correctly compare the two systems, we implemented the same type and number of feature maps in both, restricting the numerous feature maps of iNVT to intensity, orientation and flicker[4]. In

[3]Intel Core 2 Quad Cpu Q9950 @2.83GHz

[4]*ezvision –in=raster:*.ppm –display=display -T -j 4 –input-frames=@30Hz –textlog=iNVTLog.log –vc-chans=IOF –ior-type=None –use-random.*

order to remove any overhead to the computation time, the iNVT program processes a batch of camera images.

The stimulus is placed at a distance $d$ in front of the robot and centered in the fovea of both the Dragonfly and DVS cameras, such that it is completely visible and the quantity of received light is comparable for both sensors. The sensors have been configured with typical parameters (see **Table 1**) and have not been specifically tuned for the experiments, in order to assess the system's performance in typical use cases.

For each experiment we report the diffuse scene light measure[5] since the performance of both sensors and, consequently, of the two attention systems depend on the illumination level.

For all of the validation setups we report the focus of attention's scan path generated by the two systems, giving an immediate qualitative evaluation of the computation time. For a quantitative assessment the benchmark comprises a set of predefined measurements:

- Number of shifts of the focus of attention over time $F_{\text{EVA}}$ and $F_{\text{iNVT}}$ and the correspondent time interval between consecutive shifts $\Delta t_{\text{EVA}}$ and $\Delta t_{\text{iNVT}}$
- CPU utilization $U_{\text{EVA}}$ and $U_{\text{iNVT}}$ [6]
- Data rate $D_{\text{EVA}}$ and $D_{\text{iNVT}}$
- Latency time interval $L_{\text{EVA}}$ and $L_{\text{iNVT}}$

The time interval between two consecutive shifts in the selective attention is a good measure of the frequency of attentional redeployments. The latency measure gives an estimate of the minimum reaction time to the new stimuli.

We measure the latency in both systems as the time interval from the instant a novel stimulus is presented to a complete processing of the visual input. In EVA, the latency interval comprises the time interval for feature extraction and WTA selection. The former represents the time necessary to generate a new flow of events associated to feature maps from the moment a new stimulus arrives. The latter represents the time interval to process the generated trains of events and determine attentional shift. In both measures, a sequence of events is needed to alter the output of the module. The frequency of redeployment of the focus of attention depends on the time needed to acquire enough visual data and the time required to extract features, compute saliency and perform the winner-take-all selection. On the contrary, for iNVT we present a single frame and we measure the time interval necessary for the system to process the camera image.

**Table 1 | Setup parameters of DVS and Dragonfly sensors.**

| Parameter dragonfly | Value | Bias DVS | Value ($\mu$A) |
|---|---|---|---|
| Width | 320 (pixel) | cas | 0.094 |
| Height | 640 (pixel) | injg | 0.0182 |
| Shutter | 0.913 | reqPd | 3.0 |
| Gain | 0.312 | pux | 1.4401 |
| White balance A | 0.506 | diffoff | $2.378e^{-5}$ |
| White balance B | 0.494 | req | 0.0287 |
| Sharpness | 0.5 | refr | $1.688e^{-4}$ |
| Hue | 0.48 | puy | 3.0 |
| Gamma | 0.4 | diffon | 0.1143 |
| Saturation | 0.271 | diff | 0.0054 |
| Framerate | 30 (fps) | foll | $3.576e^{-6}$ |
| | | pr | $1.431e^{-6}$ |

[5]Measured by portable hand-held exposure meter Gossen Lunasix F.

[6]Measurements performed with *SAR*, a program that directly measures the computational load on the processor over a user-defined time interval.

CPU utilization and data rate give an accurate measure of the computation demand of both implementations. To obtain an unbiased measure, we normalized by the number of attentional shifts and report the computational load per shift. The benchmark comprises three test experiments. The first uses typical stimuli for visual experiments, such as oriented gratings, and is run under two different illumination conditions. The second shows the performance of the EVA system with a fast unpredictable stimulus such as a chaotic pendulum. The third indicates how performance changes with the increase of the information to process.

### 3.1. FIRST EXPERIMENT, GRATINGS WITH DIFFERENT ORIENTATIONS

**Figure 3A** shows the stimulus used in the first characterization setup: two horizontal and two vertical gratings of $4 \times 4$ cm with a gaussian profile, each positioned at the distance $d = 20$ cm from the camera. In this scenario the stimuli are static and the DVS output is generated with the use of microsaccades (see section 2.3.3).

#### *3.1.1. Case A, bright illumination*

The focus of attention locations selected by EVA and iNVT and their hit frequency are shown in **Figures 3C,B**, respectively. Both systems select conspicuous locations corresponding to the oriented gratings, with slightly different patterns. As we disabled inhibition of return, the specific focus of attention scan-path depends on the computed saliency and on the noise present in the system. Small differences in stimulus illumination and noise pattern can contribute to slightly different computed saliency for the same grating placed in different regions; the missing inhibition of selected areas over a long period of time leads to the selection of fewer stimuli with very similar salience, as shown in **Figure 3B**. Two of the oriented gratings are not selected by iNVT, despite they should have had exactly the same salience. In this scenario, EVA is capable of selecting more stimuli, reducing the latency, probably thanks to the different pattern of noise, that is intrinsically generated by the hardware.

In EVA, the data rate depends on the lighting condition and on the stimulus, under these conditions it is about 7 kAE/s. Conversely, the data rate produced by a traditional color camera only depends on intrinsic parameters of the systems such as number of pixels, color resolution and frame rate, being independent from the stimulus; for the Dragonfly used on the iCub this amounts to 530 Mbits/s. The lower amount of data corresponds to lower processing demand and, hence, in a faster computation of the focus of attention location. Consequently this results in higher shifts frequency generated by EVA with respect to iNVT,

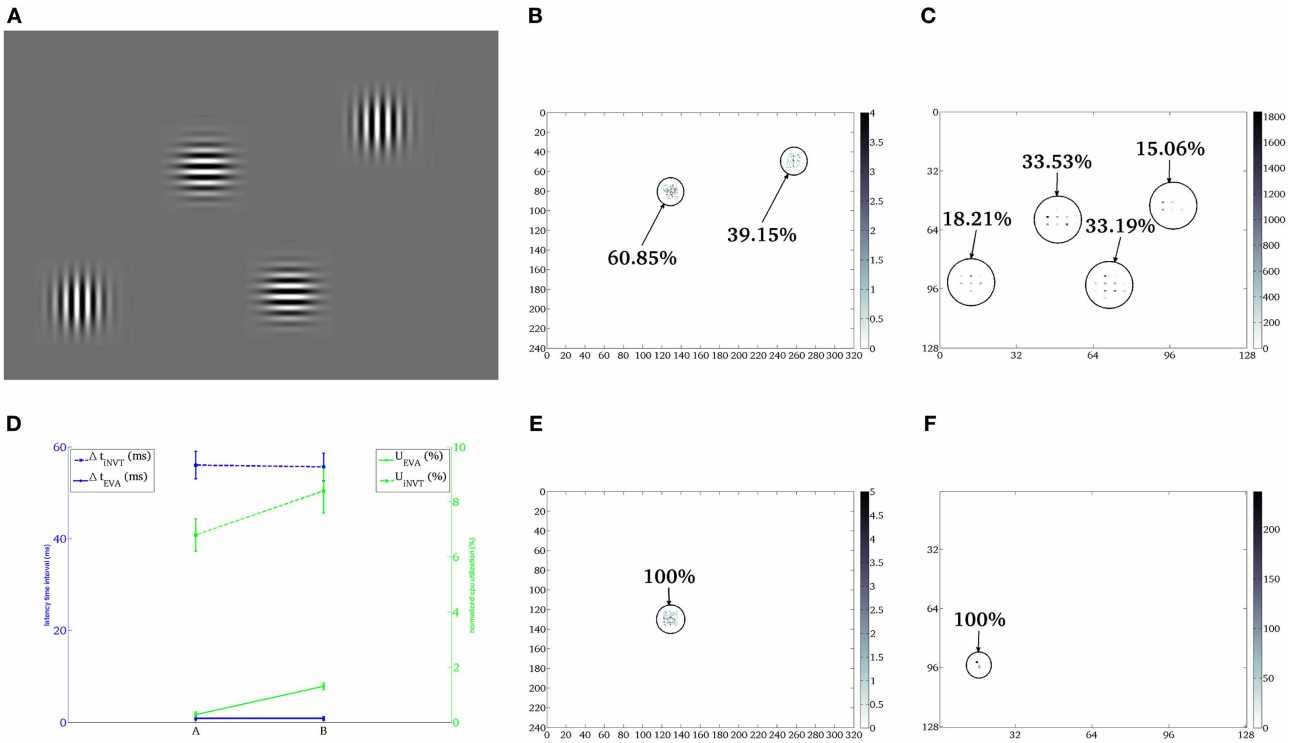



**FIGURE 3 | First scenario.** Case A: comparison of the shifts generated by iNVT **(B)** and EVA **(C)** in response to four oriented gratings **(A)** under bright background illumination (∼55.6 LUX—indoor illumination of a diffuse bright natural light). Case B: comparison of the shifts generated by iNVT **(E)** and EVA **(F)** under dim background illumination (∼2.7 LUX—dim illumination that typically would require the use of artificial light). The (x,y) coordinates of the two attention systems correspond to the image coordinates of the sensors ($240 \times 320$ for the Dragonfly and $128 \times 128$ for the DVS). **(D)** Mean and standard deviation of CPU system percentage of utilization (green), temporal distance between consecutive shifts (blue) over 10 repetitions of 10 trials in both illumination conditions.

**Table 2 | Quantitative benchmark: f : frequency of attentional shifts [shifts/s], L : Latency time interval [s], U : normalized cpu utilization [%] , D : data rate of input [Kbit/s], $\delta$ : duration of the acquisition[s].**

| | Experiment 1 | | | | Experiment 2 |
|---|---|---|---|---|---|
| | Bright (∼55.6 LUX) | | Dim (∼2.7 LUX) | | ∼27.2 LUX |
| | iNVT | EVA | iNVT | EVA | EVA |
| hor. top | 60.85% | 33.53% | 100% | 0% | . |
| hor. bot | 0% | 33.19% | 0% | 0% | . |
| ver. top | 39.15% | 15.06% | 0% | 0% | . |
| ver. bot | 0% | 18.21% | 0% | 100% | . |
| f (shifts/s) | 1.89 | 158.2 | 18.08 | 3.72 | 1708.80 |
| L (s) | $(5.60 \pm 0.3)e^{-2}$ | $(23.2 \pm 3)e^{-4}$ | $(5.56 \pm 0.3)e^{-2}$ | $(23.1 \pm 3)e^{-4}$ | $(3.72 \pm 1)e^{-3}$ |
| U (%) | 6.79 | 0.2 | 8.4 | 1.3 | 0.2 |
| D (Kbit/s) | $530e^3$ | $226.72 \pm 0.078$ | $530e^3$ | $20.32 \pm 1.2$ | $2.1e^3$ |
| $\delta t$ (s) | 100 | 100 | 100 | 100 | 2.68 |

*First experiment: number of hits clustered on the horizontally (top and bottom) and vertically (top and bottom) oriented grating stimuli under bright and dim illumination. Second experiment: performance of the EVA in details.*

and higher number of attention relocation, as shown in **Table 2**. EVA, because of the temporal resolution of the visual signal and the low computational demand, can generate a shift of attention approximately every 1.5 ms, on the contrary, iNVT is limited by the frame acquisition frequency (30 ms) and the time between two attentional shifts amounts to 50 ms. The latencies of the two systems differ of two orders of magnitude, showing the high responsiveness of EVA to sudden and potentially relevant stimuli.

**Figure 3** shows that the computation demand of EVA is lower than iNVT of at least one order of magnitude, as expected from the lower Data Rate and the different computational load of the mapping procedure. This different performance is also reflected in the shift latency that amounts to about 300 ns for EVA and 0.4 ms for iNVT.

#### *3.1.2. Case B, dim illumination*

One of the advantages of using the logarithmic encoding in the DVS is the wider dynamic range with respect to traditional sensors. Thus, we tested the attention systems in the scenario described above, but with reduced ambient light. The resulting focus of attention scan path is shown in **Figures 3E,F**. The selection of the top horizontally oriented grating in the iNVT system and the selection of the bottom vertical oriented grating in EVA are the results of a strong decrease of the response strength. The lower illumination affects both systems by drastically reducing the number of shifts. A way to improve this behavior in EVA would be the implementation of adaptive firing threshold, that can be dynamically set according to the level of background illumination (Delbruck and Oberhof, 2004).

**Figure 3** shows the aggregated performance measures for case A and B for both EVA and iNVT; Despite the latency of both systems remains largely unchanged, EVA outperforms iNVT, while the normalized computational load increases, with different slopes. In case B (low illumination), iNVT absolute CPU usage remains unchanged but it is normalized for a lower number of shifts; in EVA both the number of shifts and the CPU utilization decrease, as a result of a lower input data rate, as shown in **Table 2**. The resulting normalized computation load increases less than what observed for iNVT.

### 3.2. SECOND EXPERIMENT: CHAOTIC PENDULUM

We used a chaotic pendulum to test EVA with fast unpredictable stimuli. The chaotic pendulum shown in **Figure 4A** is composed of two black bars (22 and 18 cm) connected by a low friction joint and attached to a fixed support via a second low friction joint. In this configuration, the first bar can freely rotate with respect to the support and its movement is influenced by the second bar that revolves independently around the first joint. The pendulum is mounted over a white background and we used an

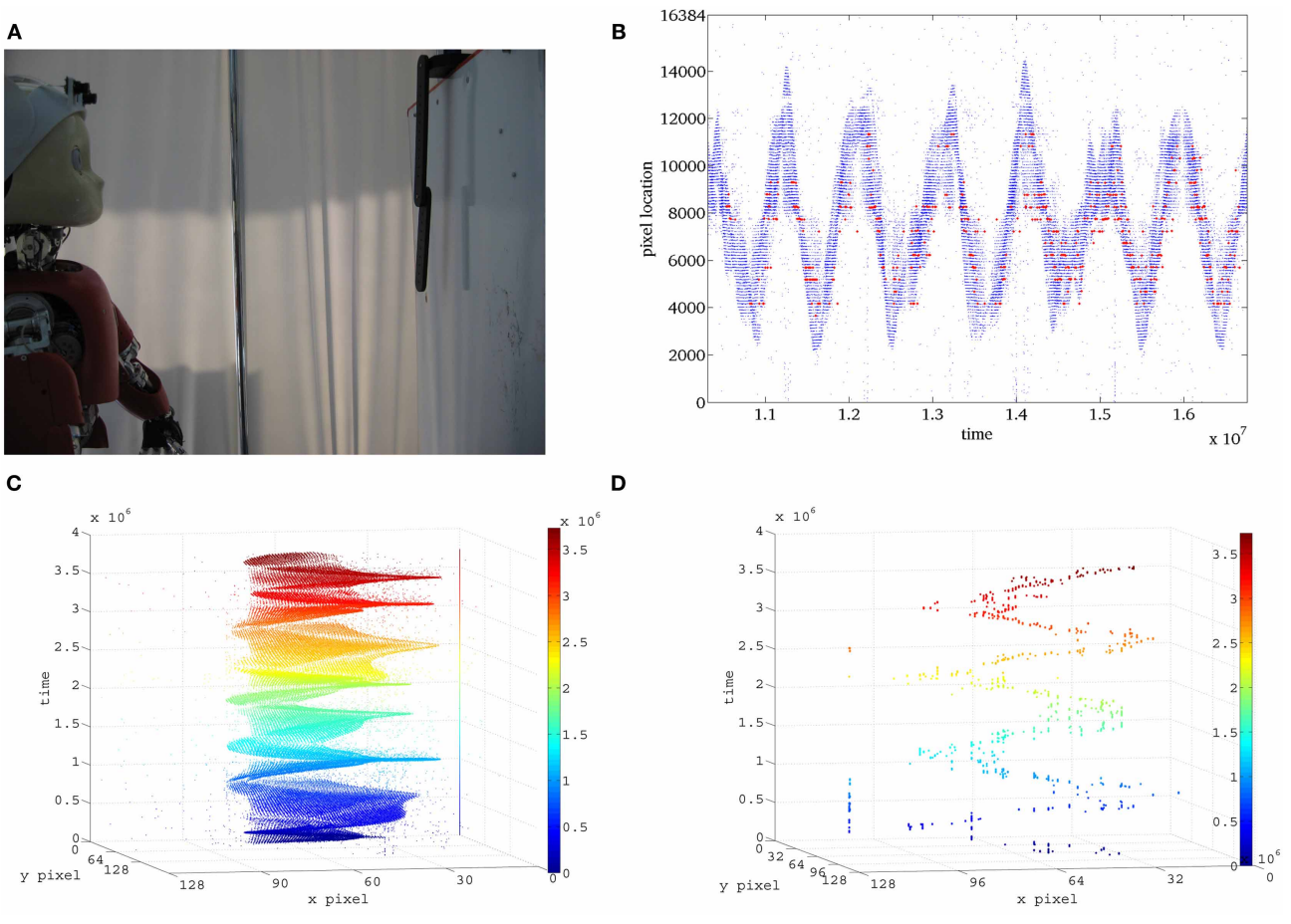



**FIGURE 4 | Second scenario. (A)** The chaotic pendulum is located 50 cm far from the robot to keep the whole stimulus in the camera's field of view. In the setup, DVS sensors are embedded in the iCub's eyes to exploit ocular movements, while the Dragonfly cameras are mounted on the head with fixed supports. **(B)** Raster representation of the activation of pixels in the DVS and relative location of the WTA selected by EVA. **(C)** Events generated by the chaotic pendulum, **(D)** focus of attention scan-path generated by EVA.

average lighting condition of 27.6LUX—corresponding to diffuse illumination where artificial light is not required.

The stimulus is so fast that neither the Dragonfly, nor the human eye, can successfully perceive its full motion. In this scenario, iNVT hardly relocates the focus of interest on the pendulum without introducing an evident delay. In iNVT, such shift is clearly shown in the video provided in Technical Materials.[7] Conversely, we accurately assess the performance of EVA as a viable to technological solution for fast dynamic stimuli in a wide range of operating conditions.

The fast movement of the pendulum generates a higher data rate with respect to the previous scenario. The resulting performance parameters are listed in **Table 2**.

To estimate the quality of the attention system, in **Figure 4** we compare the trajectory generated by the pendulum with the location of the attention shifts over time. To achieve this, we synchronized the generation of attention shifts with the batch data of the generated events, using the temporal information stored in the timestamp.

### 3.3. THIRD EXPERIMENT: PERFORMANCE SCALING WITH QUANTITY OF INFORMATION

In this scenario, we assess how the performance of EVA scales with increasing number of events. In EVA, the number of events can increase for cluttered scenes and for higher resolution sensors, increasing the computational demand of the system. This happens also for iNVT, when higher resolution sensors are used. We estimate how the computation demand expressed in CPU utilization scales with the processed information (number of bits). In order to perform such analysis we determined how the computation demand of the two systems change when the quantity of information scales. For EVA, we control the number of generated events (and then quantity of information) by increasing the number of black edges printed on a white disk rotating at constant speed. We use five different configurations, where the edge is repeated every 360°, 180°, 90°, 45°, and 22.5°. **Figure 5** shows the normalized CPU utilization measured in this experiment (red dots in the inset); we do a linear fit of the normalized CPU utilization in relation with the increasing quantity of information processed. We use this function as reference to estimate the computation demand of EVA at arbitrary number of generated events. Similarly, for iNVT, we provide different sets of images. The sets differ only for the dimension of the images, the stimulus is the same of **Figure 4A**. The computation demand increases with the quantity of processed information (green dots in **Figure 5**) and we fit a first order curve that best describes the distribution.

**Figure 5** shows that the required level of processing for EVA (in red) never exceeds the required level of processing of iNVT (in green) as it increases with a more gradual slope. **Figure 5** shows that the required level of processing for EVA (in red) never exceeds the required level of processing of iNVT (in green) as it increases with a more gradual slope. This observed divergence indicates the increasingly better performance of EVA, as more processing is required. Key points (arrows in figure) help identifying the estimated computation demand for both the systems in correspondence of quantity of information generated by different fixed size sensors (128 × 128, 320 × 240, 640 × 480).

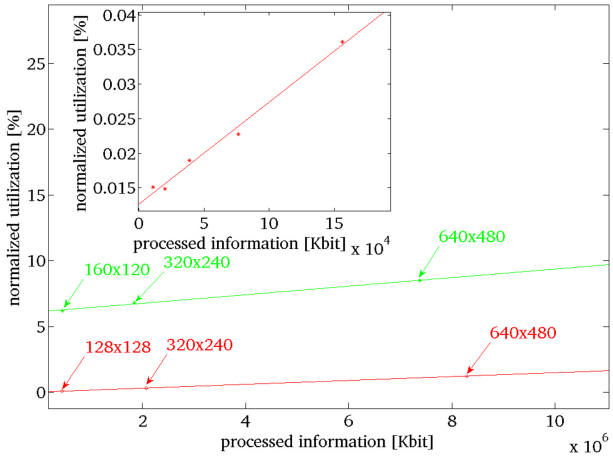



**FIGURE 5 | Expected evolution of the normalized computation demand for increasing the sensor's output by increasing the sensor's size, in green iNVT and in red EVA.** In the inset the measured data for EVA, for increasing number of events, generated by increasing number of black rotating edges repeated, respectively, at 360°, 180°, 90°, 45°, and 22.5°, in red the fit from which we extrapolate the computational demand for bigger sensors.

To estimate these numbers, we select speed of rotating bar that covers typical use (4.2284 rad/s). Even though in normal scene operation the DVS activation is about 30%, in this test, we consider the worst case scenario where all the pixels in the sensor show the maximum level of activation (27 events per pixel).

Thus, we estimate the maximum computation load associated to sensors that have dimension 128 × 128, like the DVS, 320 × 240, like the Dragonfly used for iNVT and 640 × 480. As the processing required by EVA sets well below iNVT, we conclude that for any possible situation, the required processing of EVA results less impacting on the performance than iNVT.

This confirms the assumption that relevant saving in computation demand is associated to the design of the processing in EVA and it is not limited to the hardware of the system.

## 4. DISCUSSION

In this manuscript we described EVA, a real-time implementation of selective attention based on the bio-inspired model proposed in the foundational work of Itti and Koch (2001), that uses a frame-less asynchronous vision sensor as input. The goal of the implementation was to offer a flexible and light computational paradigm for the computation of the feature maps, exploiting the *mapping* mechanism. The overall performance of the developed system takes advantage of the efficient information encoding operated of the sensor, its high dynamic range, low response latency and high temporal resolution. The use of non-conventional sensors coupled with an efficient processing results in a unprecedented fast attentional selection. We report the behavior of the system in three different experiments that highlight performance in detail. The three experiments give insights

[7]http://youtu.be/Nqd3uRbjXHE

on the three major benefits of EVA: low computation demand, high responsiveness to high frequency stimuli and favorable scalability. The attention system EVA requires lower computation utilization up to one order of magnitude with respect to iNVT when stimulated by identical stimulus. This positive characteristic does not degrade the quality of the generated attention shifts. The characteristic of low response latency and high temporal resolution resulting from the efficient design of the attention system EVA allow remarkable performance in highly dynamic scenarios. The attention system accurately and swiftly redeploys the attentional foci on the most salient regions in the visual field even in situations where frame-based algorithms of visual attention fail in obtaining clear interpretation of the stimulus. The second scenario shows that the high temporal resolution allows the attention system to track very fast stimuli, expanding the application range of the system from humanoid robotics to even more demanding use cases. We presented a solution that, by sensing via efficient event-driven hardware sensor, provides outperforming selective attention mechanism with low latency and high dynamic range. In addition, for increasing the information load, e.g., for higher resolution sensors, EVA's CPU utilization increases with lower rate than iNVT's. The design of efficient processing in EVA guarantees, when compared with iNVT, relative superior performance of growing effectiveness with the amount of processed information. Finally, the last benchmark shows that the computational advantage of EVA is not restricted to the specific stimuli and sensor dimension used in this experimental setup, rather is more general.

Most attention systems designed for real-time applications report the computational cost in terms of time needed to process a frame. The relative saliency map is often obtained in about 50–60 ms, slower than typical image frame-rate (30 frames per second) (Frintrop et al., 2007). This time scale is appropriate to reproduce typical attentional scan-paths, nevertheless, 50 ms (plus 30 ms for frame acquisition) is the lower bound for reacting to the onset of a new potentially interesting or threatening stimulus. With EVA, this limit is estimated to be as small as about 1 ms [plus 15 μs of sensor latency (Lichtsteiner et al., 2008)], thanks to the low-latency event-driven front-end data acquisition and the low computational cost of the attention system. This property is crucial in robotics systems, as it allows the robot to plan actions for reacting to unforeseen situations and sudden stimuli.

EVA has been developed to equip the iCub with a fast and low weight attention system, exploiting the event-driven vision system of the iCub (Bartolozzi et al., 2011). The mapping procedure for events filtering and feature maps generation derives from AER implementations (Bartolozzi and Indiveri, 2009; Sonnleithner and Indiveri, 2012), where a simple mapping is realized to use the sensor output as feature map. The resulting saliency map is sent to a dedicated hardware platform that implements the winner-takes-all selection enriched with dedicated inhibition of return mechanism (the Selective Attention Chip, SAC). The modules developed in this work and EVA can easily be integrated with such a system and further optimized. For example, maximization of the performance can be achieved by implementing the mapping procedure and the relative feature maps on an embedded FPGA (Bartolozzi et al., 2011; Fasnacht and Indiveri, 2011) or implementing fast convolution on ConvNet chips (Serrano-Gotarredona et al., 2008) and using the SAC (or higher resolution implementations) for WTA and IOR. Both implementations would probably be faster than the software mapping procedure described in this manuscript, for example, the ConvNet chip can start providing the output of oriented filters with a 1 μs latency and is shown to perform pseudo-simultaneous object recognition. This system, with the appropriate miniaturization and integration with top–down modules implemented on the robot, will be able to give a fast estimate of the focus of attention, leaving the computational units of the iCub free for other more complex tasks.

## ACKNOWLEDGMENTS

This work has been inspired by fruitful discussions at the Capocaccia Cognitive Neuromorphic Engineering Workshop. The present work benefited form the great support of Giacomo Indiveri who provided valuable comments and ideas. The authors would like to thank iLab and Prof. L. Itti for making the iNVT toolkit freely available.

## FUNDING

This work has been supported by the EU grant eMorph (ICT-FET-231467).

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2013.00234/abstract

## REFERENCES

Bartolozzi, C., and Indiveri, G. (2009). Selective attention in multi-chip address-event systems. *Sensors* 9, 5076–5098. doi: 10.3390/s90705076

Bartolozzi, C., Rea, F., Clercq, C., Hofstätter, M., Fasnacht, D., Indiveri, G., et al. (2011). "Embedded neuromorphic vision for humanoid robots," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (Colorado Springs, CO), 129–135. doi: 10.1109/CVPRW.2011.5981834

Borji, A., and Itti, L. (2012). State-of-the-art in visual attention modeling. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 185–207. doi: 10.1109/TPAMI.2012.89

Bumhwi, K., Hirotsugu, O., Tetsuya, Y., and Minho, L. (2011). "Implementation of visual attention system using artificial retina chip and bottom-up saliency map model," in *Neural Information Processing*. Volume 7064 of *Lecture Notes in Computer Science*, eds B.-L. Lu, L. Zhang, and J. Kwok (Berlin; Heidelberg: Springer), 416–423. doi: 10.1007/978-3-642-24965-5_47

Camunas-Mesa, L., Zamarreno-Ramos, C., Linares-Barranco, A., Acosta-Jimenez, A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2012). An event-driven multi-kernel convolution processor module for event-driven vision sensors. *IEEE J. Solid State Circ.* 47, 504–517. doi: 10.1109/JSSC.2011.2167409

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R., and Delbruck, T. (2009). "A pencil balancing robot using a pair of AER dynamic vision sensors," in *International Symposium on Circuits and Systems, (ISCAS), 2009* (Taipei: IEEE), 781–784. doi: 10.1109/ISCAS.2009.5117867

De Valois, R. L., Albrecht, D. G., and Thorell, L. G. (1982). Spatial frequency selectivity of cells in macaque visual cortex. *Vis. Res.* 22, 545–559. doi: 10.1016/0042-6989(82)90112-2

Deiss, S., Douglas, R., and Whatley, A. (1998). "A pulse-coded communications infrastructure for neuromorphic systems," in *Pulsed Neural Networks*, chapter 6, eds W. Maass and C. Bishop (Cambridge, MA: MIT Press), 157–178.

Delbruck, T. (2008). "Frame-free dynamic digital vision," in *Proceedings of the International Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society* (Tokyo, Japan), 21–26. doi: 10.5167/uzh-17620

Delbruck, T., and Oberhof, D. (2004). Self biased low power adaptive photoreceptor. *Intl. Symp. Circ. Syst.* 4, 844–847. doi: 10.1109/ISCAS.2004.1329136

Fasnacht, D., and Indiveri, G. (2011). "A PCI based high-fanout AER mapper with 2 GiB RAM look-up table, 0.8 µs latency and 66 mhz output event-rate," in *Conference on Information Sciences and Systems, CISS 2011* (Johns Hopkins University), 1–6. doi: 10.1109/CISS.2011.5766102

Ferster, D., and Koch, C. (1987). Neuronal connections underlying orientation selectivity in cat visual cortex. *Trends Neurosci.* 10, 487–492. doi: 10.1016/0166-2236(87)90126-3

Frintrop, S., and Jensfelt, P. (2008). "Active gaze control for attentional visual slam," in *IEEE International Conference on Robotics and Automation, ICRA 2008* (Pasadena, CA), 3690–3697. doi: 10.1109/ROBOT.2008.4543777

Frintrop, S., Klodt, M., and Rome, E. (2007). "A real-time visual attention system using integral images," in *In Proceedings of the 5th International Conference on Computer Vision Systems (ICVS)* (Bielefeld). doi: 10.2390/biecoll-icvs2007-66

Hofstaetter, M., Schoen, P., and Posch, C. (2010). "A SPARC-compatible general purpose address-event processor with 20-bit 10ns-resolution asynchronous sensor data interface in 0.18 µm CMOS," in *International Symposium on Circuits and Systems, ISCAS* (Paris), 4229–4232. doi: 10.1109/ISCAS.2010.5537575

Hubel, D. H., and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* 160, 106–154.

Itti, L. (2002). "Real-time high-performance attention focusing in outdoors color video streams," in *Proceedings of the SPIE 4662, Human Vision and Electronic Imaging VII* (San Jose, CA), 235. doi: 10.1117/12.469519

Itti, L., Dhavale, N., and Pighin, F. (2003). "Realistic avatar eye and head animation using a neurobiological model of visual attention," in *Proceedings of the SPIE 48th Annual International Symposium on Optical Science and Technology*. Vol. 5200, eds B. Bosacchi, D. B. Fogel, and J. C. Bezdek (Bellingham, WA: SPIE Press), 64–78. doi: 10.1117/12.512618

Itti, L., and Koch, C. (2001). Computational modelling of visual attention. *Nat. Rev. Neurosci.* 2, 194–203. doi: 10.1038/35058500

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* 20, 1254–1259. doi: 10.1109/34.730558

Kandel, E., Schwartz, J. H., and Jessell, T. M. (2000). *Principles of Neural Science*. 4th Edn. New York, NY: McGraw-Hill Medical. doi: 10.1036/0838577016

Koch, C., and Ullman, S. (1985). Shifts in selective visual-attention – towards the underlying neural circuitry. *Hum. Neurobiol.* 4, 219–227.

Kowler, E. (2011). Eye movements: the past 25 years. *Vis. Res.* 51, 1457–1483. doi: 10.1016/j.visres.2010.12.014

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). An 128 × 128 120dB 15 µs-latency temporal contrast visi on sensor. *IEEE J. Solid State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Metta, G., Fitzpatrick, P., and Natale, L. (2006). YARP: yet another robot platform. *Intl. J. Adv. Robot. Syst.* 3, 43–48. doi: 10.5772/5761

Miau, F., Papageorgiou, C., and Itti, L. (2001). Neuromorphic algorithms for computer vision and attention. *Proc. SPIE* 46, 12–23. doi: 10.1117/12.448343

Movshon, J., Thompson, I., and Tolhurst, D. (1978). Spatial summation in the receptive fields of simple cells in the cat's striate cortex. *J. Physiol.* 283, 53–77.

Navalpakkam, V., and Itti, L. (2005). Modeling the influence of task on attention. *Vis. Res.* 45, 205–231. doi: 10.1016/j.visres.2004.07.042

Ouerhani, N., Bur, A., and Hügli, H. (2005). "Robot self-localization using visual attention," in *Proceedings of the CIRA 2005*, (Ancona, Italy), 309–314. doi: 10.1109/CIRA.2005.1554295

Ouerhani, N., and Hügli, H. (2003). Real-time visual attention on a massively parallel simd architecture. *Real Time Imag.* 9, 189–196. doi: 10.1016/S1077-2014(03)00036-6

Qiaorong, Z., Guochang, G., and Huimin, X. (2009). Image segmentation based on visual attention mechanism. *J. Multimedia* 4, 363–370. doi: 10.4304/jmm.4.6.363-370

Robocup, T. (2011). RoboCup official site. URL: http://www.robocup.org/

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodriguez, F., et al. (2009). CAVIAR: a 45k neuron, 5M synapse, 12G connects/s aer hardware sensory–processing– learning–actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Serrano-Gotarredona, R., Serrano-Gotarredona, T., Acosta-Jimenez, A., Serrano-Gotarredona, C., Perez-Carrasco, J., Linares-Barranco, A., et al. (2008). On real-time aer 2d convolutions hardware for neuromorphic spike based cortical processing. *IEEE Trans. Neural Netw.* 19, 1196–1219. doi: 10.1109/TNN.2008.2000163

Siagian, C., Chang, C., Voorhies, R., and Itti, L. (2011). Beobot 2.0: cluster architecture for mobile robotics. *J. Field Robot.* 28, 278–302. doi: 10.1002/rob.20379

Sonnleithner, D., and Indiveri, G. (2012). "A real-time event-based selective attention system for active vision," in *Advances in Autonomous Mini Robots*, eds U. Ruckert, S. Joaquin, and W. Felix (Berlin; Heidelberg: Springer), 205–219. doi: 10.1007/978-3-642-27482-4_21

Walther, D., Rutishauser, U., Koch, C., and Perona, P. (2005). Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. *Comput. Vis. Image Underst.* 100, 41–63. doi: 10.1016/j.cviu.2004.09.004

Wiesmann, G., Schraml, S., Litzenberger, M., Belbachir, A., Hofstatter, M., and Bartolozzi, C. (2012). "Event-driven embodied system for feature extraction and object recognition in robotic applications," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (Providence, RI), 76–82. doi: 10.1109/CVPRW.2012.6238898

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 13 August 2013; paper pending published: 08 September 2013; accepted: 20 November 2013; published online: 13 December 2013.*

*Citation: Rea F, Metta G and Bartolozzi C (2013) Event-driven visual attention for the humanoid robot iCub. Front. Neurosci. 7:234. doi: 10.3389/fnins.2013.00234*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 31 March 2014
doi: 10.3389/fnins.2014.00048

# On the use of orientation filters for 3D reconstruction in event-driven stereo vision

***Luis A. Camuñas-Mesa[1]*, Teresa Serrano-Gotarredona[1], Sio H. Ieng[2], Ryad B. Benosman[2] and Bernabe Linares-Barranco[1]***

[1] *Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC y Universidad de Sevilla, Sevilla, Spain*
[2] *UMR_S968 Inserm/UPMC/CNRS 7210, Institut de la Vision, Université de Pierre et Marie Curie, Paris, France*

***Edited by:***
*Tobi Delbruck, INI Institute of Neuroinformatics, Switzerland*

***Reviewed by:***
*Theodore Yu, Texas Instruments Inc., USA*
*Jun Haeng Lee, Samsung Electronics, South Korea*

***Correspondence:**
*Luis A. Camuñas-Mesa, Instituto de Microelectrónica de Sevilla (IMSE-CNM), CSIC y Universidad de Sevilla, Av. Américo Vespucio, s/n, 41092 Sevilla, Spain*
*e-mail: camunas@imse-cnm.csic.es*

The recently developed Dynamic Vision Sensors (DVS) sense visual information asynchronously and code it into trains of events with sub-micro second temporal resolution. This high temporal precision makes the output of these sensors especially suited for dynamic 3D visual reconstruction, by matching corresponding events generated by two different sensors in a stereo setup. This paper explores the use of Gabor filters to extract information about the orientation of the object edges that produce the events, therefore increasing the number of constraints applied to the matching algorithm. This strategy provides more reliably matched pairs of events, improving the final 3D reconstruction.

**Keywords: stereovision, neuromorphic vision, Address Event Representation (AER), event-driven processing, convolutions, gabor filters**

## INTRODUCTION

Biological vision systems are known to outperform any modern artificial vision technology. Traditional frame-based systems are based on capturing and processing sequences of still frames. This yields a very high redundant data throughput, imposing high computational demands. This limitation is overcome in bio-inspired event-based vision systems, where visual information is coded and transmitted as events (spikes). This way, much less redundant information is generated and processed, allowing for faster and more energy efficient systems.

Address Event Representation (AER) is a widely used bio-inspired event-driven technology for coding and transmitting (sensory) information (Sivilotti, 1991; Mahowald, 1992; Lazzaro et al., 1993). In AER sensors, each time a pixel senses relevant information (like a change in the relative light) it asynchronously sends an event out, which can be processed by event-based processors (Venier et al., 1997; Choi et al., 2005; Silver et al., 2007; Khan et al., 2008; Camuñas-Mesa et al., 2011, 2012; Zamarreño-Ramos et al., 2013). This way, the most important features pass through all the processing levels very fast, as the only delay is caused by the propagation and computation of events along the processing network. Also, only pixels with relevant information send out events, reducing power and bandwidth consumption. These properties (high speed and low energy) are making AER sensors very popular, and different sensing chips have been reported for vision (Lichtsteiner et al., 2008; Leñero-Bardallo et al., 2010, 2011; Posch et al., 2011; Serrano-Gotarredona and Linares-Barranco, 2013) or auditory systems (Lazzaro et al., 1993; Cauwenberghs et al., 1998; Chan et al., 2007).

The development of Dynamic Vision Sensors (DVS) was very important for high speed applications. These devices can track extremely fast objects with standard lighting conditions, providing an equivalent sampling rate higher than 100 KFrames/s. Exploiting this fine time resolution provides a new mean for achieving stereo vision with fast and efficient algorithms (Rogister et al., 2012).

Stereovision processing is a very complex problem for conventional frame-based strategies, due to the lack of precise timing information as used by the brain to solve such tasks (Meister and Berry II, 1999). Frame-based methods usually process sequentially sets of images independently, searching for several features like orientation (Granlund and Knutsson, 1995), optical flow (Gong, 2006) or descriptors of local luminance (Lowe, 2004). However, event-based systems can compute stereo information much faster using the precise timing information to match pixels between different sensors. Several studies have applied events timing together with additional constraints to compute depth from stereo visual information (Marr and Poggio, 1976; Mahowald and Delbrück, 1989; Tsang and Shi, 2004; Kogler et al., 2009; Domínguez-Morales et al., 2012; Carneiro et al., 2013; Serrano-Gotarredona et al., 2013).

In this paper, we explore different ways to improve 3D object reconstruction using Gabor filters to extract orientation information from the retinas events. For that, we use two DVS sensors with high contrast sensitivity (Serrano-Gotarredona and Linares-Barranco, 2013), whose output is connected to a convolutional network hardware (Zamarreño-Ramos et al., 2013). Different Gabor filter architectures are implemented to reconstruct the 3D shape of objects. In section Neuromorphic Silicon Retina, we describe briefly the DVS sensor used. Section Stereo Calibration describes the calibration method used in this work. In section Event Matching, we detail the matching algorithm applied, while section 3D Reconstruction shows the method for reconstructing the 3D coordinates. Finally, section Results provides experimental results.

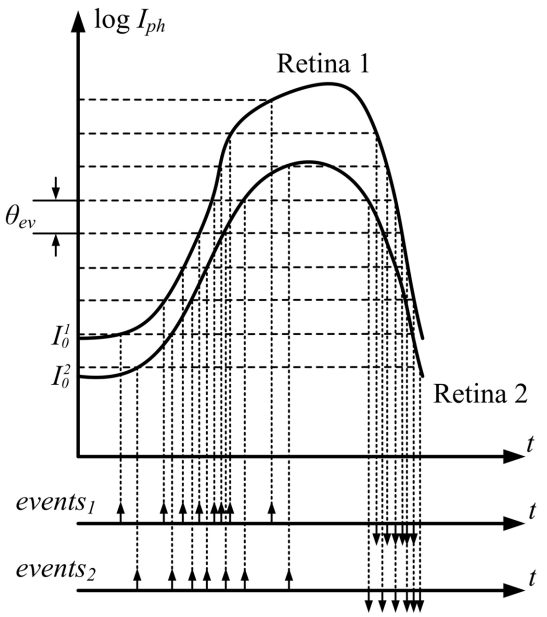



**FIGURE 1 | Data driven asynchronous event generation for two equivalent pixels in Retina 1 and Retina 2.** Because of intra-die pixel mismatch and inter-die sensor mismatch, both response curves differ.

## NEUROMORPHIC SILICON RETINA

The DVS used in this work is an AER silicon retina with 128 × 128 pixels and increased contrast sensitivity, allowing the retina to detect contrast as low as 1.5% (Serrano-Gotarredona and Linares-Barranco, 2013). The output of the retina consists of asynchronous AER events that represent a change in the sensed relative light. Each pixel independently detects changes in log intensity larger than a threshold since the last emitted event $\theta_{ev} = \left|I(t) - I(t_{\text{last-spike}})\right|/I(t)$.

The most important property of these sensors is that pixel information is obtained not synchronously at fixed frame rate $\delta t$, but asynchronously driven by data at fixed relative light increments $\theta_{ev}$, as shown in **Figure 1**. This figure represents the photocurrent transduced by two pixels in two different retinas in a stereo setup, configured so that both pixels are sensing an equivalent activity. Even though if both are sensing exactly the same light, the transduced currents are different, given the change in initial conditions ($I_0^1$ and $I_0^2$) and mismatch between retina pixels that produce a different response to the same stimulus. As a consequence, the trains of events generated by these two pixels are not identical, as represented in **Figure 1**.

The events generated by the pixels can have either positive or negative polarity, depending on whether the light intensity increased or decreased. These events are transmitted off-chip, timestamped and sent to a computer using a standard USB connection.

## STEREO CALIBRATION

Before using a pair of retinas for sensing and matching pairs of corresponding events and reconstruct each event in 3D, both retinas relative positions and orientations need to be calibrated.

Let us use lower case to denote a 2D point in the retina sensing plane as $m = [x\, y]^T$, and capital letter to denote the corresponding 3D point in real space as $M = [X\, Y\, Z]^T$. Augmented vectors are built by adding 1 as the last element: $\tilde{m} = [x\, y\, 1]^T$ and $\tilde{M} = [X\, Y\, Z\, 1]^T$. Under the assumptions of the pinhole camera model, the relationship between $\tilde{m}$ and $\tilde{M}$ is given by Hartley and Zisserman (2003):

$$\tilde{m} = P_i \cdot \tilde{M} \tag{1}$$

where $P_i$ is the projection matrix for camera $i$. In order to obtain the projection matrices of a system, many different techniques have been proposed, and they can be classified into the following two categories (Zhang, 2000):

- Photogrammetric calibration: using a calibration object with known geometry in 3D space. This calibration object usually consists of two or three planes orthogonal to each other (Faugeras, 1993).
- Self-calibration: the calibration is implemented by moving the cameras in a static scene obtaining several views, without using any calibration object (Maybank and Faugeras, 1992).

In this work, we have implemented a calibration technique based on a known 3D object, consisting of 36 points distributed in two orthogonal planes. Using this fixed pattern, we calibrate two DVS. A blinking LED was placed in each one of these 36 points. LEDs blinked sequentially one at a time, producing trains of spikes in several pixels at both sensors. From these trains of spikes, we needed to extract the 2D calibration coordinates $\tilde{m}_i^j$, where $i = 1,\ 2$ represents each silicon retina and $j = 1, \ldots 36$ represents the calibration points (see **Figure 2**). There are two different approaches to obtain these coordinates: with pixel or sub-pixel resolution. In the first one, we decided that the corresponding 2D coordinate for a single LED was represented by the pixel which responded with a higher firing rate. In the second one, we selected a small cluster of pixels which responded to that LED with a firing rate above a certain threshold, and we calculated the average coordinate, obtaining sub-pixel accuracy.

After calculating $\tilde{m}_1^j$ and $\tilde{m}_2^j$ ($j = 1, \ldots 36$) and knowing $\tilde{M}^j$, we can apply any algorithm that was developed for traditional frame-based computer vision (Longuet-Higgins, 1981) to extract $P_1$ and $P_2$ (Hartley and Zisserman, 2003). More details can be found in Calculation of Projection Matrix P in Supplementary Material.

The fundamental matrix $F$ relates the corresponding points obtained from two cameras, and is defined by the equation:

$$\tilde{m}_1^T F \tilde{m}_2 = 0 \tag{2}$$

where $\tilde{m}_1$ and $\tilde{m}_2$ are a pair of correspondent 2D points in both cameras (Luong, 1992). This system can be solved using the 36 pairs of points mentioned before (Benosman et al., 2011).

## EVENT MATCHING

In stereo vision systems, a 3D point in space $M$ is projected onto the focal planes of both cameras in pixels $m_1$ and $m_2$, therefore

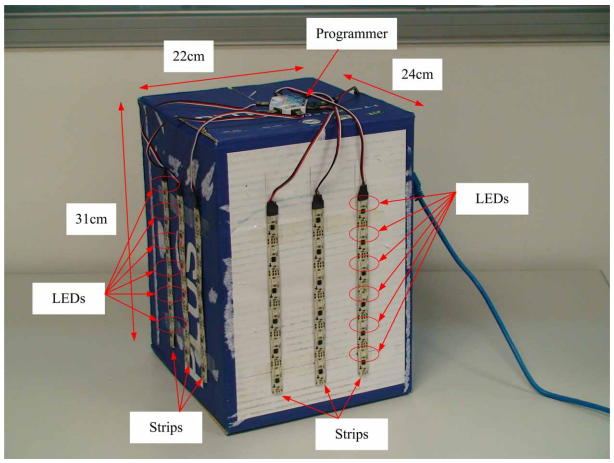



**FIGURE 2 | Photograph of the calibration structure, with 36 LEDs distributed in two orthogonal planes.** The size of the object is shown in the figure.

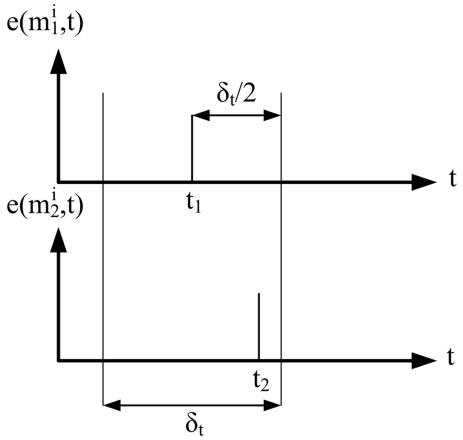



**FIGURE 3 | Temporal match.** Two events can be considered as candidates to match if they are generated within a certain time interval $\delta_t$.

generating events $e(m_1^i,\ t)$ and $e(m_2^i,\ t)$. Reconstructing the original 3D point requires matching each pair of events produced by point $M$ at time $t$ (Carneiro et al., 2013). For that, we implemented two different matching algorithms (A and B) based on a list of restrictions applied to each event in order to find its matching pair. These algorithms are described in the following subsections.

### RETINAS EVENTS MATCHING ALGORITHM (A)

This first algorithm (Carneiro et al., 2013) consists of applying the following restrictions (1–4) to the events generated by the silicon retinas. Therefore, for each event generated by retina 1 we have to find out how many events from retina 2 satisfy the 4 restrictions. If the answer is only one single event, it can be considered its matching pair. Otherwise, it is not possible to determine the corresponding event, and it will be discarded.

#### *Restriction 1: temporal match*

One of the most useful advantages of event-driven DVS based vision sensing and processing is the high temporal resolution down to fractions of micro seconds (Lichtsteiner et al., 2008; Posch et al., 2011; Serrano-Gotarredona and Linares-Barranco, 2013). Thus, in theory, two identical DVS cameras observing the same scene should produce corresponding events simultaneously (Rogister et al., 2012). However, in practice, there are many non-ideal effects that end up introducing appreciable time differences (up to many milli seconds) between corresponding events:

(a) inter-pixel and inter-sensor variability in the light-dependent latency since a luminance change is sensed by the photodiode until it is amplified, processed and communicated out of the chip;
(b) presence of noise at various stages of the circuitry;
(c) variability in inter-pixel and inter-sensor contrast sensitivity; and
(d) randomness of pixel initial conditions when a change of light happens.

Nonetheless, corresponding events occur within a milli second range time window, depending on ambient light (the lower light, the wider the time window). As a consequence, this first restriction implies that for an event $e(m_1^i,\ t_1)$, only those events $e(m_2^i,\ t_2)$ with $|t_1 - t_2| < \delta_t/2$ can be candidates to match, as shown in **Figure 3**. In our experimental setup we used a value of $\delta_t = 4$ ms, which gave the best possible result under standard interior lighting conditions.

#### *Restriction 2: epipolar restriction*

As is described in detail in (Hartley and Zisserman, 2003), when a 3D point in space $M$ is projected onto pixel $m_1$ in retina 1, the corresponding pixel $m_2$ lies on an epipolar line in retina 2 (Carneiro et al., 2013). Using this property, a second restriction is added to the matching algorithm using the fundamental matrix $F$ to calculate the epipolar line $Ep_2$ in retina 2 corresponding to event $m_1$ in retina 1 ($Ep_2\,(m_1) = F^T\tilde{m}_1$). Therefore, only those events $e(m_2^i,\ t_2)$ whose distance to $Ep_2$ is less than a given limit $\delta_{Ep_i}$ can be candidates to match. In our experiments we used a value of $\delta_{Ep_i} = 1\ pixel$.

#### *Restriction 3: ordering constraint*

For a practical stereo configuration of retinas where the angle between their orientations is small enough, a certain geometrical constraint can be applied to each pair of corresponding events. In general, the horizontal coordinate of the events generated by a retina is always larger than the horizontal coordinate of the corresponding events generated by the other retina.

#### *Restriction 4: polarity*

The silicon retinas used in our experimental setup generate output events when they detect a change in luminance in a pixel, indicating in the polarity of the event if that change means increasing or decreasing luminance (Lichtsteiner et al., 2008; Posch et al., 2011; Serrano-Gotarredona and Linares-Barranco, 2013). Using the polarity of events, we can impose the condition that two corresponding events in both retinas must have the same polarity.

### GABOR FILTER EVENTS MATCHING ALGORITHM (B)

We propose a new algorithm where we use the orientation of the object edges to improve the matching, increasing the number of correctly matched events.

If the focal planes of two retinas in a stereo vision system are roughly vertically aligned and have a small horizontal vergence, the orientation of observed edges will be approximately equal provided that the object is not too close to the retinas. A static DVS produces events when observing moving objects, or more precisely, when observing the edges of moving objects. Therefore, correspondent events in the two retinas are produced by the same moving edges, and consequently the observed orientation of the edge should be similar in both retinas. An edge would appear with a different angle in both retinas only when it is relatively close to them, and in practice this does not happen because of two reasons[1]:

(1) Since both cameras have small horizontal vergence, the object would be out of the overlapping field of view of the 2 retinas far before being so close. In that case, we do not have stereo vision anymore.
(2) The minimal focusing distance of the cameras' lenses limits the maximal vergence.

Considering that, we can assume that the orientation of an edge will be approximately the same in both retinas under our working conditions. Under different conditions, an epipolar rectification should be applied to the stereo system to ensure the orientations of the edges to be identical in the two cameras. This operation consists in estimating the homographies mapping and scaling the events of each retina into two focal planes parallel to the stereo baseline (Loop and Zhang, 1999). Lines in the rectified focal planes are precisely the epipolar lines of the stereo system. This rectification should be carried out at the same time than the retinas calibration.

The application of banks of Gabor filters to the events generated by both retinas provides information about the orientation of the object edges that produce the events as shown in **Figure 4**. This way, by using Gabor filters with different angles we can apply the previously described matching algorithm to pairs of Gabor filters with the same orientation. Thus, the new matching algorithm is as follows. The events coming out of retinas $R_1$ and $R_2$ are processed by Gabor filters $G_{1x}$ and $G_{2x}$, respectively (with $x = 1, 2, \ldots N$, being $N$ the number of orientation filters for each retina). Then, for each pair of Gabor filters $G_{1x}$ and $G_{2x}$, conditions 1–4 are applied to obtain matched events for each orientation. Therefore, the final list of matched events will be obtained as the union of all the lists of matched events obtained for each orientation.

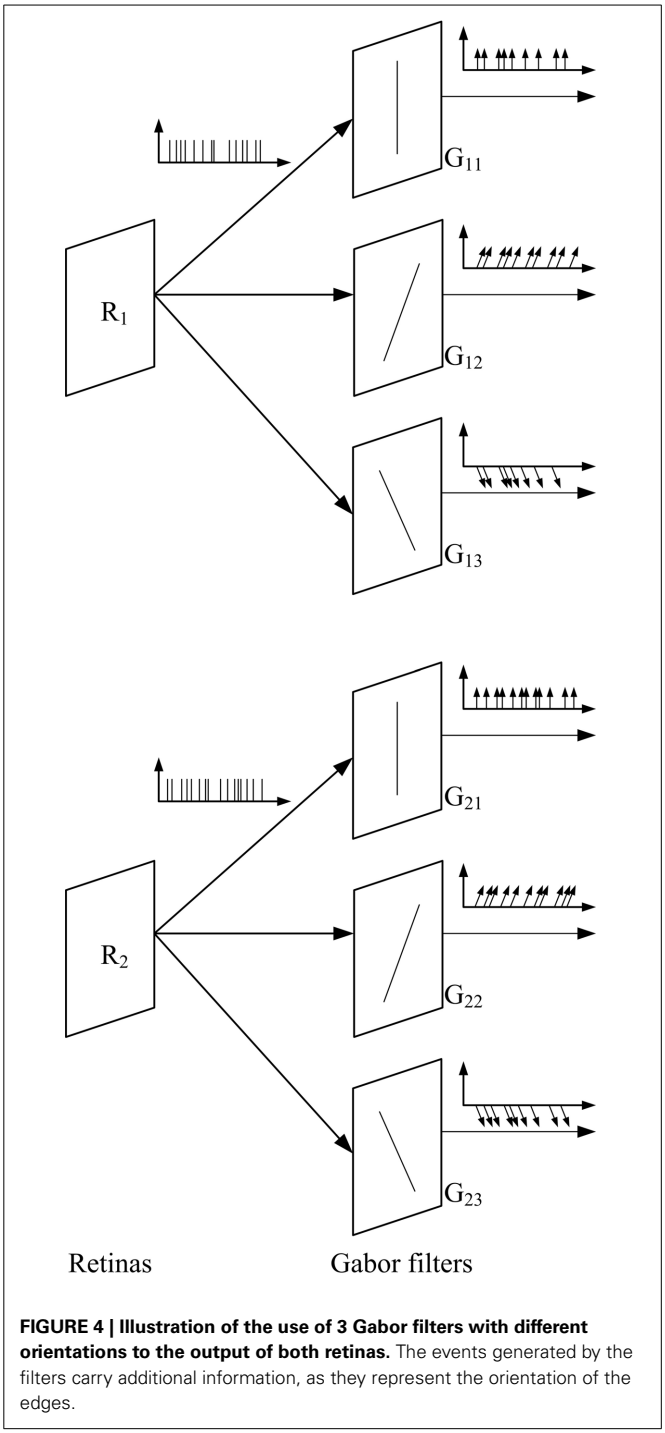



**FIGURE 4 | Illustration of the use of 3 Gabor filters with different orientations to the output of both retinas.** The events generated by the filters carry additional information, as they represent the orientation of the edges.

## 3D RECONSTRUCTION

The result provided by the previously described matching algorithm is a train of pairs of corresponding events. Each pair consists of two events with coordinates $m_1 = (x_1, y_1)^T$ and $m_2 = (x_2, y_2)^T$. The relationship between $\tilde{m}$ and $\tilde{M}$ for both retinas is given by:

$$\tilde{m}_1 \times P_1 \tilde{M} = 0 \qquad (3)$$
$$\tilde{m}_2 \times P_2 \tilde{M} = 0$$

where $P_1$ and $P_2$ represent the projection matrices calculated during calibration, and $\tilde{M}$ is the augmented vector corresponding to

[1]There is, however, a "pathological" exception: a very thin and long object, perfectly centred between the two retinas, having its long dimensión perpendicular to the retina planes, may produce different angles at both retinas.

the 3D coordinate that must be obtained. These equations can be solved as a linear least squares minimization problem (Hartley and Zisserman, 2003), giving the final 3D coordinates $M = [X\ Y\ Z]^T$ as a solution. More details can be found in Calculation of Reconstructed 3D Coordinates in Supplementary Material.

## RESULTS

In this Section, we describe briefly the hardware setup used for the experiments, then we show a comparison between the different calibration methods, after that we characterize the 3D reconstruction method, and finally we present results on the reconstruction of 3D objects.

### HARDWARE SETUP

The event-based stereo vision processing has been tested using two DVS sensor chips (Serrano-Gotarredona and Linares-Barranco, 2013) whose outputs are connected to a merger board (Serrano-Gotarredona et al., 2009) which sends the events to a 2D grid array of event-based convolution modules implemented within a Spartan6 FPGA. This scheme has been adapted from a previous one that used a Virtex6 (Zamarreño-Ramos et al., 2013). The Spartan6 was programmed to perform real-time edge extraction on the visual flow from the retinas. Finally, a USBAERmini2 board (Serrano-Gotarredona et al., 2009) was used to timestamp all the events coming out of the Spartan6 board and send them to a computer through a high-speed USB2.0 port (see **Figure 5**).

The implementation of each convolution module in the FPGA is represented in **Figure 6**. It consists of two memory blocks (one to store the pixel values, and the other to store the kernel), a control block that performs the operations, a configuration block that receives all the programmable parameters, and an output block that sends out the events. When an input event arrives, it is received by the control block, which implements the handshaking and calculates which memory positions must be affected by the operation. In particular, it must add the kernel values to the pixels belonging to the appropriate neighborhood around the address of the input event, as done in previous event-driven convolution processors (Serrano-Gotarredona et al., 1999, 2006, 2008, 2009; Camuñas-Mesa et al., 2011, 2012). At the same time, it checks if any of the updated pixels has reached its positive or negative threshold, in that case resetting the pixel and sending a signed event to the output block. A programmable forgetting process decreases linearly the value of all the pixels periodically, making the pixels behave like leaky integrate-and-fire neurons.

Several convolutional modules can be arranged in a 2D mesh, each one communicating bidirectionally with all four neighbors, as illustrated in **Figure 7** (Zamarreño-Ramos et al., 2013). Each module is characterized by its module coordinate within the array. Address events are augmented by adding either the source or destination module coordinate. Each module includes an AER router which decides how to route the events (Zamarreño-Ramos et al., 2013). This way, any network architecture can be implemented, like the one shown in **Figure 4** with any number of Gabor filters. Each convolutional module is programmed to extract a specific orientation by writing the appropriate kernel. In our experiments, the resolution of the convolutional blocks is 128 × 128 pixels.

In order to compensate the mismatch between the two DVS chips, an initial procedure must be implemented. This procedure consists of setting the values of the bias signals which control the sensitivity of the photosensors to obtain approximately the same number of events in response to a fixed stimulus in both retinas.

### CALIBRATION RESULTS

In order to calibrate the setup with both DVS retinas (with a baseline distance of 14 cm, being the retinas approximately aligned

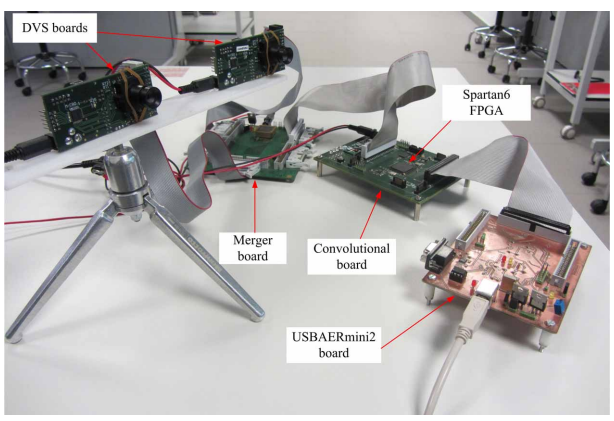



**FIGURE 5 | Experimental stereo setup.**

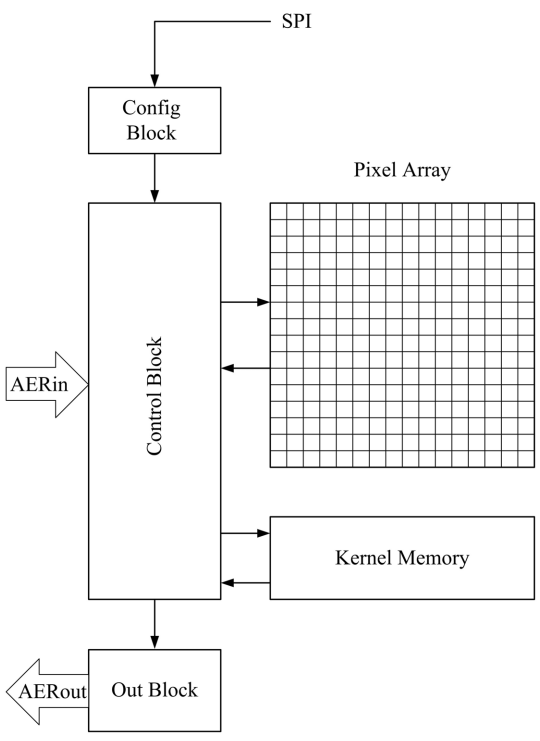



**FIGURE 6 | Block diagram for the convolutional block implemented on FPGA.**

and the focal length of the lenses 8 mm), we built a structure of 36 blinking LEDs distributed in two orthogonal planes, each with an array of 6 × 3 LEDs with known 3D coordinates in each plane (see **Figure 2**). The horizontal distance between LEDs is 5 cm, while the vertical separation is 3.5 cm. This structure was placed in front of the DVS stereo setup at approximately 1 m distance, and the events generated by the retinas were recorded by the computer. The LEDs would blink sequentially, so that when one LED produces events no other LED is blinking. This way, during a

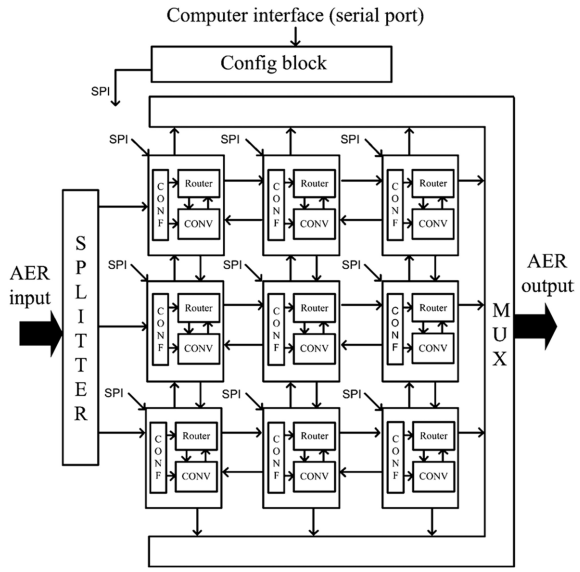



**FIGURE 7 | Block diagram for a sample network with 3 × 3 convolutional blocks implemented on FPGA.**

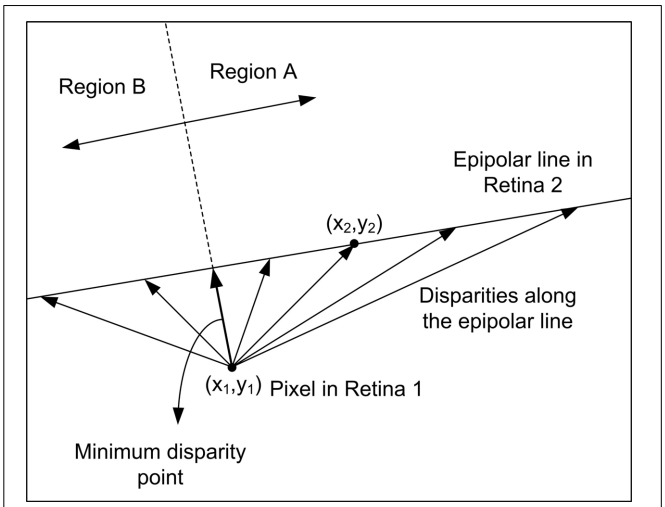



**FIGURE 9 | Measurement of the disparity (distance) between a pixel in Retina 1 and its corresponding epipolar line in Retina 2.** The minimum disparity point separates Region A and B.

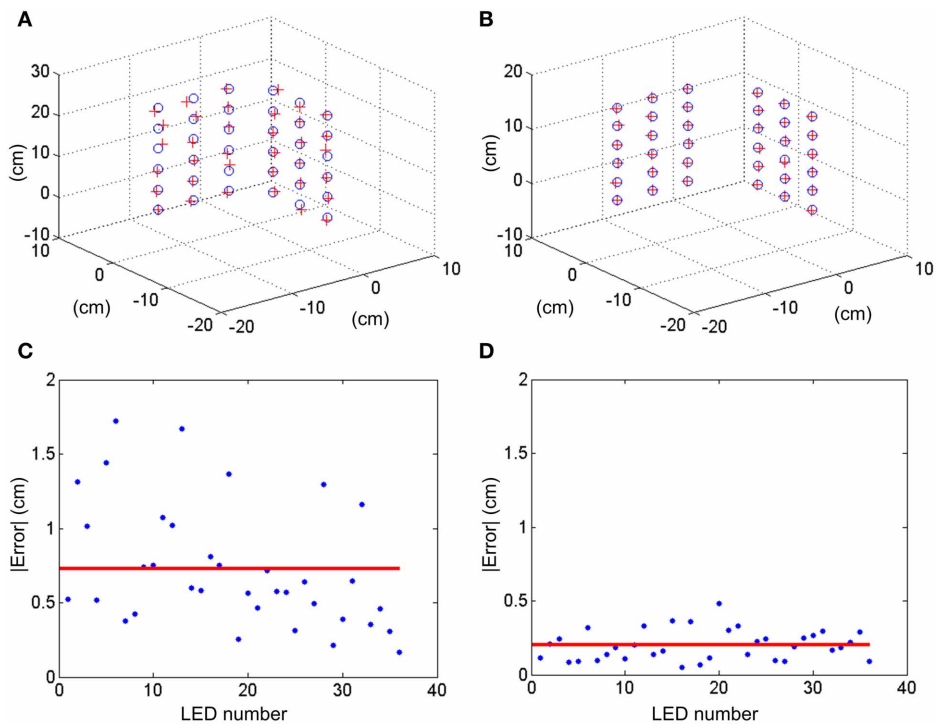



**FIGURE 8 | 3D reconstruction of the coordinates of the calibration LEDs. (A)** With pixel resolution and **(B)** with sub-pixel resolution. Blue circles represent the real location of the LEDs, while red crosses indicate the reconstructed coordinate. **(C,D)** Show the measured errors absolute value in cm for approaches 1 and 2, respectively. Red lines represent the mean error.

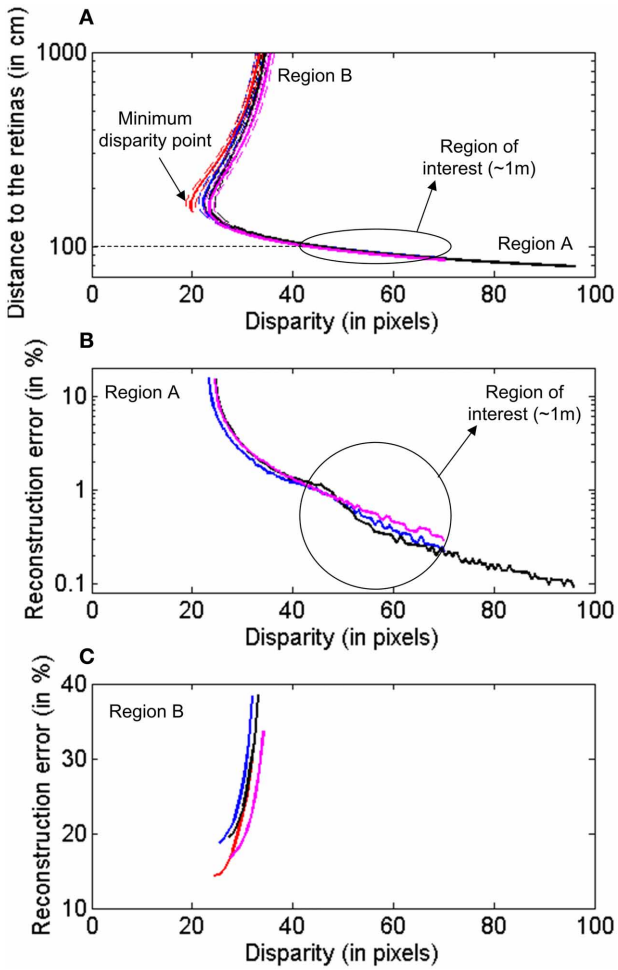



**FIGURE 10 | Characterization of the 3D reconstruction of the epipolar lines for different pixels in Retina 1.** Each color represents a different pixel. **(A)** Distance between the reconstructed points and the retinas for different disparity values. The dashed lines represent the upper and lower limits associated to the allowed deviation around the epipolar line. **(B)** Reconstruction error for 3D points closer to the retinas, Region A. **(C)** Reconstruction error for points farther from the retinas, Region B.

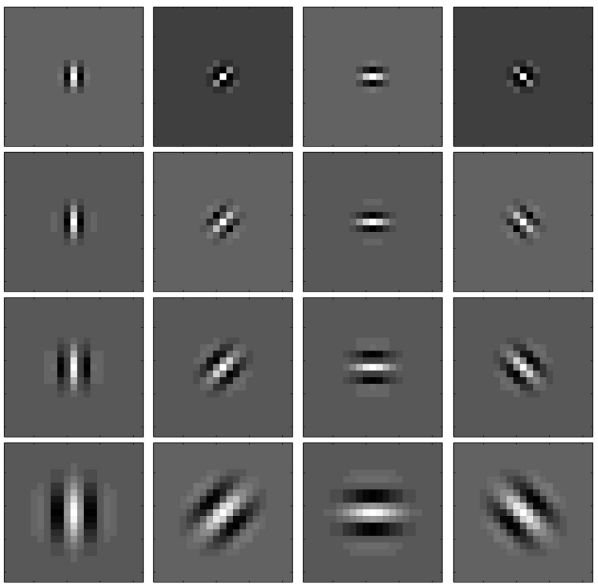


**FIGURE 11 | Kernels used for the 4-orientation configuration.** Each row represents a different scale (from smaller to larger kernels). The maximum kernel value is 15 and the minimum is −7. Kernel size is 11 × 11 pixels.

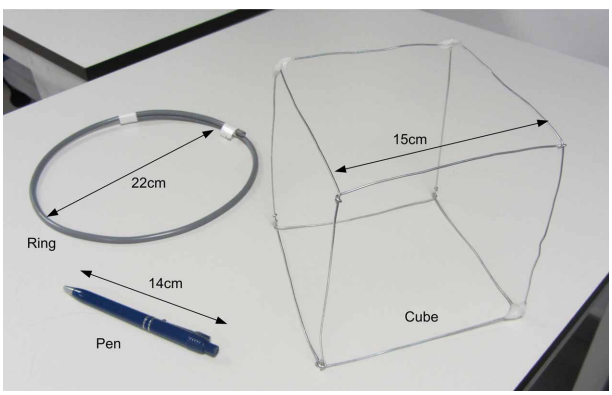



**FIGURE 12 | Photograph of the three objects used to test the 3D reconstruction algorithm: a pen, a ring, and a cube.**

simultaneous event burst in both cameras, there is only one LED in 3D space blinking, resulting in a unique spatial correspondence between the events produced in both retinas and the original 3D position. This recording was processed offline to obtain the 2D coordinates of the LEDs projected in both retinas following two different approaches:

(1) We represent a 2D image coding the number of spikes generated by each pixel. This way for each LED we obtain a cluster of pixels with large values. The coordinate of the pixel with the largest value in each cluster is considered to be the 2D projection of the LED. The accuracy of this measurement is one pixel.
(2) Using the same 2D image, the following method is applied. First, all those pixels with a number of spikes below a certain threshold are set to zero, while all those pixels above the threshold are set to one, obtaining a binarization of the image. Figure S1 in Calculation of Projection Matrix P in Supplementary Material shows an example of a 2D binarized image obtained for one DVS, where the 36 clusters represent the responses to the blinking LEDs. Then, for each cluster of pixels we calculate the mean coordinate, obtaining the 2D projection of the LEDs with sub-pixel resolution.

In both cases, these 2D coordinates together with the known 3D positions of the LEDs in space are used to calculate the projection matrices $P_1$ and $P_2$, and the fundamental matrix $F$ following the methods described in section Stereo Calibration. To validate the calibration, $P_1$ and $P_2$ were used to reconstruct the 3D calibration pattern following the method described in section 3D

**Table 1 | Comparison of the 3D reconstruction results for the pen.**

| | | Scale 1 | | | | | | | Scale 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Orientations | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | 100 | 71 | 65 | 78 | 77 | 87 | 100 | 105 | 73 | 78 | 85 | 98 | 121 | 128 | 146 |
| $N_m$ | 28 | 15 | 14 | 15 | 14 | 16 | 17 | 18 | 15 | 15 | 16 | 18 | 22 | 24 | 27 |
| Matching rate | 28 | 21 | 21 | 19 | 19 | 18 | 17 | 17 | 21 | 20 | 19 | 18 | 18 | 19 | 19 |
| Isolated events | 2.9 | 5.6 | 6.4 | 5.4 | 5.8 | 5.1 | 4.5 | 4.1 | 5.1 | 4.9 | 4.7 | 4.1 | 3.0 | 2.6 | 2.1 |
| $M_{err}$ | 8.0 | 4.1 | 3.9 | 4.2 | 3.9 | 4.1 | 3.9 | 4.1 | 3.6 | 3.6 | 3.7 | 3.6 | 3.6 | 3.6 | 3.6 |
| $N_{m-correct}$ | 24.9 | 14 | 13 | 14 | 13 | 15 | 16 | 17 | 14 | 14 | 15 | 17 | 21 | 23 | 25 |
| | | **Scale 3** | | | | | | | **Scale 4** | | | | | | |
| Orientations | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | | 74 | 80 | 87 | 106 | 131 | 154 | 169 | 77 | 79 | 85 | 99 | 129 | 145 | 170 |
| $N_m$ | | 16 | 17 | 17 | 21 | 26 | 31 | 34 | 19 | 19 | 19 | 22 | 30 | 34 | 39 |
| Matching rate | | 22 | 21 | 20 | 19 | 20 | 20 | 20 | 24 | 24 | 23 | 23 | 23 | 23 | 23 |
| Isolated events | | 5.0 | 4.7 | 4.5 | 3.3 | 2.4 | 1.8 | 1.5 | 3.3 | 3.3 | 3.2 | 2.6 | 1.6 | 1.4 | 1.0 |
| $M_{err}$ | | 5.2 | 5.2 | 5.2 | 5.1 | 4.9 | 4.9 | 5.0 | 8.3 | 8.3 | 8.3 | 8.3 | 8.2 | 8.1 | 7.8 |
| $N_{m-correct}$ | | 14 | 15 | 15 | 19 | 24 | 29 | 32 | 17 | 17 | 17 | 20 | 27 | 31 | 36 |

*The first column (0 orientations) presents the results obtained applying the matching algorithm to the retinas events (algorithm A, section Event Matching), while the rest of the columns are related to the pair-wise application of the matching algorithm to the outputs of the Gabor filters (algorithm B, section Event Matching), from Scale 1 (smaller kernels) to Scale 4 (larger kernels). For each scale, different numbers of orientations are considered (from 2 to 8), as indicated in the first row (Orientations). Second row ($N_{ev}$) shows the number of events processed (in Kevents) by the matching algorithm in each case (i.e., the total number of events generated by all the filters). Third row ($N_m$) presents the number of matched events (in Kevents) produced by the algorithm, while fourth row (Matching Rate) shows the ratio of matched events over the total number of events generated by the Gabor filters (Matching Rate* $= 100 \cdot N_m/N_{ev}$ *, in %). Fifth row (Isolated events) shows the ratio of isolated events over the total number of matched events (in %). Sixth row ($M_{err}$) presents the ratio of wrongly matched events over the total number of matched events (in %). The last row ($N_{m-correct}$) encapsulates the number of matched events with the ratio of isolated and wrongly matched events, presenting the number of correctly matched events* ($N_{m-correct} = N_m - \left(\frac{Isolated\ events}{100} \cdot N_m\right) - \left(\frac{M_{err}}{100} \cdot N_m\right)$*, in Kevents).*

Reconstruction, obtaining the results shown in **Figures 8A,B.** The reconstruction error is measured as the distance between each original 3D point and its corresponding reconstructed position, giving the results shown in **Figures 8C,D.** As can be seen in the figure, the mean reconstruction error for approach 1 is 7.3 mm with a standard deviation of 4.1 mm, while for approach 2 it is only 2 mm with a standard deviation of 1 mm. This error is comparable to the size of each LED (1 mm).

## PRECISION CHARACTERIZATION

Using the calibration results obtained in the previous subsection, we performed the following evaluation of the 3D reconstruction method. For a fixed pixel $m_1^1$ in Retina 1, we used the fundamental matrix $F$ to calculate the corresponding epipolar line in Retina 2 $Ep_2^1$, as represented in **Figure 9**. Although a perfect alignment between the two retinas would produce an epipolar line parallel to the x-axis and crossing the pixel position [minimum disparity point coincident with $(x_1, y_1)$], we represent a more general case, where the alignment is performed manually and is not perfect. This case is illustrated in Figure S1 (see Calculation of Projection Matrix P in Supplementary Material), where we show the 2D images representing the activity recorded by both retinas during calibration. The orientations of the epipolar lines indicate that the alignment is not perfect. The mean disparity for the LEDs coordinates is 24.55 pixels. Considering that we admit a deviation around the epipolar line of $\delta_{Ep_i} = 1$ *pixel* in the matching algorithm, we calculated two more lines, an upper and a lower limit, given by the distance of $\pm 1$ *pixel* to the epipolar line. Using projection matrices $P_1$ and $P_2$, we reconstructed the 3D coordinates for all the points in these three lines. We repeated the procedure for a total of four different pixels in Retina 1 $m_1^i$ ($i = 1, 2, 3, 4$) distributed around the visual space, obtaining four sets of 3-dimensional lines. In **Figure 10A**, we represent the distance between these 3D points and the retinas for each disparity value [the disparity measures the 2D euclidean distance between the projections of a 3D point in both retinas $(x_1, y_1)$ and $(x_2, y_2)$], where each color corresponds to a different pixel $m_1^i$ in Retina 1, and the dashed lines represent the upper and lower limits given by the tolerance of 1 pixel around the epipolar lines. As can be seen in the figure, each disparity has two different values of distance associated, which represent the two possible points in $Ep_2^i$ which are at the same distance from $m_1^i$. This effect results in two different zones in each trace (regions A and B in **Figure 9**), which correspond to two different regions in the 3D space, where the performance of the reconstruction changes drastically. Therefore, we consider both areas separately in order to estimate the reconstruction error. Using the range of distances given by **Figure 10A** between each pair of dashed lines, we calculate the reconstruction error for each disparity value as $(d_{\max} - d_{\min})/\mu_d$, where $d_{\max}$ and $d_{\min}$ represent the limits of the range of distance at that point, and $\mu_d$ is the mean value. **Figure 10B** shows the obtained error for the 3D points located in the closer region (A), while **Figure 10C**

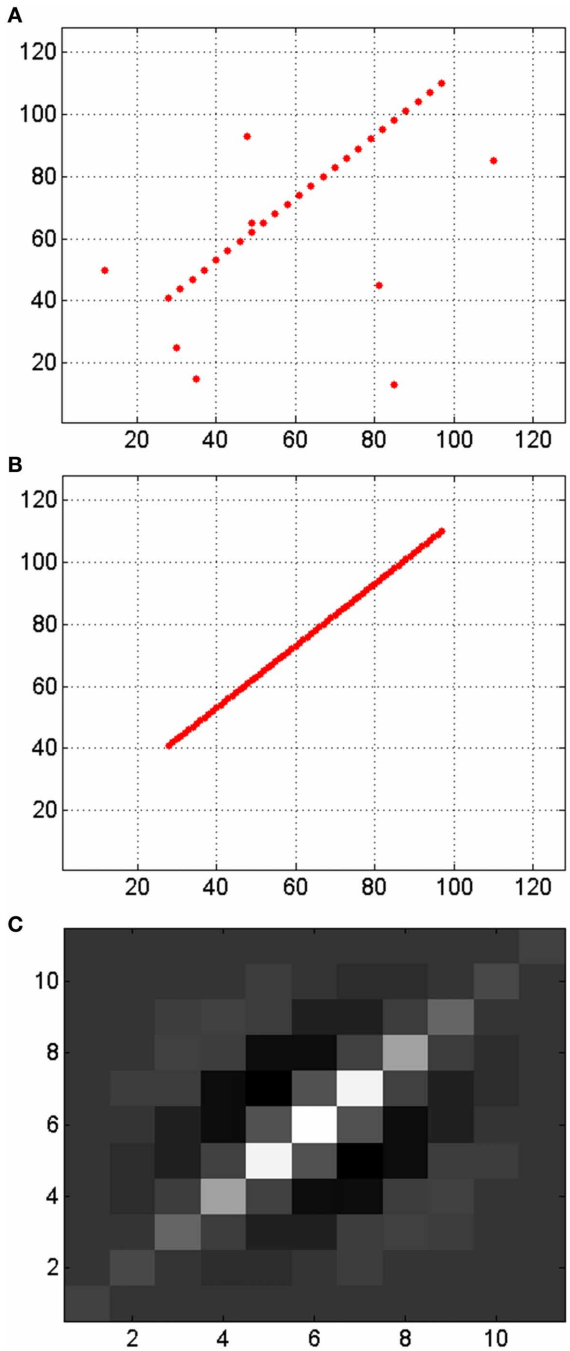



**FIGURE 13 | Illustration of enhancing edges and noise reduction by a Gabor filter. (A)** Input events representing a discontinuous edge with noise. **(B)** Output events generated by the Gabor filter, with the reconstructed edge without noise. **(C)** Gabor kernel. All axes represent pixels, being the visual space in **(A,B)** 128 × 128 and the size of the kernel in **(C)** 11 × 11.

corresponds to the points farther from the retinas (Region B). In both figures, each line represents a different pixel $m_1^i$ in Retina 1. As shown in **Figure 10B**, the reconstruction error in the area of interest (around $1m$ distance from the retinas) is less than 1.5%. Note that the minimum disparity value is around 20 pixels (while a perfect alignment would give 0), showing the robustness of the method for manual approximate alignment.

## 3D RECONSTRUCTION

For the experimental evaluation of the 3D reconstruction, we analyzed the effect of several configurations of Gabor filters on the event matching algorithm B in order to compare them to algorithm A. For each configuration, we tested different numbers of orientation Gabor filters (from 2 to 8). All filters had always the same spatial scale, and we tested 4 different scales. Identical filters were applied to both retina outputs. Each row in **Figure 11** shows an example of the kernels used in a configuration of 4 orientations (90, 45, 0, −45°), each configuration for a given spatial scale. In general, the different angles implemented in each case are uniformly distributed between 90 and −90°. This strategy was used to reconstruct in 3D the three objects shown in **Figure 12**: a 14 cm pen, a 22 cm diameter ring, and a 15 cm side metal wire cube structure.

### *Pen*

A swinging pen of 14 cm length was moved in front of the two retinas for half a minute, with a number of approximately 100 Kevents generated by each retina. **Table 1** summarizes the results of the 3D reconstruction, in terms of events. The column labeled "Orientations 0" corresponds to applying the matching algorithm directly to the retina pair outputs (algorithm A). When using Gabor filters (algorithm B), experiments with four different scales were conducted. For each scale, a different number of simultaneous filter orientations were tested, ranging from 2 to 8. In order to compare the performance of the stereo matching algorithm applied directly to the retinas (algorithm A, see section Event Matching) and applied to the outputs of the Gabor filters (algorithm B, see section Event Matching), the second row in **Table 1** ($N_{ev}$) shows the number of events processed by the algorithm in both cases. We show only the number of events coming originally from Retina 1, as they both have been configured to generate approximately the same number of events for a given stimulus.

When the algorithm is applied directly to the output of the retinas, the number of matched pairs of events obtained is around 28 Kevents (28% of success rate). The third row in **Table 1** ($N_m$) shows the number of matched events for the different configurations of Gabors. If we calculate the percentage of success obtained by the algorithm for each configuration of filters in order to compare it with the 28% provided by the retinas alone, we obtain the values shown in the fourth row of **Table 1** (*Matching Rate*).

Although these results show that the matching rate of the algorithm is smaller when we use Gabor filters to extract information about the orientation of the edges that generated the events, we should consider that the performance of 3D reconstruction is determined by the total number of matched events, not the relative proportion. Note that the Gabor filters are capable of edge filling when detecting somewhat sparse or incomplete edges from the retina, thus enhancing edges and providing more events for these edges. **Figure 13** shows an example where a weak edge (in **Figure 13A**) produced by a retina together with noise events is

filled by a Gabor filter (with the kernel shown in **Figure 13C**) producing the enhanced noise-less edge in **Figure 13B**, and increasing the number of edge events from 24 to 70 while removing all retina-noise events. The more matched events, the better 3D reconstruction. For that reason, we consider that a bank of 8 Gabor filters with kernels of scale 4 gives the best result, with more than 39 Kevents that can be used to reconstruct the 3D sequence, using 100 Kevents generated by the retinas. This application of Gabor filters for edges filling was first demonstrated in (Lindenbaum et al., 1994), and has also been used for fingerprint image enhancement (Hong et al., 1998; Greenberg et al., 2002).

Another parameter that can be used to measure the quality of the 3D reconstruction is the proportion of "isolated" events in the matched sequence. We define an isolated event as an event which is not correlated to any other event in a certain spatio-temporal window, meaning that no other event has been generated in its neighbor region within a limited time range. A non-isolated event (an event generated by an edge of the object) will be correlated

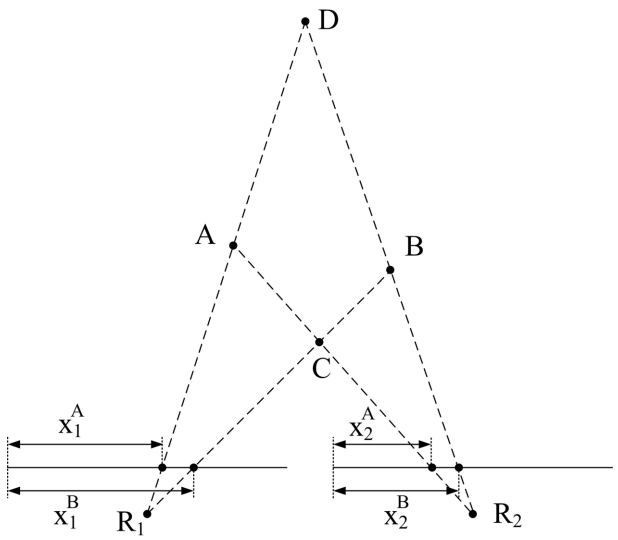



**FIGURE 14 | Illustration of matching errors.**

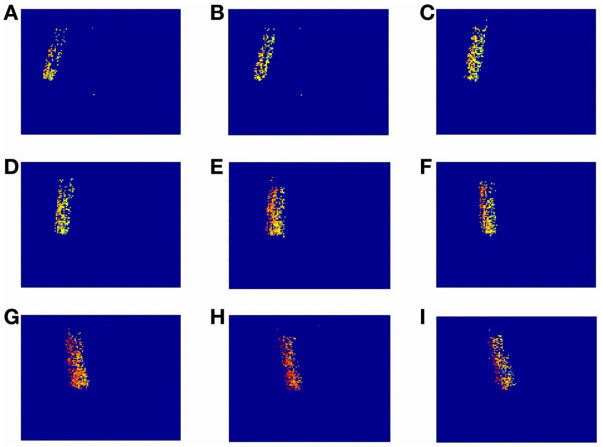



**FIGURE 16 | Sequence of disparity maps.** They were reconstructed with $T_{frame} = 50$ ms and they correspond to the movement of the swinging pen (from **A**–**I**). The disparity scale goes from dark blue to red to encode events from far to near.

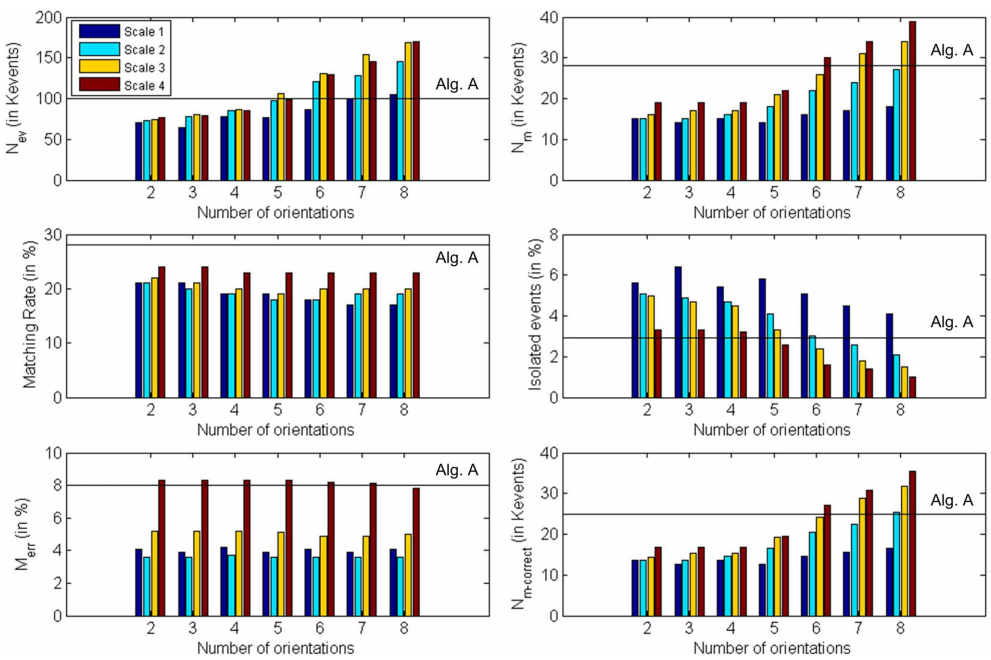



**FIGURE 15 | Graphical representation of Table 1.** Each subplot corresponds to a different row of the table, showing the obtained values for each number of orientations and scale. The black horizontal lines indicate the values obtained using algorithm A (0 orientations).

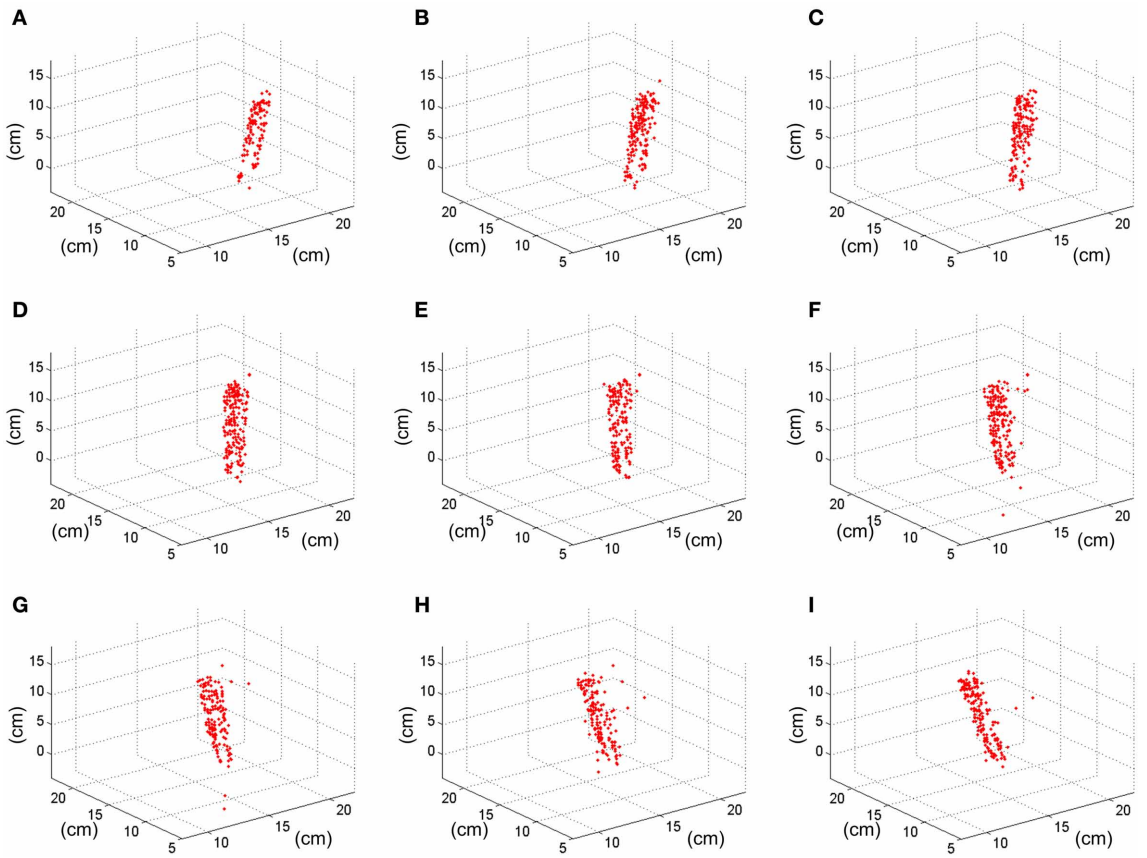



**FIGURE 17 | Result of the 3D reconstruction of the swinging pen recording.** Each plot (from **A–I**) corresponds to a 50 ms-frame representation of the 3D coordinates of the matched events.

to some other events generated by the same edge, which will be close in space and time. Note that these isolated matched events correspond to false matches. These false matches can be produced when an event in one retina is matched by mistake with a noise event in the other retina, or when two or more events that happen very simultaneously in 3D space are cross-matched by the matching algorithm. With this definition of isolated events, the 28 Kevents that were matched for the retinas without any filtering were used to reconstruct the 3D coordinates of these events, resulting in only 2.93% of isolated events. After the application of the same methodology to all the Gabor filters configurations, the results in the fifth row in **Table 1** (*Isolated events*) are obtained. These results show that several configurations of Gabor filters give a smaller proportion of isolated events.

In order to remove the retina-noise events, it is also possible to insert a noise removal block directly at the output of the retina (jAER, 2007). However, this introduces a small extra latency before the events can be processed, thus limiting event-driven stereo vision for very high speed applications (although it can be a good solution when timing restrictions are not too critical). The effect of Gabor filters on noise events is also illustrated in **Figure 13**, where all the events that were not part of an edge with the appropriate orientation are removed by the filter. However, it is possible that some noise events add their contributions together producing noise events at the output of the Gabor filters. Two different things can happen with these events: (1) the stereo matching algorithm does not find a corresponding event in the other retina; (2) there is a single event which satisfies all restrictions, so a 3D point will be reconstructed from a noise event, producing a wrongly matched event, as is described in the next paragraph.

Although the object used in this first example is very simple, we must consider the possibility that the algorithm matches wrongly some events. In particular, if we think about a wide object we can have events generated simultaneously by two far edges: the left and the right one. Therefore, it can happen that an event corresponding to the left edge in Retina 1 does not have a proper partner in Retina 2, but another event generated by the right edge in Retina 2 might satisfy all the restrictions imposed by the matching algorithm. **Figure 14** illustrates the mechanism that produces this error. Let us assume that the 3D object has its left and right edges located at positions A and B in 3D space. Locations A and B produce events at $x_1^A$ and $x_1^B$ in Retina 1, and at $x_2^A$ and $x_2^B$ in Retina 2. These events are the projections onto the focal points $R_1$ and $R_2$ of both retinas, activating pixels $\left(x_i^j,\ y_i^j\right)$, with $i = 1, 2$ and $j = A, B$. Therefore, an event generated in Retina

**Table 2 | Comparison of the 3D reconstruction results for the ring.**

| | | Scale 1 | | | | | | | Scale 2 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Orientations | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | 115 | 78 | 75 | 100 | 109 | 131 | 151 | 168 | 78 | 95 | 119 | 143 | 177 | 197 | 229 |
| $N_m$ | 17 | 8 | 8 | 9 | 10 | 12 | 14 | 16 | 8 | 10 | 12 | 15 | 19 | 21 | 25 |
| Matching rate | 15 | 10 | 11 | 9 | 10 | 10 | 9 | 9 | 10 | 11 | 10 | 10 | 11 | 11 | 11 |
| Isolated events | 5.9 | 7.8 | 7.1 | 6.5 | 5.4 | 4.9 | 4.1 | 3.9 | 7.6 | 6.2 | 5.0 | 3.9 | 3.0 | 2.6 | 1.9 |
| $M_{err}$ | 12.0 | 9.9 | 9.5 | 9.3 | 8.7 | 8.5 | 8.7 | 8.9 | 9.3 | 9.0 | 8.4 | 8.1 | 8.2 | 8.0 | 7.8 |
| $N_{m-correct}$ | 14 | 7 | 7 | 8 | 9 | 10 | 12 | 14 | 7 | 8 | 10 | 13 | 17 | 19 | 23 |
| | | **Scale 3** | | | | | | | **Scale 4** | | | | | | |
| Orientations | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | | 82 | 103 | 122 | 157 | 185 | 217 | 245 | 83 | 107 | 131 | 161 | 201 | 229 | 266 |
| $N_m$ | | 8 | 10 | 12 | 16 | 19 | 22 | 25 | 6 | 9 | 11 | 14 | 17 | 20 | 23 |
| Matching rate | | 9 | 10 | 10 | 10 | 10 | 10 | 10 | 8 | 8 | 8 | 9 | 9 | 9 | 9 |
| Isolated events | | 7.5 | 6.3 | 4.8 | 3.5 | 2.9 | 2.3 | 2.0 | 7.7 | 6.3 | 5.1 | 3.9 | 3.0 | 2.5 | 2.0 |
| $M_{err}$ | | 8.9 | 7.7 | 7.3 | 6.8 | 6.5 | 6.6 | 6.4 | 8.4 | 6.5 | 6.2 | 5.7 | 5.9 | 5.8 | 5.6 |
| $N_{m-correct}$ | | 7 | 9 | 11 | 14 | 17 | 20 | 23 | 5 | 8 | 10 | 13 | 15 | 18 | 21 |

*The meaning of the columns and rows is as in* ***Table 1****.*

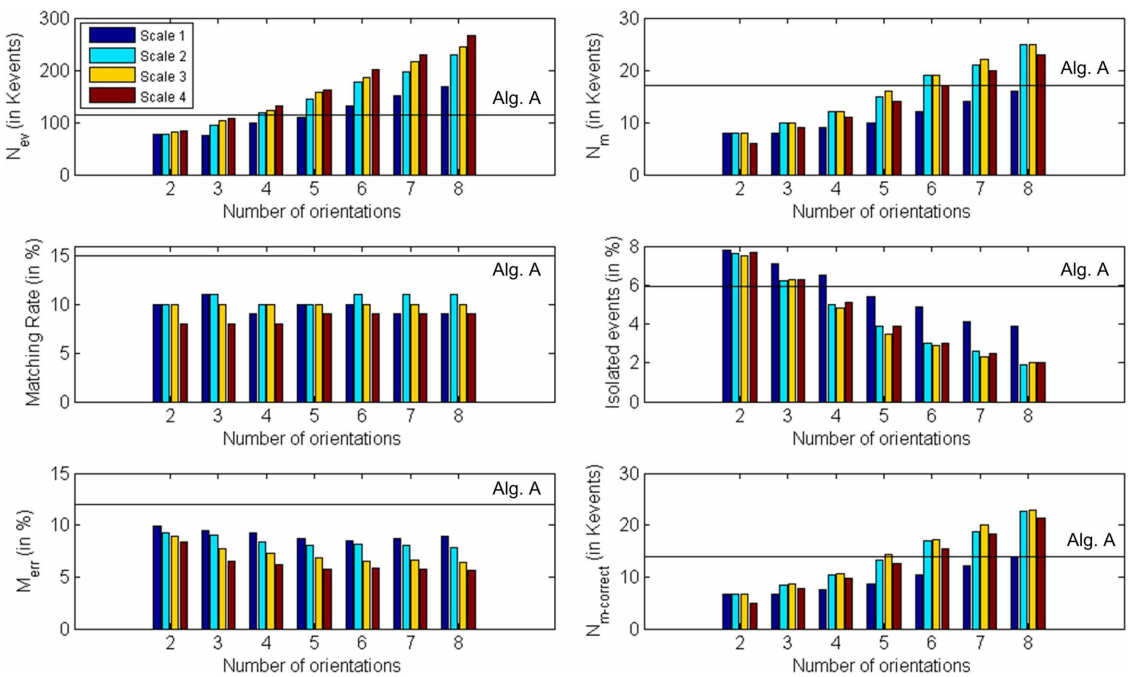



**FIGURE 18 | Graphical representation of Table 2.** Each subplot corresponds to a different row of the table, showing the obtained values for each number of orientations and scale. The black horizontal lines indicate the values obtained using algorithm A (0 orientations).

1 with coordinates $(x_1^A,\ y_1^A)$ should match another event generated in Retina 2 with coordinates $(x_2^A,\ y_2^A)$. However, note that in **Figure 13**, an edge at position D is captured by Retina 1 at the same pixel that an edge at A, and in Retina 2 they would be on the same epipolar lines. The same happens for edges at positions B and C. Consequently, it can happen that no event is produced in Retina 2 at coordinate $(x_2^A,\ y_2^A)$ at the same time, but another event with coordinates $(x_2^B,\ y_2^B)$ is generated within a short time range by the opposite simultaneously moving edge, being those coordinates in the same epipolar line. In that case, the algorithm might match $(x_1^A,\ y_1^A)$ with $(x_2^B,\ y_2^B)$, reconstructing a wrong 3D point in coordinate D. The opposite combination would produce a wrong 3D event in point C. This effect could produce false edges in the 3D reconstruction, especially when

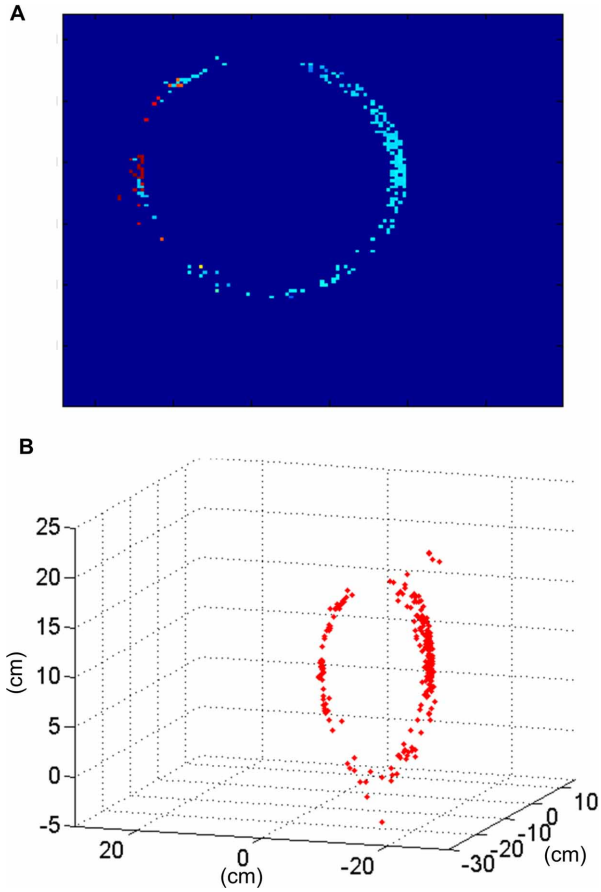



**FIGURE 19 | Results obtained for the rotating ring. (A)** Disparity map reconstructed with $T_{\text{frame}} = 50$ ms corresponding to the rotation of the ring. **(B)** Result of the 3D reconstruction of the same frame of the ring recording.

processing more complex objects. However, the introduction of the Gabor filters to extract the orientation of the edges will reduce the possibility of matching wrong pairs of events. In order to measure the proportion of wrongly matched events, we consider that all the good pairs of events will follow certain patterns of disparity, so all the events which are close in time will be included within a certain range of disparity values. Calculating continuously the mean and standard deviation of the distribution of disparities, we define the range of acceptable values, and we identify as wrongly matched all those events whose disparity is outside that range. Using this method, we calculate the proportion of wrongly matched events and present it (in %) in the sixth row of **Table 1** ($M_{err}$). Finally, the last row presents the number of correctly matched events, subtracting both the isolated and wrongly matched events from the total number of matched events: $N_{\text{m-correct}} = N_m - \left(\frac{\text{Isolated events}}{100} \cdot N_m\right) - \left(\frac{M_{\text{err}}}{100} \cdot N_m\right)$. All these results are presented graphically in **Figure 15**, where the colored vertical bars represent the results obtained applying algorithm B with different number of orientations and scales, while the black horizontal lines indicate the values obtained using algorithm A (no Gabor filters). From this figure, we decide that the best case is 8 orientations and Scale 4, as it provides the largest number of correctly matched events. However, it could also be argued that 8 orientations and Scale 3 gives a smaller number of wrongly matched events, but in that case the number of correctly matched events is also smaller.

Using the sequence of matched events provided by the algorithm in the best case (8 orientations, Scale 4), we computed the disparity map. The underlying reasons why this configuration provides the best result are: (a) Scale 4 matches better the scale of the object edges in this particular case, and (b) given the object geometry and its tilting in time, a relatively fine orientation angle detection was required. If we compare this case with the results obtained applying algorithm A without Gabor filters (first column in **Table 1**), we observe an increase of 39% in the number of matched events, while the proportions of isolated events and wrongly matched pairs have decreased by 65 and 2.5%, respectively. Moreover, the number of correctly matched events has increased by 44%. In order to compute the disparity map, we calculated the euclidean distance between both pixels in each pair of events (from Retina 1 and Retina 2). This measurement is inversely proportional to the distance between the represented object and the retinas, as further objects produce a small disparity and closer objects produce a large disparity value. **Figure 16** shows 9 consecutive frames of the obtained disparity sequence, with a frame time of 50 ms. The disparity scale goes from dark blue to red to encode events from far to close.

Applying the method described in section 3D Reconstruction, the 3 dimensional coordinates of the matched events are calculated. **Figure 17** shows 9 consecutive frames of the resultant 3D reconstruction, with a frame time of 50 ms. The shape of the pen is clearly represented as it moves around 3D space. Using this sequence, we measured manually the approximate length of the pen by calculating the distance between the 3D coordinates of pairs of events located in the upper and lower limits of the pen, respectively. This gave an average length of 14.85 cm, being the real length 14 cm, which means an error of 0.85 cm. For an approximate distance to the retinas of 1 m, the maximum error predicted in **Figure 10** would be below 1.5%, resulting in 1.5 cm. Therefore, we can see that the 0.85 cm error is smaller than the maximum predicted by **Figure 10**.

### *Ring*

A ring with a diameter of 22 cm was rotating slowly in front of the two retinas for half a minute, with a number of approximately 115 Kevents generated by each retina. As in the previous example, the matching algorithm was applied both to the events generated by the retinas (see section Event Matching, algorithm A) and to the events generated by the Gabor filters (see section Event Matching, algorithm B), in order to compare both methods. **Table 2** shows all the results for all the configurations of Gabor filters (from 2 to 8 orientations, with scales 1–4). All these results are presented graphically in **Figure 18**, where the colored vertical bars represent the results obtained applying algorithm B with different number of orientations and scales, while the black horizontal lines indicate the values obtained using algorithm A (no Gabor filters). We can see in the table how the largest number of matched events (25 K) is obtained for 8 orientations and both

**Table 3 | Comparison of the 3D reconstruction results for the cube.**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Scale 1** | | | | | | | **Scale 2** | | | | | | |
| Orientations | 0 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | 118 | 54 | 68 | 100 | 112 | 132 | 153 | 178 | 50 | 93 | 125 | 152 | 183 | 205 | 243 |
| $N_m$ | 11 | 6 | 10 | 13 | 15 | 18 | 21 | 24 | 6 | 11 | 14 | 17 | 21 | 24 | 28 |
| Matching rate | 9 | 12 | 14 | 13 | 14 | 14 | 14 | 14 | 11 | 12 | 11 | 11 | 11 | 11 | 12 |
| Isolated events | 14.0 | 5.2 | 5.0 | 4.5 | 4.3 | 3.8 | 3.7 | 3.3 | 5.0 | 5.0 | 4.1 | 4.1 | 3.9 | 3.4 | 3.1 |
| $M_{err}$ | 20.3 | 17.0 | 15.5 | 15.1 | 15.0 | 15.1 | 15.8 | 14.1 | 17.9 | 14.2 | 11.9 | 11.1 | 13.3 | 12.0 | 10.3 |
| $N_{m-correct}$ | 6 | 5 | 8 | 10 | 12 | 15 | 17 | 20 | 5 | 9 | 12 | 14 | 17 | 20 | 24 |
| | | **Scale 3** | | | | | | | **Scale 4** | | | | | | |
| Orientations | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $N_{ev}$ | | 54 | 130 | 170 | 219 | 256 | 300 | 346 | 51 | 145 | 190 | 235 | 285 | 329 | 386 |
| $N_m$ | | 5 | 12 | 14 | 20 | 23 | 27 | 31 | 3 | 10 | 12 | 16 | 19 | 21 | 25 |
| Matching rate | | 9 | 9 | 8 | 9 | 9 | 9 | 9 | 6 | 7 | 6 | 7 | 7 | 7 | 7 |
| Isolated events | | 5.2 | 4.2 | 4.1 | 3.6 | 3.1 | 3.2 | 3.0 | 4.8 | 3.7 | 3.2 | 3.1 | 2.9 | 2.7 | 2.8 |
| $M_{err}$ | | 19.0 | 15.1 | 12.7 | 11.3 | 11.9 | 11.2 | 10.9 | 27.4 | 15.0 | 12.9 | 11.4 | 13.7 | 12.2 | 10.7 |
| $N_{m-correct}$ | | 4 | 10 | 12 | 17 | 20 | 23 | 27 | 2 | 8 | 10 | 14 | 16 | 18 | 22 |

*The meaning of the columns and rows is as in* ***Table 1****.*

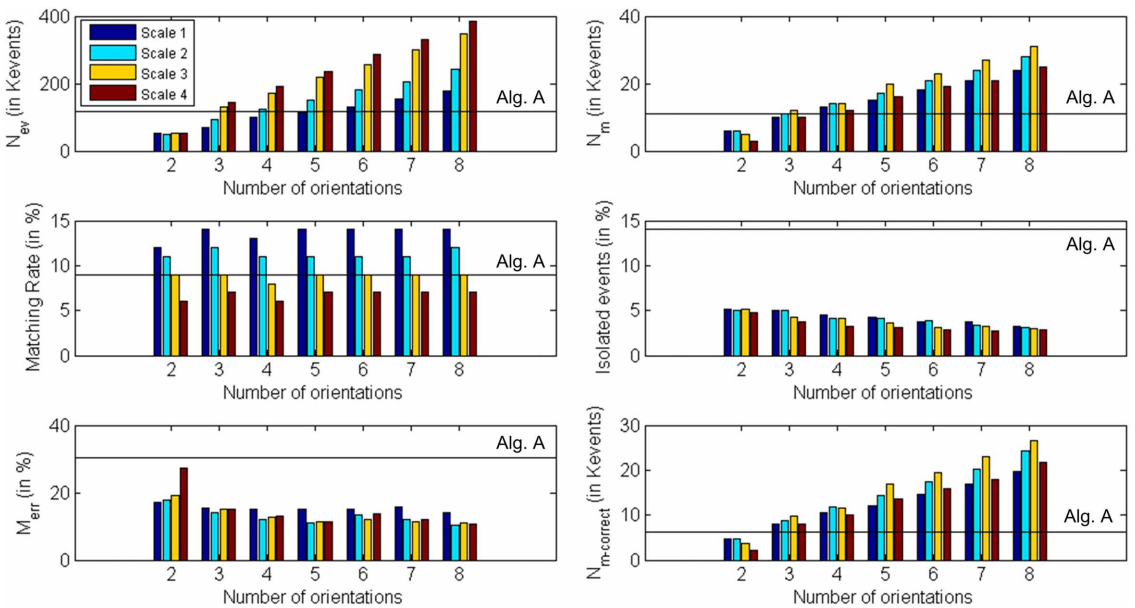



**FIGURE 20 | Graphical representation of Table 3.** Each subplot corresponds to a different row of the table, showing the obtained values for each number of orientations and scale. The black horizontal lines indicate the values obtained using algorithm A (0 orientations).

scales 2 and 3. Although the ratio of noise events is very similar for both of them (1.9% for Scale 2 and 2.0% for Scale 3), Scale 3 provides a smaller ratio of wrongly matched events (7.8% for Scale 2 and 6.4% for Scale 3). Therefore, we conclude that the best performance is found with 8 orientations and Scale 3, as it is more appropriate to the geometry of the object. If we compare this case with the results obtained applying algorithm A without Gabor filters (first column in **Table 2**), we observe an increase of 47% in the number of matched events, while the proportions of isolated events and wrongly matched pairs have decreased by 66 and 46%, respectively. Therefore, the number of correctly matched events has increased by 64%. A frame reconstruction of the disparity map and the 3D sequence are shown in **Figure 19**.

The diameter of the reconstructed ring was measured manually by selecting pairs of events with the largest possible separation. This gave an average diameter of 21.40 cm, which implies a reconstruction error of 0.6 cm. This error is also smaller than the maximum predicted in **Figure 10**.

### *Cube*

Finally, a cube with an edge length of 15 cm was rotating in front of the retinas, with a number of approximately 118 Kevents generated by each retina in approximately 20 s. The same procedure performed in previous examples was repeated, obtaining the results shown in **Table 3**. All these results are presented graphically in **Figure 20**, where the colored vertical bars represent the results obtained applying algorithm B with different number of orientations and scales, while the black horizontal lines indicate the values obtained using algorithm A (no Gabor filters). In this case, the largest number of matched events (31 K) is given by 8 orientations and Scale 3, while both the ratio of isolated events and the ratio of wrongly matched events are very similar for the four different scales with 8 orientations (around 3% noise and 10.9% wrong matches). Therefore, the best performance is given by 8 orientations and Scale 3. If we compare this case with the results obtained applying algorithm A without Gabor filters (first column in **Table 3**), we observe an increase of 181% in the number of matched events, while the proportions of isolated events and wrongly matched pairs have decreased by 78 and 46%, respectively. The number of correctly matched events has increased by 350%.

A reconstruction of the disparity map and the 3D sequence is shown in **Figure 21**. The ratio of wrongly matched events is much larger than on the ring example (about twice as much). That is because this object has many parallel edges, increasing the number of events in the same epipolar line which are candidates to be matched and which the orientation filters do not discriminate. While **Figure 14** shows a situation where 2 different positions in 3D space (A and B) can generate events that could be wrongly matched, in this case we could find at least 4 different positions in 3D space (as we have 4 parallel edges) with the same properties.

The edge length of the reconstructed 3D cube was measured manually on the reconstructed events, giving an average length of 16.48 cm, which implies a reconstruction error of 1.48 cm. This error is smaller than the maximum predicted in **Figure 10**.

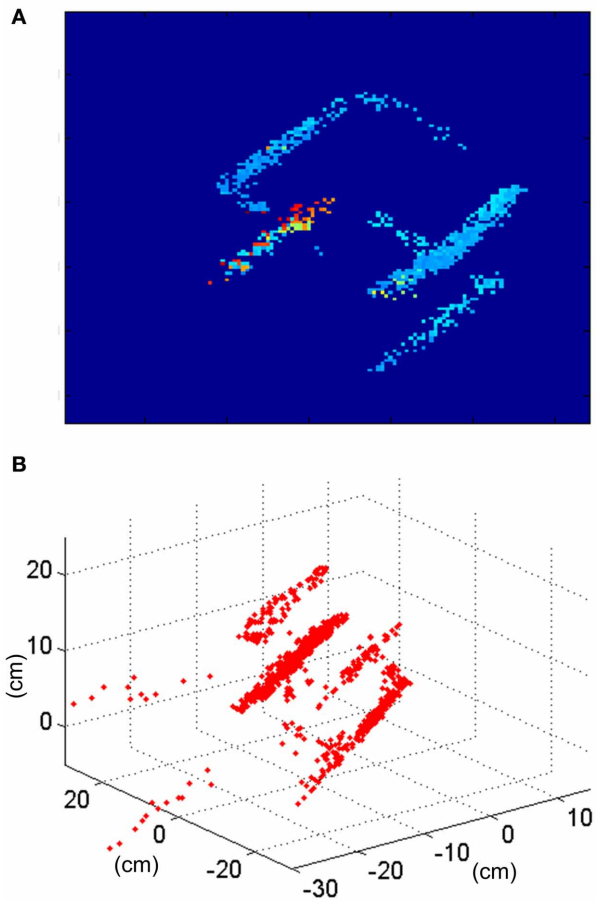



**FIGURE 21 | Results obtained for the cube. (A)** Disparity map reconstructed with $T_{frame}$ = 50 ms corresponding to the rotation of the cube. **(B)** Result of the 3D reconstruction of the same frame of the cube recording.

## CONCLUSION

This paper analyzes different strategies to improve 3D stereo reconstruction in event-based vision systems. First of all, a comparison between stereo calibration methods showed that by using a calibration object with LEDs placed in known locations and measuring their corresponding 2D projections with sub-pixel resolution, we can extract the geometric parameters of the stereo setup. This method was tested by reconstructing the known coordinates of the calibration object, giving a mean error comparable to the size of each LED.

Event matching algorithms have been proposed for stereo reconstruction, taking advantage of the precise timing information provided by DVS sensors. In this work, we have explored the benefits of using Gabor filters to extract the orientation of the object edges and match events from pair wise filters directly. This imposes the restriction that the distance from the stereo cameras to the objects must be much larger than the focal length of the lenses, so that edge orientations appear similar in both cameras. By analyzing different numbers of filters with several spatial scales, we have shown that we can increase the number of reconstructed events for a given sequence, reducing the number of both noise events and wrong matches at the same time. This improvement has been validated by reconstructing in 3D three different objects. The size of these objects was estimated from the 3D reconstruction, with an error smaller than theoretically predicted by the method (1.5%).

## ACKNOWLEDGMENTS

This work has been funded by ERANET grant PRI-PIMCHI-2011-0768 (PNEUMA) funded by the Spanish Ministerio de Economía y Competitividad, Spanish research grants (with support from the European Regional Development Fund) TEC2009-10639-C04-01 (VULCANO) and TEC2012-37868-C04-01 (BIOSENSE), Andalusian research grant TIC-6091 (NANONEURO) and by the French national Labex program "Life-senses". The authors also benefited from both the CapoCaccia Cognitive Neuromorphic Engineering Workshop, Sardinia, Italy, and the Telluride Neuromorphic Cognition Engineering Workshop, Telluride, Colorado.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2014.00048/abstract

## REFERENCES


Benosman, R., Ieng, S., Rogister, P., and Posch, C. (2011). Asynchronous event-based Hebbian epipolar geometry. *IEEE Trans. Neural Netw.* 22, 1723–1734. doi: 10.1109/TNN.2011.2167239

Camuñas-Mesa, L., Acosta-Jiménez, A., Zamarreño-Ramos, C., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). A 32x32 pixel convolution processor chip for address event vision sensors with 155ns event latency and 20Meps throughput. *IEEE Trans. Circuits Syst. I* 58, 777–790. doi: 10.1109/TCSI.2010.2078851

Camuñas-Mesa, L., Zamarreño-Ramos, C., Linares-Barranco, A., Acosta-Jiménez, A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2012). An event-driven multi-kernel convolution processor module for event-driven visión sensors. *IEEE J. Solid State Circuits* 47, 504–517. doi: 10.1109/JSSC.2011.2167409

Carneiro, J., Ieng, S., Posch, C., and Benosman, R. (2013). Asynchronous event-based 3D reconstruction from neuromorphic retinas. *Neural Netw.* 45, 27–38. doi: 10.1016/j.neunet.2013.03.006

Cauwenberghs, G., Kumar, N., Himmelbauer, W., and Andreou, A. G. (1998). An analog VLSI chip with asynchronous interface for auditory feature extraction. *IEEE Trans. Circuits Syst. II* 45, 600–606. doi: 10.1109/82.673642

Chan, V., Liu, S. C., and van Schaik, A. (2007). AER EAR: a matched silicon cochlea pair with address event representation interface. *IEEE Trans. Circuits Syst. I* 54, 48—59. doi: 10.1109/TCSI.2006.887979

Choi, T. Y. W., Merolla, P., Arthur, J., Boahen, K., and Shi, B. E. (2005). Neuromorphic implementation of orientation hypercolumns. *IEEE Trans. Circuits Syst. I* 52, 1049–1060. doi: 10.1109/TCSI.2005.849136

Domínguez-Morales, M. J., Jiménez-Fernández, A. F., Paz-Vicente, R. Jiménez-Moreno, G., and Linares-Barranco, A. (2012). Live demonstration: on the distance estimation of moving targets with a stereo-vision AER system. *Int. Symp. Circuits Syst.* 2012, 721–725. doi: 10.1109/ISCAS.2012.6272137

Faugeras, O. (1993). Three-*Dimensional Computer Vision: a Geometric Viewpoint*. Cambridge, MA: MIT Press.

Gong, M. (2006). Enforcing temporal consistency in real-time stereo estimation. *ECCV* 2006, Part III, 564–577. doi: 10.1007/11744078_44

Granlund, G. H., and Knutsson, H. (1995). *Signal Processing for Computer Vision*. Dordrecht: Kluwer. doi: 10.1007/978-1-4757-2377-9

Greenberg, S., Aladjem, M., and Kogan, D. (2002). Fingerprint image enhancement using filtering techniques. *Real Time Imaging* 8, 227–236. doi: 10.1006/rtim.2001.0283

Hartley, R., and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. (New York, NY: Cambridge University Press). doi: 10.1017/CBO9780511811685

Hong, L., Wan, Y., and Jain, A. (1998). Fingerprint image enhancement: algorithm and performance evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* 20, 777–789. doi: 10.1109/34.709565

jAER Open Source Project. (2007). Available online at: http://jaer.wiki.sourcefourge.net

Khan, M. M., Lester, D. R., Plana, L. A., Rast, A. D., Jin, X., Painkras, E., et al. (2008). "SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *Proceedings International Joint Conference on Neural Networks, IJCNN 2008* (Hong Kong), 2849–2856. doi: 10.1109/IJCNN.2008.4634199

Kogler, J., Sulzbachner, C., and Kubinger, W. (2009). "Bio-inspired stereo vision system with silicon retina imagers," in *7th ICVS International Conference on Computer Vision Systems*, Vol. 5815, (Liege), 174–183. doi: 10.1007/978-3-642-04667-4_18

Lazzaro, J., Wawrzynek, J., Mahowald, M., Sivilotti, M., and Gillespie, D. (1993). Silicon auditory processors as computer peripherals. *IEEE Trans. Neural Netw.* 4, 523–528. doi: 10.1109/72.217193

Leñero-Bardallo, J. A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2010). A five-decade dynamic-range ambient-light-independent calibrated signed-spatial-contrast AER Retina with 0.1-ms latency and optional time-to-first-spike mode. *IEEE Trans. Circuits Syst. I* 57, 2632–2643. doi: 10.1109/TCSI.2010.2046971

Leñero-Bardallo, J. A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). A 3.6s latency asynchronous frame-free event-driven dynamic-vision-sensor. *IEEE J. Solid State Circuits* 46, 1443–1455. doi: 10.1109/JSSC.2011.2118490

Lichtsteiner, P., Posch, C., and Delbrück, T. (2008). A 128x128 120dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid State Circuits* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Longuet-Higgins, H. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature* 293, 133–135. doi: 10.1038/293133a0

Loop, C., and Zhang, Z. (1999). Computing rectifying homographies for stereo vision. *IEEE Conf. Comp. Vis. Pattern Recognit.* 1, 125–131. doi: 10.1109/CVPR.1999.786928

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* 60, 91–110. doi: 10.1023/B:VISI.0000029664.99615.94

Luong, Q. T. (1992). *Matrice Fondamentale et Auto-Calibration en Vision Par Ordinateur*. Ph.D. Thesis, Universite de Paris-Sud, Centre d'Orsay.

Mahowald, M. (1992). *VLSI Analogs of Neural Visual Processing: a Synthesis of form and Function*. Ph.D. dissertation, California Institute of Technology, Pasadena, CA.

Mahowald, M., and Delbrück, T. (1989). "Cooperative stereo matching using static and dynamic image features," in *Analog VLSI Implementation of Neural Systems*, eds C. Mead and M. Ismail (Boston, MA: Kluwer Academic Publishers), 213–238. doi: 10.1007/978-1-4613-1639-8_9

Marr, D., and Poggio, T. (1976). Cooperative computation of stereo disparity. *Science* 194, 283–287. doi: 10.1126/science.968482

Maybank, S. J., and Faugeras, O. (1992). A theory of self-calibration of a moving camera. *Int. J. Comp. Vis.* 8, 123–152. doi: 10.1007/BF00127171

Meister, M., and Berry II, M. J. (1999). The neural code of the retina. *Neuron* 22, 435–450. doi: 10.1016/S0896-6273(00)80700-X

Lindenbaum, M., Fischer, M., and Bruckstein, A. M. (1994). On Gabor's contribution to image enhancement. *Pattern Recognit.* 27, 1–8. doi: 10.1016/0031-3203(94)90013-2

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid State Circuits* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Rogister, P., Benosman, R., Ieng, S., Lichsteiner, P., and Delbruck, T. (2012). Asynchronous event-based binocular stereo matching. *IEEE Trans. Neural Netw.* 23, 347–353. doi: 10.1109/TNNLS.2011.2180025

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., et al. (2009). CAVIAR: a 45k-Neuron, 5M-Synapse, 12G-connects/sec AER hardware sensory-processing-learning-actuating system for high speed visual object recognition and tracking. *IEEE Trans. Neural Netw.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Serrano-Gotarredona, R., Serrano-Gotarredona, T., Acosta-Jimenez, A., and Linares-Barranco, B. (2006). A neuromorphic cortical-layer microchip for spike-based event processing vision systems. *IEEE Trans. Circuits and Systems I* 53, 2548–2556. doi: 10.1109/TCSI.2006.883843

Serrano-Gotarredona, R., Serrano-Gotarredona, T., Acosta-Jimenez, A., Serrano-Gotarredona, C., Perez-Carrasco, J. A., Linares-Barranco, A., et al. (2008). On real-time AER 2D convolutions hardware for neuromorphic spike based cortical processing. *IEEE Trans. Neural Netw.* 19, 1196–1219. doi: 10.1109/TNN.2008.2000163

Serrano-Gotarredona, T., Andreou, A. G., and Linares-Barranco, B. (1999). AER image filtering architecture for vision processing systems. *IEEE Trans. Circuits Syst. I* 46, 1064–1071. doi: 10.1109/81.788808

Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). A 128x128 1.5% contrast sensitivity 0.9% FPN 3μs latency 4mW asynchronous frame-free dynamic vision sensor using transimpedance amplifiers. *IEEE J. Solid State Circuits* 48, 827–838. doi: 10.1109/JSSC.2012.2230553

Serrano-Gotarredona, T., Park, J., Linares-Barranco, A., Jiménez, A., Benosman, R., and Linares-Barranco, B. (2013). Improved contrast sensitivity DVS and its application to event-driven stereo vision. *IEEE Int. Symp. Circuits Syst.* 2013, 2420–2423. doi: 10.1109/ISCAS.2013.6572367

Silver, R., Boahen, K., Grillner, S., Kopell, N., and Olsen, K. L. (2007). Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools. *J. Neurosci.* 27, 11807–11819. doi: 10.1523/JNEUROSCI.3575-07.2007

Sivilotti, M. (1991). *Wiring Considerations in Analog VLSI Systems With Application to Field-Programmable Networks*. Ph.D. dissertation, California Institute of Technology, Pasadena, CA.

Tsang, E. K. C., and Shi, B. E. (2004). "A neuromorphic multi-chip model of a disparity selective complex cell," in *Advances in Neural Information Processing Systems*, Vol. 16, eds S. Thrun, L. K. Saul, and B. Schölkopf (Vancouver, BC: MIT Press), 1051–1058.

Venier, P., Mortara, A., Arreguit, X., and Vittoz, E. A. (1997). An integrated cortical layer for orientation enhancement. *IEEE J. Solid State Circuits* 32, 177–186. doi: 10.1109/4.551909

Zamarreño-Ramos, C., Linares-Barranco, A., Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). Multi-casting mesh AER: a scalable assembly approach for reconfigurable neuromorphic structured AER systems. Application to ConvNets. *IEEE Trans. Biomed. Circuits Syst.* 7, 82–102. doi: 10.1109/TBCAS.2012.2195725

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* 22, 1330–1334. doi: 10.1109/34.888718

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 25 September 2013; accepted: 23 February 2014; published online: 31 March 2014.*

*Citation: Camuñas-Mesa LA, Serrano-Gotarredona T, Ieng SH, Benosman RB and Linares-Barranco B (2014) On the use of orientation filters for 3D reconstruction in event-driven stereo vision. Front. Neurosci.* ***8****:48. doi: 10.3389/fnins.2014.00048*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 07 February 2014
doi: 10.3389/fnins.2014.00009

# Asynchronous visual event-based time-to-contact

**Xavier Clady[1]*, Charles Clercq[1,2], Sio-Hoi Ieng[1], Fouzhan Houseini[2], Marco Randazzo[2], Lorenzo Natale[2], Chiara Bartolozzi[2] and Ryad Benosman[1,2]**

[1] Vision Institute, Université Pierre et Marie Curie, UMR S968 Inserm, UPMC, CNRS UMR 7210, CHNO des Quinze-Vingts, Paris, France
[2] iCub Facility, Istituto Italiano di Tecnologia, Genova, Italia

**Edited by:**
Jennifer Hasler, Georgia Insitute of Technology, USA

**Reviewed by:**
Ueli Rutishauser, California Institute of Technology, USA
Leslie S. Smith, University of Stirling, UK
Scott M. Koziol, Baylor University, USA

***Correspondence:**
Xavier Clady, Vision Institute, Université Pierre et Marie Curie, UMR S968 Inserm, UPMC, CNRS UMR 7210, CHNO des Quinze-Vingts, 17 rue Moreau, 75012 Paris, France
e-mail: xavier.clady@upmc.fr

Reliable and fast sensing of the environment is a fundamental requirement for autonomous mobile robotic platforms. Unfortunately, the frame-based acquisition paradigm at the basis of main stream artificial perceptive systems is limited by low temporal dynamics and redundant data flow, leading to high computational costs. Hence, conventional sensing and relative computation are obviously incompatible with the design of high speed sensor-based reactive control for mobile applications, that pose strict limits on energy consumption and computational load. This paper introduces a fast obstacle avoidance method based on the output of an asynchronous event-based time encoded imaging sensor. The proposed method relies on an event-based Time To Contact (TTC) computation based on visual event-based motion flows. The approach is event-based in the sense that every incoming event adds to the computation process thus allowing fast avoidance responses. The method is validated indoor on a mobile robot, comparing the event-based TTC with a laser range finder TTC, showing that event-based sensing offers new perspectives for mobile robotics sensing.

**Keywords: neuromorphic vision, event-based computation, time to contact, robotics, computer vision**

## 1. INTRODUCTION

A fundamental navigation task for autonomous mobile robots is to detect and avoid obstacles in their path. This paper introduces a full methodology for the event-based computation of Time To Contact (TTC) for obstacle avoidance, using an asynchronous event-based sensor.

Sensors such as ultrasonic sensors, laser range finders or infrared sensors are often mounted on-board of robotic platforms in order to provide distance to obstacles. Such active devices are used to measure signals transmitted by the sensor and reflected by the obstacle(s). Their performance is essentially dependent on how the transmitted energy (ultrasonic waves, light,...) interacts with the environment Everett (1995); Ge (2010).

These sensors have limitations. In the case of ultrasonic sensors, corners and oblique surfaces, or even temperature variations can provide artifacts in the measurements. Infrared-based sensors (including recently emerged Time-Of-Light or RGB-D cameras) are sensitive to sunlight and can fail if the obstacle absorbs the signal. Laser range finder readings may also be erroneous because of specular reflections; additionally, the potential problems of eye-safety limit the use of many laser sensors to environments where humans are not present. In addition, most of the sensors have restrictions in terms of field-of-view and/or spatial resolution, requiring a mechanical scanning system or a network of several sensors. This leads to severe restrictions in terms of temporal responsiveness and computational load.

Vision can potentially overcome many of these restrictions; visual sensors often provide better resolution, wider range at faster rates than active scanning sensors. Their capacity to detect the natural light reflected by the objects or the surrounding areas paves the way to biological-inspired approaches.

Several navigation strategies using vision have been proposed, the most common consist of extracting depth information from visual information. Stereo-vision techniques can also produce accurate depth maps if the stability of the calibration parameters and a relative sufficient inter-camera distance can be ensured. However, these are strong requirements for high-speed and small robots. Another class of algorithms (Lorigo et al., 1997; Ulrich and Nourbakhsh, 2000), is based on color or texture segmentation of the ground plane. Even if this approach works on a single image, it requires the assumption that the robot is operating on a flat and uni-colored/textured surface and all objects have their bases on the ground.

Another extensively studied strategy is based on the evaluation of the TTC, noted $\tau$. This measure, introduced by Lee (1976), corresponds to the time that would elapse before the robot reaches an obstacle if the current relative motion between the robot and the obstacle itself were to continue without change. As the robot can navigate through the environment following a trajectory decomposed into straight lines (which is a classic and efficient strategy for autonomous robots in most environments), a general definition of TTC can be expressed as follows:

$$\tau = -\frac{Z}{\frac{dZ}{dt}} \tag{1}$$

where $Z$ is the distance between the camera and the obstacle, and $\frac{dZ}{dt}$ corresponds to the relative speed.

The Time-to-contact can be computed considering only visual information, without extracting relative depth information and speed, as demonstrated by Camus (1995) (see Section 3.2). Its computation has the advantage to work with a single camera,

without camera calibration or binding assumptions about the environment. Several techniques for the measure of TTC have been proposed. In Negre et al. (2006); Alyena et al. (2009), it is approximated measuring the local scale change of the obstacle, under the assumption that the obstacle is planar and parallel to the image plane. This approach requires either to precisely segment the obstacle in the image or to compute complex features in multi-scales representation of the image. Most studied methods of TTC rely on the estimation of optical flow. Optical flow conveys all necessary information from the environment Gibson (1978), but its estimation on natural scenes is well-known to be a difficult problem. Existing techniques are computationally expensive and are mostly used off line (Negahdaripour and Ganesan, 1992; Horn et al., 2007). Real-time implementations, using gradient-based, feature matching-based (Tomasi and Shi, 1994) or differential ones, do not deal with large displacements. Multi-scale process, as proposed by Weber and Malik (1995), can manage with this limitation, at the cost of computing time and hardware memory to store and process frames at different scales and timings.

Rind and Simmons (1999) proposed a bio-inspired neural network modeling the lobula giant movement detector (LGMD), a visual part of the optic lobe of the locust that responds most strongly to approaching objects. In order to process the frames provided by a conventional camera, existing implementations proposed by Blanchard et al. (2000) and Yue and Rind (2006) required a distributed computing environment (three PCs connected via ethernet). Another promising approach consists in VLSI architecture implementing functional models of similar neural networks, but it will require huge investments to go beyond the single proof of concept, such as the 1-D architecture of 25 pixels proposed by Indiveri (1998) modeling locust descending contralateral movement detector (DCMD) neurons. The hardware systems constructed in Manchester and Heidelberg, and described respectively by Bruderle et al. (2011) and Furber et al. (2012), could be an answer to this issue.

Globally, most of these approaches suffer from the limitations imposed by frame-based acquisition of visual information in the conventional cameras, that output large and redundant data flow, at a relative low temporal frequency. Most of the calculations are operated on uninformative parts of the images, or are dedicated to compensate for the lack of temporal precision. Existing implementations are often a trade off between accuracy and efficiency and are restricted to mobile robots moving relatively slowly. For example, Low and Wyeth (2005) and Guzel and Bicker (2010) present experiments on the navigation of a wheeled mobile robotic platform using optical flow based TTC computation applied with an embedded conventional camera. Their softwares run at approximatively 5 Hz and the maximal speed of the mobile robot is limited to 0.2 m/s.

In this perspective, free-frame acquisition of the neuromorphic cameras (Guo et al., 2007; Lichtsteiner et al., 2008; Lenero-Bardallo et al., 2011; Posch et al., 2011), can introduce significant improvements in robotic applications. The operation of such sensors is based on independent pixels that asynchronously collect and send their own data, when the processed signal exceeds a tunable threshold. The resulting compressed stream of events includes the spatial location of active pixels and an accurate time stamping at which a given signal change occurs. Events can be processed locally while encoding the additional temporal dynamics of the scene.

This article presents an event-based methodology to measure the TTC from the events stream provided by a neuromorphic vision sensor mounted on a wheeled robotic platform. The TTC is computed and then updated for each incoming event, minimizing the computational load of the robot. The performance of the developed event-based TTC is compared with a laser range finder, showing that event-driven sensing and computation, with their sub-microsecond temporal resolution and the inherent redundancy suppression, are a promising solution to vision-based technology for high-speed robots.

In the following we briefly introduce the used neuromorphic vision sensor (Section 2), describe the event-based approach proposed to compute the TTC (Section 3) and present experimental results validating the accuracy and the robustness of the proposed technique on a mobile robots moving in an indoor environment (Section 4).

## 2. TIME ENCODED IMAGING

Biomimetic, event-based cameras are a novel type of vision devices that—like their biological counterparts—are driven by "events" happening within the scene, and not by artificially created timing and control signals (i.e., frame clock of conventional image sensors) that have no relation whatsoever with the source of the visual information. Over the past few years, a variety of these event-based devices, reviewed in Delbruck et al. (2010), have been implemented, including temporal contrast vision sensors that are sensitive to relative light intensity change, gradient-based sensors sensitive to static edges, edge-orientation sensitive devices and optical-flow sensors. Most of these vision sensors encode visual information about the scene in the form of asynchronous address events (AER) Boahen (2000)using time rather than voltage, charge or current.

The ATIS ("Asynchronous Time-based Image Sensor") used in this work is a time-domain encoding image sensor with QVGA resolution Posch et al. (2011). It contains an array of fully autonomous pixels that combine an illuminance change detector circuit and a conditional exposure measurement block.

As shown in the functional diagram of the ATIS pixel in **Figure 1**, the change detector individually and asynchronously initiates the measurement of an exposure/gray scale value only if a brightness change of a certain magnitude has been detected in the field-of-view of the respective pixel. The exposure measurement circuit encodes the absolute instantaneous pixel illuminance into the timing of asynchronous event pulses, more precisely into inter-event intervals.

Since the ATIS is not clocked, the timing of events can be conveyed with a very accurate temporal resolution in the order of microseconds. The time-domain encoding of the intensity information automatically optimizes the exposure time separately for each pixel instead of imposing a fixed integration time for the entire array, resulting in an exceptionally high dynamic range and an improved signal to noise ratio. The pixel-individual change detector driven operation yields almost ideal temporal

redundancy suppression, resulting in a sparse encoding of the image data.

**Figure 2** shows the general principle of asynchronous imaging in a spatio-temporal representation. Frames are absent from this acquisition process. They can however be reconstructed, when needed, at frequencies limited only by the temporal resolution of the pixel circuits (up to hundreds of kiloframes per second) (**Figure 2** top). Static objects and background information, if required, can be recorded as a snapshot at the start of an acquisition. And henceforward moving objects in the visual scene describe a spatio-temporal surface at very high temporal resolution (**Figure 2** bottom).

## 3. EVENT-BASED TTC COMPUTATION

### 3.1. EVENT-BASED VISUAL MOTION FLOW

The stream of events from the silicon retina can be mathematically defined as follows: let $e(\mathbf{p}, t) = (\mathbf{p}, t)^T$ a triplet giving the position $\mathbf{p} = (x, y)^T$ and the time $t$ of an event. We can then define locally the function $\Sigma_e$ that maps to each $\mathbf{p}$, the time $t$:

$$\begin{aligned} \Sigma_e &: \mathbb{N}^2 \rightarrow \mathbb{R} \\ \mathrm{p} &\mapsto \Sigma_e(\mathrm{p}) = t \end{aligned} \tag{2}$$

Time being an increasing function, $\Sigma_e$ is a monotonically increasing surface in the direction of the motion.

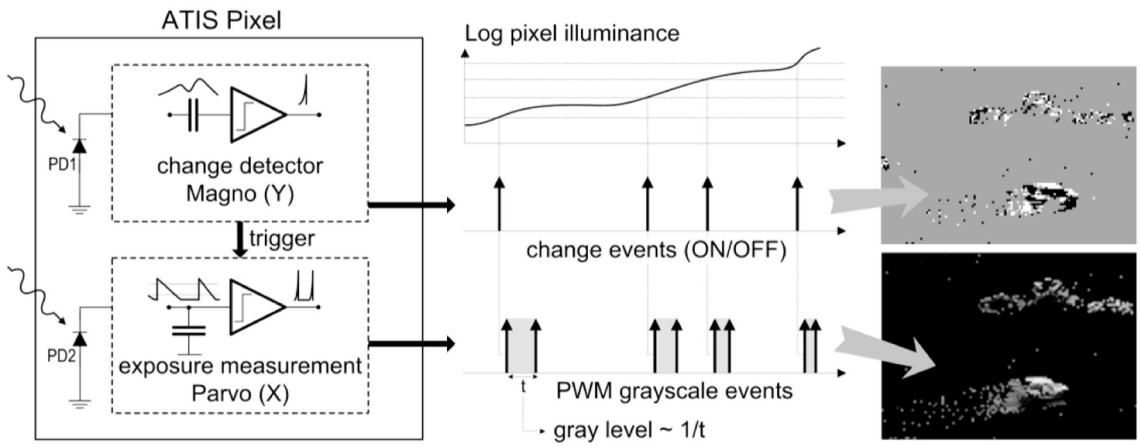



**FIGURE 1 | Functional diagram of an ATIS pixel Posch (2010).** Two types of asynchronous events, encoding change and brightness information, are generated and transmitted individually by each pixel in the imaging array.

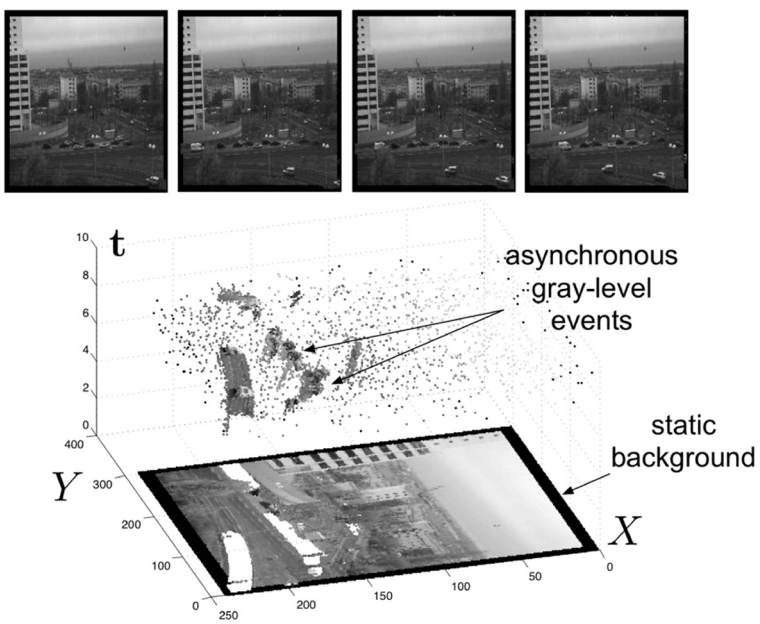



**FIGURE 2 | Lower part The spatio-temporal space of imaging events: static objects and scene background are acquired first.** Then, dynamic objects trigger pixel-individual, asynchronous gray-level events after each change. Frames are absent from this acquisition process. Samples of generated images from the presented spatio-temporal space are shown in the upper part of the figure.

We then set the first partial derivatives with respect to the parameters as: $\Sigma_{e_x} = \frac{\partial \Sigma_e}{\partial x}$ and $\Sigma_{e_y} = \frac{\partial \Sigma_e}{\partial y}$ (see **Figure 3**). We can then write $\Sigma_e$ as:

$$\Sigma_e(\mathbf{p} + \Delta\mathbf{p}) = \Sigma_e(\mathbf{p}) + \nabla\Sigma_e^T \Delta\mathbf{p} + o(||\Delta\mathbf{p}||) \tag{3}$$

with $\nabla\Sigma_e = \left(\frac{\partial \Sigma_e}{\partial x}, \frac{\partial \Sigma_e}{\partial y}\right)^T$.

The partial functions of $\Sigma_e$ are functions of a single variable, whether $x$ or $y$. Time being a strictly increasing function, $\Sigma_e$ is a nonzero derivatives surface at any point. It is then possible to use the inverse function theorem to write around a location $\mathbf{p} = (x, y)^T$ :

$$\left(\frac{\partial \Sigma_e}{\partial x}(x, y_0), \frac{\partial \Sigma_e}{\partial y}(x_0, y)\right)^T = \left(\frac{\mathrm{d}\Sigma_e|_{y=y_0}}{\mathrm{d}x}(x), \frac{\mathrm{d}\Sigma_e|_{x=x_0}}{\mathrm{d}y}(y)\right)^T = \left(\frac{1}{v_{nx}(x, y_0)}, \frac{1}{v_{ny}(x_0, y)}\right)^T \tag{4}$$

$\Sigma_e|_{x=x_0}$, $\Sigma_e|_{y=y_0}$ being $\Sigma_e$ restricted respectively to $x = x_0$ and $y = y_0$, and $\mathbf{v}_n(x, y) = (v_{nx}, v_{ny})^T$ represents the normal component of the visual motion flow; it is perpendicular to the object boundary (describing the local surface $\Sigma_e$).

The gradient of $\Sigma_e$ or $\nabla\Sigma_e$, is then:

$$\nabla\Sigma_e(\mathrm{p}, t) = \left(\frac{1}{v_{nx}(x, y_0)}, \frac{1}{v_{ny}(x_0, y)}\right)^T \tag{5}$$

The vector $\nabla\Sigma_e$ measures the rate and the direction of change of time with respect to the space, its components are also the inverse of the components of the velocity vector estimated at $\mathbf{p}$.

The flow definition given by Equation 5 is sensitive to noise since it consists in estimating the partial derivatives of $\Sigma_e$ at each individual event. One way to make the flow estimation robust against noise is to add a regularization process to the estimation. To achieve this, we assume a local velocity constancy. This hypothesis is satisfied in practice for small clusters of events. It is then equivalent to assume $\Sigma_e$ being locally planar since its partial spatial derivatives are the inverse of the speed, hence constant velocities produce constant spatial rate of change in $\Sigma_e$. Finally, the slope of the fitted plane with respect to the time axis is directly proportional to the motion velocity. The regularization also compensates for absent events in the neighborhood of active events where motion is being computed. The plane fitting provides an approximation of the timing of still non active spatial locations due the non idealities and the asynchronous nature of the sensor. The reader interested in the computation of motion flow can refer to Benosman et al. (2014) for more details. A full characterization of its computational cost is proposed; it shows that the event-based calculation required much less computation time than the frame-based one.

### 3.2. TIME-TO-CONTACT

Assuming parts of the environment are static, while the camera is moving forward, the motion flow diverges around a point called the focus of expansion (FOE). The visual motion flow field and the corresponding focus of expansion can be used to determine the time-to-contact (TTC) or time-to-collision. If the camera is embedded on an autonomous robot moving with a constant velocity, the TTC can be determined without any knowledge of the distance to be traveled or the velocity the robot is moving.

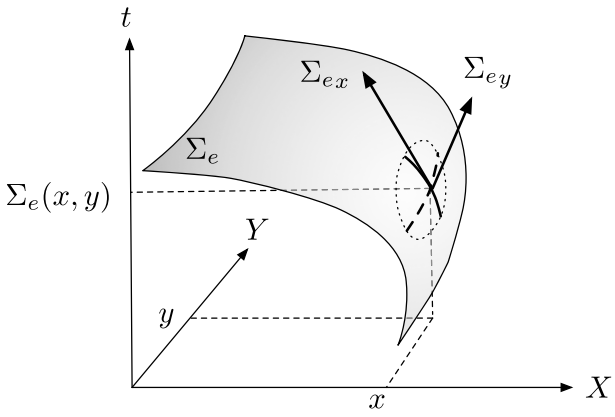



**FIGURE 3 | General principle of visual flow computation, the surface of active events $\Sigma_e$ is derived to provide an estimation of orientation and amplitude of motion.**

We assume the obstacle is at $\mathbf{P} = (X_c, Y_c, Z_c)^T$ in the camera coordinate frame and $\mathbf{p} = (x, y)^T$ is its projection into the camera's focal plane coordinate frame (see **Figure 4**). The velocity vector $\mathbf{V}$ is also projected into the focal plane as $\mathbf{v} = (\dot{x}, \dot{y})^T$.

By deriving the pinhole model's equations, Camus (1995) demonstrates that, if the coordinates $\mathbf{p}_f = (x_f, y_f)^T$ of the FOE are known, the following relation is satisfied:

$$\tau = -\frac{Z_c}{\dot{Z}_c} = \frac{y - y_f}{\dot{y}} = \frac{x - x_f}{\dot{x}}, \text{ where}$$
$$\dot{Z}_c = \frac{dZ_c}{dt}, \quad \dot{x} = \frac{dx}{dt}, \quad \dot{y} = \frac{dy}{dt}. \tag{6}$$

With our notation, this is equivalent to:

$$\tau(\mathbf{p}, t)\mathrm{v}(\mathbf{p}, t) = \mathbf{p} - \mathbf{p}_f \tag{7}$$

The TTC is then obtained at pixel $\mathbf{p}$ according to the relation:

$$\tau(\mathbf{p}, t) = \frac{\mathrm{v}^T(\mathbf{p}, t)(\mathbf{p} - \mathbf{p}_f)}{||\mathrm{v}(\mathbf{p}, t)||^2} \tag{8}$$

The TTC as defined is a signed real value because of the scalar product. Its sign refers to the direction of the motion: when $\tau$ is positive, the robot is going toward the obstacle and, vice-versa, for negative $\tau$ it is getting away. This equality shows also that $\tau$ can be determined only if the velocity $\mathbf{v}$ at $\mathbf{p}$ is known or can be estimated for any $\mathbf{p}$ at anytime $t$. There is unfortunately no general technique for estimating densely the velocity $\mathbf{v}$ from the visual information. However, optical flow techniques allow to compute densely the vector field of velocities normal to the

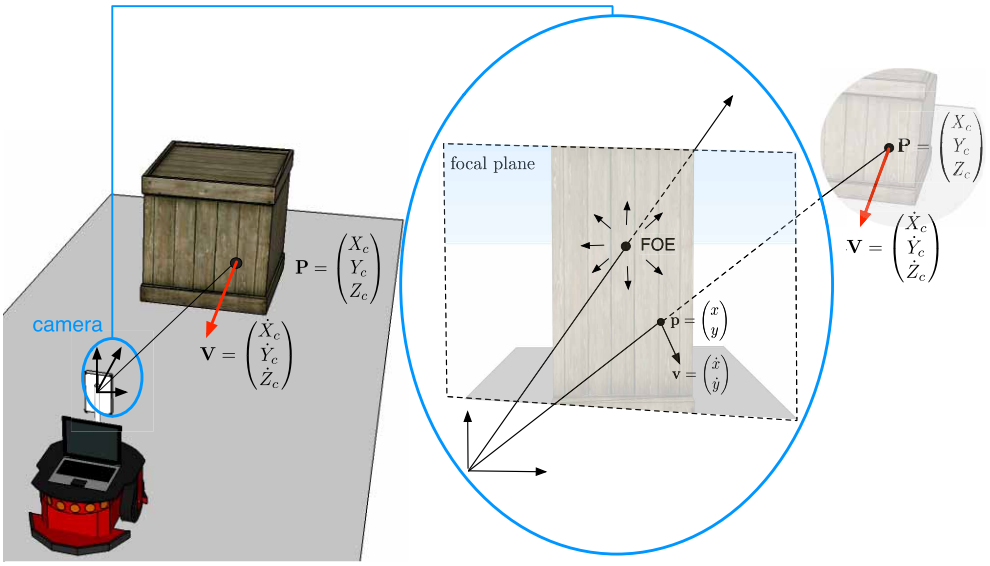



**FIGURE 4 | 3D obstacle velocity *V* projected into the camera focal plane as *v*.** The dotted letters refer to temporal derivatives of each component.

edges, noted as $\mathbf{v}_n$. The visual flow technique presented in subsection 3.2 is the ideal technique to compute $\tau$, not only because of its event-based formulation, but it is also showing that the normal to the edge component of $\mathbf{v}$ is sufficient for $\tau$ determination. From Equation 7, we apply the scalar product of both end sides with $\nabla\Sigma_e$:

$$\tau(\mathbf{p}, t)\mathrm{v}(\mathbf{p}, t)^T\nabla\Sigma_e(\mathbf{p}, t) = (\mathbf{p} - \mathbf{p}_f)^T\nabla\Sigma_e(\mathbf{p}, t) \quad (9)$$

Because $\mathbf{v}$ can be decomposed as the sum of a tangential vector $\mathbf{v}_t$, and a normal vector $\mathbf{v}_n$, the left end side of Equation 9 simplifies into:

$$\begin{aligned}\tau\mathrm{v}^T(\mathbf{p}, t)\nabla\Sigma_e(\mathbf{p}, t) &= \tau\left(\mathrm{v}_t(\mathbf{p}, t) + \mathrm{v}_n(\mathbf{p}, t)\right)^T\nabla\Sigma_e(\mathbf{p}, t) \\ &= \tau\mathrm{v}_n^T(\mathbf{p}, t)\nabla\Sigma_e(\mathbf{p}, t) = 2\tau \quad (10)\end{aligned}$$

$\mathbf{v}_t^T\nabla\Sigma_e = 0$ since the tangential component is orthogonal to $\nabla\Sigma_e$. Therefore $\tau$ is given by:

$$\tau(\mathbf{p}, t) = \frac{1}{2}(\mathbf{p} - \mathbf{p}_f)^T\nabla\Sigma_e(\mathbf{p}, t) \quad (11)$$

### 3.3. FOCUS OF EXPANSION

The FOE is the projection of the observer's direction of translation (or heading) on the sensor's image plane. The radial pattern of flows depends only on the observer's heading and is independent of 3D structure, while the magnitude of flow depends on both heading and depth. Thus, in principle, the FOE could be obtained by triangulation of two vectors in a radial flow pattern. However, such a method would be vulnerable to noise. To calculate the FOE, we used the redundancy in the flow pattern to reduce errors.

The principle of the approach is described in Algorithm 1. We consider a probability map of the visual field, where each point represents the likelihood of the FOE to be located on the corresponding point in the field. Every flow provides an estimation of the location of the FOE in the visual field; indeed, because the visual flow is diverging from the FOE, it belongs to the negative semi-plane defined by the normal motion flow vector. So, for each incoming event, all the corresponding potential locations of the FOE are also computed (step 3 in Algorithm 1) and their likelihood is increased (step 4). Finding the location of the probability map with maximum value, the FOE is shifted toward this location (step 5)). This principe is illustrated in **Figure 5A**. The area with the maximum of probability is highlighted as the intersection of the negative semi-planes defined by the normal motion flow vectors. Finally, an exponential decreasing function is applied on the probability map; it allows updating the location of the FOE, giving more importance to the contributions provided by the most recent events and their associated flow.

**Figures 5B,C** show real results obtained viewing a densely textured pattern (the same as used in Experiment 1, see **Figure 7**). **Figure 5B** shows the probability map defined as an accumulative table and the resulting FOE. The corresponding motion flow is given in **Figure 5C**; the normal motion vectors (with an amplitude superior than a threshold) computed in a time interval $\Delta t = 10$ ms are represented as yellow arrows. Globally, the estimated FOE is consistent with the motion flow. However, some small groups of vectors (an example is surrounded by a white dotted ellipse) that seems converging, instead of diverging, to the FOE. Such flow events do not occur at the same time as the others; they are most probably generated by a temporary micro-motion (vibration, unexpected roll-, pitch- or yaw-motion). The cumulative process allows to filter such noise motions and to keep a FOE stable.

For an incoming event $e(\mathbf{p}, t)$ with a velocity vector $\mathbf{v}_n$, we can define the following algorithm to estimate the FOE:

**Algorithm 1 | Computation of the Focus Of Expansion.**

**Require** $M_{prob} \in \mathbb{R}^m \times \mathbb{R}^n$ and $M_{time} \in \mathbb{R}^m \times \mathbb{R}^n$ ($M_{prob}$ is the probability map and holds the likelihood for each spatial location and $M_{time}$ the last time when its likelihood has been increased).

1: Initiate the matrices $M_{prob}$ and $M_{time}$ to 0

2: **for** every incoming $e(\mathbf{p}, t)$ at velocity $\mathbf{v_n}$ **do**

3: Determine all spatial locations $\mathbf{p}_i$ such as $(\mathbf{p} - \mathbf{p_i})^T.\mathbf{v_n} > 0$

4: for all $\mathbf{p}_i$ : $M_{prob}(\mathbf{p}_i) = M_{prob}(\mathbf{p}_i) + 1$ and $M_{time}(\mathbf{p}_i) = t_i$

5: $\forall\, \mathbf{p}_i \in \mathbb{R}^m \times \mathbb{R}^n$, update the probability map $M_{prob}(\mathbf{p}_i) = M_{prob}(\mathbf{p}_i)\, e^{-\frac{t_i - M_{time}(\mathbf{p}_i)}{\Delta t}}$

6: Find $\mathbf{p}_f = (x_f, y_f)^T$ the spatial location of the maximum value of $M_{prob}$ corresponding to the FOE location

7: **end for**

## 4. EXPERIMENTAL RESULTS

The method proposed in the previous sections is validated in the experimental setup illustrated in **Figure 6**. The neuromorphic camera is mounted on a Pioneer 2 robotic platform, equipped with a Hokuyo laser range finder (LRF) providing the actual distance between the platform and the obstacles. In experimental environment free of specular or transparent objects (as in the first proposed experiment), the TTC based on the LRF can be estimated using the Equation 1 and is used as ground truth measure against which the event-based TTC is benchmarked.

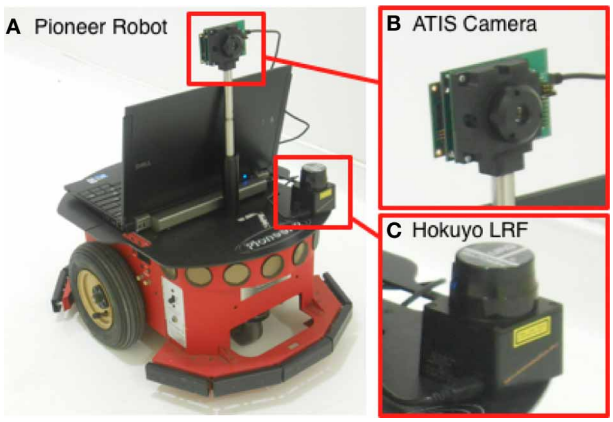



**FIGURE 6 | Experimental setup: (A) the Pioneer 2, (B) the asynchronous event-based ATIS camera, (C) the Hokuyo laser range finder (LRF).**

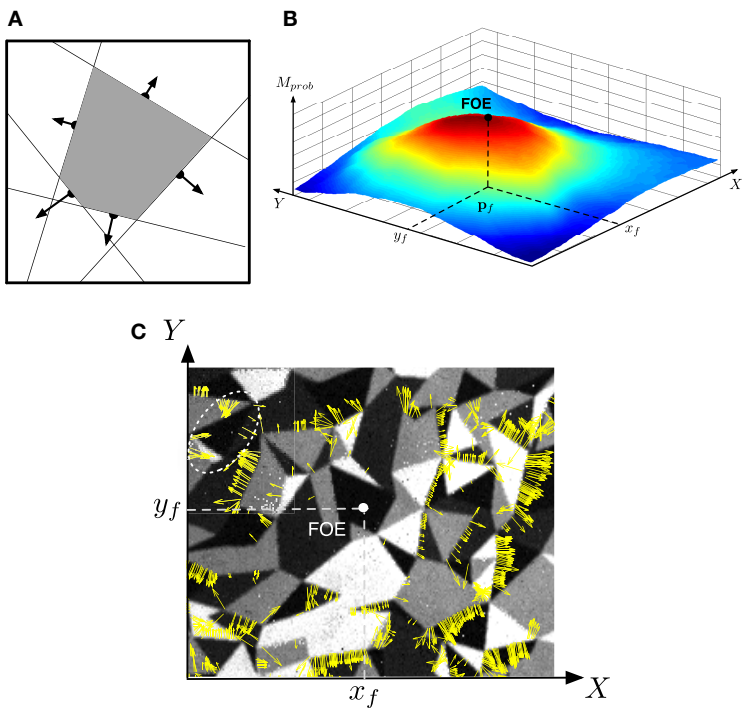



**FIGURE 5 | Computation of the focus of expansion: (A) the focus of expansion lies under the normal flow, we can then vote for an area of the focal plane shown in (B) the FOE is the max of this area (C) Motion flow vectors obtained during a time period of $\Delta t = 10$ ms and superimposed over a corresponding snapshot (realized using the PWM grayscale events; the viewed pattern is the same as used in Experiment 1, cf. Figure 7).** Note that only the vectors with high amplitude are represented in order to enhance the readability of the Figure. Most of the motion flow vectors are diverging from the estimated FOE. The white ellipse in the up left corner shows a group of inconsistent motion flow vectors: they are probably due to a temporary noise micro-motion (vibration, unexpected roll-, pitch-, or yaw-motion).

In the first experiment, the robot is moving forward and backward in the direction of a textured obstacle as shown in **Figure 7**, the corresponding TTC estimated by both sensors (LRF and ATIS) is shown in **Figure 8A.** The TTC is expressed in the coordinate system of the obstacle: the vertical axis corresponds to time and the horizontal axis to the size of the obstacle. The extremes (and the white parts of the plots) correspond to the changes of direction of the robot: when its speed tends to 0, the LRF based TTC tends to infinity and the vision based TTC cannot be computed because too few events are generated. In order to show comparable results, only the TTC obtained with a robot speed superior to 0.1 m/s are shown; under this value, the robot motion is relatively unstable, the robot tilting during the acceleration periods.

**Figure 8B** shows the relative error of the event-based TTC with respect of the ground truth calculated with the LRF TTC. The error is large during the phases of positive and negative accelerations of the robot. There are two potential explanations. The estimation of the speed of the robot based on the LRF is relatively inaccurate during the change of velocity. In addition, brutal changes of velocity could generate fast pitch motions which produce unstable FOE. Globally, more than the 60% of the relative errors are inferior to 20% showing that that the event-based approach is robust and accurate when the motion of the robot is stable.

In the second experiment, the robot moves along a corridor. In this conditions, multiple objects reflect the light from the LRF, that fails to detect obstacles, on the contrary the event-based algorithm succeeds in estimating the TTC relative to the obstacles. **Figure 8** shows the robot's trajectory: during the first stage the robot navigates toward an obstacle (portion A-B of the trajectory). An avoidance maneuver is performed during portion B-C that leads the robot to continue its trajectory to enter the warehouse (portion C-D). The estimated TTC to the closest obstacle, is shown as red plots in **Figure 9** and compared to the ground truth given by the odometer's data (in

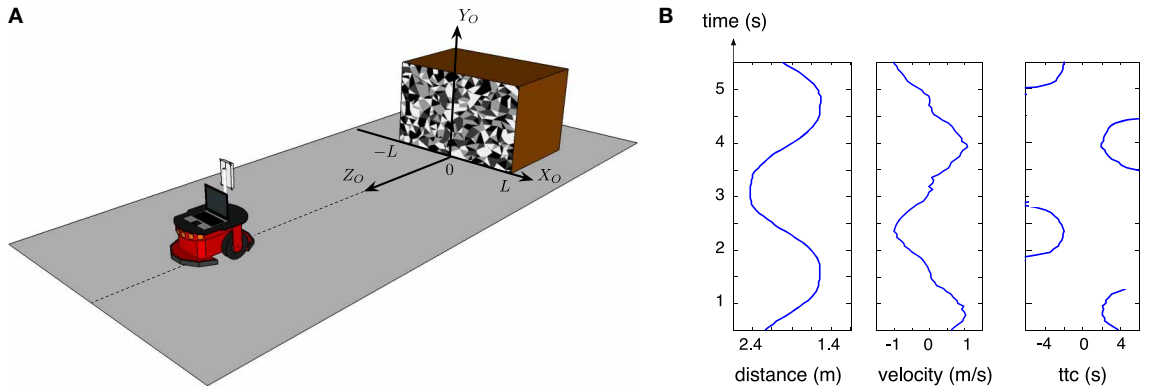



**FIGURE 7 | First experiment: (A) setup and location of the coordinate system ($X_O$, $Y_O$, $Z_O$) related to the obstacle; (B) distance between the robot and the obstacle, velocity of the robot and the relative estimated TTC over time are computed based on the odometer of the robot.** Only the TTC computed while the velocity of the robot is superior to 0.1 m/s is given, because it tends to infinity when velocity tends to 0.

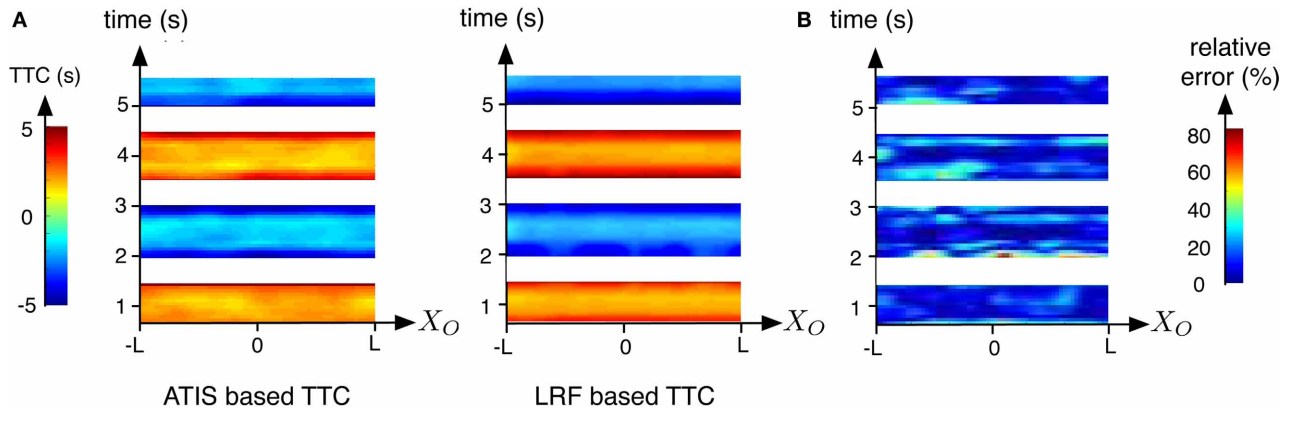



**FIGURE 8 | Comparison of the results obtained while the robot is moving forward and backward in the direction of an obstacle.** Results are expressed related to time and the coordinates system of the obstacle. **(A)** TTC computed using the LRF (right) and the ATIS (left). **(B)** Relative errors between bothTTC estimations. illustrated using a color map, blue to red for increasing TTC.

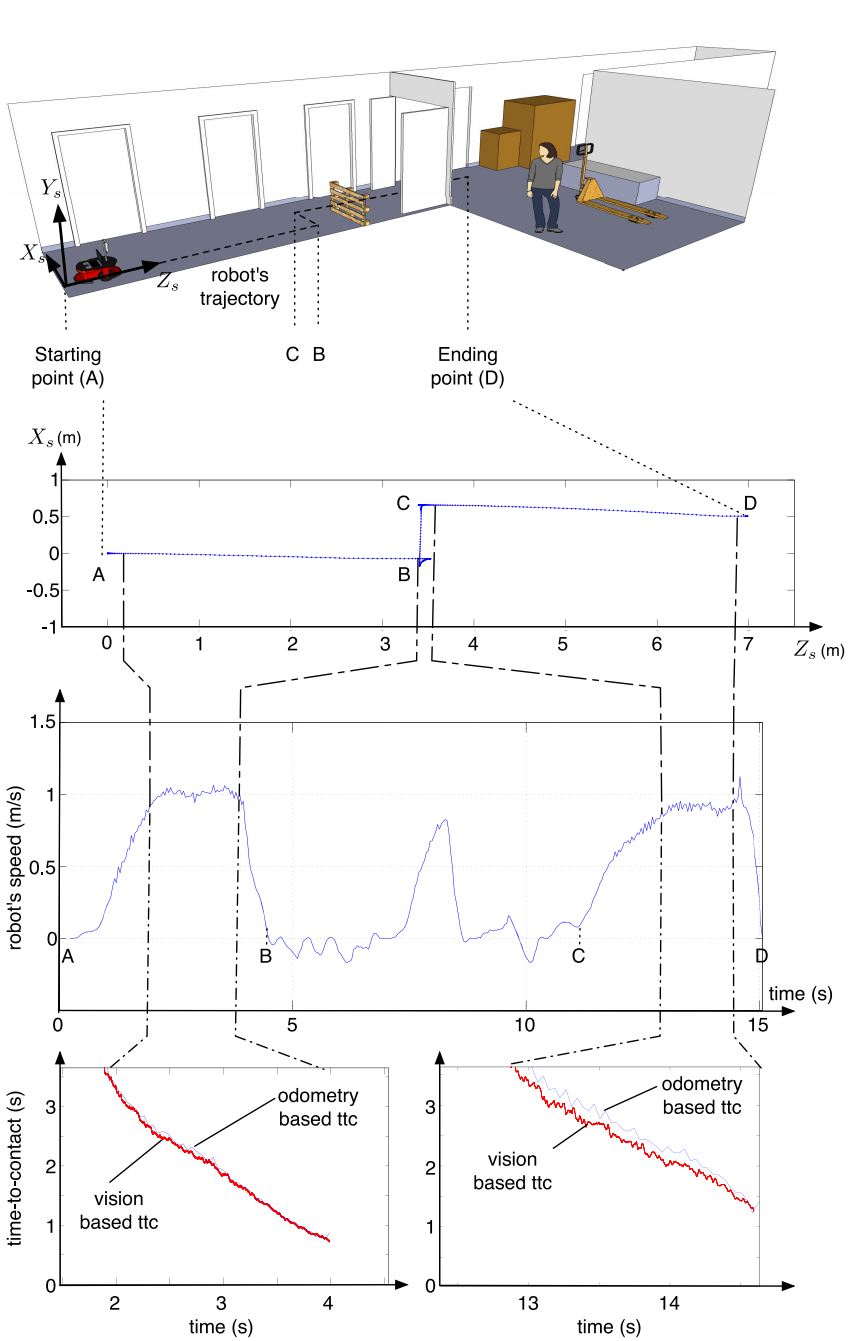



**FIGURE 9 | Results of second experiment: the top Figure represents the trajectory followed by the the robot, on a schematic view of the warehouse, both middle Figures represents data collected from the odometer (the trajectory and the speed of the robot) and finally, the bottom Figures represent the time-to-contact estimated during the two time intervals during which the TTC is estimated; the red curves correspond to the TTC estimated from the neuromorphic camera's data, compared to an estimation of the ttc (blue curves) using the odometer's data and the knowledge of the obstacles' locations in the map.**

blue). It corresponds to the TTC collected in a region of interest of $60 \times 60$ pixels, matching with the closest obstacle. The image plane is segmented into four regions of interest (ROI) of $60 \times 60$ pixels (4 squares represented in the **Figure 10**) around the x-coordinate of the FOE. Only the normal flow vectors into the lower ROI, in which the activity, expressed as the number of events per second, is superior to a threshold (>5000 events/s), are considered, assuming that the closest obstacle is on the ground and so viewed in the bottom part of the vision field.

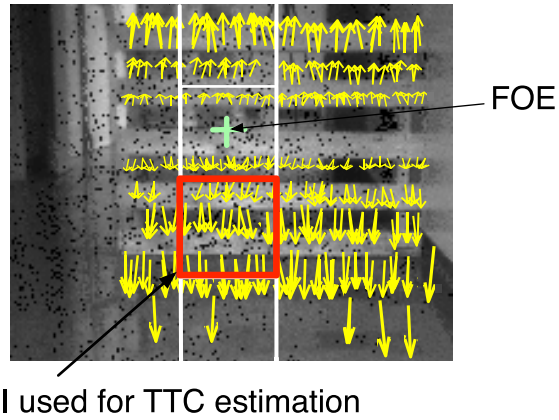



**FIGURE 10 | TTC computation: the yellow arrows represent the motion flow vectors obtained during a time period of 1 ms.** These flow vectors are superimposed over a corresponding image of the scene (realized using the PWM grayscale events). In order to enhance the readability of the Figure, only 10% vectors with high strengths and orientations close to $\pm\pi/2$ have been draw. The red square corresponds to the ROI where the measure of TTC is estimated.

The low shift between them can be explained by the drift in odometer's data (especially after the avoidance maneuver) Everett (1995); Ge (2010); a difference of 0.5 m. has been observed between the real position of the robot and the odometer-based estimate of the ending location. This is an expected effect, as odometer always drifts in the same measured proportions Ivanjko et al. (2007). In addition, the estimations are slightly less precise once the robot is in the warehouse, where the poor environment with white walls without texture or objects produces less events and the computation degrades. This shows the robustness of the technique even in poorly textured.

All programs have been written in C++ under linux and run in real time. We estimated the average time per event spent to compute the Time-to-Contact: it is approximately 20 µs per event on the computer used in the experiments (Intel Core i7 at 2.40 GHz). When the robot is at its maximum speed, data stream acquired during 1 s is processed in 0.33 s . The estimation of the visual flow is the most computationally expensive task (>99% of the total computational cost), but could be easily run in parallel to further accelerate it.

The most significant result of this work is that the TTC can be processed at an unprecedented rate and with a low computational cost. The output frequency of our method reaches over 16 kHz, which is largely superior to the ones which can be expected from any other conventional cameras, limited by their frame-based acquisitions and processing load needed to process data.

## 5. CONCLUSIONS AND PERSPECTIVES

The use of vision based navigation using conventional frame-based cameras is impractical for the limited available resources usually embedded on autonomous robots. The corresponding large amount of data to process is not compatible with fast and reactive navigation commands, especially when parts of the processing are allocated to extract the useful information. Such computational requirements are out of the reach of most small robots. Additionally, the temporal resolution of frame-based cameras trades off with the quantity of data that need to be processed, posing limits on the robot's speed and computational demand. In this paper, we gave an example of a simple collision avoidance technique based on the estimation of the TTC by combining the use of an event-based vision sensor and a recent previously developed event-based optical flow. We showed that event-based techniques can solve vision tasks in a more efficient way than traditional approaches that are used to do, by means of complex and hungry algorithms.

One remarkable highlight of this work is how well the event-based optical flow presented in Benosman et al. (2014) helped in estimating the TTC. This is because we have ensured the preservation of the high temporal dynamics of the signal from its acquisition to its processing. The precise timing conveyed by the neuromorphic camera allows to process locally around each event for a low computational cost, whilst ensuring a precise computation of the visual motion flow and thus, of the TTC. The experiments carried out on a wheeled robotic platform support this statement, as the results are as reliable as the ones obtained with a laser range finder, at a much higher frequency. With event-based vision, the motion behavior of a robot could be controlled with a time delay far below the one that is inherent to the frame-based acquisition in conventional cameras.

The method described in this work stands on the constant velocity hypothesis since Equation 1 is a result derived from that assumption. for this reason, the normal to the edges velocity is sufficient for the TTC estimation. For more general motion, the proposed method should be modified by for example assuming the velocity to be constant only locally.

This work supports the observation that event-driven (bio-inspired) asynchronous sensing and computing are opening promising perspectives for autonomous robotic applications. The event-based approaches would allow small robots to avoid obstacles in natural environment with high speed that has never been achieved until now. Extending our approach to more complex scenarios than those exposed in this paper, and proposing a complete navigation system able to deal with motion or uncontrolled environment, requires to combine the visual information with other provided from top-down process and proprioceptive sensing, as for humans or animals.

## ACKNOWLEDGMENTS

This work benefitted from the fruitful discussions and collaborations fostered by the CapoCaccia Cognitive Neuromorphic Engineering Workshop and the NSF Telluride Neuromorphic Cognition workshops.

## FUNDING

This work has been supported by the EU grant eMorph (ICT-FET-231467). The authors are also grateful to the Lifesense Labex.

## REFERENCES

Alyena, G., Negre, A., and Crowley, J. L. (2009). "Time to contact for obstacle avoidance," in *European Conference on Mobile Robotics* (Dubrovnik).

Benosman, R., Clercq, C., Lagorce, X., Ieng, S.-H., and Bartolozzi, C. (2014). Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.* 25, 407–417. doi: 10.1109/TNNLS.2013.2273537

Blanchard, M., Rind, F., and Verschure, P. F. (2000). Collision avoidance using a model of the locust LGMD neuron. *Robot. Auton. Syst.* 30, 17–38. doi: 10.1016/S0921-8890(99)00063-9

Boahen, K. A. (2000). Point-to-point connectivity between neuromorphic chips using address-events. *Circuits and Sys. II: Analog and Digit. Signal Process. IEEE Trans.* 47, 416–434. doi: 10.1109/82.842110

Bruderle, D., Petrovici, M. A., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., et al. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern.* 104, 263–296. doi: 10.1007/s00422-011-0435-9

Camus, T. (1995). Calculating time-to-contact using real-time quantized optical flow. *National Institute of Standards and Technology NISTIR 5609*.

Delbruck, T., Linares-Barranco, B., Culurciello, E., and Posch, C. (2010). "Activity-driven, event-based vision sensors," in *IEEE International Symposium on Circuits and Systems* (Paris), 2426–2429. doi: 10.1109/ISCAS.2010.5537149

Everett, H. (1995). *Sensors for Mobile Robots: Theory and Applications*. Natick, MA: A K Peters/CRC Press.

Furber, S., Lester, D., Plana, L., Garside, J., Painkras, E., Temple, S., et al. (2012). Overview of the spinnaker system architecture. *IEEE Trans. Comput.* 62, 2454–2467. doi: 10.1109/TC.2012.142

Ge, S. (2010). *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*. Automation and Control Engineering. Boca Raton, FL: Taylor and Francis.

Gibson, J. J. (1978). The ecological approach to the visual perception of pictures. *Leonardo* 11, 227–235. doi: 10.2307/1574154

Guo, X., Qi, X., and Harris, J. (2007). A time-to-first-spike cmos image sensor. *Sens. Jo. IEEE* 7, 1165–1175. doi: 10.1109/JSEN.2007.900937

Guzel, M., and Bicker, R. (2010). "Optical flow based system design for mobile robots," in *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on* (Singapour), 545–550. doi: 10.1109/RAMECH.2010.5513134

Horn, B., Fang, Y., and Masaki, I. (2007). "Time to contact relative to a planar surface," in *Intelligent Vehicles Symposium, 2007 IEEE* (Istanbul), 68–74. doi: 10.1109/IVS.2007.4290093

Indiveri, G. (1998) Analog vlsi model of locust dcmd neuron response for computation of object approach. *Prog. Neural Process.* 10, 47–60. doi: 10.1142/9789812816535_0005

Ivanjko, E., Komsic, I., and Petrovic, I. (2007). "Simple off-line odometry calibration of differential drive mobile robots," in *Proceedings of 16th Int. Workshop on Robotics in Alpe-Adria-Danube Region-RAAD.* (Ljubljana).

Lee, D. N. (1976). A theory of visual control of braking based on information about time-to-collision. *Perception* 5, 437–459. doi: 10.1068/p050437

Lenero-Bardallo, J., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). A 3.6 μs latency asynchronous frame-free event-driven dynamic-vision-sensor. *J. Solid-State Circ.* 46, 1443–1455. doi: 10.1109/JSSC.2011.2118490

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor. *J. Solid-State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Lorigo, L., Brooks, R., and W. Grimsou, W. (1997). "Visually-guided obstacle avoidance in unstructured environments," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems* Vol. 1 (Grenoble), 373–379. doi: 10.1109/IROS.1997.649086

Low, T., and Wyeth, G. (2005). "Obstacle detection using optical flow," in *Proceedings of Australasian Conference on Robotics and Automation* (Sydney, NSW). doi: 10.1109/IVS.1992.252254

Negahdaripour, S., and Ganesan, V. (1992). "Simple direct computation of the FOE with confidence measures," in *Computer Vision and Pattern Recognition* (Champaign, IL), 228–235. doi: 10.1109/CVPR.1992.22327

Negre, A., Braillon, C., Crowley, J. L., and Laugier, C. (2006). "Real-time time-to-collision from variation of intrinsic scale," in *International Symposium of Experimental Robotics* (Rio de Janeiro), 75–84. doi: 10.1007/978-3-540-77457-0_8

Posch, C. (2010). "High-dr frame-free pwm imaging with asynchronous aer intensity encoding and focal-plane temporal redundancy suppression," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (Paris). doi: 10.1109/ISCAS.2010.5537150

Posch, C., Matolin, D., and Wohlgenannt, R. (2011). A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *J. Solid-State Circ.* 46, 259–275. doi: 10.1109/JSSC.2010.2085952

Rind, F. C., and Simmons, P. J. (1999). Seeing what is coming: building collision-sensitive neurones. *Trends Neurosci.* 22. 215–220. doi: 10.1016/S0166-2236(98)01332-0

Tomasi, C., and Shi, J. (1994). "Good features to track," in *Proceedings CVPR '94., IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994*, (Seattle, WA), 593–600. doi: 10.1109/CVPR.1994.323794

Ulrich, I., and Nourbakhsh, I. R. (2000). "Appearance-based obstacle detection with monocular color vision," in *Proceedings of the International Conference on AAAI/IAAI* (Austin, TX), 866–871.

Weber, J., and Malik, J. (1995). Robust computation of optical flow in a multi-scale differential framework. *Int. J. Comput. Vis.* 14, 67–81. doi: 10.1007/BF01421489

Yue, S., and Rind, F. C. (2006). Collision detection in complex dynamic scenes using an lgmd-based visual neural network with feature enhancement. *Neural Netw. IEEE Trans.* 17, 705–716. doi: 10.1109/TNN.2006.873286

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 30 September 2013; accepted: 16 January 2014; published online: 07 February 2014.*

*Citation: Clady X, Clercq C, Ieng S-H, Houseini F, Randazzo M, Natale L, Bartolozzi C and Benosman R (2014) Asynchronous visual event-based time-to-contact. Front. Neurosci. **8**:9. doi: 10.3389/fnins.2014.00009*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 08 October 2013
doi: 10.3389/fnins.2013.00178

# Real-time classification and sensor fusion with a spiking deep belief network

***Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck and Michael Pfeiffer****

*Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland*

***Edited by:***
*André Van Schaik, The University of Western Sydney, Australia*

***Reviewed by:***
*Bernabe Linares-Barranco, Instituto de Microelectrónica de Sevilla, Spain*
*Eugenio Culurciello, Purdue University, USA*

***Correspondence:**
*Michael Pfeiffer, Institute of Neuroinformatics, University of Zurich and ETH Zurich, Winterthurerstrasse 190, CH-8057, Zurich, Switzerland*
*e-mail: pfeiffer@ini.phys.ethz.ch*

Deep Belief Networks (DBNs) have recently shown impressive performance on a broad range of classification problems. Their generative properties allow better understanding of the performance, and provide a simpler solution for sensor fusion tasks. However, because of their inherent need for feedback and parallel update of large numbers of units, DBNs are expensive to implement on serial computers. This paper proposes a method based on the Siegert approximation for Integrate-and-Fire neurons to map an offline-trained DBN onto an efficient event-driven spiking neural network suitable for hardware implementation. The method is demonstrated in simulation and by a real-time implementation of a 3-layer network with 2694 neurons used for visual classification of MNIST handwritten digits with input from a 128 × 128 Dynamic Vision Sensor (DVS) silicon retina, and sensory-fusion using additional input from a 64-channel AER-EAR silicon cochlea. The system is implemented through the open-source software in the jAER project and runs in real-time on a laptop computer. It is demonstrated that the system can recognize digits in the presence of distractions, noise, scaling, translation and rotation, and that the degradation of recognition performance by using an event-based approach is less than 1%. Recognition is achieved in an average of 5.8 ms after the onset of the presentation of a digit. By cue integration from both silicon retina and cochlea outputs we show that the system can be biased to select the correct digit from otherwise ambiguous input.

**Keywords: deep belief networks, spiking neural network, silicon retina, sensory fusion, silicon cochlea, deep learning, generative model**

## 1. INTRODUCTION

Deep Learning architectures, which subsume convolutional networks (LeCun et al., 1998), deep autoencoders (Hinton and Salakhutdinov, 2006), and in particular DBNs (Bengio et al., 2006; Hinton et al., 2006; Hinton and Salakhutdinov, 2006) have excelled among machine learning approaches in pushing the state-of-the-art in virtually all relevant benchmark tasks to new levels. In this article we focus on DBNs, which are constructed as hierarchies of recurrently connected simpler probabilistic graphical models, so called Restricted Boltzmann Machines (RBMs). Every RBM consists of two layers of neurons, a hidden and a visible layer, which are fully and symmetrically connected between layers, but not connected within layers (see **Figure 1**). Using unsupervised learning, each RBM is trained to encode in its weight matrix a probability distribution that predicts the activity of the visible layer from the activity of the hidden layer. By stacking such models, and letting each layer predict the activity of the layer below, higher RBMs learn increasingly abstract representations of sensory inputs, which matches well with representations learned by neurons in higher brain regions e.g., of the visual cortical hierarchy (Gross et al., 1972; Desimone et al., 1984). The success of Deep Learning rests on the unsupervised layer-by-layer pre-training with the Contrastive Divergence (CD) algorithm (Hinton et al., 2006; Hinton and Salakhutdinov, 2006), on which supervised learning and inference can be efficiently performed (Bengio et al., 2006; Erhan et al., 2010). This avoids typical problems of training large neural networks with error backpropagation, where overfitting and premature convergence pose problems (Hochreiter et al., 2001; Bengio et al., 2006). The data required for pre-training does not have to be labeled, and can thus make use of giant databases of images, text, sounds, videos, etc. that are now available as collections from the Internet. An additional attractive feature is that the performance of deep networks typically improves with network size, and there is new hope of achieving brain-like artificial intelligence simply by scaling up the computational resources.

With the steady increase in computing power, DBNs are becoming increasingly important for an increasing number of commercial *big data* applications. Using gigantic computational resources industry leaders like Google or Microsoft have started to invest heavily in this technology, which has thus been recently named one of the Breakthrough Technologies of 2013 (MIT Technology Review, 2013), and has led to what has been called the "second reNNaissance of neural networks" (Ciresan et al., 2010). This is the result of the success stories of Deep Learning approaches for computer vision (Larochelle et al., 2007; Lee et al., 2009; Ciresan et al., 2010; Le et al., 2012), voice recognition (Dahl et al., 2012; Hinton et al., 2012; Mohamed et al., 2012), or machine transcription and translation (Seide et al., 2011; MIT Technology Review, 2013). Despite this potential, the sheer number of neurons and connections in deep neural networks requires massive computing power, time, and energy, and thus makes their use in real-time applications e.g., on mobile devices or autonomous robots infeasible. Instead of speculating on Moore's

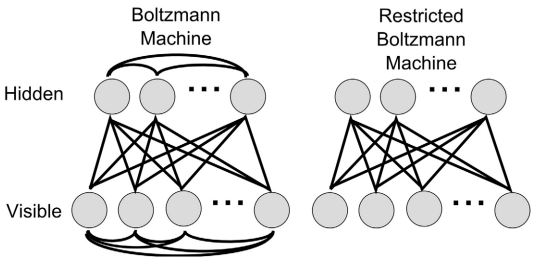



**FIGURE 1 | Boltzmann and Restricted Boltzmann Machines.** A Boltzmann machine is fully connected within and between layers, whereas in a RBM, the lateral connections in the visible and hidden layers are removed. As a result, the random variables encoded by hidden units are conditionally independent given the states of the visible units, and vice versa.

law to achieve progress through faster and cheaper computing resources in the future, we argue that fast and energy efficient inference in DBNs is already possible now, and is an ideal use case for neuromorphic circuits (Indiveri et al., 2011), which emulate neural circuits and event-based, asynchronous communication architectures in silicon. This is motivated by the fact that in the brain, having many neurons and connections is not a factor that constrains the processing time, since all units operate in parallel, and only the arrival of spike events triggers processing, so the neural circuits can adapt the processing speed to the rate at which input spikes occur. This scheme would allow the system to remain silent, consuming little power, in potentially long silent periods, and still allow fast recognition when bursts of input activity arrive, a scenario that is realistic for natural organisms. These advantages have been recently realized for event-based convolutional networks using convolution chips (Camuñas Mesa et al., 2010; Farabet et al., 2012), but a principled way of building DBNs models out of spiking neurons, in which both feed-forward and feed-back processing are implemented has been lacking.

This paper presents the first proof-of-concept of how to transform a DBN model trained offline into the event-based domain. This allows exploiting the aforementioned advantages in terms of processing efficiency, and provides a novel and computationally powerful model for performing recognition, sampling from the model distribution, and fusion of different sensory modalities. Although our current implementation is in software, and not on neuromorphic VLSI, inference with small DBNs runs in real time on a standard laptop, and thus provides the first necessary step toward the goal of building neuromorphic hardware systems that efficiently implement deep, self-configuring architectures. In particular, the novel framework allows us to apply state-of-the-art computer vision and machine learning techniques directly to data coming from neuromorphic sensors that naturally produce event outputs, like silicon retinas (Lichtsteiner et al., 2008) and cochleas (Liu et al., 2010).

Our main contribution is a novel method for adapting conventional CD training algorithms for DBNs with spiking neurons, using an approximation of the firing rate of a Leaky Integrate-and-Fire (LIF) spiking neuron (Siegert, 1951; Jug et al., 2012). After training with a time-stepped model, the learned parameters are transferred to a functionally equivalent spiking neural network, in which event-driven real-time inference is performed. In this article we explicitly perform learning of the network offline, rather than with spike-based learning rules, but note that there is a high potential for future event-driven DBNs that could exploit spike-timing based learning for recognizing dynamical inputs. We evaluate the spiking DBNs by demonstrating that networks constructed in this way are able to robustly and efficiently classify handwritten digits from the MNIST benchmark task (LeCun et al., 1998), given either simulated spike-train inputs encoding static images of digits, or live inputs from neuromorphic vision sensors. In addition we present an event-based DBN architecture that can associate visual and auditory inputs, and combine multiple uncertain cues from different sensory modalities in a near-optimal way. The same architecture that is used for inference of classes can also be used in a generative mode, in which samples from the learned probability distribution are generated through feed-back connections.

The aspect of combining feed-back and feed-forward streams of information is an important deviation from traditional purely feed-forward hierarchical models of information processing in the brain (Van Essen and Maunsell, 1983; Riesenhuber and Poggio, 1999), and DBNs provide a first step toward linking state-of-the-art machine learning techniques and modern models of Bayesian inference and predictive coding in the brain (Rao and Ballard, 1999; Hochstein and Ahissar, 2002; Friston, 2010; Markov and Kennedy, 2013). The importance of recurrent local and feed-back connections in the cortex seems obvious from the anatomy (da Costa and Martin, 2009; Douglas and Martin, 2011; Markov et al., 2012), and *in vivo* experiments (Lamme et al., 1998; Kosslyn et al., 1999; Bullier, 2001; Murray et al., 2002), but the precise role of feed-back processing is still debated (Lamme et al., 1998; Bullier, 2001; Kersten and Yuille, 2003). One hypothesized role is in multisensory integration, and as generative Bayesian models, DBNs are very well suited to perform such tasks, e.g., by combining visual and auditory cues for improved recognition (Hinton et al., 2006). We will thus discuss the potential impact of DBNs as abstract functional models for cortical computation and learning.

The structure of this article is as follows: The mathematical framework and the algorithms used for training and converting conventional DBNs into spiking neural networks are presented in Section 2. Section 3 shows the application of the framework to simulated spike train inputs and real visual and auditory inputs from neuromorphic sensors. Implications of this new framework are discussed in Section 4.

## 2. MATERIALS AND METHODS

### 2.1. DEEP BELIEF NETWORKS

A DBN (Bengio et al., 2006; Hinton et al., 2006) is a multi-layered probabilistic generative model. The individual layers consist of simpler undirected graphical models, so called *Restricted Boltzmann Machines* (RBMs), typically with stochastic binary units. A RBM has a bottom layer of "visible" units, and a top layer of "hidden" units, which are fully and bidirectionally connected with symmetric weights. The difference between standard Boltzmann machines and RBMs is that in the restricted model

units within the same layer are not connected (see **Figure 1**), which makes inference and learning within this graphical model tractable. The visible layers of RBMs at the bottom of a DBN are clamped to the actual inputs when data is presented. When RBMs are stacked to form a DBN, the hidden layer of the lower RBM becomes the visible layer of the next higher RBM. Through this process, higher level RBMs can be trained to encode more and more abstract features of the input distribution.

In a binary RBM the units stochastically take on states 0 or 1, depending on their inputs from the other layer. Denoting the states of visible units with $v_i$, the states of hidden units with $h_j$, the weights connecting these units with $w_{ij}$, and the biases of visible and hidden units with $b_i^{(v)}$ and $b_j^{(h)}$ respectively, a RBM encodes a joint probability distribution $p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta})$, defined via the energy function

$$E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}) = -\sum_i \sum_j w_{ij} v_i h_j - \sum_i b_i^{(v)} v_i - \sum_j b_j^{(h)} h_j, \quad (1)$$

where $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{b}^{(v)}, \mathbf{b}^{(h)})$. The encoded joint probability can then be written as

$$p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta}) = \frac{\exp(-E(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta}))}{\sum_{\mathbf{v}'} \sum_{\mathbf{h}'} \exp(-E(\mathbf{v}', \mathbf{h}'; \boldsymbol{\theta}))}. \quad (2)$$

From equations (1) and (2) the following stochastic update rules for the states of units were derived (Hinton and Sejnowski, 1986), such that on average every update results in a lower energy state, and ultimately settles into an equilibrium:

$$p(v_i = 1|\mathbf{h}, \boldsymbol{\theta}) = \sigma\left(\sum_j w_{ij} h_j + b_i^{(v)}\right) \quad (3)$$

$$p(h_j = 1|\mathbf{v}, \boldsymbol{\theta}) = \sigma\left(\sum_i w_{ij} v_i + b_j^{(h)}\right), \quad (4)$$

where $\sigma(x) = 1/\left(1 + \exp(-x)\right)$ is the sigmoid function, and the units will switch to state 0 otherwise. When left to run freely, the network will generate samples over all possible states $(\mathbf{v}, \mathbf{h})$ according to the joint probability distribution in (2). This holds for any arbitrary initial state of the network, given that the network has enough time to become approximately independent of the initial conditions.

#### 2.1.1. Training a RBM

During learning, the visible units are clamped to the actual inputs, which are seen as samples from the "data distribution." The task for learning is to adapt the parameters $\boldsymbol{\theta}$ such that the marginal distribution $p(\mathbf{v}|\boldsymbol{\theta}) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}|\boldsymbol{\theta})$ becomes maximally similar to the true observed data distribution $p^*(\mathbf{v})$, i.e., the log-likelihood of generating the observed data needs to be maximized. Hinton et al. (2006) have shown that this gradient ascent on the log-likelihood w.r.t. the weights $w_{ij}$ can be efficiently approximated by a Gibbs-sampling procedure, which alternates between stochastically updating the hidden and visible units respectively. For the RBM this leads to the learning rule

$$\Delta w_{ij} = \eta\left(\langle v_i h_j\rangle_{\text{data}} - \langle v_i h_j\rangle_{\text{model}}\right), \quad (5)$$

where $\langle.\rangle_{\text{data}}$ denotes an average over samples with visible units clamped to actual inputs, $\langle.\rangle_{\text{model}}$ denotes an average over samples when the network is allowed to sample all units freely, and $\eta$ is the learning rate.

Using a sampling approximation normally requires creating enough samples such that the network can settle into an equilibrium. However, for a RBM the CD algorithm (Hinton et al., 2006) has been developed, which uses only a single sample for the data and model distribution, and performs very well in practice. CD first samples new values for all hidden units in parallel, conditioned on the current input, which gives a complete sample $(\mathbf{v}_{\text{data}}, \mathbf{h}_{\text{data}})$ for the data distribution. It then generates a sample for the visible layer, conditioned on the hidden states $\mathbf{h}_{\text{data}}$ sampled in the first step, and then samples the hidden layer again, conditioned on this new activity in the visible layer. This generates a sample $(\mathbf{v}_{\text{model}}, \mathbf{h}_{\text{model}})$ from the model distribution. The weight update can then be computed as

$$\Delta w_{ij} = \eta\left(v_{i,\text{data}} h_{j,\text{data}} - v_{i,\text{model}} h_{j,\text{model}}\right). \quad (6)$$

#### 2.1.2. Persistent CD and transient weights

Since the form of sampling induced by CD strongly biases the samples from the model distribution toward the most recently seen data, one can alternatively use so-called Persistent Contrastive Divergence (Tieleman, 2008). In this approach, the model distribution is initialized arbitrarily, and at every iteration of the training process further samples are created by sampling conditioned on the most recently sampled hidden states, which are maintained between data points.

There is a delicate balance between sampling and learning in Persistent CD: Although fast learning is generally desirable, too fast learning can result in too fast changes of the encoded joint probability distribution, which can cause the equilibrium distribution to change too fast for the Markov chain of model states to ever settle in. Nevertheless, high learning rates have turned out to be beneficial in practice, since they increase the mixing rates of the persistent Markov chains (Tieleman and Hinton, 2009). Following the suggestions in Tieleman and Hinton (2009) we used so called "fast weights," which are added to the regular weights of the network, and decay exponentially with each training step. When sampling from the model distribution, the fast weights are updated with the rule:

$$\Delta w_{ij}^{\text{fast}} = -\alpha\left\langle v_i h_j\right\rangle_{\text{model}}. \quad (7)$$

We will later show that such transient weight changes can be interpreted as short-term plasticity in a spiking neural network implementation.

#### 2.1.3. Constructing DBNs by stacking RBMs

As discussed previously, DBNs can be constructed by stacking RBMs and interpreting the hidden layer of the lower RBM as the visible layer of the next layer. It has been shown that adding

hidden layers and applying the previously discussed unsupervised learning methods for RBMs is guaranteed to increase the lower bound on the log-likelihood of the training data (Hinton et al., 2006). Higher layers will tend to encode more abstract features, which are typically very informative for classification tasks. The top-layer of the DBN can then be trained with supervised learning methods, and the whole multi-layer network can be optimized for the task through error backpropagation (Hinton and Salakhutdinov, 2006; Hinton et al., 2006).

DBNs can also be used for associating different sets of inputs, e.g., from different sensory modalities. In this case one can build pre-processing hierarchies for both inputs independently, and then treat the top layers of these hierarchies as a common visible layer for a new association layer on top of them (Hinton et al., 2006). DBNs are therefore not necessarily single hierarchies, but can also exhibit tree-like architectures.

### 2.2. DISCRETE-TIME AND EVENT-DRIVEN NEURON MODELS

Traditional RBMs are, like most machine-learning models, simulated in time-stepped mode, where every neuron in a layer gets updated at every time step, and the size of this time step $\Delta t$ is fixed throughout the simulation. While training is typically easier to achieve with continuous and time-stepped neuron models, the event-driven model has the potential to run faster and more precisely. This is because the states of LIF neurons in the event-based network are only updated upon the arrival of input spikes, and only at these times the neurons decide whether to fire or not. Temporal precision is limited only by the numerical representation of time in the system (as opposed to the duration of the time-step parameter). A drawback is that not all neuron models, e.g., smooth conductance-based models, can be easily converted into event-driven models.

In the standard formulation (Hinton et al., 2006), units within RBMs are binary, and states are sampled according to the sigmoidal activation probabilities from Equations (3) and (4). We call such neuron models sigmoid-binary units. In Nair and Hinton (2010) it was shown that an equivalent threshold-linear model can be formulated, in which zero-mean Gaussian noise $\mathcal{N}(0, \sigma_n^2)$ with variance $\sigma_n^2$ is added to the activation functions:

$$h_j = \max\left(0, \sum_i w_{ij} v_i + b_j^h + \mathcal{N}(0, \sigma_n^2)\right), \tag{8}$$

and similarly for the sampling of visible units.

A threshold-linear function can also be used to approximate the expected firing rates of simplified spiking neurons under constant current stimulation, such as the LIF neuron (Gerstner and Kistler, 2002), which is one of the simplest, yet biologically relatively plausible models for spiking neurons. In this model each incoming event adds to the membrane potential $V_m$ according to the strength $w_{ij}$ of the synapse along which the event occurred. Incoming spikes within an absolute refractory period $t_{ref}$ after an output spike are ignored. Spikes are generated deterministically whenever the membrane potential crosses the firing threshold $V_{th}$, otherwise the membrane potential decays exponentially with time constant $\tau$. Simple versions of LIF neurons can be simulated in an event-based way, since membrane potentials only need to be updated upon the arrival of input spikes, and spikes can only be created at the times of such input events. For a LIF neuron representing $h_j$, which receives a constant input current $s_j = \sum_i w_{ij} v_i$ corresponding to the weighted sum of inputs from connected visible units, the expected firing rate $\rho_j(s_j)$ is:

$$\rho_j(s_j) = \begin{cases} \left(t_{ref} - \tau \log\left(1 - \frac{V_{th}}{s_j}\right)\right)^{-1} & \text{if } s_j \geq V_{th} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The above equation holds when the neuron is injected with a constant input, but under realistic conditions the neuron receives a continuous stream of input spike trains, each arriving to first approximation as samples from a Poisson process with some underlying firing rate. For this case, a more accurate prediction of the average firing rate can be obtained using **Siegert neurons** (Siegert, 1951; Jug et al., 2012). Siegert neurons have transfer functions that are mathematically equivalent to the input-rate output-rate transfer functions of LIF neurons with Poisson-process inputs. In order to compute the Siegert transformation for a neuron receiving excitatory and inhibitory inputs with rates $(\vec{\rho}_e, \vec{\rho}_i)$ and weights $(\vec{w}_e, \vec{w}_i)$ respectively, we first have to compute the auxiliary variables

$$\mu_Q = \tau \sum (\vec{w}_e \vec{\rho}_e + \vec{w}_i \vec{\rho}_i) \qquad \sigma_Q^2 = \tfrac{\tau}{2} \sum (\vec{w_e^2} \vec{\rho}_e + \vec{w_i^2} \vec{\rho}_i)$$
$$\Upsilon = V_{rest} + \mu_Q \qquad \Gamma = \sigma_Q$$
$$k = \sqrt{\tau_{syn}/\tau} \qquad \gamma = |\zeta(1/2)|$$

where $\tau_{syn}$ is the synaptic time constant (for our purposes considered to be zero), and $\zeta$ is the Riemann zeta function. Then the average firing rate $\rho_{out}$ of the neuron with resting potential $V_{rest}$ and reset potential $V_{reset}$ can be computed as (Jug et al., 2012):

$$\rho_{out} = \left(t_{ref} + \frac{\tau}{\Gamma}\sqrt{\frac{\pi}{2}} \cdot \int_{V_{reset} + k\gamma\Gamma}^{V_{th} + k\gamma\Gamma} \exp\left[\frac{(u - \Upsilon)^2}{2\Gamma^2}\right] \cdot \left[1 + \operatorname{erf}\left(\frac{u - \Upsilon}{\Gamma\sqrt{2}}\right)\right] du\right)^{-1}. \tag{10}$$

A RBM trained using Siegert units can thus be easily converted into an equivalent network of spiking LIF neurons: By normalizing the firing rate in Equation (10) relative to the maximum firing rate $1/t_{ref}$, $\rho_{out}$ can be converted into activation probabilities as required to sample RBM units in Equations (3, 4) during standard CD learning with continuous units. After learning, the parameters and weights are retained, but instead of sampling every time step, the units generate Poisson spike trains with rates computed by the Siegert formula Equation (10).

### 2.3. TRAINING THE NETWORK

#### *2.3.1. Task*

The network was trained on a visual classification task on the MNIST benchmark dataset for machine learning (LeCun et al., 1998). This set consists of a collection of 28 × 28 gray-scale images of handwritten digits, of which 60,000 form a training set,

and 10,000 an independent test set. In order to make the network more robust, we modified the training set by adding small random translations (±15%), rotations (±3°) and scalings (±10%). The modified training set contains 120,000 images.

#### 2.3.2. Network Architecture

For the visual classification task we trained a DBN with one input layer of 784 visual input units (corresponding to the pixels of 28 × 28 input images), a 500-unit "Visual Abstraction Layer," a 500-unit "Association Layer," and a 10-unit "Label Layer," with units corresponding to the 10 digit-classes. The architecture of the network is shown in **Figure 2**. Since our goal in this article is to demonstrate a proof-of-concept for spiking DBNs, the 785-500-500-10 network we used is substantially smaller than the 784-500-500-2000-10 network used previously for the MNIST task (Hinton et al., 2006), or the state-of-the-art network in Ciresan et al. (2010).

#### 2.3.3. Training

Each RBM in **Figure 2** was first trained in a time-stepped mode with Siegert neurons as individual units, for which we fixed the parameters for resting and reset potential, membrane time constants, and refractory period. Since the output rates of Siegert neurons are not constrained to the interval [0, 1] like in Sigmoid-Binary units, the outputs were normalized, such that the maximum possible firing rate (given by $1/t_{ref}$) had a value of 1. As training algorithm for RBMs we applied persistent Contrastive Divergence learning (Hinton et al., 2006) and the fast weights heuristics described in Section 2.1.2. We also applied a modification to the training process proposed by Goh et al. (2010) to encourage sparse and selective receptive fields in the hidden layer.

Learning proceeded in a bottom-up fashion, starting by training the weights between the Visual Input and the Visual Abstraction Layers. Next, the weights of the Associative Layer were trained, using input from the previously trained Visual Abstraction Layer and the supervised information in the Label Layer as the joint visible layer of the RBM. For each layer we trained for 50 iterations over the complete training set.

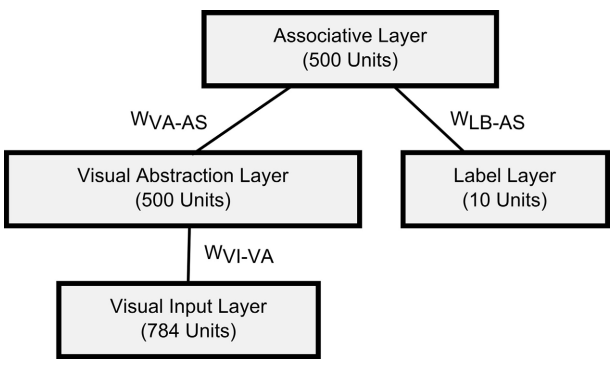



**FIGURE 2 | Architecture of the DBN for handwritten digit recognition.** The connections between layers represent the weights of a RBM.

### 2.4. SIMULATION OF AN EVENT-DRIVEN DBN

We created simulators for arbitrary event-driven DBNs in Matlab and Java. The simulation can be either run in *Recognition mode*, where input is applied at the bottom layer, and the label has to be inferred through bottom-up processing, or in *Generation mode*, where the activity of the label layer is fixed, and the network samples activity in the Visual Input Layer through top-down connections, according to the learned generative model. Bottom-up and top-down processing can also be activated simultaneously.

In *Recognition mode*, the DBN is shown a number of test images, which are transformed into spike trains that activate the Visual Input Layer. A Poisson spike train is created for each pixel with a rate proportional to the pixel intensity, and all firing rates are scaled such that the total input rate summed over all 28 × 28 pixels is constant (between 300 and 3000 spikes per second). The goal is to compute the correct classification in the Label Layer. For every input image, the input activations are sampled as Poisson spike trains with rates proportional to the pixel intensities. Classification can be done in one of two ways: first, we can turn on only bottom-up connections from the Visual Input Layer toward the Label Layer, and observe which of the neurons in the Label Layer spikes the most within a fixed time interval. The second variant is to use only bottom-up connections between Visual Input and Visual Abstraction Layer, but activate all recurrent connections in the other RBMs. Information about previous inputs is stored both within the membrane potentials and the recurrent spiking activity within the network. Recognition is thus achieved through a modulation of the persistent network activity by input spike trains. In the absence of input, the network will continue to be active and drift randomly through the space of possible states according to the encoded generative model.

This principle is exploited in the *Generation mode*, where units within the Label Layer are stimulated, and activation propagates recurrently through the top-level RBM, and top-down to the Visual Input Layer. Thus, analyzing these samples from the generative model provides a way to visualize what the network has learned so far. If the DBN is activated in this way, it might settle into a particular state, but could become stuck there, if this state corresponds to a local minimum of the Energy landscape according to (1). This can be avoided by using a short-term depressing STDP kernel in Generation mode, which temporarily reduces the weights of synapses where pre- and post-synaptic neurons are active within the same short time window (see **Figure 3**). These short-term modifications vanish over time, and the weights return to their original values. This modification is inspired by the idea of using auxiliary "fast-weights" for learning (Tieleman and Hinton, 2009), which transiently raise the energy of any state that the network is currently in, thereby slightly pushing it out of that state. The effect is that the network, instead of settling into an energy well and remaining there, constantly explores the whole space of low-energy states. This is a useful feature for search and associative memory tasks, where the network represents a cost function through the encoded energy landscape, and the task is to converge to a maximally likely state starting from an arbitrary initial state, e.g., an incomplete or ambiguous input. We demonstrate this in Section 3.4 in the context of multi-sensory integration.

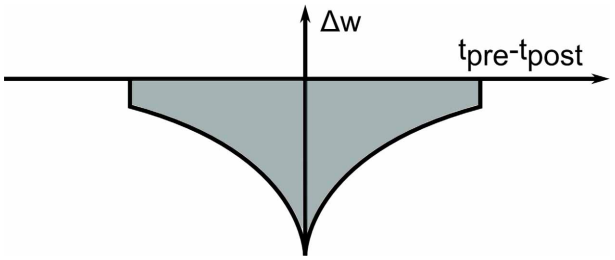



**FIGURE 3 | Short-term plasticity kernel for Generation mode.** The "fast-weight" STDP kernel temporarily depresses all synapses in which the pre- and post-synaptic neurons were active shortly after each other, depending on the spike timing difference $t_{pre} - t_{post}$. As a result, the network is constantly being pushed out of its present state.

### 2.5. REAL-TIME IMPLEMENTATION

#### *2.5.1. Neuromorphic visual input*

We developed a real-time variant of the event-driven DBN which receives inputs from neuromorphic sensors. Visual input was obtained from the DVS (Lichtsteiner et al., 2008), an event-generating image sensor consisting of 128 × 128 pixels, which asynchronously outputs streams of address events in response to local relative light-intensity changes. The events are tagged with the address of the creating pixel, a time-stamp, and an ON or OFF polarity tag, which indicates whether the event was created in response to an increase or decrease of light intensity over that pixel. Events are transmitted via a USB port to a computer, and processed in the open-source jAER software framework written in Java (Delbruck, 2013). The networks were first trained in Matlab, and then transferred into the jAER software, where they could run in real-time in response to event stream inputs. We did not use the polarity information for our purposes, and down-sampled the 128 × 128 pixels to a resolution of 28 × 28, which matched the resolution of the images in the MNIST training set. These events were fed into the Visual Input Layer (see **Figure 2**) while the DVS was moved by hand across several hand-drawn images.

#### *2.5.2. Multi-sensory fusion*

We also created a task in which visual stimuli from a silicon retina and auditory stimuli from a silicon cochlea (see Section 2.5.3) were associated with each other in real-time. During training the presentation of a pure tone was always paired with the presentation of an image of a handwritten digit. **Table 1** shows the tones and frequencies that were used, and the visual-auditory pairing scheme. The network thus had to learn to associate the two sensory domains, e.g., by resolving ambiguity in one sensory stream through information from the other stream.

The DBN architecture for sensory fusion is described in detail in Section 3.4 and shown in **Figure 8**.

#### *2.5.3. Neuromorphic Auditory Input*

Auditory input was received from the AER-EAR2 (Liu et al., 2010) neuromorphic auditory sensor, which was built to mimic the biological cochlea. The device transforms input sounds into streams of spikes in 64 channels responsive to different frequency ranges. We found that since spikes of the silicon cochlea tend to be phase-locked to the sound waveform to which they are responding, the distribution of Inter-spike Intervals (ISIs) was a more precise indicator of the frequency of pure input tones than the distributions of channels from which the spikes originated. We preprocessed the auditory spikes with an event-based ISI histogramming method wherein 100 ISI bins were distributed logarithmically between 0.833 and 2.85 ms (350–1200 Hz), and for each bin an input LIF unit was assigned which was stimulated every time an ISI occurred on any channel that was within the unit's designated frequency-range. The output events of these units were then routed to the Auditory Input Layer (see Section 3.4 and **Figure 8**).

**Table 1 | Paired tones and digits in multi-sensory fusion task.**

| Tone | $A_4$ | $B_4$ | $C_5$ | $D_5$ | $E_5$ | $F_5$ | $G_5$# | $A_5$ | $B_5$ | $C_6$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Freq.(Hz) | 440.0 | 493.9 | 523.3 | 587.3 | 659.3 | 698.5 | 830.6 | 880.0 | 987.8 | 1046.5 |
| Digit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

*During training pure tones with given frequencies (upper rows) were paired with an associated digit (bottom row).*

As stimuli we chose the pure tones from **Table 1** from the A-minor harmonic scale, ranging from A4 (440 Hz) to C6 (1046.5 Hz), which were played for 1 s each into the silicon cochlea. We recorded the spike response of neurons in the Auditory Input Layer, which fired whenever enough input events from AER-EAR2 in their ISI range were received. For training in the time-stepped domain we constructed data vectors for auditory data by computing the average firing rates of Auditory Input Layer neurons over time bins of 100 ms, evaluated every 30 ms.

## 3. RESULTS

This section presents the classification performance, shows the generative mode of operation, and presents sensor fusion experiments. For the results in sections 3.1 and 3.2 we use simulated spike-train input (see Section 2.4). Inputs from neuromorphic sensors (Section 2.5) are directly used in the results of sections 3.3 and 3.4.

### 3.1. CLASSIFICATION PERFORMANCE

Three variants of DBNs were trained, using the architecture shown in **Figure 2** for the MNIST visual classification task: the first two variants are time-stepped models using sigmoid-binary or Siegert neurons respectively (see Section 2.2), the third is an event-driven DBN using LIF neurons that were converted from Siegert neurons used during training. The networks were all trained in time-stepped mode for 50 iterations over the modified 120,000 example MNIST dataset using a variant of Contrastive Divergence learning (see Section 2.3). **Figure 4** shows the features learned by a subset of the neurons in the RBM for the Visual Abstraction Layer. One can see that this first layer has learned through unsupervised learning to extract useful features for the discrimination of handwritten digits, in this case parts of digits. The classification performance shown in **Table 2** was evaluated on images from the MNIST test set, using simulated Poisson spike trains with a total rate of 300 spikes per second for the whole image as input for event-based models. The size of our

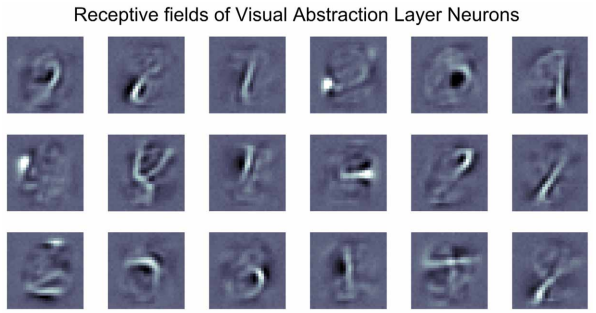



**FIGURE 4 | Analysis of weights learned in the DBN.** Visualization of the weights learned by a subset of neurons in the Visual Abstraction Layer for 28 × 28 images in the MNIST task. Each image shows the vector of weights feeding into one neuron.

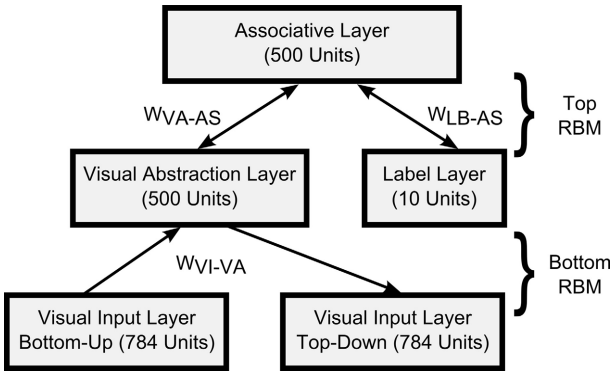



**FIGURE 5 | DBN architecture for recognition and generation.** The Visual Input Layer was split into a bottom-up and a top-down part, used for projecting inputs in Recognition mode, or visualizing top-down activity in Generation mode.

**Table 2 | Classification performance on the MNIST test set for two time-stepped and one event-based LIF neuron model.**

| Neuron model | Domain | % correct |
|---|---|---|
| Sigmoid-Binary | time-step | 97.48 |
| Siegert | time-step | 95.2 |
| LIF | event-based | 94.09 |

*Inputs for the event-based model were simulated Poisson spike trains (see Section 2.4).*

DBN is substantially smaller than in current state-of-the-art deep network approaches for MNIST, e.g., (Ciresan et al., 2010), but **Table 2** shows that the performance is in a very good range (above 94%). More importantly for this proof-of-concept study, the performance loss when switching to spiking neuron models is small (on the order of 1%), and can possibly be further improved when going to larger network sizes.

### 3.2. GENERATION MODE

In *Generation mode* the network does not receive external input at the bottom layers. Instead one of the top layers (in our case the Label Layer in **Figure 2**) is stimulated, and activity spreads in top-down direction through the network. This provides a way to visualize what has been learned in the probabilistic generative model encoded in the bi-directional weights.

Since the network is event-driven, and neurons fire only upon the arrival of input spikes, an initial stimulus in at least one of the layers is needed to push the network from a silent state into one of self-sustaining activity, provided that the neuron parameters and recurrent connectivity allow this. We performed empirical exhaustive parameter search over firing thresholds $V_{th}$ and membrane time constants $\tau$ in a fully trained network of LIF neurons and measured the mean firing rate within the network after 1 s of 100 Hz stimulation of one Label Layer unit, and 5 s without external stimulation. This allowed us to identify parameter regimes that allow self-sustained activity of about 20 Hz average activity in *Generation mode* ($\tau = 800$ ms, $V_{reset} = 0$, $V_{th} = 0.005$).

To visualize the activity of the DBN in Generation mode we modified the architecture in **Figure 2** that was used for training on the MNIST dataset. In the new architecture shown in **Figure 5** the lowest layer is split up after training into two Visual Input Layers, one projecting only bottom-up from inputs to the Visual Abstraction Layer, and another copy that is purely driven by top-down connections. The weight matrices for bottom-up and top-down connections are identical. Thus, the top layers of the network form the recurrent model that encodes the data distribution, whereas the bottom part either projects inputs through bottom-up connections in Recognition mode, or visualizes the activity of the top layers through top-down connections in Generation mode. If both bottom-up and top-down connections are activated at the same time, the top-down Visual Input Layer visualizes a processed image of what the network 'believes' it is seeing in the bottom-up Visual Input Layer. This process performs probabilistic inference by which evidence from the current input is combined with the prior distribution over likely MNIST images encoded in the DBN weights, and a posterior estimate of the most likely input is generated.

**Figure 6A** illustrates the generation of samples from the encoded probabilistic model after activating a unit in the Label Layer. This induces spiking activity in the intermediate Associative and Visual Abstraction Layer, and ultimately stimulates units in the top-down Visual Input Layer, which can be visualized. **Figure 6A** shows the response of the network when the label unit corresponding to the class "4" is stimulated. The snapshot shows the induced activity in the lower layers, and one can clearly see that the response in the Visual Input Layer resembles closely the handwritten digits in the MNIST set that were used for training. By using short-term depressing synapses as described in Section 2.1.2 and in **Figure 3** the network not just samples one single example of a "4," but iterates through different variations that are compatible with the variance over inputs in the learned generative model. This can be best seen in Video 1 of the supplementary material.

**Figure 6B** shows the spiking activity in the different layers of the network in generation mode, both during a forced stimulation, and in a free self-sustained mode. The network is initially stimulated for 2 s by forcing firing of neurons in the Label Layer

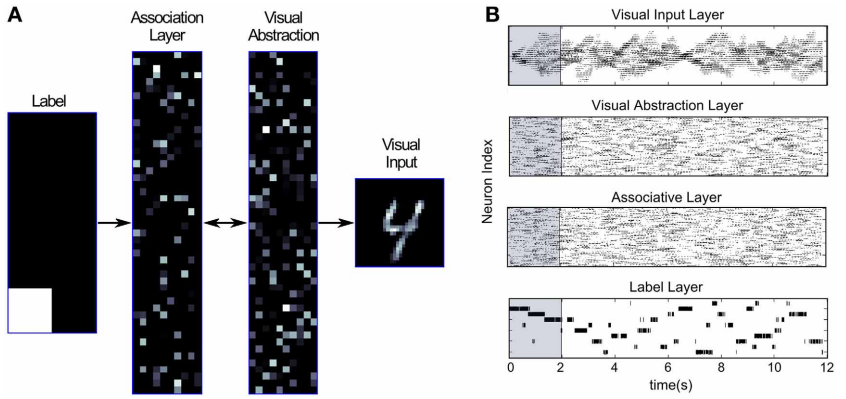



**FIGURE 6 | Generation mode of the event-driven DBN. (A)** Screen capture of the network while generating samples of input activations corresponding to class "4." The neuron corresponding to label "4" was stimulated in the Label Layer (left), and activity propagated through the whole network. The snapshot shows a single example of activity in the Visual Input Layer (right) that is sampled from the generative model encoded in the weights of the DBN. Through short-term depressing synapses (see **Figure 3**) the network starts to drift through the space of all potential compatible input activations. **(B)** Raster plot of the DBN in Generation mode. The Label Layer (bottom) is initially stimulated for 2 s (shaded region) to fire in sequence for digits 1, 2, and 3. Afterwards, the network freely samples from the encoded generative model. Although activity in the Label Layer jumps between digits, activity in the Visual Input Layer transitions smoothly.

corresponding to digit classes "1," "2," and "3" (shaded region). One can see that through the recurrent connectivity activity spreads throughout the layers of the network. After 2 s the input to the Label Layer is turned off, and the network is allowed to freely generate samples from the encoded probability distribution. We can see that in the Label Layer the network jumps between different digits, whereas in the other layers, more smooth transitions are found. Switches between visually similar digits (e.g., 4 and 9) occurred more often on average than between very different digits (e.g., 0 and 1).

### 3.3. REAL-TIME VISUAL RECOGNITION

For this task the event-driven DBN was connected to a neuromorphic vision sensor, the 128 × 128 pixel DVS (Lichtsteiner et al., 2008). Events indicating local light intensity changes are used as inputs in the bottom-up Visual Input Layer. The whole system works in real-time, i.e., while the DVS is recording visual input, the DBN simultaneously computes the most likely interpretation of the input. By splitting up the connections between Visual Input and Visual Abstraction Layer into a bottom-up and a top-down pathway as in **Figure 5** we can simultaneously classify the input in real-time, and also visualize in the top-down Visual Input Layer the interpretation of the input after recurrent processing in the DBN.

The real-time system runs as a filter in the jAER software package (Delbruck, 2013) on a standard laptop, after training the weights of the DBN offline in Matlab on the MNIST database. **Figure 7** shows snapshots of the activity within the different layers of the network during operation on various stimuli recorded in real-time with the DVS. In **Figure 7A** the DVS was moved over a hand-drawing of the digit "5" which was not included in the training set. The left panel shows the input into the Visual Input Layer. The digit was correctly classified as a "5" in the Label Layer. On the right we can see the reconstruction of the image, which closely resembles the actual input. In **Figure 7B** an ambiguous input was presented, which can either be interpreted as a "3" or a "5." The network iterated between both interpretations, in this snapshot the reconstruction on the right shows that the network currently interprets the input as a "3," adding the missing parts of the input to match the actual shape of a digit. In **Figure 7C** the network is shown an input from an unknown input class, namely the letter "A." Since the generative model learned in the

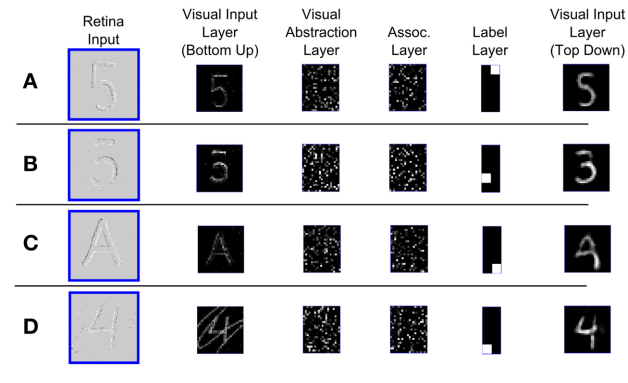



**FIGURE 7 | Screen captures of the real-time spiking DBN in operation during visual handwritten digit recognition.** Each row displays a snapshot of the activity in the different layers of the network (see **Figure 5**) for a different visual input recorded with the DVS (left column). Neurons in the Label Layer (column 5) are arranged such that the first column represent classes 0–4 (top to bottom), and the second column classes 5–9. The rightmost column shows the top-down reconstruction of the Visual Input Layer. **(A)** The network recognizes the digit 5. **(B)** For an ambiguous input, the network alternates between the two possible interpretations "3" and "5." The top-down reconstruction shows the current interpretation. **(C)** For an unfamiliar input (letter "A"), the network classifies it as the closest resembling digit class "9," and reconstructs a mixture between the actual input and the generative model for class "9." **(D)** For an input containing a distractor, the network still classifies it as the most likely input, and reconstructs an image without the distractor.

network knows only digits, it classifies the input as the most similar digit, in this case "9," and reconstructs the input as a mixture between the actual DVS input and the entrained model of the digit. In **Figure 7D** a digit "4" with a distracting stimulus on top was shown. It was correctly classified and reconstructed in the top-down Visual Input Layer without the distracting stimulus.

In general, the network recognized reliably all tested classes of handwritten digits in real-time, even in the presence of strong distractors, with slightly rotated images, or variations in scale or translation of the image. It can also do so very quickly: at a typical low-rate input firing rate of 3000 input spikes per second over the whole image, the DBN submits its first correct guess of the output label within an average of 5.8 ms after the onset of the simulated Poisson spike train input. Firing rates in the intermediate layers are higher, resulting in 58800 spikes/s in the 500 neuron Visual Abstraction Layer (see **Figure 2**), 147600 spikes/s in the 500 neuron Association Layer, and 1800 spikes/s in the 10 neuron Label Layer.

### 3.4. REAL-TIME SENSORY FUSION

We trained a DBN to associate visual stimuli from a silicon retina, and auditory stimuli from a silicon cochlea, in order to classify them in real-time by integrating both input streams. **Table 1** shows the respective association of digit images recorded with the DVS (Lichtsteiner et al., 2008), and tones of different frequencies recorded with the AER-EAR2 silicon cochlea (Liu et al., 2010). We used the DBN architecture shown in **Figure 8**, in which a bidirectional connection between the top-level Association Layer and the Auditory Input Layer is added.

During training a network of Siegert neurons were presented with input images from the MNIST database and pre-recorded activations of Auditory Input Layer neurons in response to the tones in **Table 1** (see Section 2.5.3). After the training phase, the DBN was converted into an event-driven DBN as described previously, which was run in real-time in the jAER software package.

One key aspect of sensory fusion is the ability to integrate multiple, possibly noisy or ambiguous cues from different sensory domains to decide on the actual state of the world. We tested this by providing simultaneous visual and auditory stimuli to the DBN, such that the combination of both stimuli would provide more conclusive evidence of the true label than the single modalities. The auditory stimulus was a mixture of $A_4$ and $F_5$ tones corresponding to "0" and "5" digits, with four times as many input spikes corresponding to class "0" as to class "5". Thus, if given only the audio input, the DBN should identify a "0". Conversely, the visual input shows an ambiguous input that is consistent with either a "3" or a "5," but very unlikely for a "0". **Figure 9** demonstrates the audio-visual fusion using an ambiguous visual input and the auditory input favoring class "0." However, while each input stream favors an incorrect interpretation of either "3" or "0," class "5" is correctly chosen as the most consistent representation for the combined visual-auditory input stream.

In **Figure 10** we analyzed how this depends on the relative strength of visual and auditory input streams and the ambiguity of the visual input by (1) changing the relative proportion of input spikes coming from the audio stream, and (2) interpolating the visual input between an image showing "3" and another one showing "5." We varied the mixture of firing rates of input neurons such that 80% (**Figure 10A**), 20% (**Figure 10B**), and 10% (**Figure 10C**) of all input spikes came from the auditory stream, and measured the proportion of output spikes for the three classes "0," "3," and "5". In panels **A** and **C** the classes that are inconsistent with the dominating auditory respectively visual input are almost completely suppressed, and class "5" is favored. One can also see from the difference between **Figures 10A,B** that an increase of a few spikes favoring an alternative interpretation can dramatically adjust the output choice: In this case 10% more of spikes favoring the interpretation "5" are enough to bias the classification toward the interpretation consistent with both visual and auditory input over a wide range of visual ambiguity.

## 4. DISCUSSION

The great potential of DBNs is widely recognized in the machine learning community and industry (MIT Technology Review, 2013). However, due to the high computational costs, and the capability to integrate large amounts of unlabeled data that is

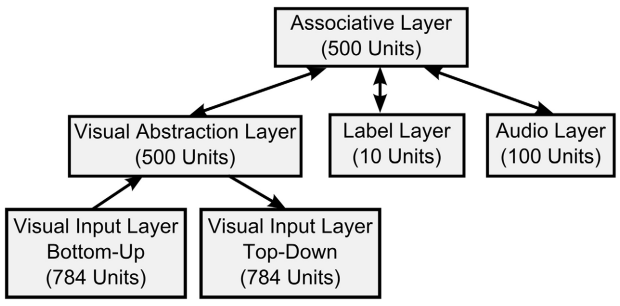



**FIGURE 8 | DBN architecture of the multi-sensory fusion network.** In addition to the architecture for visual recognition in **Figure 5** the Auditory Input Layer is bidirectionally connected to the top-level Association Layer. Thus, associations between visual inputs, auditory inputs, and classification results in the Label Layer can be learned during training, and classification can be achieved in real-time.

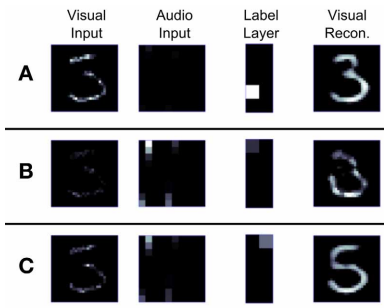



**FIGURE 9 | Cue integration with a multi-sensory spiking DBN. (A)** When presenting only an ambiguous visual input to the DVS, the network in the absence of auditory input will alternate between recognizing a "3" or a "5" (see also **Figure 7**). **(B)** When presenting only an ambiguous auditory input to the cochlea, the network in the absence of visual input will alternate between recognizing a "0" or a "5." **(C)** By combining the two inputs (mixing at 50%), the network reliably classifies the two ambiguous patterns as class "5," which is the only consistent interpretation.

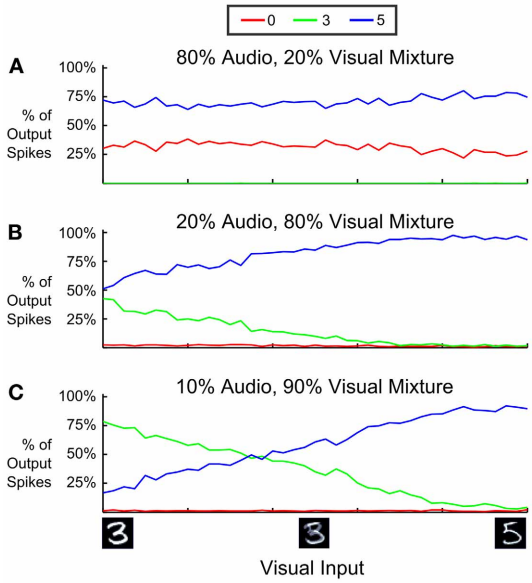



**FIGURE 10 | Proportion of output spikes for 3 different mixture ratios of auditory and visual input in a multi-sensory spiking DBN.** Red, green, and blue encode the ratio of 0, 3, and 5 choices (spikes) relative to the total number of spikes emitted from the Label Layer (averaged over 10 trials). The horizontal axis sweeps the probability that visual input spikes are chosen from either a "3" digit or an aligned "5" digit. Auditory input consists of a mixture of "0" and "5" inputs, with four times more spikes indicating a "0." Over a wide range of mixture values, the network correctly infers the only consistent interpretation of the multi-modal input, which is class "5." Inputs that are inconsistent with the dominating sensory domain ("3" in **A**, "0" in **B,C**) are mostly suppressed.

freely available on the web, applications so far have strongly concentrated on big data problems (Le et al., 2012). Surprisingly little effort has gone into making this technology available for real-time applications, although the scenarios in which DBNs excel, e.g., visual object recognition, speech recognition, or multi-sensory fusion, are extremely important tasks in fields like robotics or mobile computing. An exception is the work of (Hadsell et al., 2009), who use small and mostly feed-forward deep networks for long-range vision in an autonomous robot driving off road. In general, previous attempts to reduce the running time have mostly attempted to restrict the connectivity of networks (Lee et al., 2008; Le et al., 2012), e.g., by introducing weight-sharing, pooling, and restricted receptive fields. In speech processing on mobile phones, data is first communicated to a central server where it is processed by a large DBN before the result is sent back to the mobile device (Acero et al., 2008). Online and on-board processing would be very important for mobile applications where such communication infrastructure is not available, e.g., for exploration robots in remote areas, underwater, or other planets, but this requires fast and efficient processing architectures, that conventional DBNs currently cannot provide.

We think that this presents a big opportunity for neuromorphic engineering, which has always pursued the goal of providing fast and energy efficient alternatives to conventional digital computing architectures for real-time brain-inspired cognitive systems (Indiveri et al., 2011). Here we have presented a novel method how to convert a fully trained DBN into a network of spiking LIF neurons. Even though we have only shown a proof-of-concept in software, this provides the necessary theoretical framework for an architecture that can in the future be implemented on neuromorphic VLSI chips, and first experiments in this direction are promising. The event-driven approach can be energy efficient, in particular since the required processing power depends dynamically on the data content, rather than on the constant dimensionality of the processed data. Furthermore, as we have shown, spiking DBNs can process data with very low latency, without having to wait for a full frame of data, which can be further improved if individual units of the DBN compute in parallel, rather than updating each unit in sequence. This advantage has been recognized for many years for feed-forward convolutional networks, in which almost all operations can be efficiently parallelized, and has led to the development of custom digital hardware solutions and spike-based convolution chips (Camuñas Mesa et al., 2010; Farabet et al., 2012), which through the use of the Address Event Representation (AER) protocol, can also directly process events coming from event-based dynamic vision sensors (Lichtsteiner et al., 2008). For such architectures (P'erez-Carrasco et al., 2013) have recently developed a similar mapping methodology between frame-based and event-driven networks that translates the weights and other parameters of a fully trained frame-based feed-forward network into the event-based domain, and then optimizes them with simulated annealing. In comparison, this offers increased flexibility to change neuronal parameters after training, whereas our method uses the accurate Siegert-approximation of spike rates already during the training of a bi-directional network, and does not require an additional optimization phase. The advantages of spike-based versus digital frame-based visual processing in terms of processing speed and scalability have been compared in Farabet et al. (2012), where it was also suggested that spike-based systems are more suitable for systems that employ both feed-forward and feed-back processing.

Although our model is event-based, the Siegert model (Siegert, 1951) does not make use of the precise timing of spikes. The basic theoretical framework of DBNs is not suitable for inputs that vary in time, and thus requires modifications to the network architecture (Taylor et al., 2007), or a transformation of inherently time-dependent inputs (Dahl et al., 2012). Learning with STDP-like rules in spiking DBNs provides an intriguing future possibility for a direct handling of dynamic inputs. In our current network, the short-time memory of previously seen inputs carried in the membrane potential of LIF neurons allows us to process inputs from asynchronous neuromorphic sensors, in which complete frames are never available (Lichtsteiner et al., 2008; Liu et al., 2010). We can therefore for the first time apply the state-of-the-art machine learning technique of DBNs directly to inputs from event-based sensors, without any need to convert input signals, and can classify the input while also completing the input signals using feed-back connections.

Feed-back connections are rarely used in models of biologically inspired vision, e.g., HMAX (Riesenhuber and Poggio, 1999), but as we show e.g., in **Figure 7**, feed-back and recurrency are essential for implementing general probabilistic inference,

e.g., to infer missing, ambiguous, or noisy values in the input. Only in recent years have models become available that directly link spiking activity in recurrent neural networks to inference and learning in probabilistic graphical models. Nessler et al. (2013) have shown that learning via STDP in cortical microcircuits can lead to the emergence of Bayesian computation for the detection of hidden causes of inputs. They interpret spikes as samples from a posterior distribution over hidden variables, which is also the essential idea for neural sampling approaches (Büsing et al., 2011), in which spiking neurons implement inference in a Boltzmann machine via Markov Chain Monte Carlo sampling. Using clock-like waves of inhibition, Merolla et al. (2010) showed an alternative implementation of single Boltzmann machines with spiking neurons.

In biology, the precise role of feed-back processing is still debated, but the deficiencies of purely feed-forward architectures for processing the kind of clutter, occlusions, and noise inherent to natural scenes point at least to a role in modulation by attention signals, and in the integration of multiple cues, possibly from different modalities as well as memory and high-level cognitive areas (Lamme et al., 1998; Bullier, 2001; Kersten and Yuille, 2003). A proposal from Hochstein and Ahissar (2002) even suggests a reverse hierarchy for conscious vision, whereby fast feed-forward perception is used for a quick estimate of the *gist* of the scene, and for activating top-down signals that focus attention on low-level features necessary to resolve the details of the task. Such a model can explain the fast pop-out effect of image parts that violate model expectations, and also provides a model for fast learning without changes in the early sensory processing stages. This is consistent with a variety of theories that the brain encodes Bayesian generative models of its natural environment (Kersten and Yuille, 2003; Knill and Pouget, 2004). The hierarchical organization of sensory cortices would then naturally correspond to a hierarchy of prior distributions from higher to lower areas that can be optimally adapted to statistics of the real world in order to minimize surprise (Friston, 2010). Rao and Ballard (1999) suggested that inference in such hierarchical generative models could be efficiently performed through predictive coding. In this framework, feed-back connections would signal a prediction from higher to lower layers, whereas feed-forward connections would encode the error between prediction and actual input. In Rao and Ballard (1999) it was shown that such a model can account for several phenomena concerning the non-linear interaction of center and surround of receptive fields, and fMRI data support the theory by reporting reduced V1 activity when recognition-related activity in higher areas increases (Murray et al., 2002).

The framework of Bayesian generative models also provides a principled way of associating and integrating potentially uncertain cues from different sources, e.g., across sensory modalities (Knill and Pouget, 2004). It is well known that humans use all available cues for solving tasks, e.g., by using visual cues to improve their understanding of speech (Kayser and Logothetis, 2007; Stein and Stanford, 2008). Although traditional models have assumed that multi-sensory integration occurs only at higher association areas like superior colliculus (Felleman and Van Essen, 1991), feed-back connections from higher to lower areas or between sensory streams are likely to be involved in sensory fusion tasks. Recent studies have revealed the existence of anatomical connections that would enable cross-modal interactions also at lower levels (Falchier et al., 2002; Markov et al., 2012), and functional studies have provided some (but not conclusive) evidence of co-activations of early sensory areas by stimulation of different modalities [see (Kayser and Logothetis, 2007) for a review]. Integration might also be required within the same sensory modality, since e.g., the visual pathway splits up into at least two separate major ventral and dorsal streams.

All these arguments indicate that the traditional concept of sensory processing in the cortex as a feed-forward hierarchy of feature detectors with increasing levels of abstraction in higher layers (Gross et al., 1972; Van Essen and Maunsell, 1983; Desimone et al., 1984) needs to be reassessed (Markov and Kennedy, 2013). A closer look at the anatomy of intra- and inter-areal cortical connectivity reveals an abundance of feed-back and recurrent connections. Every brain area receives inputs from a large number of cortical and subcortical sources (Douglas and Martin, 2011; Markov et al., 2012), and feed-forward connections actually make up only a relatively small fraction of inputs to neurons along the hypothesized pathways (da Costa and Martin, 2009). Many studies have demonstrated feed-back effects, in which the activation or deactivation of a higher area alters activity in lower sensory areas (Lamme et al., 1998; Bullier, 2001; Murray et al., 2002), e.g., activation of V1 through a high-level cognitive process like visual imagery (Kosslyn et al., 1999).

DBN models can play an important role in capturing many of those effects, and the event-based framework presented in this article provides a model in which the dynamics and short-term memory properties of spiking neurons can be exploited for dealing with realistic input sequences, in our case coming from bio-inspired sensors. There are still plenty of open research questions, in particular concerning the integration of spike-timing based learning in the DBN framework, and the exploitation of spike-timing for dealing with sequences of inputs. This will likely require an adaptation of the simple RBM model used as the building block of DBNs, and will have to include recurrent lateral connections. Similar mechanisms for the processing of input sequences have been proposed in the framework of Hierarchical Temporal Memory (Hawkins and Blakeslee, 2004), which opens up new directions for combining machine learning approaches with cortical modeling.

## ACKNOWLEDGMENTS

This project was partially supported by the FP7 SeeBetter (FP7-ICT-2009-6), Swiss National Foundation EARS (200021_126844), and the Samsung Advanced Institute of Technology. Michael Pfeiffer has been supported by a Forschungskredit grant of the University of Zurich. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2013.00178/abstract

## REFERENCES


Acero, A., Bernstein, N., Chambers, R., Ju, Y.-C., Li, X., Odell, J., et al. (2008). "Live search for mobile: web services by voice on the cellphone," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (IEEE), (Las Vegas, NV), 5256–5259.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems 19*. Vancouver: MIT Press.

Bullier, J. (2001). Integrated model of visual processing. *Brain Res. Rev.* 36, 96–107. doi: 10.1016/S0165-0173(01)00085-6

Büsing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol.* 7:e1002211. doi: 10.1371/journal.pcbi.1002211

Camuñas Mesa, L., Pérez-Carrasco, J., Zamarreño Ramos, C., Serrano-Gotarredona, T., and Linares-Barranco, B. (2010). "On scalable spiking ConvNet hardware for cortex-like visual sensory processing systems," in *Processing of IEEE International Symposium on Circuits and Systems (ISCAS)*, (Paris), 249–252.

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Comput.* 22, 3207–3220. doi: 10.1162/NECO_a_00052

da Costa, N. M., and Martin, K. A. C. (2009). The proportion of synapses formed by the axons of the lateral geniculate nucleus in layer 4 of area 17 of the cat. *J. Comp. Neurol.* 516, 264–276. doi: 10.1002/cne.22133

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans. Audio Speech Lang. Process.* 20, 30–42. doi: 10.1109/TASL.2011.2134090

Delbruck, T. (2013). *jAER Open Source Project*. Available online at: http://sourceforge.net/apps/trac/jaer/wiki.

Desimone, R., Albright, T. D., Gross, C. G., and Bruce, C. (1984). Stimulus-selective properties of inferior temporal neurons in the macaque. *J. Neurosci.* 4, 2051–2062. doi: 10.1007/s12021-011-9106-1

Douglas, R. J., and Martin, K. A. C. (2011). What's black and white about the grey matter? *Neuroinformatics* 9, 167–179. doi: 10.1007/s12021-011-9106-1

Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.* 11, 625–660.

Falchier, A., Clavagnier, S., Barone, P., and Kennedy, H. (2002). Anatomical evidence of multimodal integration in primate striate cortex. *J. Neurosci.* 22, 5749–5759.

Farabet, C., Paz, R., Pérez-Carrasco, J., Zamarreño, C., Linares-Barranco, A., LeCun, Y., et al. (2012). Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel ConvNets for visual processing. *Front. Neurosci.* 6:32. doi: 10.3389/fnins.2012.00032

Felleman, D. J., and Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex* 1, 1–47. doi: 10.1093/cercor/1.1.1

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nat. Rev. Neurosci.* 11, 127–138. doi: 10.1038/nrn2787

Gerstner, W., and Kistler, W. (2002). *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511815706

Goh, H., Thome, N., and Cord, M. (2010). "Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity," in *NIPS workshop on deep learning and unsupervised feature learning*, (Whistler, BC).

Gross, C. G., Roche-Miranda, G. E., and Bender, D. B. (1972). Visual properties of neurons in the inferotemporal cortex of the macaque. *J. Neurophysiol.* 35, 96–111.

Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Scoffier, M., Kavukcuoglu, K., et al. (2009). Learning long-range vision for autonomous off-road driving. *J. Field Robot.* 26, 120–144. doi: 10.1002/rob.20276

Hawkins, J., and Blakeslee, S. (2004). *On intelligence*. New York, NY: Times Books.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Process. Mag.* 29, 82–97. doi: 10.1109/MSP.2012.2205597

Hinton, G., Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 1527–1554. doi: 10.1162/neco.2006.18.7.1527

Hinton, G. E., and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science* 313, 504–507. doi: 10.1126/science.1127647

Hinton, G. E., and Sejnowski, T. J. (1986). *Learning and Relearning in Boltzmann Machines*. Cambridge, MA: MIT Press 1, 282–317.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., and Elvezia, C. (2001). "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. eds S. C. Kremer and J. F. Kolen (New York, NY: IEEE Press), 237–244.

Hochstein, S., and Ahissar, M. (2002). View from the top: hierarchies and reverse hierarchies review. *Neuron* 36, 791–804. doi: 10.1016/S0896-6273(02)01091-7

Indiveri, G., Linares-Barranco, B., Hamilton, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Jug, F., Cook, M., and Steger, A. (2012). "Recurrent competitive networks can learn locally excitatory topologies," in *International Joint Conference on Neural Networks (IJCNN)*, (Brisbane), 1–8.

Kayser, C., and Logothetis, N. K. (2007). Do early sensory cortices integrate cross-modal information? *Brain Struct. Funct.* 212, 121–132. doi: 10.1007/s00429-007-0154-0

Kersten, D., and Yuille, A. (2003). Bayesian models of object perception. *Curr. Opin. Neurobiol.* 13, 150–158. doi: 10.1016/S0959-4388(03)00042-4

Knill, D. C., and Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends Neurosci.* 27, 712–719. doi: 10.1016/j.tins.2004.10.007

Kosslyn, S. M., Pascual-Leone, A., Felician, O., Camposano, S., Keenan, J. P., Thompson, W. L., et al. (1999). The role of Area 17 in visual imagery: Convergent evidence from PET and rTMS. *Science* 284, 167–170. doi: 10.1126/science.284.5411.167

Lamme, V. A. F., Supér, H., and Spekreijse, H. (1998). Feedforward, horizontal, and feedback processing in the visual cortex. *Curr. Opin. Neurobiol.* 8, 529–535. doi: 10.1016/S0959-4388(98)80042-1

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of ICML*, 473–480. ACM.

Le, Q. V., Ranzato, M. A., Monga, R., Devin, M., Chen, K., Corrado, G., et al. (2012). "Building high-level features using large scale unsupervised learning," in *Proceedings of ICML*, (Edinburgh).

LeCun, Y. L., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Lee, H., Ekanadham, C., and Ng, A. (2008). "Sparse deep belief net model for visual area V2," in *Advances in Neural Information Processing Systems*, Vol 20, (Vancouver), 873–880.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of ICML*, (Montreal), 609–616. doi: 10.1145/1553374.1553453

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Liu, S., Van Schaik, A., Minch, B., and Delbruck, T. (2010). "Event-based 64-channel binaural silicon cochlea with Q enhancement mechanisms," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, (Paris), 2027–2030.

Markov, N., and Kennedy, H. (2013). The importance of being hierarchical. *Curr. Opin. Neurobiol.* 23, 187–194. doi: 10.1016/j.conb.2012.12.008

Markov, N. T., Ercsey-Ravasz, M. M., Ribeiro Gomes, A. R., Lamy, C., Magrou, L., Vezoli, J., et al. (2012). A weighted and directed interareal connectivity matrix for macaque cerebral cortex. *Cereb. Cortex* 1–20. doi: 10.1093/cercor/bhs270

Merolla, P., Ursell, T., and Arthur, J. (2010). *The Thermodynamic Temperature of a Rhythmic Spiking Network*. Arxiv preprint arXiv:1009.5473.

MIT Technology Review (2013). 10 breakthrough technologies 2013: Deep learning. Available online at: http://www.technologyreview.com/featuredstory/513696/deep-learning/

Mohamed, A.-R., Dahl, G. E., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Trans. Audio Speech Lang. Process.* 20, 14–22. doi: 10.1109/TASL.2011.2109382

Murray, S. O., Kersten, D., Olshausen, B. A., Schrater, P., and Woods, D. L. (2002). Shape perception reduces activity in human primary visual cortex. *Proc. Natl. Acad. Sci. U.S.A.* 99, 15164–15169. doi: 10.1073/pnas.192579399

Nair, V., and Hinton, G. (2010). "Rectified linear units improve Restricted Boltzmann Machines," in *Proceedings of ICML*, (Haifa), 807–814.

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput. Biol.* 9:e1003037. doi: 10.1371/journal.pcbi.1003037

Pérez-Carrasco, J., Zhao, B., Serrano, C., Acha, B., Serrano-Gotarredona, T., Chen, S., et al. (2013). Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate-coding and coincidence processing. application to feed forward ConvNets. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 2706–2719. doi: 10.1109/TPAMI.2013.71

Rao, R. P. N., and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nat. Neurosci.* 2, 79–87. doi: 10.1038/4580

Riesenhuber, M., and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat. Neurosci.* 2, 1019–1025. doi: 10.1038/14819

Seide, F., Li, G., and Yu, D. (2011). "Conversational speech transcription using context-dependent deep neural networks," in *Proceedings of Interspeech*, (Florence), 437–440.

Siegert, A. J. F. (1951). On the first passage time probability problem. *Phys. Rev.* 81:617. doi: 10.1103/PhysRev.81.617

Stein, B. E., and Stanford, T. R. (2008). Multisensory integration: current issues from the perspective of the single neuron. *Nat. Rev. Neurosci.* 9, 255–266. doi: 10.1038/nrn2331

Taylor, G., Hinton, G., and Roweis, S. (2007). "Modeling human motion using binary latent variables," in *Advances in Neural Information Processing Systems*, (Vancouver), 1345–1352.

Tieleman, T. (2008). "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proceedings of ICML*, (Helsinki: ACM), 1064–1071.

Tieleman, T., and Hinton, G. (2009). "Using fast weights to improve persistent contrastive divergence," in *Proceedings of ICML*, (Montreal: ACM), 1033–1040.

Van Essen, D. C., and Maunsell, J. H. R. (1983). Hierarchical organization and functional streams in the visual cortex. *Trends Neurosci.* 6, 370–375. doi: 10.1016/0166-2236(83)90167-4

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 12 June 2013; accepted: 17 September 2013; published online: 08 October 2013.*

*Citation: O'Connor P, Neil D, Liu SC, Delbruck T and Pfeiffer M (2013) Real-time classification and sensor fusion with a spiking deep belief network. Front. Neurosci. **7**:178. doi: 10.3389/fnins.2013.00178*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 30 January 2014
doi: 10.3389/fnins.2013.00272

# Event-driven contrastive divergence for spiking neuromorphic systems

***Emre Neftci[1]*, Srinjoy Das[1,2], Bruno Pedroni[3], Kenneth Kreutz-Delgado[1,2] and Gert Cauwenberghs[1,3]***

[1] *Institute for Neural Computation, University of California, San Diego, La Jolla, CA, USA*
[2] *Electrical and Computer Engineering Department, University of California, San Diego, La Jolla, CA, USA*
[3] *Department of Bioengineering, University of California, San Diego, La Jolla, CA, USA*

***Edited by:***
*André Van Schaik, The University of Western Sydney, Australia*

***Reviewed by:***
*Michael Schmuker, Freie Universität Berlin, Germany*
*Philip De Chazal, University of Western Sydney, Australia*

****Correspondence:***
*Emre Neftci, Institute for Neural Computation, University of California, San Diego, 9500 Gilman Drive - 0523, La Jolla, CA-92093, USA*
*e-mail: nemre@ucsd.edu*

Restricted Boltzmann Machines (RBMs) and Deep Belief Networks have been demonstrated to perform efficiently in a variety of applications, such as dimensionality reduction, feature learning, and classification. Their implementation on neuromorphic hardware platforms emulating large-scale networks of spiking neurons can have significant advantages from the perspectives of scalability, power dissipation and real-time interfacing with the environment. However, the traditional RBM architecture and the commonly used training algorithm known as Contrastive Divergence (CD) are based on discrete updates and exact arithmetics which do not directly map onto a dynamical neural substrate. Here, we present an event-driven variation of CD to train a RBM constructed with Integrate & Fire (I&F) neurons, that is constrained by the limitations of existing and near future neuromorphic hardware platforms. Our strategy is based on neural sampling, which allows us to synthesize a spiking neural network that samples from a target Boltzmann distribution. The recurrent activity of the network replaces the discrete steps of the CD algorithm, while Spike Time Dependent Plasticity (STDP) carries out the weight updates in an online, asynchronous fashion. We demonstrate our approach by training an RBM composed of leaky I&F neurons with STDP synapses to learn a generative model of the MNIST hand-written digit dataset, and by testing it in recognition, generation and cue integration tasks. Our results contribute to a machine learning-driven approach for synthesizing networks of spiking neurons capable of carrying out practical, high-level functionality.

**Keywords: synaptic plasticity, neuromorphic cognition, Markov chain monte carlo, recurrent neural network, generative model**

## 1. INTRODUCTION

Machine learning algorithms based on stochastic neural network models such as RBMs and deep networks are currently the state-of-the-art in several practical tasks (Hinton and Salakhutdinov, 2006; Bengio, 2009). The training of these models requires significant computational resources, and is often carried out using power-hungry hardware such as large clusters (Le et al., 2011) or graphics processing units (Bergstra et al., 2010). Their implementation in dedicated hardware platforms can therefore be very appealing from the perspectives of power dissipation and of scalability.

Neuromorphic Very Large Scale Integration (VLSI) systems exploit the physics of the device to emulate very densely the performance of biological neurons in a real-time fashion, while dissipating very low power (Mead, 1989; Indiveri et al., 2011). The distributed structure of RBMs suggests that neuromorphic VLSI circuits and systems can become ideal candidates for such a platform. Furthermore, the communication between neuromorphic components is often mediated using asynchronous address-events (Deiss et al., 1998) enabling them to be interfaced with event-based sensors (Liu and Delbruck, 2010; Neftci et al., 2013; O'Connor et al., 2013) for embedded applications, and to be implemented in a very scalable fashion (Silver et al., 2007; Joshi et al., 2010; Schemmel et al., 2010).

Currently, RBMs and the algorithms used to train them are designed to operate efficiently on digital processors, using batch, discrete-time, iterative updates based on exact arithmetic calculations. However, unlike digital processors, neuromorphic systems compute through the continuous-time dynamics of their components, which are typically Integrate & Fire (I&F) neurons (Indiveri et al., 2011), rendering the transfer of such algorithms on such platforms a non-trivial task. We propose here a method to construct RBMs using I&F neuron models and to train them using an online, event-driven adaptation of the Contrastive Divergence (CD) algorithm.

We take inspiration from computational neuroscience to identify an efficient neural mechanism for sampling from the underlying probability distribution of the RBM. Neuroscientists argue that brains deal with uncertainty in their environments by encoding and combining probabilities optimally (Doya et al., 2006), and that such computations are at the core of cognitive function (Griffiths et al., 2010). While many mechanistic theories of how the brain might achieve this exist, a recent *neural sampling* theory postulates that the spiking activity of the neurons encodes samples of an underlying probability distribution (Fiser et al., 2010). The advantage for a neural substrate in using such a strategy over the alternative one, in which neurons encode probabilities, is that it requires exponentially

fewer neurons. Furthermore, abstract model neurons consistent with the behavior of biological neurons can implement Markov Chain Monte Carlo (MCMC) sampling (Buesing et al., 2011), and RBMs sampled in this way can be efficiently trained using CD, with almost no loss in performance (Pedroni et al., 2013). We identify the conditions under which a dynamical system consisting of I&F neurons performs neural sampling. These conditions are compatible with neuromorphic implementations of I&F neurons (Indiveri et al., 2011), suggesting that they can achieve similar performance. The calibration procedure necessary for configuring the parameters of the spiking neural network is based on firing rate measurements, and so is easy to realize in software and in hardware platforms.

In standard CD, weight updates are computed on the basis of alternating, feed-forward propagation of activities (Hinton, 2002). In a neuromorphic implementation, this translates to reprogramming the network connections and resetting its state variables at every step of the training. As a consequence, it requires two distinct dynamical systems: one for normal operation (i.e., testing), the other for training, which is highly impractical. To overcome this problem, we train the neural RBMs using an online adaptation of CD. We exploit the recurrent structure of the network to mimic the discrete "construction" and "reconstruction" steps of CD in a spike-driven fashion, and Spike Time Dependent Plasticity (STDP) to carry out the weight updates. Each sample (spike) of each random variable (neuron) causes synaptic weights to be updated. We show that, over longer periods, these microscopic updates behave like a macroscopic CD weight update. Compared to standard CD, no additional connectivity programming overhead is required during the training steps, and both testing and training take place in the same dynamical system.

Because RBMs are generative models, they can act simultaneously as classifiers, content-addressable memories, and carry out probabilistic inference. We demonstrate these features in a MNIST hand-written digit task (LeCun et al., 1998), using an RBM network consisting of one layer of 824 "visible" neurons and one layer of 500 "hidden" neurons. The spiking neural network was able to learn a generative model capable of recognition performances with accuracies up to 91.9%, which is close to the performance obtained using standard CD and Gibbs sampling, 93.6%.

## 2. MATERIALS AND METHODS

### 2.1. NEURAL SAMPLING WITH NOISY I&F NEURONS

We describe here conditions under which a dynamical system composed of I&F neurons can perform neural sampling. It has been proven that abstract neuron models consistent with the behavior of biological spiking neurons can perform MCMC sampling of a Boltzmann distribution (Buesing et al., 2011). Two conditions are sufficient for this. First, the instantaneous firing rate of the neuron verifies:

$$\rho(u(t), t - t') = \begin{cases} 0 & \text{if } t - t' < \tau_r \\ r(u(t)) & t - t' \geq \tau_r \end{cases}, \tag{1}$$

with $r(u(t))$ proportional to $\exp(u(t))$, where $u(t)$ is the membrane potential and $\tau_r$ is an absolute refractory period during which the neuron cannot fire. $\rho(u(t), t - t')$ describes the neuron's instantaneous firing rate as a function of $u(t)$ at time $t$, given that the last spike occurred at $t'$. It can be shown that the average firing rate of this neuron model for stationary $u(t)$ is the sigmoid function:

$$\rho(u) = (\tau_r + \exp(-u))^{-1}. \tag{2}$$

Second, the membrane potential of neuron $i$ is equal to the linear sum of its inputs:

$$u_i(t) = b_i + \sum_{j=1}^{N} w_{ij} z_j(t), \forall i = 1, \ldots, N, \tag{3}$$

where $b_i$ is a constant bias, and $z_j(t)$ represents the pre-synaptic spike train produced by neuron $j$ defined as being equal to 1 when the pre-synaptic neuron spikes for a duration $\tau_r$, and equal to zero otherwise. The terms $w_{ij} z_j(t)$ are identified with the time course of the Post–Synaptic Potential (PSP), i.e., the response of the membrane potential to a pre-synaptic spike. The two conditions above define a neuron model, to which we refer as the "abstract neuron model." Assuming the network states are binary vectors $[z_1, \ldots, z_k]$, it can be shown that, after an initial transient, the sequence of network states can be interpreted as MCMC samples of the Boltzmann distribution:

$$\begin{aligned} p(z_1, \ldots, z_k) &= \frac{1}{Z} \exp\big(-E(z_1, \ldots, z_k)\big), \text{ with} \\ E(z_1, \ldots, z_k) &= -\frac{1}{2} \sum_{ij} W_{ij} z_i z_j - \sum_i b_i z_i, \end{aligned} \tag{4}$$

where $Z = \sum_{z_1, \ldots, z_k} \exp\big(-E(z_1, \ldots, z_k)\big)$ is a constant such that $p$ sums up to unity, and $E(z_1, \ldots, z_k)$ can be interpreted as an energy function (Haykin, 1998).

An important fact of the abstract neuron model is that, according to the dynamics of $z_j(t)$, the PSPs are "rectangular" and non-additive since no two presynaptic spikes can occur faster than the refractive period. The implementation of synapses producing such PSPs on a large scale is very difficult to realize in hardware, when compared to first-order linear filters that result in "alpha"-shaped PSPs (Destexhe et al., 1998; Bartolozzi and Indiveri, 2007). This is because, in the latter model, the synaptic dynamics are linear, such that a single hardware synapse can be used to generate the same current that would be generated by an arbitrary number of synapses (see also next section). As a consequence, we will use alpha-shaped PSPs instead of rectangular PSPs in our models. The use of the alpha PSP over the rectangular PSP is the major source of degradation in sampling performance, as we will discuss in section 2.2.

#### 2.1.1. Stochastic I&F neurons

A neuron whose instantaneous firing rate is consistent with Equation (1) can perform neural sampling. Equation (1) is a generalization of the Poisson process to the case when the firing probability depends on the time of the last spike (i.e., it is a renewal

process), and so can be verified only if the neuron fires stochastically (Cox, 1962). Stochasticity in I&F neurons can be obtained through several mechanisms, such as a noisy reset potential, noisy firing threshold, or noise injection (Plesser and Gerstner, 2000). The first two mechanisms necessitate stochasticity in the neuron's parameters, and therefore may require specialized circuitry. But noise injection in the form of background Poisson spike trains requires only synapse circuits, which are present in many neuromorphic VLSI implementation of spiking neurons (Bartolozzi and Indiveri, 2007; Indiveri et al., 2011). Furthermore, Poisson spike trains can be generated self-consistently in balanced excitatory-inhibitory networks (van Vreeswijk and Sompolinsky, 1996), or using finite-size effects and neural mismatch (Amit and Brunel, 1997).

We show that the abstract neuron model in Equation (1) can be realized in a simple dynamical system consisting of leaky I&F neurons with noisy currents. The neuron's membrane potential below firing threshold $\theta$ is governed by the following differential equation:

$$C\frac{\mathrm{d}}{\mathrm{d}t}u_i = -g_L u_i + I_i(t) + \sigma\xi(t), \quad u_i(t) \in (-\infty, \theta), \tag{5}$$

where $C$ is a membrane capacitance, $u_i$ is the membrane potential of neuron $i$, $g_L$ is a leak conductance, $\sigma\xi(t)$ is a white noise term of amplitude $\sigma$ (which can for example be generated by background activity), $I_i(t)$ its synaptic current and $\theta$ is the neuron's firing threshold. When the membrane potential reaches $\theta$, an action potential is elicited. After a spike is generated, the membrane potential is clamped to the reset potential $u_{\mathrm{rst}}$ for a refractory period $\tau_r$.

In the case of the neural RBM, the currents $I_i(t)$ depend on the layer the neuron is situated in. For a neuron $i$ in layer $\upsilon$

$$\begin{aligned} I_i(t) &= I_i^d(t) + I_i^\upsilon(t), \\ \tau_{\mathrm{syn}}\frac{\mathrm{d}}{\mathrm{d}t}I_i^\upsilon &= -I_i^\upsilon + \sum_{j=1}^{N_h} q_{h_{ji}} h_j(t) + q_{b_i} b_{\upsilon_i}(t), \end{aligned} \tag{6}$$

where $I_i^d(t)$ is a current representing the data (i.e., the external input), $I^\upsilon$ is the feedback from the hidden layer activity and the bias, and the $q$'s are the respective synaptic weights, and $b_\upsilon(t)$ is a Poisson spike train implementing the bias. Spike trains are represented by a sum of Dirac delta pulses centered on the respective spike times:

$$b_{\upsilon_i}(t) = \sum_{k \in Sp_i} \delta(t - t_k), \quad h_j(t) = \sum_{k \in Sp_j} \delta(t - t_k) \tag{7}$$

where $Sp_i$ and $Sp_j$ are the set of the spike times of the bias neuron $b_{\upsilon_i}$ and the hidden neuron $h_j$, respectively, and $\delta(t) = 1$ if $t = 0$ and 0 otherwise.

For a neuron $j$ in layer $h$,

$$\begin{aligned} I_j(t) &= I_j^h(t), \\ \tau_{\mathrm{syn}}\frac{\mathrm{d}}{\mathrm{d}t}I_j^h &= -I_j^h + \sum_{i=1}^{N_\upsilon} q_{\upsilon_{ij}} \upsilon_i(t) + q_{b_j} b_{h_j}(t), \end{aligned} \tag{8}$$

where $I^h$ is the feedback from the visible layer, and $\upsilon(t)$ and $b_h(t)$ are Poisson spike trains of the visible neurons and the bias neurons, defined similarly as in Equation (7). The dynamics of $I^h$ and $I^\upsilon$ correspond to a first-order linear filter, so each incoming spike results in PSPs that rise and decay exponentially (i.e., alpha-PSP) (Gerstner and Kistler, 2002).

Can this neuron verify the conditions required for neural sampling? The membrane potential is already assumed to be equal to the sum of the PSPs as required by neural sampling. So to answer the above question we only need to verify whether Equation (1) holds. Equation (5) is a Langevin equation which can be analyzed using the Fokker–Planck equation (Gardiner, 2012). The solution to this equation provides the neuron's input/output response, i.e., its transfer curve (for a review, see Renart et al., 2003):

$$\rho(u_0) = \left(\tau_r + \tau_m\sqrt{\pi}\int_{\frac{u_{\mathrm{rst}}-u_0}{\sigma_V}}^{\frac{\theta-u_0}{\sigma_V}} \mathrm{d}x \exp(x^2)(1 + \mathrm{erf}(x))\right)^{-1}, \tag{9}$$

where erf is the error function (the integral of the normal distribution), $u_0 = \frac{I}{g_L}$ is the stationary value of the membrane potential when injected with a constant current $I$, $\tau_m = \frac{C}{g_L}$ is the membrane time constant, $u_{\mathrm{rst}}$ is the reset voltage, and $\sigma_V^2(u) = \sigma^2/(g_L C)$.

According to Equation (2), the condition for neural sampling requires that the average firing rate of the neuron to be the sigmoid function. Although the transfer curve of the noisy I&F neuron Equation (9) is not identical to the sigmoid function, it was previously shown that with an appropriate choice of parameters, the shape of this curve can be very similar to it (Merolla et al., 2010). We observe that, for a given refractory period $\tau_r$, the smaller the ratio $\frac{\theta - u_{\mathrm{rst}}}{\sigma_V}$ in Equation (5), the better the transfer curve resembles a sigmoid function (**Figure 1**). With a small $\frac{\theta - u_{\mathrm{rst}}}{\sigma_V}$, the transfer function of a neuron can be fitted to

$$\nu(I) = \frac{1}{\tau_r}\left(1 + \frac{\exp(-I\beta)}{\gamma\tau_r}\right)^{-1}, \tag{10}$$

where $\beta$ and $\gamma$ are the parameters to be fitted. The choice of the neuron model described in Equation (5) is not critical for neural sampling: A relationship that is qualitatively similar to Equation (9) holds for neurons with a rigid (reflective) lower boundary (Fusi and Mattia, 1999) which is common in VLSI neurons, and for I&F neurons with conductance-based synapses (Petrovici et al., 2013).

This result also shows that synaptic weights $q_{\upsilon_i}$, $q_{h_j}$, which have the units of charge are related to the RBM weights $W_{ij}$ by a factor $\beta^{-1}$. To relate the neural activity to the Boltzmann distribution, Equation (4), each neuron is associated to a binary random variable which is assumed to take the value 1 for a duration $\tau_r$ after the

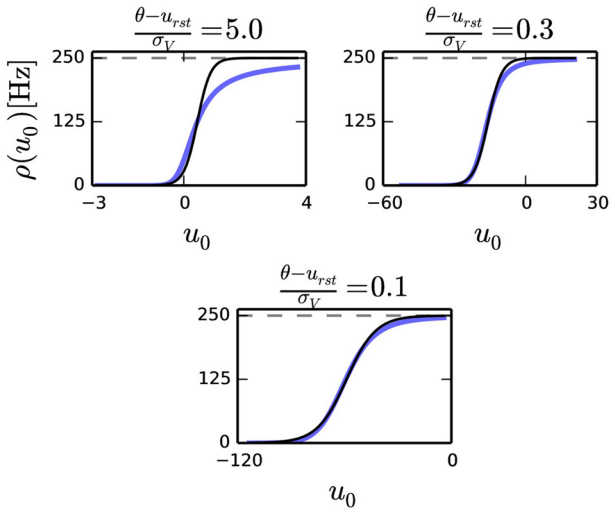



**FIGURE 1 | Transfer curve of a leaky I&F neuron for three different parameter sets where $u_0 = \frac{I}{g_L}$, and $\frac{1}{\tau_r} = 250$ [Hz] (dashed gray).** In this plot, $\sigma_V$ is varied to produce different ratios $\frac{\theta - u_{\rm rst}}{\sigma_V}$. The three plots above shows that the fit with the sigmoid function (solid black) improves as the ratio decreases.

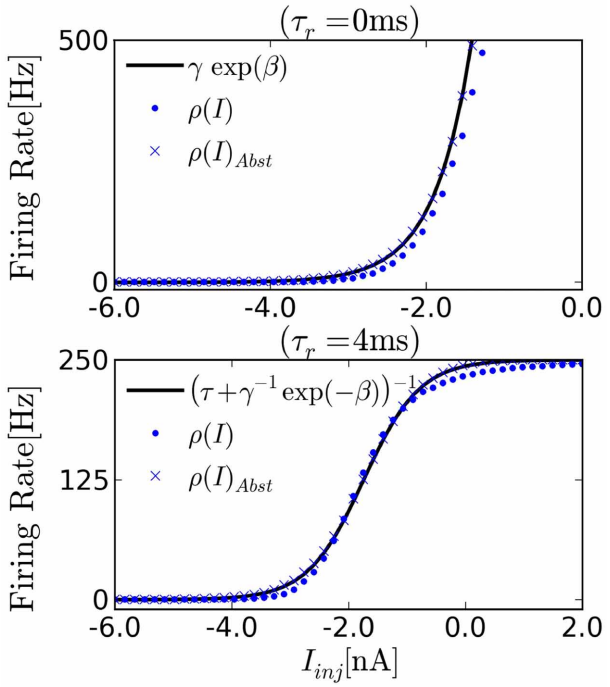



**FIGURE 2 | Transfer function of I&F neurons driven by background white noise Equation (5).** We measure the firing rate of the neuron as a function of a constant current injection to estimate $\rho(u_0)$, where for constant $I_{\rm inj}$, $u_0 = I_{\rm inj}/g_L$. (Top) The transfer function of noisy I&F neurons in the absence of refractory period [$\rho(u) = r(u)$, circles]. We observe that $\rho$ is approximately exponential over a wide range of inputs, and therefore compatible with neural sampling. Crosses show the transfer curve of neurons implementing the abstract neuron Equation (1), exactly. (Bottom) With an absolute refractory period the transfer function approximates the sigmoid function. The firing rate saturates at [250]Hz due to the refractory period chosen for the neuron.

neuron has spiked, and zero otherwise, similarly to Buesing et al. (2011). With this encoding, the network state is characterized by a binary vector having the same number of entries as the number of neurons in the network. The relationship between this random vector and the I&F neurons' spiking activity is illustrated in **Figure 3**. The membrane potential of the neuron (black) evolves in a random fashion until it spikes, after which it is clamped to $u_{\rm rst}$ for a duration $\tau_r$ (gray). While the neuron is in the refractory period, the random variable associated to it is assumed to takes the value 1. This way, the state of the network can always be associated with a binary vector. According to the theory, the dynamics in the network guarantees that the binary vectors are samples drawn from a Boltzmann distribution.

#### *2.1.2. Calibration protocol*

In order to transfer the parameters from the probability distribution Equation (4) to those of the I&F neurons, the parameters $\gamma$, $\beta$ in Equation (10) need to be fitted. An estimate of a neuron's transfer function can be obtained by computing its spike rate when injected with different values of constant inputs $I$. The refractory period $\tau_r$ is the inverse of the maximum firing rate of the neuron, so it can be easily measured by measuring the spike rate for very high input current $I$. Once $\tau_r$ is known, the parameter estimation can be cast into a simple linear regression problem by fitting $\log(\rho(i)^{-1} - \tau_r)$ with $\beta I + \log(\gamma)$. **Figure 2** shows the transfer curve when $\tau_r = 0$ ms, which is approximately exponential in agreement with Equation (1).

The shape of the transfer curse is strongly dependent on the noise amplitude. In the absence of noise, the transfer curve is a sharp threshold function, which softens as the amplitude of the noise is increased (**Figure 1**). As a result, both parameters $\gamma$ and $\beta$ are dependent on the variance of the input currents from other neurons $I(t)$. Since $\beta q = w$, the effect of the fluctuations on the network is similar to scaling the synaptic weights and the biases which can be problematic. However, by selecting a large enough noise amplitude $\sigma$ and a slow enough input synapse time constant, the fluctuations due to the background input are much larger than the fluctuations due to the inputs. In this case, $\beta$ and $\gamma$ remain approximately constant during the sampling.

Neural mismatch can cause $\beta$ and $\gamma$ to differ from neuron to neuron. From Equation (10) and the linearity of the postsynaptic currents $I(t)$ in the weights, it is clear that this type of mismatch can be compensated by scaling the synaptic weights and biases accordingly. The calibration of the parameters $\gamma$ and $\beta$ quantitatively relate the spiking neural network's parameters to the RBM. In practice, this calibration step is only necessary for mapping pre-trained parameters of the RBM onto the spiking neural network.

Although we estimated the parameters of software simulated I&F neurons, parameter estimation based on firing rate measurements were shown to be an accurate and reliable method for VLSI I&F neurons as well (Neftci et al., 2012).

### 2.2. VALIDATION OF NEURAL SAMPLING USING I&F NEURONS

The I&F neuron verifies Equation (1) only approximately, and the PSP model is different from the one of Equation (3). Therefore,

the following two important questions naturally arise: how accurately does the I&F neuron-based sampler outlined above sample from a target Boltzmann distribution? How well does it perform in comparison to an exact sampler, such as the Gibbs sampler? To answer these questions we sample from several neural RBM consisting of five visible and five hidden units for randomly drawn weight and bias parameters. At these small dimensions, the probabilities associated to all possible values of the random vector $\mathbf{z}$ can be computed exactly. These probabilities are then compared to those obtained through the histogram constructed with the sampled events. To construct this histogram, each spike was extended to form a box of length $\tau_r$ (as illustrated in **Figure 3**), the spiking activity was sampled at 1 kHz, and the occurrences of all the possible $2^{10}$ states of the random vector $\mathbf{z}$ were counted. We added 1 to the number of occurrences of each state to avoid zero probabilities. The histogram obtained from a representative run is shown in **Figure 4** (left).

A common measure of similarity between two distributions $p$ and $q$ is the KL divergence:

$$D(p||q) = \sum_i p_i \log \frac{p_i}{q_i}.$$

If the distributions $p$ and $q$ are identical then $D(p||q) = 0$, otherwise $D(p||q) > 0$. The right panel of **Figure 4** shows $D(p||P_{\text{exact}})$ as a function of sampling duration, for distributions $p$ obtained from three different samplers: the abstract neuron based sampler with alpha PSPs ($P_{\text{NS,Abstract}}$), the I&F neuron-based sampler ($P_{\text{NS}}$), and the Gibbs sampler ($P_{\text{Gibbs}}$).

In the case of the I&F neuron-based sampler, the average KL divergence for 48 randomly drawn distributions after 1000 s of sampling time was $0.059 \pm 0.049$. This result is not significantly different if the abstract neuron model Equation (1) with alpha PSPs is used (average KL divergence $0.10 \pm 0.049$), and in both cases the KL divergence did not tend to zero as the number of samples increased. The only difference in the latter neuron model compared to the abstract neuron model of Buesing et al. (2011), which tends to zero when sampling time tends to infinity, is the PSP model. This indicates that the discrepancy is largely due to the use of alpha-PSPs, rather than the approximation of Equation (1) with I&F neurons.

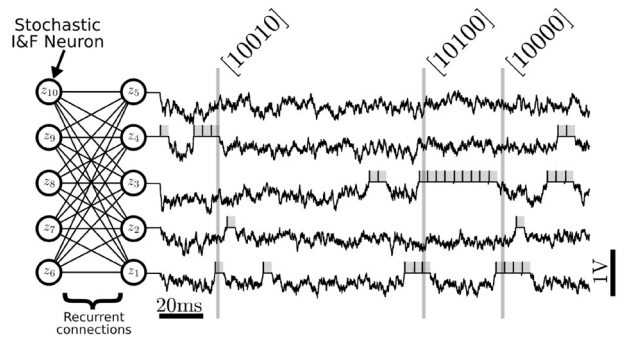



**FIGURE 3 | Neural Sampling in an RBM consisting of 10 stochastic I&F neurons, with five neurons in each layer.** Each neuron is associated to a binary random variable which take values 1 during a refractory period $\tau_r$ after the neuron has spiked (gray shadings). The variables are sampled at 1 kHz to produce binary vectors that correspond to samples of the joint distribution $p(z)$. In this figure, only the membrane potential and the samples produced by the first five neurons are shown. The vectors inside the brackets are example samples of the marginalized distribution $p(z_1, z_2, z_3, z_4, z_5)$ produced at the time indicated by the vertical lines. In the RBM, there are no recurrent connections within a layer.

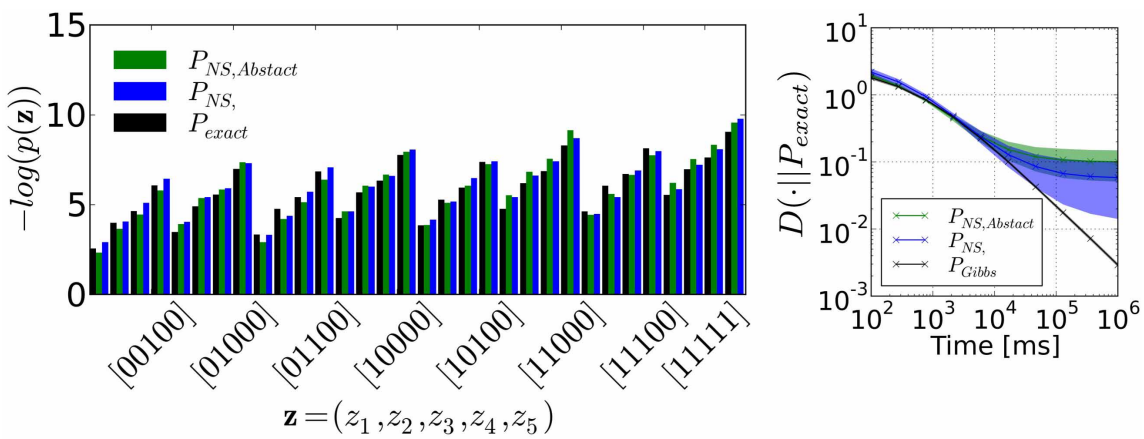



**FIGURE 4 | (Left)** Example probability distribution obtained by neural sampling of the RBM of **Figure 3**. The bars are marginal probabilities computed by counting the events [00000], [00001], . . . , [11110], [11111], respectively. $P_{\text{NS}}$ is the distribution obtained by neural sampling and $P$ is the exact probability distribution computed with Equation (4). **(Right)** The degree to which the sampled distribution resembles the target distribution is quantified by the KL divergence measured across 48 different distributions, and the shadings correspond to its standard deviation. This plot also shows the KL divergence of the target distribution sampled by Gibbs Sampling ($P_{\text{Gibbs}}$), which is the common choice for RBMs. For comparison with the neural sampler, we identified the duration of one Gibbs sampling iteration with one refractory period $\tau_r = 4$ ms. The plot shows that up to $10^4$ms, the two methods are comparable. After this, the KL divergence of the neural sampler tends to a plateau due to the fact that neural sampling with our I&F neural network is approximate. In both figures, $P_{\text{NS, Abstract}}$ refers to the marginal probability distribution obtained by using the abstract neuron model Equation (1). In this case, the KL divergence is not significantly different from the one obtained with the I&F neuron model-based sampler.

The standard sampling procedure used in RBMs is Gibbs Sampling: the neurons in the visible layer are sampled simultaneously given the activities of the hidden neurons, then the hidden neurons are sampled given the activities of the visible neurons. This procedure is iterated a number of times. For comparison with the neural sampler, the duration of one Gibbs sampling iteration is identified with one refractory period $\tau_r = 4$ ms. At this scale, we observe that the speed of convergence of the neural sampler is similar to that of the Gibbs sampler up to $10^4$ms, after which the neural sampler plateaus above the $D(p||q) = 10^{-2}$ line. Despite the approximations in the neuron model and the synapse model, these results show that in RBMs of this size, the neural sampler consisting of I&F neurons sample from a distribution that has the same KL divergence as the distribution obtained after $10^4$ iterations of Gibbs sampling, which is more than the typical number of iterations used for MNIST hand-written digit tasks in the literature (Hinton et al., 2006).

### 2.3. NEURAL ARCHITECTURE FOR LEARNING A MODEL OF MNIST HAND-WRITTEN DIGITS

We test the performance of the neural RBM in a digit recognition task. We use the MNIST database, whose data samples consist of centered, gray-scale, $28 \times 28$-pixel images of hand-written digits 0–9 (LeCun et al., 1998). The neural RBM's network architecture consisted of two layers, as illustrated in **Figure 5**. The visible layer was partitioned into 784 sensory neurons ($\nu_d$) and 40 class label neurons ($\nu_c$) for supervised learning. The pixel values of the digits were discretized to two values, with low intensity pixel values ($p \leq 0.5$) mapped to $10^{-5}$ and high intensity values ($p > 0.5$) mapped to 0.98. A neuron $i$ in $d$ stimulated each neuron $i$ in layer $\nu$, with synaptic currents $f_i$ such that $P(\upsilon_i = 1) = \nu(f_i)\tau_r = p_i$, where $0 \leq p_i \leq 1$ is the value of pixel $i$. The value $f_i$ is calculated by inverting the transfer function of the neuron: $f_i = \nu^{-1}(s) = \log\left(\frac{s}{\gamma - s\gamma\tau_r}\right)\beta^{-1}$. Using this RBM, classification is performed by choosing the most likely label given the input, under the learned model. This equals to choosing the population of class neurons associated to the same label that has the highest population firing rate.

To reconstruct a digit from a class label, the class neurons belonging to a given digit are clamped to a high firing rate. For testing the discrimination performance of an energy-based model such as the RBM, it is common to compute the free-energy $F(\mathbf{v_c})$ of the class units (Haykin, 1998), defined as:

$$\exp(-F(v_c)) = \sum_{v_d, h} \exp(-E(v_d, v_c, h)), \tag{11}$$

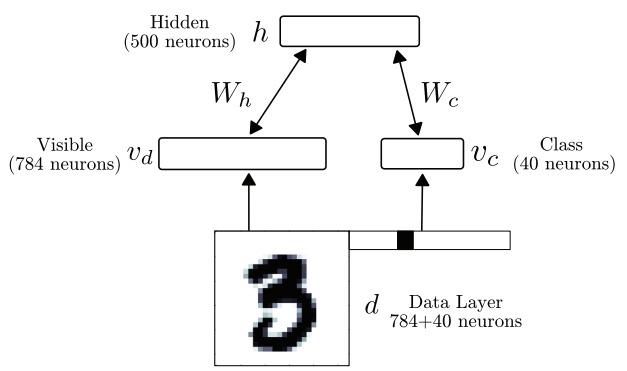



**FIGURE 5 | The RBM network consists of a visible and a hidden layer.** The visible layer is partitioned into 784 sensory neurons ($\nu_d$) and 40 class label neurons ($\nu_c$) for supervised learning. During data presentation, the activities in the visible layer are driven by a data layer d, consisting of a digit and its label (1 neuron per label). In the RBM, the weight matrix between the visible layer and the hidden layer is symmetric.

**Table 1 | List of parameters used in the software simulations.**[a]

| | | | |
|---|---|---|---|
| $\nu_{\text{bias}}$ | Mean firing rate of bias Poisson spike train | All figures | 1000 Hz |
| $\sigma$ | Noise amplitude | All figures, except **Figure 1** | $3 \cdot 10^{-11}$ A/s$^{0.5}$ |
| | | **Figure 1** (left) | $2 \cdot 10^{-11}$ A/s$^{0.5}$ |
| | | **Figure 1** (right) | $3 \cdot 10^{-10}$ A/s$^{0.5}$ |
| | | **Figure 1** (bottom) | $1 \cdot 10^{-9}$ A/s$^{0.5}$ |
| $\beta$ | Exponential factor (fit) | All figures | $2.044 \cdot 10^9$ A$^{-1}$ |
| $\gamma$ | Baseline firing rate (fit) | All figures | 8808 Hz |
| $\tau_r$ | Refractory period | All figures | 4 ms |
| $\tau_{\text{syn}}$ | Time constant of recurrent, and bias synapses | All figures | 4 ms |
| $\tau_{\text{br}}$ | "Burn-in" time of the neural sampling | All figures | 10 ms |
| $g_L$ | Leak conductance | All figures | 1 nS |
| $u_{\text{rst}}$ | Reset potential | All figures | 0 V |
| $C$ | Membrane capacitance | All figures | $10^{-12}$ F |
| $\theta$ | Firing threshold | All figures | 100 mV |
| $W$ | RBM weight matrix ($\in \mathbb{R}^{N_\upsilon \times N_h}$) | **Figure 4** | $N(-0.75, 1.5)$ |
| $b_\upsilon$, $b_h$ | RBM bias for layer $\upsilon$ and $h$ | **Figure 4** | $N(-1.5, 0.5)$ |
| $N_\upsilon$, $N_h$ | Number of visible and hidden units | **Figure 4** | 5, 5 |
| | in the RBM | **Figures 7**, **8**, **7** | 824, 500 |
| $N_c$ | Number of class label units | **Figures 7**, **8**, **7** | 40 |
| $2T$ | Epoch duration | **Figures 4**, **7**, **8** | 100 ms |
| | | **Figure 9** | 300 ms |
| $T_{\text{sim}}$ | Simulation time | **Figure 2** | 5 s |
| | | **Figure 4** | 1000 s |
| | | **Figure 7** | 0.2 s |
| | | **Figure 9** | 0.85 s |
| | | **Figure 8** (testing) | 1.0 s |
| | | **Figure 8** (learning) | 2000 s |
| $\tau_{\text{STDP}}$ | Learning time window | **Figure 7** | 4 ms |
| $\eta$ | Learning rate | Standard CD | $0.1 \cdot 10^{-2}$ |
| | | Event-driven CD | $3.2 \cdot 10^{-2}$ |

[a] *Software simulation scripts are available online (https://github.com/eneftci/eCD).*

and selecting $v_c$ such that the free-energy is minimized. The spiking neural network is simulated using the BRIAN simulator (Goodman and Brette, 2008). All the parameters used in the simulations are provided in **Table 1**.

## 3. RESULTS

### 3.1. EVENT-DRIVEN CONTRASTIVE DIVERGENCE

A Restricted Boltzmann Machine (RBM) is a stochastic neural network consisting of two symmetrically interconnected layers composed of neuron-like units—a set of visible units $v$ and a set of hidden units $h$, but has no connections within a layer.

The training of RBMs commonly proceeds in two phases. At first the states of the visible units are clamped to a given vector from the training set, then the states of the hidden units are sampled. In a second "reconstruction" phase, the network is allowed to run freely. Using the statistics collected during sampling, the weights are updated in a way that they maximize the likelihood of the data (Hinton, 2002). Collecting equilibrium statistics over the data distribution in the reconstruction phase is often computationally prohibitive. The CD algorithm has been proposed to mitigate this (Hinton, 2002; Hinton and Salakhutdinov, 2006): the reconstruction of the visible units' activity is achieved by sampling them conditioned on the values of the hidden units (**Figure 6**). This procedure can be repeated $k$ times (the rule is then called $\mathrm{CD}_k$), but relatively good convergence is obtained for the equilibrium distribution even for one iteration. The CD learning rule is summarized as follows:

$$\Delta w_{ij} = \epsilon(\langle \upsilon_i h_j \rangle_{\mathrm{data}} - \langle \upsilon_i h_j \rangle_{\mathrm{recon}}), \tag{12}$$

where $\upsilon_i$ and $h_j$ are the activities in the visible and hidden layers, respectively. This rule can be interpreted as a difference of Hebbian and anti-Hebbian learning rules between the visible and hidden neurons sampled in the data and reconstruction phases. In practice, when the data set is very large, weight updates are calculated using a subset of data samples, or "mini-batches." The above rule can then be interpreted as a stochastic gradient descent (Robbins and Monro, 1951). Although the convergence properties of the CD rule are the subject of continuing investigation, extensive software simulations show that the rule often converges to very good solutions (Hinton, 2002).

The main result of this paper is an online variation of the CD rule for implementation in neuromorphic hardware. By virtue of neural sampling the spikes generated from the visible and hidden units can be used to compute the statistics of the probability distributions online (further details on neural sampling in the Materials and Methods section 2.1). Therefore a possible neural mechanism for implementing CD is to use synapses whose weights are governed by synaptic plasticity. Because the spikes cause the weight to update in an online, and asynchronous fashion, we refer to this rule as *event-driven* CD.

The weight update in event-driven CD is a modulated, pair-based STDP rule:

$$\frac{\mathrm{d}}{\mathrm{d}t} q_{ij} = g(t)\,\mathrm{STDP}_{ij}(\upsilon_i(t), h_j(t)) \tag{13}$$

where $g(t) \in \mathbb{R}$ is a zero-mean global gating signal controlling the data vs. reconstruction phase, $q_{ij}$ is the weight of the synapse and $\upsilon_i(t)$ and $h_j(t)$ refer to the spike trains of neurons $\upsilon_i$ and $h_j$, defined as in Equation (7).

As opposed to the standard CD rule, weights are updated after every occurrence of a pre-synaptic and post-synaptic event. While this online approach slightly differentiates it from standard CD, it is integral to a spiking neuromorphic framework where the data samples and weight updates cannot be stored. The weight update is governed by a symmetric STDP rule with a symmetric temporal window $K(t) = K(-t), \forall t$:

$$\begin{aligned} \mathrm{STDP}_{ij}(\upsilon_i(t), h_j(t)) &= \upsilon_i(t) A_{h_j}(t) + h_j(t) A_{\upsilon_i}(t), \\ A_{h_j}(t) &= A \int_{-\infty}^{t} \mathrm{d}s K(t-s) h_j(s), \\ A_{\upsilon_i}(t) &= A \int_{-\infty}^{t} \mathrm{d}s K(s-t) \upsilon_i(s), \end{aligned} \tag{14}$$

with $A > 0$ defining the magnitude of the weight updates. In our implementation, updates are additive and weights can change polarity.

#### 3.1.1. Pairwise STDP with a global modulatory signal approximates CD

The modulatory signal $g(t)$ switches the behavior of the synapse from LTP to LTD (i.e., Hebbian to Anti-Hebbian). The temporal average of $g(t)$ must vanish to balance LTP and LTD, and must

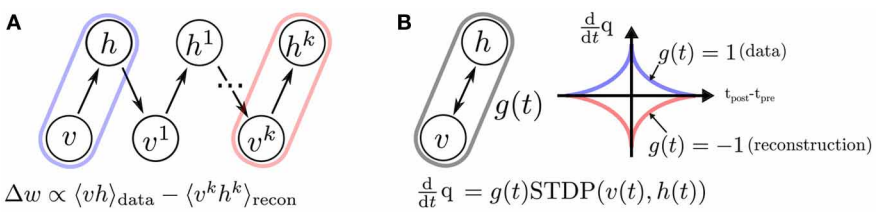



**FIGURE 6 | The standard Contrastive Divergence $(\mathrm{CD})_k$ procedure, compared to event-driven CD. (A)** In standard CD, learning proceeds iteratively by sampling in "construction" and "reconstruction" phases (Hinton, 2002), which is impractical in a continuous-time dynamical system. **(B)** We propose a spiking neural sampling architecture that folds these updates on a continuous time dimension through the recurrent activity of the network. The synaptic weight update follows a STDP rule modulated by a zero mean signal $g(t)$. This signal switches the behavior of the synapse from Long-Term Potentiation (LTP) to Long-Term Depression (LTD), and partitions the training into two phases analogous to those of the original CD rule. The spikes cause microscopic weight modifications, which on average behave as the macroscopic CD weight update. For this reason, the learning rule is referred to as event-driven CD.

vary on much slower time scales than the typical times scale of the network dynamics, denoted $\tau_{\text{br}}$, so that the network samples from its stationary distribution when the weights are updated. The time constant $\tau_{\text{br}}$ corresponds to a "burn-in" time of MCMC sampling and depends on the overall network dynamics and cannot be computed in the general case. However, it is reasonable to assume $\tau_{\text{br}}$ to be in the order of a few refractory periods of the neurons (Buesing et al., 2011). In this work, we used the following modulation function $g(t)$:

$$g(t) = \begin{cases} 1 & \text{if mod}(t, 2T) \in (\tau_{\text{br}}, T) \\ -1 & \text{if mod}(t, 2T) \in (T + \tau_{\text{br}}, 2T) \\ 0 & \text{otherwise} \end{cases}, \tag{15}$$

where mod is the modulo function and $T$ is a time interval. The data is presented during the time intervals $(2iT, (2i+1)T)$, where $i$ is a positive integer. With the $g(t)$ defined above, no weight update is undertaken during a fixed period $\tau_{\text{br}}$. This allows us to neglect the transients after the stimulus is turned on and off (respectively in the beginning of the data and reconstruction phases). In this case and under further assumptions discussed below, the event-driven CD rule can be directly compared with standard CD as we now demonstrate. The average weight update during $(0, 2T)$ is:

$$\begin{aligned} \left\langle \frac{\mathrm{d}}{\mathrm{d}t} q_{ij} \right\rangle_{(0,2T)} &= C_{ij} + R_{ij}, \\ C_{ij} &= \frac{T - \tau_{\text{br}}}{2T} (\langle \upsilon_i(t) A_{h_j}(t) \rangle_{t_d} + \langle h_j(t) A_{\upsilon_i}(t) \rangle_{t_d}) \\ R_{ij} &= -\frac{T - \tau_{\text{br}}}{2T} (\langle \upsilon_i(t) A_{h_j}(t) \rangle_{t_r} + \langle h_j(t) A_{\upsilon_i}(t) \rangle_{t_r}), \end{aligned} \tag{16}$$

where $t_d = (\tau_{\text{br}}, T)$ and $t_r = (T + \tau_{\text{br}}, 2T)$ denote the intervals during the positive and negative phases of $g(t)$, and $\langle \cdot \rangle_{(a,b)} = \frac{1}{b-a} \int_a^b \mathrm{d}t \cdot$.

We write the first average in $C_{ij}$ as follows:

$$\begin{aligned} \langle \upsilon_i(t) A_{h_j}(t) \rangle_{t_d} &= A \frac{1}{T - \tau_{\text{br}}} \int_{\tau_{\text{br}}}^{T} \mathrm{d}t \int_{-\infty}^{t} \mathrm{d}s K(t-s) \upsilon_i(t) h_j(s), \\ &= A \frac{1}{T - \tau_{\text{br}}} \int_{\tau_{\text{br}}}^{T} \mathrm{d}t \int_{0}^{\infty} \mathrm{d}\Delta K(\Delta) \upsilon_i(t) h_j(t - \Delta), \\ &= A \int_0^\infty \mathrm{d}\Delta K(\Delta) \langle \upsilon_i(t) h_j(t - \Delta) \rangle_{t_d}. \end{aligned} \tag{17}$$

If the spike times are uncorrelated the temporal averages become a product of the average firing rates of a pair of visible and hidden neurons (Gerstner and Kistler, 2002):

$$\langle \upsilon_i(t) h_j(t - \Delta) \rangle_{t_d} = \langle \upsilon_i(t) \rangle_{t_d} \langle h_j(t - \Delta) \rangle_{t_d} =: \bar{\upsilon}_i^+ \bar{h}_j^+.$$

If we choose a temporal window that is much smaller than $T$, and assume the network activity is stationary in the interval $(\tau_{\text{br}}, T)$, we can write (up to a negligible error Kempter et al., 2001)

$$\langle \upsilon_i(t) A_{h_j}(t) \rangle_{t_d} = A \bar{\upsilon}_i^+ \bar{h}_j^+ \int_0^\infty \mathrm{d}\Delta K(\Delta). \tag{18}$$

In the uncorrelated case, the second term in $C_{ij}$ contributes the same amount, leading to:

$$C_{ij} = \eta \bar{\upsilon}_i^+ \bar{h}_j^+.$$

with $\eta := 2A \frac{T - \tau_{\text{br}}}{2T} \int_0^\infty \mathrm{d}\Delta K(\Delta)$. Similar arguments apply to the averages in the time interval $t_r$:

$$R_{ij} = 2A \int_0^\infty \mathrm{d}\Delta K(\Delta) \langle \upsilon_i(t) h_j(t - \Delta) \rangle_{t_r} = \eta \bar{\upsilon}_i^- \bar{h}_j^-.$$

with $\bar{\upsilon}_i^- \bar{h}_j^- := \langle \upsilon_i(t) \rangle_{t_r} \langle h_j(t - \Delta) \rangle_{t_r}$. The average update in $(0, 2T)$ then becomes:

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}t} q_{ij} \right\rangle_{(0,2T)} = \eta \left( \bar{\upsilon}_i^+ \bar{h}_j^+ - \bar{\upsilon}_i^- \bar{h}_j^- \right). \tag{19}$$

According to Equation (18), any symmetric temporal window that is much shorter than $T$ can be used. For simplicity, we choose an exponential temporal window $K(\Delta) = \exp(-|\Delta / \tau_{\text{STDP}}|)$ with decay rate $\tau_{\text{STDP}} \ll T$ (**Figure 6B**). In this case, $\eta = 2A \frac{T - \tau_{\text{br}}}{2T} \tau_{\text{STDP}}$.

The modulatory function $g(t)$ partitions the training into epochs of duration $2T$. Each epoch consists of a LTP phase during which the data is presented (construction), followed by a free-running LTD phase (reconstruction). The weights are updated asynchronously during the time interval in which the neural sampling proceeds, and Equation (19) tells us that its average resembles Equation (12). However, it is different in two ways: the averages are taken over one data and reconstruction phase rather than a mini-batch of data samples and their reconstructions; and more importantly, the synaptic weights are updated during the data and the reconstruction phase, whereas in the CD rule, updates are carried out at the end of the reconstruction phase. In the derivation above the effect of the weight change on the network during an epoch $2T$ was neglected for mathematical simplicity. In the following, we verify that despite this approximation, the event-driven CD performs nearly as well as standard CD in the context of a common benchmark task.

### 3.2. LEARNING A GENERATIVE MODEL OF HAND-WRITTEN DIGITS

We train the RBM to learn a generative model of the MNIST handwritten digits using event-driven CD (see section 2.3 for details). For training, 20,000 digits selected randomly (with repetition) from a training set consisting of 10,000 digits were presented in sequence, with an equal number of samples for each digit.

The raster plots in **Figure 7** show the spiking activity of each layer before and after learning for epochs of duration 100 ms. The top panel shows the population-averaged weight. After training, the sum of the upwards and downward excursions of the average

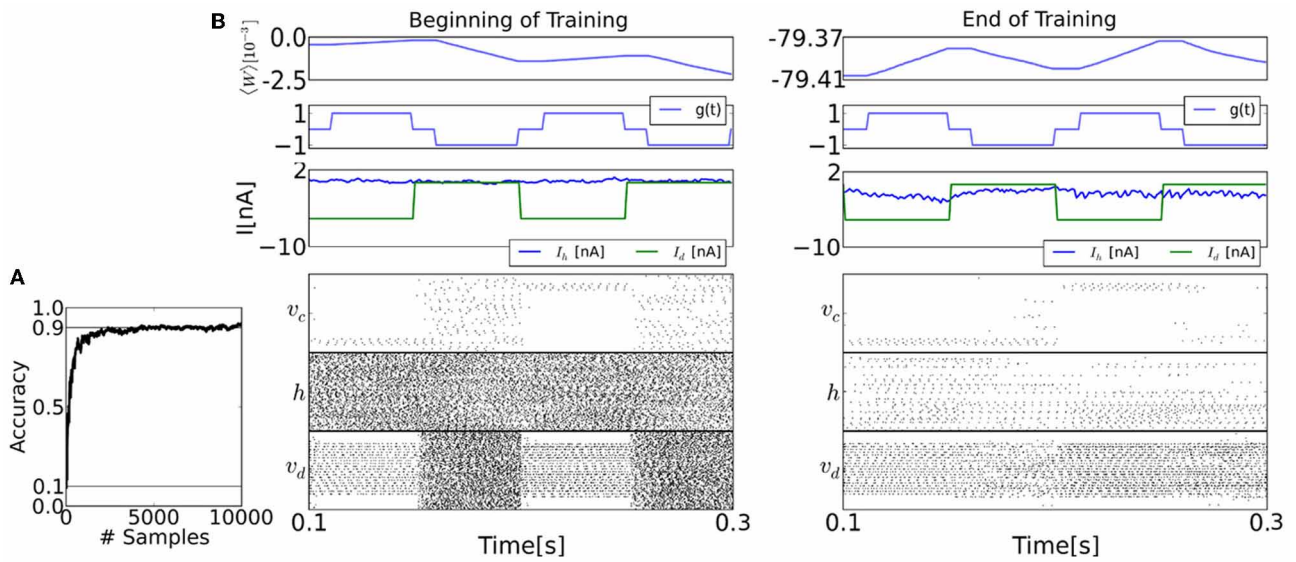



**FIGURE 7 | The spiking neural network learns a generative model of the MNIST dataset using the event-driven CD procedure. (A)** Learning curve, shown here up to 10, 000 samples. **(B)** Details of the training procedure, before and after training (20,000 samples). During the first half of each 0.1 s epoch, the visible layer $v$ is driven by the sensory layer, and the gating variable $g$ is 1, meaning that the synapses undergo LTP. During the second half of each epoch, the sensory stimulus is removed, and $g$ is set to $-1$, so the synapses undergo LTD. The top panels of both figures show the mean of the entries of the weight matrix. The second panel shows the values of the modulatory signal $g(t)$. The third panel shows the synaptic currents of a visible neuron, where $I_h$ is caused by the feedback from the hidden and the bias, and $I_d$ is the data. The timing of the clamping ($I_d$) and $g$ differ due to an interval $\tau_{\text{br}}$ where no weight update is undertaken to avoid the transients (see section 2). Before learning and during the reconstruction phase, the activity of the visible layer is random. But as learning progresses, the activity in the visible layer reflects the presented data in the reconstruction phase. This is very well visible in the layer class label neurons $v_c$, whose activity persists after the sensory stimulus is removed. Although the firing rates of the hidden layer neurons before training is high (average 113 Hz), this is only a reflection of the initial conditions for the recurrent couplings $W$. In fact, at the end of the training, the firing rates in both layers becomes much sparser (average 9.31 Hz).

weight is much smaller than before training, because the learning is near convergence. The second panel shows the value of the modulatory signal $g(t)$. The third panel shows the input current ($I_d$) and the current caused by the recurrent couplings ($I_h$).

Two methods can be used to estimate the overall recognition accuracy of the neural RBM. The first is to sample: the visible layer is clamped to the digit only (i.e., $\upsilon_d$), and the network is run for 1s. The known label is then compared with the position of the group of class neurons that fired at the highest rate. The second method is to minimize free-energy: the neural RBMs parameters are extracted, and for each data sample, the class label with the lowest free-energy (see section 2) is compared with the known label. In both cases, recognition was tested for 1000 data samples that were not used during the training. The results are summarized in **Figure 8**.

As a reference we provide the best performance achieved using the standard CD and one unit per class label ($N_c = 10$) (**Figure 8**, table row 1), 93.6%. By mapping the these parameters to the neural sampler, the recognition accuracy reached 92.6%. The discrepancy is expected since the neural sampler does not exactly sample from the target Boltzmann distribution (see section 2.2).

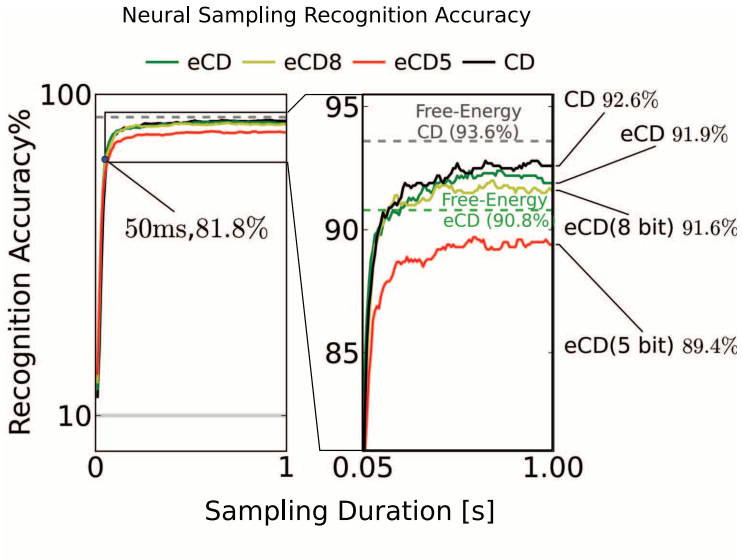
When training a neural RBM of I&F neurons using event-driven CD, the recognition result was 91.9% (**Figure 8**, table row 2). The performance of this RBM obtained by minimizing its free-energy was 90.8%. The learned parameters performed well for classification using the free-energy calculation which suggests that the network learned a model that is consistent with the mathematical description of the RBM.

In an energy-based model like the RBM the free-energy minimization should give the upper bound on the discrimination performance (Haykin, 1998). For this reason, the fact that the recognition accuracy is higher when sampling as opposed to using the free-energy method may appear puzzling. However, this is possible because the neural RBM does not exactly sample from the Boltzmann distribution, as explained in section 2.2. This suggests that event-driven CD compensates for the discrepancy between the distribution sampled by the neural RBM and the Boltzmann distribution, by learning a model that is tailored to the spiking neural network.

Excessively long training durations can be impractical for real-time neuromorphic systems. Fortunately, the learning using event-driven CD is fast: Compared to the off-line RBM training (250, 000 presentations, in mini-batches of 100 samples) the event-driven CD training succeeded with a smaller number of data presentations (20, 000), which corresponded to 2000 s of simulated time. This suggests that the training durations are achievable for real-time neuromorphic systems.

#### 3.2.1. The choice of the number of class neurons $N_c$

Event-driven CD underperformed in the case of 1 neuron per class label ($N_c = 10$), which is the common choice for standard CD and Gibbs sampling. This is because a single neuron firing




| | Accuracy Neural Sampler | Accuracy Free-energy |
|---|---|---|
| Standard CD | 92.6% | 93.6% |
| Event-driven CD | 91.9% | 90.8% |
| Event-driven CD (8 bits) | 91.6% | 91.0% |
| Event-driven CD (5 bits) | 89.4% | 89.2% |

**FIGURE 8 | To test recognition accuracy, the trained RBMs are sampled using the I&F neuron-based sampler for up to 1 s.** The classification is read out by identifying the group of class label neurons that had the highest activity. This experiment is run for RBM parameter sets obtained by standard CD (black, CD) and event-driven CD (green, eCD). To test for robustness to finite precision weights, the RBM was run with parameters obtained by event-driven CD discretized to 8 and 5 bits. In all scenarios, the accuracy after 50 ms of sampling was above 80% and after 1 s the accuracies typically reached their peak at around 92%. The dashed horizontal lines show the recognition accuracy obtained by minimizing the free-energy (see text). The fact that the eCD curve (solid green) surpasses its free-energy line suggests that a model that is tailored to the I&F spiking neural network was learned.

at its maximum rate of 250 Hz cannot efficiently drive the rest of the network without tending to induce spike-to-spike correlations (e.g., synchrony), which is incompatible with the assumptions made for sampling with I&F neurons and event-driven CD. As a consequence, the generative properties of the neural RBM degrade. This problem is avoided by using several neurons per class label (in our case four neurons per class label) because the synaptic weight can be much lower to achieve the same effect, resulting in smaller spike-to-spike correlations.

#### *3.2.2. Neural parameters with finite precision*

In hardware systems, the parameters related to the weights and biases cannot be set with floating-point precision, as can be done in a digital computer. In current neuromorphic implementations the synaptic weights can be configured at precisions of about 8 bits (Yu et al., 2012). We characterize the impact of finite-precision synaptic weights on performance by discretizing the weight and bias parameters to 8 bits and 5 bits. The set of possible weights were spaced uniformly in the interval $(\mu - 4.5\sigma, \mu + 4.5\sigma)$, where $\mu, \sigma$ are the mean and the standard deviation of the parameters across the network, respectively. The classification performance of MNIST digits degraded gracefully. In the 8 bit case, it degrades only slightly to 91.6%, but in the case of 5 bits, it degrades more substantially to 89.4%. In both cases, the RBM still retains its discriminative power, which is encouraging for implementation in hardware neuromorphic systems.

### 3.3. GENERATIVE PROPERTIES OF THE RBM

We test the neural RBM as a generative model of the MNIST dataset of handwritten digits, using parameters obtained by running the event-driven CD. The RBM's generative property enables it to classify and generate digits, as well as to infer digits by combining partial evidence. These features are clearly illustrated in the following experiment (**Figure 9**). First the digit 3 is presented (i.e., layer $v_d$ is driven by layer d) and the correct class label in $v_c$ activated. Second, the neurons associated to class label 5 are clamped, and the network generated its learned version of the digit. Third, the right-half part of a digit 8 is presented, and the class neurons are stimulated such that only 3 or 6 are able to activate (the other class neurons are inhibited, indicated by the gray shading). Because the stimulus is inconsistent with 6, the network settled to 3 and reconstructed the left part of the digit.

The latter part of the experiment illustrates the integration of information between several partially specified cues, which is of interest for solving sensorimotor transformation or multi-modal sensory cue integration problems (Deneve et al., 2001; Doya

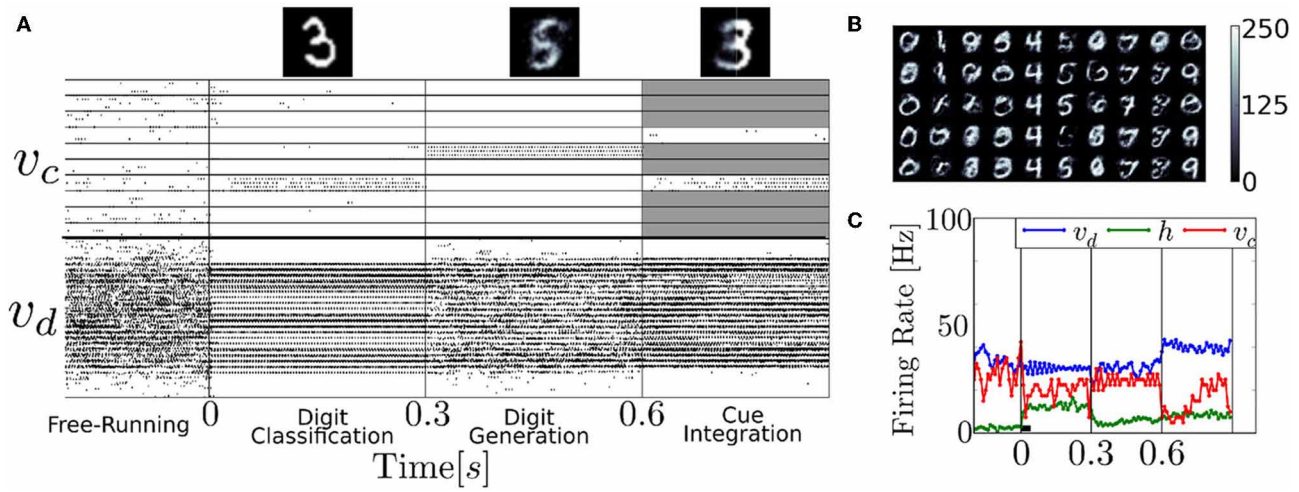



**FIGURE 9 | The recurrent structure of the network allows it to classify, reconstruct and infer from partial evidence. (A)** Raster plot of an experiment illustrating these features. Before time 0*s*, the neural RBM runs freely, with no input. Due to the stochasticity in the network, the activity wanders from attractor to attractor. At time 0*s*, the digit 3 is presented (i.e., layer $\upsilon_d$ is driven by *d*), activating the correct class label in $\upsilon_c$; At time $t = 0.3$ s, the class neurons associated to 5 are clamped to high activity and the rest of the class label neurons are strongly inhibited, driving the network to reconstruct its version of the digit in layer $\upsilon_d$; At time $t = 0.6$ s, the right-half part of a digit 8 is presented, and the class neurons are stimulated such that only 3 or 6 can activate (all others are strongly inhibited as indicated by the gray shading). Because the stimulus is inconsistent with 6, the network settles to a 3 and attempts to reconstruct it. The top figures show the digits reconstructed in layer $\upsilon_d$. **(B)** Digits 0–9, reconstructed in the same manner. The columns correspond to clamping digits 0–9, and each is different, independent run. **(C)** Population firing rate of the experiment presented in **(A)**. The network activity is typically at equilibrium after about $10\tau_r = 40$ ms (black bar).

et al., 2006; Corneil et al., 2012). This feature has been used for auditory-visual sensory fusion in a spiking Deep Belief Network (DBN) model (O'Connor et al., 2013). There, the authors trained a DBN with visual and auditory data, which learned to associate the two sensory modalities, very similarly to how class labels and visual data are associated in our architecture. Their network was able to resolve a similar ambiguity as in our experiment in **Figure 9**, but using auditory inputs instead of a class label.

During digit generation, the trained network had a tendency to be globally bistable, whereby the layer $\upsilon_d$ completely deactivated layer $h$. Since all the interactions between $\upsilon_d$ and $\upsilon_c$ take place through the hidden layer, $\upsilon_c$ could not reconstruct the digit. To avoid this, we added populations of I&F neurons that were wired to layers $\upsilon$ and $h$, respectively. The parameters of these neurons and their couplings were tuned such that each layer was strongly excited when it's average firing rate fell below 5 Hz.

## 4. DISCUSSION

Neuromorphic systems are promising alternatives for large-scale implementations of RBMs and deep networks, but the common procedure used to train such networks, Contrastive Divergence (CD), involves iterative, discrete-time updates that do not straightforwardly map on a neural substrate. We solve this problem in the context of the RBM with a spiking neural network model that uses the recurrent network dynamics to compute these updates in a continuous-time fashion. We argue that the recurrent activity coupled with STDP dynamics implements an event-driven variant of CD. Event-driven CD enables the system to learn on-line, while being able to carry out functionally relevant tasks such as recognition, data generation and cue integration.

The CD algorithm can be used to learn the parameters of probability distributions other than the Boltzmann distribution (even those without any symmetry assumptions). Our choice for the RBM, whose underlying probability distribution is a special case of the Boltzmann distribution, is motivated by the following facts: They are universal approximators of discrete distributions (Le Roux and Bengio, 2008); the conditions under which a spiking neural circuit can naturally perform MCMC sampling of a Boltzmann distribution were previously studied (Merolla et al., 2010; Buesing et al., 2011); and RBMs form the building blocks of many deep learning models such as DBNs, which achieve state-of-the-art performance in many machine learning tasks (Bengio, 2009). The ability to implement RBMs with spiking neurons and train then using event-based CD paves the way toward on-line training of DBNs of spiking neurons (Hinton et al., 2006).

We chose the MNIST handwritten digit task as a benchmark for testing our model. When the RBM was trained with standard CD, it could recognize up to 926 out of 1000 of out-of-training samples. The MNIST handwritten digit recognition task was previously shown in a digital neuromorphic chip (Arthur et al., 2012), which performed at 89% accuracy, and in a software simulated visual cortex model (Eliasmith et al., 2012). However, both implementations were configured using weights trained off-line. A recent article showed the mapping of off-line trained DBNs onto spiking neural network (O'Connor et al., 2013). Their results demonstrated hand-written digit recognition using neuromorphic event-based sensors as a source of input spikes. Their performance reached up to 94.1% using leaky I&F neurons. The

use of an additional layer explains to a large extent their better performance compared to ours (91.9%). Our work extends (O'Connor et al., 2013) with on-line training that is based on synaptic plasticity, testing its robustness to finite weight precision, and providing an interpretation of spiking activity in terms of neural sampling.

To achieve the computations necessary for sampling from the RBM, we have used a neural sampling framework (Fiser et al., 2010), where each spike is interpreted as a sample of an underlying probability distribution. Buesing et al. proved that abstract neuron models consistent with the behavior of biological spiking neurons can perform MCMC, and have applied it to a basic learning task in a fully visible Boltzmann Machine. We extended the neural sampling framework in three ways: First, we identified the conditions under which a dynamical system consisting of I&F neurons can perform neural sampling; Second, we verified that the sampling of RBMs was robust to finite-precision parameters; Third, we demonstrated learning in a Boltzmann Machine with hidden units using STDP synapses.

In neural sampling, neurons behave stochastically. This behavior can be achieved in I&F neurons using noisy input currents, created by a Poisson spike train. Spike trains with Poisson-like statistics can be generated with no additional source of noise, for example by the following mechanisms: balanced excitatory and inhibitory connections (van Vreeswijk and Sompolinsky, 1996), finite-size effects in a large network, and neural mismatch (Amit and Brunel, 1997). The latter mechanism is particularly appealing, because it benefits from fabrication mismatch and operating noise inherent to neuromorphic implementations (Chicca and Fusi, 2001).

Other groups have also proposed to use I&F neuron models for computing the Boltzmann distribution. (Merolla et al., 2010) have shown that noisy I&F neurons' activation function is approximately a sigmoid as required by the Boltzmann machine, and have devised a scheme whereby a global inhibitory rhythm drives the network to generate samples of the Boltzmann distribution. O'Connor et al. (2013) have demonstrated a deep belief network of I&F neurons that was trained off-line, using standard CD and tested it using the MNIST database. Independently and simultaneously to this work, Petrovici et al. (2013) demonstrated that conductance-based I&F neurons in a noisy environment are compatible with neural sampling as described in Buesing et al. (2011). Similarly, Petrovici et al. find that the choice of non-rectangular PSPs and the approximations made by the I&F neurons are not critical to the performance of the neural sampler. Our work extends all of those above by providing an online, STDP-based learning rule to train RBMs sampled using I&F neurons.

### 4.1. APPLICABILITY TO NEUROMORPHIC HARDWARE

Neuromorphic systems are sensible to fabrication mismatch and operating noise. Fortunately, the mismatch in the synaptic weights and the activation function parameters $\gamma$ and $\beta$ are not an issue if the biases and the weights are learned, and the functionality of the RBM is robust to small variations in the weights caused by discretization. These two findings are encouraging for neuromorphic implementations of RBMs. However, at least two conceptual problems of the presented RBM architecture must be solved in order to implement such systems on a larger scale. First, the symmetry condition required by the RBM does not necessarily hold. In a neuromorphic device, the symmetry condition is impossible to guarantee if the synapse weights are stored locally at each neuron. Sharing one synapse circuit per pair of neurons can solve this problem. This may be impractical due to the very large number of synapse circuits in the network, but may be less problematic when using Resistive Random-Access Memorys (RRAMs) (also called *memristors*) crossbar arrays to emulate synapses (Kuzum et al., 2011; Cruz-Albrecht et al., 2013; Serrano-Gotarredona et al., 2013).RRAM are a new class of nanoscale devices whose current-voltage relationship depends on the history of other electrical quantities (Strukov et al., 2008), and so act like programmable resistors. Because they can conduct currents in both directions, one RRAM circuit can be shared between a pair of neurons. A second problem is the number of recurrent connections. Even our RBM of modest dimensions involved almost two million synapses, which is impractical in terms of bandwidth and weight storage. Even if a very high number of weights are zero, the connections between each pair of neurons must exist in order for a synapse to learn such weights. One possible solution is to impose sparse connectivity between the layers (Murray and Kreutz-Delgado, 2007; Tang and Eliasmith, 2010) and implement synaptic connectivity in a scalable hierarchical address-event routing architecture (Joshi et al., 2010; Park et al., 2012).

### 4.2. OUTLOOK: A CUSTOM LEARNING RULE

Our method combines I&F neurons that perform neural sampling and the CD rule. Although we showed that this leads to a functional model, we do not know whether event-driven CD is optimal in any sense. This is partly due to the fact that $CD_k$ is an approximate rule (Hinton, 2002), and it is still not entirely understood why it performs so well, despite extensive work in studying its convergence properties (Carreira-Perpinan and Hinton, 2005). Furthermore, the distribution sampled by the I&F neuron does not exactly correspond to the Boltzmann distribution, and the average weight updates in event-driven CD differ from those of standard CD, because in the latter they are carried out at the end of the reconstruction step.

A very attractive alternative is to derive a custom synaptic plasticity rule that minimizes some functionally relevant quantity (such as Kullback-Leibler divergence or Contrastive Divergence), *given* the encoding of the information in the I&F neuron (Deneve, 2008; Brea et al., 2013). A similar idea was recently pursued in Brea et al. (2013), where the authors derived a triplet-based synaptic learning rule that minimizes an upper bound of the Kullback–Leibler divergence between the model and the data distributions. Interestingly, their rule had a similar global signal that modulates the learning rule, as in event-driven CD, although the nature of this resemblance remains to be explored. Such custom learning rules can be very beneficial in guiding the design of on-chip plasticity in neuromorphic VLSI and RRAM nanotechnologies, and will be the focus of future research.

## ACKNOWLEDGMENTS

This work was partially funded by the National Science Foundation (NSF EFRI-1137279, CCF-1317560), the Office of

Naval Research (ONR MURI 14-13-1-0205), and the Swiss National Science Foundation (PA00P2_142058).

## REFERENCES


Amit, D., and Brunel, N. (1997). Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cereb. Cortex* 7, 237–252. doi: 10.1093/cercor/7.3.237

Arthur, J., Merolla, P., Akopyan, F., Alvarez, R., Cassidy, A., Chandra, S., et al. (2012). "Building block of a programmable neuromorphic substrate: a digital neurosynaptic core," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane, QLD: IEEE), 1–8. doi: 10.1109/IJCNN.2012.6252637

Bartolozzi, C., and Indiveri, G. (2007). Synaptic dynamics in analog VLSI. *Neural Comput.* 19, 2581–2603. doi: 10.1162/neco.2007.19.10.2581

Bengio, Y. (2009). Learning deep architectures for ai. *Found. Trends Mach. Learn.* 2, 1–127. doi: 10.1561/2200000006

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., et al. (2010). "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Vol. 4 (Austin, TX). Available online at: http://deeplearning.net/software/theano/

Brea, J., Senn, W., and Pfister, J.-P. (2013). Matching recall and storage in sequence learning with spiking neural networks. *J. Neurosci.* 33, 9565–9575. doi: 10.1523/JNEUROSCI.4098-12.2013

Buesing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput. Biol.* 7:e1002211. doi: 10.1371/journal.pcbi.1002211

Carreira-Perpinan, M. A., and Hinton, G. E. (2005). On contrastive divergence learning. *Artif. Intell. Stat.* 2005, 17. Available online at: http://www.gatsby.ucl.ac.uk/aistats/AIabst.htm

Chicca, E. and Fusi, S. (2001). "Stochastic synaptic plasticity in deterministic aVLSI networks of spiking neurons," in *Proceedings of the World Congress on Neuroinformatics, ARGESIM Reports*, ed. F. Rattay (Vienna: ARGESIM/ASIM Verlag), 468–477.

Corneil, D., Sonnleithner, D., Neftci, E., Chicca, E., Cook, M., Indiveri, G., et al. (2012). "Function approximation with uncertainty propagation in a VLSI spiking neural network," in *International Joint Conference on Neural Networks, IJCNN* (Brisbane: IEEE), 2990–2996. doi: 10.1109/IJCNN.2012.6252780

Cox, D. (1962). *Renewal Theory.* Vol. 1. London: Methuen.

Cruz-Albrecht, J. M., Derosier, T., and Srinivasa, N. (2013). A scalable neural chip with synaptic electronics using cmos integrated memristors. *Nanotechnology* 24, 384011. doi: 10.1088/0957-4484/24/38/384011

Deiss, S., Douglas, R., and Whatley, A. (1998). "A pulse-coded communications infrastructure for neuromorphic systems, chapter 6," in *Pulsed Neural Networks*, eds W. Maass and C. Bishop (Cambridge, MA: MIT Press), 157–178.

Deneve, S. (2008). Bayesian spiking neurons I: inference. *Neural Comput.* 20, 91–117. doi: 10.1162/neco.2008.20.1.91

Deneve, S., Latham, P., and Pouget, A. (2001). Efficient computation and cue integration with noisy population codes. *Nature Neurosci.* 4, 826–831. doi: 10.1038/90541

Destexhe, A., Mainen, Z., and Sejnowski, T. (1998). "Kinetic models of synaptic transmission ," in *Methods in Neuronal Modelling, from Ions to Networks*, eds C. Koch and I. Segev (Cambridge, MA: MIT Press), 1–25.

Doya, K., Ishii, S., Pouget, A., and Rao, R. (2006). *Bayesian Brain Probabilistic Approaches to Neural Coding*. Cambridge, MA: MIT Press. doi: 10.7551/mitpress/9780262042383.001.0001

Eliasmith, C., Stewart, T., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science* 338, 1202–1205. doi: 10.1126/science.1225266

Fiser, J., Berkes, P., Orbán, G., and Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations: perceptual learning, motor learning, and automaticity. *Trends Cogn. Sci.* 14, 119. doi: 10.1016/j.tics.2010.01.003

Fusi, S., and Mattia, M. (1999). Collective behavior of networks with linear (VLSI) integrate and fire neurons. *Neural Comput.* 11, 633–652. doi: 10.1162/089976699300016601

Gardiner, C. W. (2012). *Handbook of Stochastic Methods*. Berlin: Springer. doi: 10.1007/978-3-662-02377-8

Gerstner, W., and Kistler, W. (2002). *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge: Cambridge University Press. doi: 10.1017/CBO9780511815706

Goodman, D., and Brette, R. (2008). Brian: a simulator for spiking neural networks in Python. *Front. Neuroinform.* 2:5. doi: 10.3389/neuro.11.005.2008

Griffiths, T., Chater, N., Kemp, C., Perfors, A., and Tenenbaum, J. B. (2010). Probabilistic models of cognition: exploring representations and inductive biases. *Trends Cogn. Sci.* 14, 357–364. doi: 10.1016/j.tics.2010.05.004

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. 2nd Edn. Prentice Hall. Available online at: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0132733501

Hinton, G., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.* 18, 1527–1554. doi: 10.1162/neco.2006.18.7.1527

Hinton, G., and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science* 313, 504–507. doi: 10.1126/science.1127647

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14, 1771–1800. doi: 10.1162/089976602760128018

Indiveri, G., Linares-Barranco, B., Hamilton, T., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5, 1–23. doi: 10.3389/fnins.2011.00073

Joshi, S., Deiss, S., Arnold, M., Park, J., Yu, T., and Cauwenberghs, G. (2010). "Scalable event routing in hierarchical neural array architecture with global synaptic connectivity," in *12th International Workshop on Cellular Nanoscale Networks and Their Applications* (Berkeley, CA: IEEE), 1–6. doi: 10.1109/CNNA.2010.5430296

Kempter, R., Gerstner, W., and Van Hemmen, J. (2001). Intrinsic stabilization of output rates by spike-based hebbian learning. *Neural Comput.* 13, 2709–2741. doi: 10.1162/089976601317098501

Kuzum, D., Jeyasingh, R. G., Lee, B., and Wong, H.-S. P. (2011). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Lett.* 12, 2179–2186. doi: 10.1021/nl201040y

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., et al. (2011). Building high-level features using large scale unsupervised learning. arXiv preprint: arXiv:1112.6209.

Le Roux, N., and Bengio, Y. (2008). Representational power of restricted boltzmann machines and deep belief networks. *Neural Comput.* 20, 1631–1649. doi: 10.1162/neco.2008.04-07-510

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324. doi: 10.1109/5.726791

Liu, S.-C., and Delbruck, T. (2010). Neuromorphic sensory systems. *Curr. Opin. Neurobiol.* 20, 288–295. doi: 10.1016/j.conb.2010.03.007

Mead, C. (1989). *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley. doi: 10.1007/978-1-4613-1639-8

Merolla, P., Ursell, T., and Arthur, J. (2010). The thermodynamic temperature of a rhythmic spiking network. *CoRR*. abs/1009.5473, ArXiv e-prints. Available online at: http://arxiv.org/abs/1009.5473

Murray, J. F., and Kreutz-Delgado, K. (2007). Visual recognition and inference using dynamic over complete sparse learning. *Neural Comput.* 19, 2301–2352. doi: 10.1162/neco.2007.19.9.2301

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. J. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proc. Natl. Acad. Sci. U.S.A.* 110, E3468–E3476. doi: 10.1073/pnas.1212083110

Neftci, E., Toth, B., Indiveri, G., and Abarbanel, H. (2012). Dynamic state and parameter estimation applied to neuromorphic systems. *Neural Comput.* 24, 1669–1694. doi: 10.1162/NECO_a_00293

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Front. Neurosci.* 7:178. doi: 10.3389/fnins.2013.00178

Park, J., Yu, T., Maier, C., Joshi, S., and Cauwenberghs, G. (2012). "Live demonstration: Hierarchical address-event routing architecture for reconfigurable large scale neuromorphic systems," in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, (Seoul), 707, 711, 20–23. doi: 10.1109/ISCAS.2012.6272133. Available online at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6272133

Pedroni, B., Das, S., Neftci, E., Kreutz-Delgado, K., and Cauwenberghs, G. (2013). "Neuromorphic adaptations of restricted boltzmann machines and deep belief networks," in *International Joint Conference on Neural Networks, IJCNN*. (Dallas, TX).

Petrovici, M. A., Bill, J., Bytschok, I., Schemmel, J., and Meier, K. (2013). Stochastic inference with deterministic spiking neurons. arXiv preprint: arXiv:1311.3211.

Plesser, H. E., and Gerstner, W. (2000). Noise in integrate-and-fire neurons: from stochastic input to escape rates. *Neural Comput.* 12, 367–384. doi: 10.1162/089976600300015835

Renart, A., Song, P., and Wang, X.-J. (2003). Robust spatial working memory through homeostatic synaptic scaling in heterogeneous cortical networks. *Neuron* 38, 473–485. doi: 10.1016/S0896-6273(03)00255-1

Robbins, H., and Monro, S. (1951). A stochastic approximation method. *Ann. Math. Stat.* 22, 400–407. doi: 10.1214/aoms/1177729586

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., and Millner, S. (2010). "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *International Symposium on Circuits and Systems, ISCAS* (Paris: IEEE), 1947–1950. doi: 10.1109/ISCAS.2010.5536970

Serrano-Gotarredona, T., Masquelier, T., Prodromakis, T., Indiveri, G., and Linares-Barranco, B. (2013). Stdp and stdp variations with memristors for spiking neuromorphic learning systems. *Front. Neurosci.* 7:2. doi: 10.3389/fnins.2013.00002

Silver, R., Boahen, K., Grillner, S., Kopell, N., and Olsen, K. (2007). Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools. *J. Neurosci.* 27, 11807. doi: 10.1523/JNEUROSCI.3575-07.2007

Strukov, D. B., Snider, G. S., Stewart, D. R., and Williams, R. S. (2008). The missing memristor found. *Nature* 453, 80–83. doi: 10.1038/nature06932

Tang, Y. and Eliasmith, C. (2010). "Deep networks for robust visual recognition," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (Haifa), 1055–1062. Available online at: http://www.icml2010.org/papers/370.pdf

van Vreeswijk, C., and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* 274, 1724–1726. doi: 10.1126/science.274.5293.1724

Yu, T., Park, J., Joshi, S., Maier, C., and Cauwenberghs, G. (2012) "65k-neuron integrate-and-fire array transceiver with address-event reconfigurable synaptic routing," in *Biomedical Circuits and Systems Conference (BioCAS), IEEE*, (Hsinch), 21, 24. 28–30. doi: 10.1109/BioCAS.2012.6418479. Available online at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6418479

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 07 October 2013; accepted: 22 December 2013; published online: 30 January 2014.*

*Citation: Neftci E, Das S, Pedroni B, Kreutz-Delgado K and Cauwenberghs G (2014) Event-driven contrastive divergence for spiking neuromorphic systems. Front. Neurosci. 7:272. doi: 10.3389/fnins.2013.00272*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 07 May 2014
doi: 10.3389/fnins.2014.00086

# Compiling probabilistic, bio-inspired circuits on a field programmable analog array

***Bo Marr*[1]* and *Jennifer Hasler*[2]**

[1] *Raytheon, Space and Airborne Systems, Manhattan Beach, CA, USA*
[2] *Georgia Institute of Technology, Atlanta, GA, USA*

***Edited by:***
*Tobi Delbruck, University of Zurich and ETH Zurich, Switzerland*

***Reviewed by:***
*Emre O. Neftci, ETH Zurich, Switzerland*
*Davide Badoni, University Tor Vergata, Italy*

***Correspondence:**
*Bo Marr, Raytheon, Space and Airborne Systems, Manhattan Beach, 2000 E. El Segundo Blvd, El Segundo, CA 90245 USA*
*e-mail: harry.b.marr@raytheon.com*

A field programmable analog array (FPAA) is presented as an energy and computational efficiency engine: a mixed mode processor for which functions can be compiled at significantly less energy costs using probabilistic computing circuits. More specifically, it will be shown that the core computation of any dynamical system can be computed on the FPAA at significantly less energy per operation than a digital implementation. A stochastic system that is dynamically controllable via voltage controlled amplifier and comparator thresholds is implemented, which computes Bernoulli random variables. From Bernoulli variables it is shown exponentially distributed random variables, and random variables of an arbitrary distribution can be computed. The Gillespie algorithm is simulated to show the utility of this system by calculating the trajectory of a biological system computed stochastically with this probabilistic hardware where over a 127X performance improvement over current software approaches is shown. The relevance of this approach is extended to any dynamical system. The initial circuits and ideas for this work were generated at the 2008 Telluride Neuromorphic Workshop.

**Keywords: FPAA, probabilistic hardware, reconfigurable analog, dymamical system, bio-inspired, Hardware accelerator, biological computational model, probability theory**

## 1. INTRODUCTION

Due to the large computational efficiency gap that is theorized between classic digital computing and neuromorphic style computing, particularly in biological systems, this work seeks to explore the potential efficiency gains of a neuromorphic approach using stochastic circuits to solve dynamical systems.

There is wide demand for a technology to compute dynamical systems much more efficiently with some recent examples being to calculate quantum equations to aid in the development of quantum computers or in the search for new meta-materials and pharmaceuticals using high computational throughput search methods for new meta-compounds. Standard digital computers—even super computers—have proven to be inefficient at these tasks limiting our ability to innovate here.

Stochastic functions can be computed in a more efficient way if these stochastic operations are done natively in probabilistic hardware. Neural connections in the cortex of the brain is just such an example and occur on a stochastic basis (Douglas, 2008). The neurotransmitter release is probabilistic in regards to synapse firings (Goldberg et al., 2001) in neural communication. Most germaine to this work, many chemical and biological reactions occur on a stochastic basis and are modeled here via probabilistic circuits compiled on a reconfigurable field programmable analog array (Gillespie, 1976).

Gillespie effectively showed that molecular reactions occur probabilistically and gave a method for translating a system of $N$ chemical equations, normally specified by deterministic differential equations, into a system of probabilistic, Markov processes. This will be the dynamic system described and computed herein. It has been shown that in a system with a sufficiently small number of molecules, the stochastic method is *more accurate* than its deterministic counterpart. It was further proven that in the thermodynamic limit (large number of molecules) of such a system, the deterministic and stochastic forms are mathematically equivalent (Oppenheim et al., 1969; Kurtz, 1972).

Mathematical results will be extended from Gillespie's algorithm to show that *any dynamical system* can be computed using a system of stochastic equations. The efficiency in computing such a system is greatly increased by a direct, probabilistic hardware implementation that can be done in the analog co-processor we present. However, *how* to express a general dynamical system stochastically will not be discussed, only that a formulation exists that is more efficient when computed with natively probabilistic hardware.

Several encryption algorithms and other on-chip solutions for a uniformly random number generator in hardware have been shown for microprocessors (Ohba et al., 2006). Generating static Bernoulli trials, where a 1 is generated with fixed probability $p$ and 0 is generated with fixed probability $1 - p$, were proposed using amplified thermal noise across digital gates and is also not a novel concept, but the work in which this concept was described was not fabricated, measured, or applied in hardware, and only existed in theory (Chakrapani et al., 2006). Static probabilities, or those with a fixed $p$-value, will not allow the performance gains that are possible in many stochastic systems because without dynamic $p$-values stochastic processes cannot be fully realizable in hardware.

There has been a hardware solution proposed for dynamic Bernoulli trial generators, where the probability $p$ can be dynamically reconfigured via reprogramming a floating gate

transistor (Xu et al., 1972). While this latter work illustrates a note-worthy solution and is a unique implementation to provide dynamic probability adjustment, there is an overhead cost in terms of time to readjust the probability $p$ due to the nature of floating gate programming, which were not designed in that work for the continuous updates that are required for the applications presented here.

The topic of generating stochastic variables with hardware circuits has also been addressed previously in Genov and Cauwenberghs (2001), but not in this manner. We make a contribution to the literature by showing that not only can we produce stochastic variables, but we can tune the probability of these stochastic variables in real time through a software controllable input to the Bernoulli trial generator circuit via the FPAA. Further, we show how an array of these can be compiled on hardware and where outputs are input to a priority encoder to create an exponentially random variable for the first time known to the authors. Finally, this paper shows how the FPAA, with tunable stochastic variables which can be dynamically tuned in real time, results in significant performance gains of 127X for computing dynamical systems.

In short, this paper will present several novel contributions including

- A novel circuit for fast dynamic Bernoulli random number generation.
- A compiled chaos circuit to generate environment independent probabilistic variables.
- A novel circuit for fast dynamic exponentially distributed random number generation.
- Analysis of the performance gains provided by the latter circuits over current methods for Gillespie's algorithm that apply to many biological applications and stochastic applications in general.
- Extension of the latter methods for applicability to any dynamical system and the result that all dynamic systems calculated stochastically require exponentially distributed random numbers.
- A method for going from concept to circuit measurements in 2 weeks using a novel reconfigurable chipset developed in part by the authors.

Section 2 introduces the technology behind building probabilistic function generators in hardware using thermal noise characteristics. Section 3 reviews implementation of dynamical systems in general and specifically Gillespie's Algorithm to give a context for why these circuits are important. Section 4 will review the chipset that was built in part by the authors and how it can used for faster stochastic computation. Section 5 will discuss the hardware results and experimental measurements. Section 6 will conclude the paper and discuss future directions.

## 2. EXTREMELY EFFICIENT STOCHASTIC CIRCUITS

Stochastic computation has shown to be a powerful tool to compute solutions to systems that would otherwise require complex continuous-time differential equations. However, the efficacy of stochastic methods are lost if complex computations are needed to produce the digital representation of these stochastic results.

We present several circuits to solve these issues that can uniquely be compiled in analog technology, and thus our FPAA co-processor.

### 2.1. A PROGRAMMABLE THERMAL NOISE CIRCUIT

Thermal noise is a well-defined, well-behaved phenomenon that we show can be used as a computational resource within the FPAA fabric. This work will show that it can be used not only as a resource for random number generation, but for the generation of arbitrarily complex probabilistic functions.

The current through a transistor, and hence the voltage at the drain or source node of a transistor, shows the probabilistic thermal noise effect as shown in **Figure 1**, and can be used as the basic building block of any probabilistic function generator.

Thermal noise present in integrated circuits, also known as Johnson Noise, is generated by the natural thermal excitation of the electrons in a circuit. When modeled on a capacitive load, the root-mean-square voltage level of the thermal noise is given by $U_T = \sqrt{\frac{kT}{C}}$ where $k$ is Boltzmann's constant, $T$ is temperature, and $C$ is the capacitance of the load. The likelihood of the voltage level of thermal noise is modeled as a Gaussian probability function and has an equivalent magnitude throughout its frequency spectrum and is thus known as white Gaussian noise (Kish, 2002).

To take advantage of the well-defined properties of thermal noise trapped on a capacitor, the circuit in **Figure 2A** was developed. This circuit can be broken down into the circuit components seen in **Figure 2B**.

Thermal noise voltage on a 0.35 $\mu$m process, which is the process size used for testing in this paper, with capacitor values in the range of $C = 500$ fF has RMS noise level of roughly 100 $\mu$V. Even with a 100 mV supply, the thermal noise voltage that would cause a probabilistic digital bit flip is $1000\sigma$ down from supply, giving the probabilistic function designer limited options. Hence, in these experiments the thermal voltage signal was routed through two operational transconductance amplifiers (OTA) with gain $A_i$. These circuits are shown in **Figure 3**.

By routing the amplified thermal noise signal through a comparator, and then comparing this signal to a user selectable voltage signal, a Bernoulli trial is created where the comparator outputs

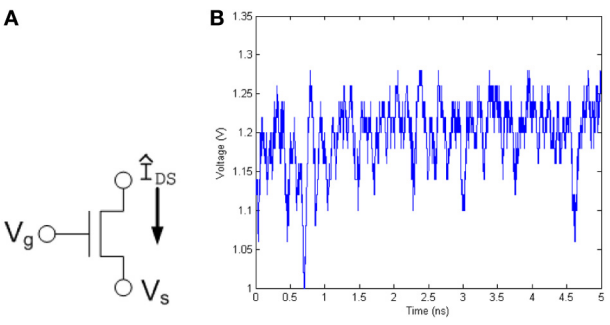



**FIGURE 1 | (A)** The noise characteristics generated by a single transistor can be used as well-defined noise source. The current through and thus the voltage at the drain or source node of the transistor produces noise due to the thermal agitation of charge carriers. **(B)** Voltage measured from a transistor on 0.35 $\mu$m test chip.

a digital "1" if the amplified thermal voltage signal is greater than the probability select and a "0" is output otherwise. A probability $p$ can be set for the Bernoulli trial by setting the input voltage to the comparator such that it is less or more likely for a randomly varying thermal voltage to surpass this value, "Probability Select." The integral from the input voltage to $V_{dd}$ of the thermal noise function is the probability of a digital 1, and the probability of a digital 0 is the integral from ground to the input voltage. This concept is illustrated in **Figure 4**.

Note that the reconfigurable FPAA used to program this circuit has floating-gate controllable current biases such that they can be used to offset temperature effects.

A Bernoulli random variable is generated with a probability $p$ dynamically selectable by the user using these techniques. A useful property of Bernoulli trials is that an arbitrary probability distribution can be created when used in large numbers (as the number of Bernoulli trials $N \rightarrow \infty$) they can be used to create an arbitrary probability distribution. This phenomenon is illustrated in **Figure 5**.

An exponential random variable is generated from a Bernoulli variable in the following way. $X$ is defined here as the number of Bernoulli trials needed to produce a success, and this variable $X$ is exponentially distributed. For example, to require six coin flips to produce a heads is exponentially less likely than to require two flips to get a head, since this is nothing more than a standard geometric sequence. The shape of this exponential distribution is controlled by the probability $p$ of the Bernoulli trials.

$$Pr(X = k) = (1 - p)^{k-1} p \tag{1}$$

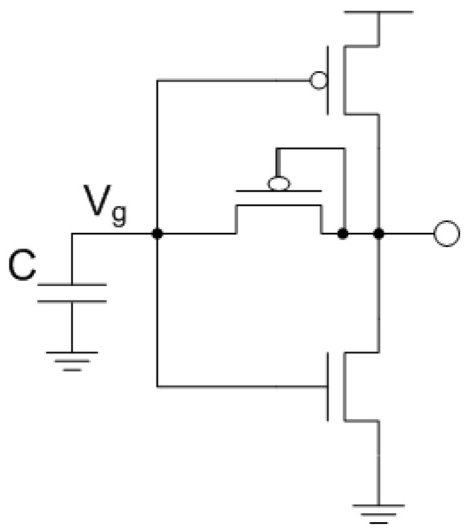



**FIGURE 2 | Circuit used to take advantage of the well-defined thermal noise properties on a capacitor where the root-mean-square (RMS) voltage of the noise on a capacitor is $V_n = \sqrt{\frac{kT}{C}}$.** The thermal noise on this capacitor is used as the gate voltage for PMOS and NMOS transistors where a current with the same spectrum as the thermal noise is generated. This ultimately produces a voltage controlled current through the diode-connected transistor.

**Figure 6** shows how these Bernoulli random variables are used to create an exponentially distributed number by being placed as inputs to a priority encoder. Recall that a priority encoder works by encoding the output to represent in binary which input was the first in priority order (from top to bottom for example) to be a 1. Also recall from Equation (1) that the number of Bernoulli trials needed to get a 1 is exponentially distributed. !The top Bernoulli trial in the figure is considered our first trial, the second from the top, our second trial etc. So the priority encoder in **Figure 6** is encoding for us how many trials are needed to get a success (1), exactly our exponential distribution.

Using these methods, an exponentially distributed random number can be generated in two clock cycles, a vast improvement

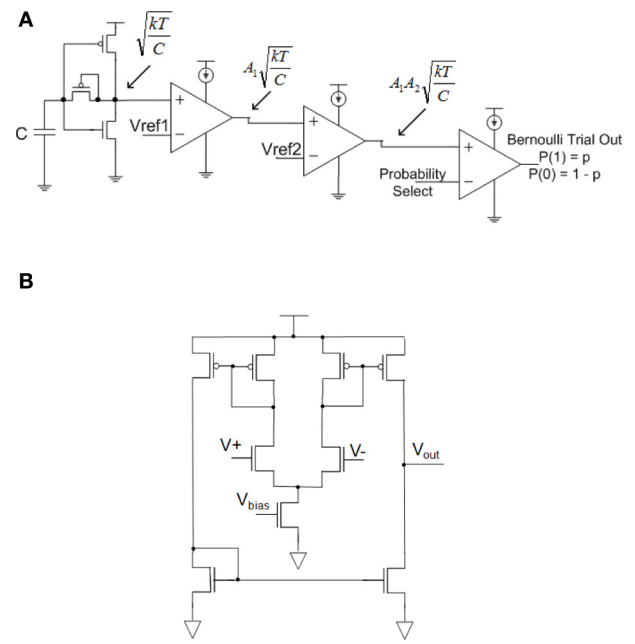



**FIGURE 3 | Circuits for generating a Bernoulli random variable (1 with probability *p* and 0 with probability 1-*p*). (A)** Dynamic Bernoulli Probability Circuit. Thermal noise is trapped on the capacitor, amplified twice through two operational transconductance amplifiers (OTA) with gain of $A_i$ then put through a comparator with a probability select input. Note that these three OTA's are the same basic circuit programmed to different functionality. **(B)** Nine-transistor OTA used in **(A)** for amplifier and comparator circuits.

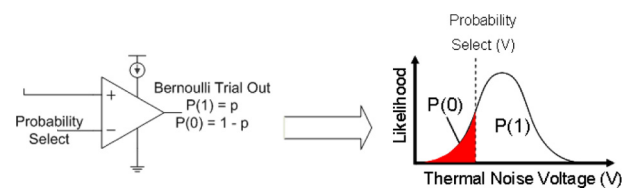



**FIGURE 4 | The probability distribution function of thermal noise and the probability of generating a *P*(1) or *P*(0) in the Bernoulli variable generating circuit.** The probability of a 1 is the integral under the probability distribution function of the thermal noise from the comparator voltage to the supply voltage of the circuit, $V_{dd}$.

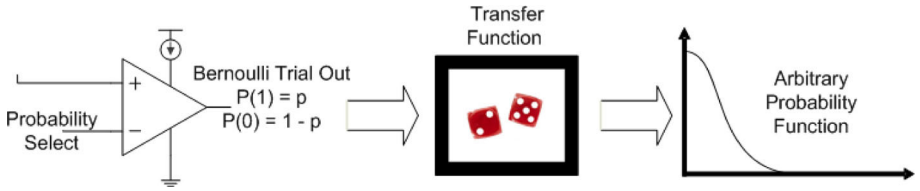



**FIGURE 5 | A Bernoulli probability trial can be used to generate an arbitrary probability distribution with the correct transfer function.** This result will be taken advantage of to greatly increase the performance of dynamical systems in this paper.

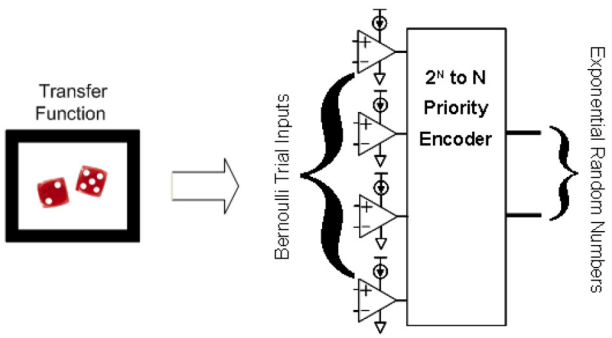



**FIGURE 6 | Illustration of how to transform *N* Bernoulli trials into an exponential distribution.** Bernoulli trials put through a priority encoder as inputs results in an exponentially distributed probability function, where the shape of the exponential function can be tuned through the threshold inputs to the Bernoulli trials.

over software and other current methods, which will be explained in the next section.

### 2.2. PROGRAMMING BERNOULLI TRIALS AT TELLURIDE WORKSHOP

All of the above circuits from the previous section were conceived of, designed, built, measured, and compiled in about in the 2 weeks of the 2008 Telluride Neuromorphic workshop. This accomplishment is both a testament to the productivity that the Telluride Neuromorphic workshop allows as well as a testament to the quick prototyping capability of the FPAA.

The Telluride Neuromorphic workshop is unique in that some of the most brilliant minds in the country get together for an extended several week session dedicated to teaching, working with real hardware, and producing results such that students are completely immersed in a world class engineering environment, day and night, for 2–3 weeks.

The FPAA is analogous to the FPGA for logic circuits in that circuits can be conceived of, programmed onto the chip, and measured in the same day. The FPAA has a mature toolset where an analog designer can conceive of a circuit, such as during a Telluride lecture session on analog design, and can simply create a netlist. The FPAA tools automatically take this netlist and optimally compile it to the fabric using a bitstream to program the on-board floating gate devices to set switches allowing networks of active and passive devices, set current sources, bias currents, amplifier characteristics, and calibrate out device mismatch. A standard multi-meter is connected to the FPAA test board via built-for-test pinned out circuit leads. The multi-meter in this instance was connected back to the computer via GPIB that was producing the netlist to allow a full hardware in the loop programmable environment. Current toolsets are even more advanced allowing Simulink and other Simulink-like tools to build the circuit netlists.

### 2.3. TEMPERATURE INVARIANT BERNOULLI TRIALS

The thermal noise circuits used to create Bernoulli trials shown in the previous section have the well known side effect that their accuracy is highly dependent on temperature. And although methods such as adjusting bias currents with temperature are available on the FPAA, we present a temperature invariant method here to address this potential variability in the Bernoulli trials presented previously. These chaos circuits were built as follow-up work to the Telluride workshop.

Chaos circuits were chosen to exemplify a more temperature invariant method. The model and explanation for the low-power chaos circuit used in this paper is first presented in Dudek and Juncu (2005).

A chaos circuit works by introducing a seed value to a non-linear "chaos map" circuit which is itself chaotic. The sample and hold circuit then captures a continuous voltage value for consumption by a stochastic algorithm. The chaos map from Dudek and Juncu (2005) was chosen because of its proven results, but also because it only requires nine transistors and is extremely energy efficient.

The resulting chaos map with a tunable control voltage to dictate the probability characteristics is shown in **Figure 7**.

While further reading may be needed to understand the chaos circuit map shown in **Figure 7**, this map is very close to the results expected as shown in the literature. The general idea is that a given output voltage will result in a random assignment to the chaos map, allowing us to generate random variables in a temperature invariant way. The idea is that this chaos map could be used in place of the thermal noise circuits should the designer be concerned about temperature.

These circuits all have something in common: they can be used to directly compute a stochastic function, cannot be compiled on a digital chip, and compute more efficiently than a digital system.

Next we show how the usefulness of these circuits in a dynamical system.

## 3. GILLESPIE'S ALGORITHM FOR STOCHASTIC COMPUTATION

The previous findings are used to generate the results of a chemical and biological system using Gillespie's algorithm in this

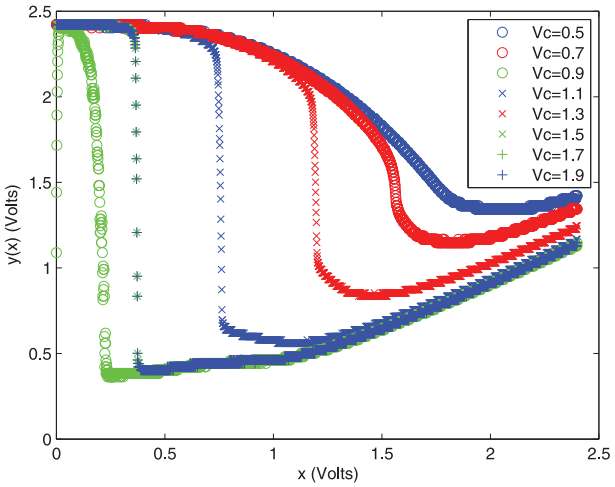



**FIGURE 7 | Measured y(x) voltage vs. $V_c$ for the chaos map circuit compiled on the FPAA.**

section. This section will also review the expense to calculate the trajectory of stochastic systems in software as a comparison. Gillespie's algorithm is a natively probabilistic algorithm that takes advantage of the naturally stochastic trajectories of molecular reactions (Gillespie, 1976), this algorithm is described below.

### 3.1. GILLESPIE'S ALGORITHM

1. Initialize. Set the initial number of each type of molecule in the system and time, $t = 0$.
2. For each reaction **i**, calculate the propensity function to obtain parameter value, $a_i$.
3. For each **i**, generate a reaction time $\tau_i$ according to an exponential distribution with parameter $a_i$.
4. Let $\mu$ be the reaction time whose time is least, $\tau_\mu$.
5. Change the number of molecules to reflect execution of reaction $\mu$. Set $t = t + \tau_\mu$.
6. If initialized time or reaction constraints met, finished. If not, go to step 2.

We use complexity analysis, or big-Oh, analysis to analyze the algorithms here where $O(x)$ gives an expression, $x$, that represents the worst case running time of the algorithm. , Only algorithmic improvements in software have been made to computing Gillespie's algorithm, until this work, such as Gibson *et al.* who have improved the running time of the algorithm from $O(Er)$ to $O(r + Elogr)$ where $E$ is the number of reaction events in the trajectory and $r$ is the number of different reaction types (Gibson and Bruck, 1998). Several orders of magnitude improvement in energy efficiency and performance can be realized by computing the exponentially distributed random variable $\tau_i$ in hardware. Note that the big-Oh function does not change, just the way we implement this algorithm is much improved.

The generation of the exponentially distributed random number is the bottleneck of the algorithm, and the computational complexity of each step is calculated to show this. The metric used to judge computational cost is the number of clock cycles it takes to do a given calculation on a modern general purpose CPU as described in Patterson and Hennessy (2004).

A load instruction is used to initialize a variable in Step 1. With the best case with a multiple data fetch scheme such as in the cell processor, this requires a single computational step. The propensity function $a_i$ in Step 2 is calculated by a floating point multiplication (FPMUL), which takes five computational steps in a modern processor per reaction (Gillespie, 1976; Patterson and Hennessy, 2004). All $r$ reactions assuming $\delta$ FPMUL units are available takes $\frac{5r}{\delta}$ total computational steps. In Step 4, the minimum reaction time $\tau_\mu$ takes $r - 1$ compare operations. Assuming $\delta$ ALU (compare) units are available, Step 4 takes $\frac{r-1}{\delta}$ computational steps. Step 5 involves $r - 1$ integer addition/subtraction operations taking again $\frac{r-1}{\delta}$ computational steps. Step 6 is an update in the program counter resulting in a single step. Step 3 is a key step where each $\tau_i$ according to an exponential distribution. Generating an exponentially distributed random number is complex and deserves a bit more treatment.

This is believed to be the first method to generate an exponentially distributed random number in hardware, and the random numbesr generated by other current methods is only *pseudo-random*. The Park–Miller algorithm on a modern advanced processor is the best known software method where a uniformly pseudo-random number is generated in 88 computational steps (Park and Miller, 1998; Patterson and Hennessy, 2004). The equation to transform a uniformly distributed random variable $U$ to one with an exponential distribution $E$ with parameter $\lambda$ is shown in Equation (2).

$$E = \frac{-\ln U}{\lambda} \tag{2}$$

The natural logarithm function, ln is extremely expensive, and even in the best case of computing this on a modern digital signal processor (DSP) takes 136 computational steps by itself according to Yang et al. (2002). Thus counting the FP multiply and the FP divide taking 5 steps and 32 steps, respectively (Patterson and Hennessy, 2004), it takes a total of 261 computational steps to generate a single exponentially distributed pseudo-random variable in software. Thus Step 3 alone takes $261r$ computational steps to generate $\tau_i$ for all $i$ reactions. To review the number of computational steps for each part of the algorithm is shown below.

### 3.2. COMPUTATIONAL STEPS IN GILLESPIE'S ALGORITHM

| Algorithmic step | Computational steps |
|---|---|
| (1)Initialize. | 1 |
| (2)Multiply to find each $a_i$. | $\frac{5r}{\delta}$ |
| (3)Generate each $\tau_i$. | $261r$ |
| (4)Find $\tau_\mu$. | $\frac{r-1}{\delta}$ |
| (5)Update. | $\frac{r-1}{\delta}$ |
| (6)Go to step 2 | 1 |

Thus for a conservative value of $\delta = 2$, generating each exponentially distributed $\tau_i$ in Step 3 takes approximately 98% of the computational steps for a single iteration of Gillespie's algorithm. Seen in this light, the problem of improving exponential random variable generation becomes quite an important one.

### 3.3. EXPANSION TO ANY DYNAMICAL SYSTEM

It has been shown in Gillespie (1976) and Kurtz (1972) that the trajectory of a chemical system consisting of $N$ different reactant concentrations can be expressed as both a system of deterministic differential equations and a system of stochastic processes. For deeper understanding of Equations (3–5) that follows, the reader is encouraged to read these aforementioned references. This concept can be generalized to any dynamical system where the time evolution of a system of $N$ state variables that has been described in a classical, state-based deterministic model as

$$\frac{dx_1}{dt} = f_1(x_1, x_2, x_3, \ldots)$$
$$\frac{dx_2}{dt} = f_2(x_1, x_2, x_3, \ldots)$$
$$\ldots$$

and in general as:

$$\dot{\mathbf{X}} = F(\mathbf{X}) \tag{3}$$

This system can also be expressed as a stochastic, Markov process.

For the stochastic system, the probability that state variable $x_\mu$ is updated during the next time interval $\tau$ is assigned, $P(\tau, \mu)d\tau$. More formally, this is the joint probability density function at time $t$ expressing the probability that the next state update will occur in the differential time interval $(t + \tau, t + \tau + d\tau)$ and that this update will occur to state variable $x_\mu$ for $\mu = 1 \ldots N$ and $0 \leq \tau < \infty$.

Given probability $P_0(\tau)$, the probability that no state-space update occurs in the interval $(t, t + \tau)$, and the probability, $\alpha_\mu$ that an update to state $x_\mu$ will occur in the differential time interval $(t + \tau, t + \tau + d\tau)$ we have the general form for the joint probability function:

$$P(\tau, \mu)d\tau = P_0(\tau) \cdot \alpha_\mu d\tau \tag{4}$$

Note that $\alpha_\mu$ is based on the state of the system $\mathbf{X}$. Also note that determining $\alpha_\mu$ is the critical factor in determining the Markov process representation and no general method for this is given here. In the chemical system example, $\alpha_\mu$ is the probability that reaction $R_\mu$ is going to occur in the differential time interval and is a function of the number of each type of molecule currently in the system. The probability that more than one state update will occur during the differential time interval is shown to be small and thus ignored (Kurtz, 1972; Gillespie, 1976). Finally some function $g_\mu$ must be given describing the update to state variable $x_\mu$ once an update occurs. We then have the stochastic, Markov process defined for the system:

$$Pr[\mathbf{X(t} + \tau + \mathbf{d}\tau) = \mathbf{G(X(t))} \mid \mathbf{X(t)}] = \mathbf{P}(\tau, \mu) \tag{5}$$

Note that this formulation does not make the assumption that infinitesimal $d\tau$ need be approximated by a finite time step $\Delta\tau$, which is a source of error in many Monte Carlo formulations.

To solve this system using computational methods, random numbers are generated according to the probability distributions described by Equation (5). No matter what dynamical system is involved, exponentially distributed random numbers will always be needed. To calculate $P_0(\tau)$ from Equation (4), we break the interval $(t, t + \tau)$ into $K$ subintervals of equal length $\epsilon = \tau/K$, and calculate the probability that no state update occurs in the first $\epsilon$ subinterval $(t, t + \epsilon)$ which is:

$$\prod_{i=1}^{N} [1 - \alpha_i\epsilon + o(\epsilon)] = 1 - \sum_{i=1}^{N} \alpha_i\epsilon + o(\epsilon) \tag{6}$$

This probability is equal for every subinterval $(t, t + 2\epsilon)$, $(t, t + 3\epsilon)$, and so on. Thus the probability $P_0(\tau)$ for all $K$ subintervals is:

$$P_0(\tau) = \lim_{K\to\infty} \left[1 - \sum_{i=1}^{N} \alpha_i\epsilon + o(\epsilon)\right]^K \tag{7}$$

$$= \lim_{K\to\infty} \left[1 - \sum_{i=1}^{N} \alpha_i\tau/K + o(\tau/K)\right]^K \tag{8}$$

Where $o(\epsilon)$ is the probability that more than one event occurs in the time interval $\epsilon$. Following the analysis in Gillespie (1976), we assume as our incremental time interval goes to zero our function $o(\epsilon) \to 0$ as well. With $o(\tau/K) \to 0$ we are left with the probabilistic, exponential function in Equation (9).

$$P_0(\tau) = exp\left[-\sum_{i=1}^{N} \alpha_i\tau\right] \tag{9}$$

Thus we prove how this work can be extended to any dynamical system.

## 4. RECONFIGURABLE ANALOG HARDWARE FOR STOCHASTIC COMPUTATION

These complex probability functions are generated on a reconfigurable platform, reviewed in this section. More specifically, a dynamic *Bernoulli Trial* generator is illustrated on chip. This novel method involves using the reconfigurable analog signal processor (RASP) chip that was recently introduced (Basu et al., 2008). This device allows one to go from concept to full functionality for analog circuits in a matter of weeks or even days instead of months or years for large-scale, integrated circuits. The design presented went from concept to measured hardware in a matter of 2 weeks. The other useful feature is that many FPGA type architectures allow a designer to build a subset of the possible circuits available with the RASP. Circuits such probabilistic function generators could not be produced on strictly digital reconfigurable architectures although digital designs can be built on the RASP. The RASP chip is shown in **Figure 8**.

### 4.1. STOCHASTIC CIRCUIT ARCHITECTURE

The macroarchitecture and details of the algorithmic implementation via Bernoulli trials and how this is built on the RASP chip is explored here. The RASP has 32 reconfigurable, computational analog blocks (CABs). The elements of a CAB and the elements that are used in this design are shown in **Figure 9**.

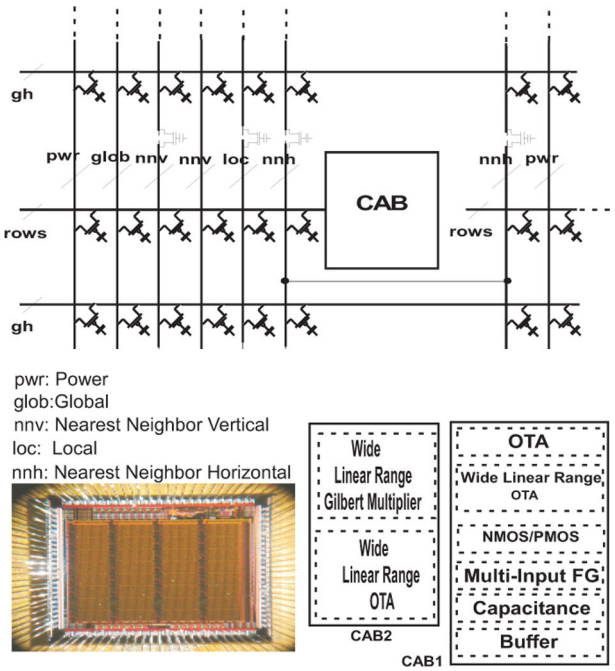



**FIGURE 8 | Micrograph of the Reconfigurable Analog Signal Processor (RASP), also referred to the Field Programmable Analog Array (FPAA).** The circuits were compiled onto this device. Computational Analog Blocks (CAB) populate the chip where both the computational elements and the routing is configured via floating gates. Buffers, capacitors, transmission gates, NMOS, PMOS, floating gates, current biases, adaptive amplifiers, and analog multiply arrays are all available and fully programmable on the chip. Development is shared by many in the CADSP group at Georgia Tech.

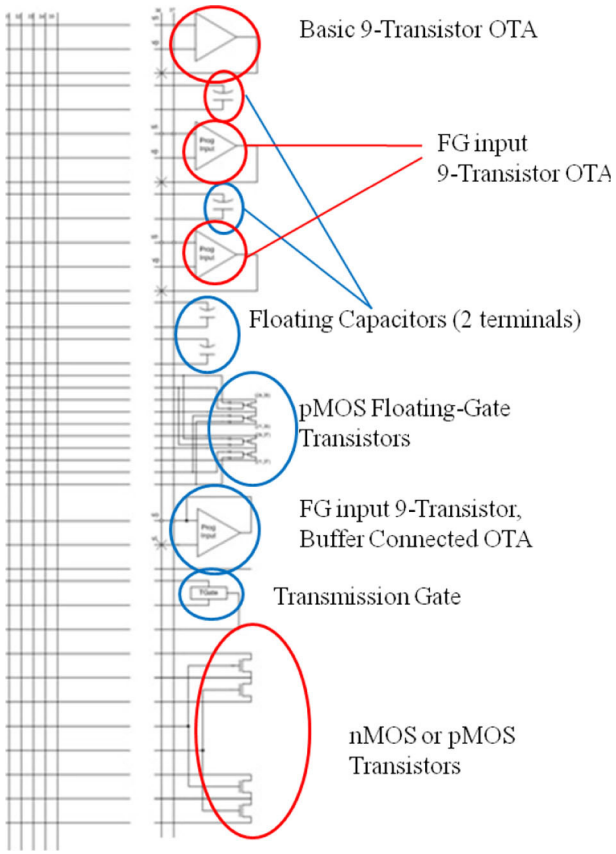



**FIGURE 9 | The typical routing structure and available analog computational elements in one of the 32 computational analog blocks (CABs) present on the RASP chip.** The elements used in the present design are highlighted in red. Alternating CABs have NMOS transistors and PMOS as the bottom CAB element, and although only three of the MOSFETs are used in our circuit from **Figure 3**, instead of the four that are circled, this is meant to show several different combinations of these four transistors can be used to make the three transistor circuit. The top triangle elements are an OTAs showing the two inputs and output being routed back into the fabric which is a mesh of lines with switches able to connect any two lines that cross in the mesh. The element below the OTAs is a capacitor and the bottom elements circled are NMOS and PMOS available to be routed.

An array of up to 32 Bernoulli trials can be calculated simultaneously on a single RASP device. New versions of the RASP have been fabricated in 350, 130, and 45 nm; an entire family of RASP 2.9 chip variants exist for different applications spaces, which allow as much as 10X this number of Bernoulli trials, and this scales with Moore's law. The RASP chipset and accompanying tools also have the ability to be linked together easily for a multi-core RASP chipset should more Bernoulli generators be needed. The RASP chipset is useful for a proof of concept here. Since each Bernoulli generator only takes 30 transistors, many thousands of these circuits could be built in custom hardware if needed.

## 5. CHIP MEASUREMENTS AND EXPERIMENTAL RESULTS

To gather data, the Probability Select line described in **Figure 3A** and the output producing the random numbers were routed to the primary inputs/outputs of the chip. A 40-channel, 14-bit digital-to-analog-converter (DAC) chip was interfaced with the RASP chip on a printed circuit board, which we used as our testing apparatus, so that any arbitrary voltage could be input to the Probability Select line. This chip is 40 channel in the sense that there are 40 independent outputs of the DAC. The outputs of the RASP chip were connected to oscilloscope probes so that the noise spectrum and random numbers could be captured.

The distribution of the Bernoulli trial circuits were measured in the following way: 2500 random numbers were captured at each Probability Select voltage. The number of successes was divided by the total number of samples captured at each voltage to calculate the probability of a Bernoulli success (probability of randomly generating a 1). The results are shown in **Figure 10**.

An example of a voltage signal from the Dynamic Bernoulli Probability Circuit producing a digital "1" with probability $p = 0.90$ is shown in **Figure 11**. The voltage signal was recorded via on-chip measurement circuits and transmitted to a PC through a USB connection to the chipset.

The array of Bernoulli trials was encoded and the exponential distribution of reaction times, $\tau_i$, was generated. The resulting distribution is what one would expect and matches a true, exponential distribution as shown in **Figure 12**.

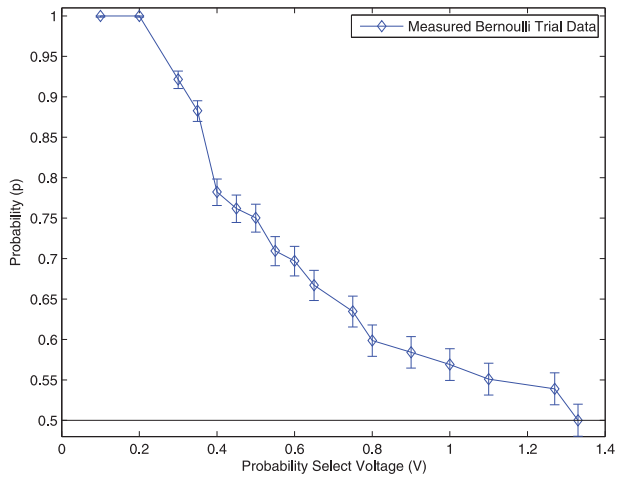



**FIGURE 10 | Measurements from the output of the Dynamic Bernoulli Probability Circuit shown in Figure 3 were taken.** The "Probability Select" voltage was adjusted from 0.1 volts up to 1.4 volts and the resulting probability of a digital "1" being produced was recorded with a 95% confidence interval. Measurements were only taken down to $p = 0.5$ since a Bernoulli trial is symmetric about this value.

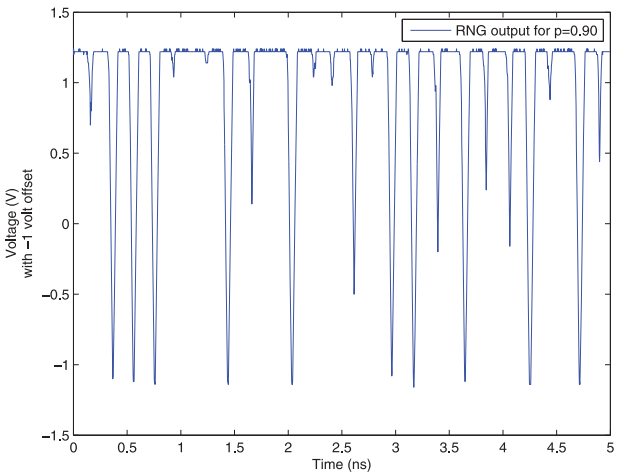



**FIGURE 11 | Voltage output from Dynamic Bernoulli Probability Circuit also called a random number generator (RNG) circuit as labeled in the graph, corresponding to the output out of the third (final) OTA circuit from Figure 3.** A 1 volt offset was arbitrarily chosen for the comparator, but other voltage offset values than this were anecdotally observed to have undesired noise effects resulting in spurious switching at the output of the comparator. The noise amplifiers, and thus comparators, were observed to switch at approximately 208 ps as a maximum rate, thus independent Bernoulli variables could be produced at most at this time period in this particular device. A digital "1" is produced with probability $p = 0.90$ and "0" with $1 - p = 0.10$. When the voltage is above the threshold $V_{out} > \frac{V_{dd}-V_{ss}}{2}$ it is considered a digital "1" and otherwise a "0," where the threshold happens to be 0 volts in this case. The samples in this circuit change much faster than can be consumed, and thus random samples are taken from the output of this circuit at a slower rate than the rate at which it changes state, but preserving randomness.

### 5.1. VALIDATION OF RANDOMNESS

A *probabilistic* output and a *random* output are differing concepts, and the ability to control this difference is the strength of the proposed circuits. They are linked together and defined through Shannon's entropy (Shannon, 1949). Formally, entropy and thus the randomness of a function are defined by $H$ in Equations (10, 11).

Let

$$H_n = -\frac{1}{n} \sum_{i,j,\ldots,s} p(i, j, \ldots, s) \log_2 p(i, j, \ldots, s) \tag{10}$$

Then entropy is

$$H = \lim_{n \to \infty} H_n \tag{11}$$

where $p(i, j, \ldots, s)$ is the probability of the sequence of symbols $i, j, \ldots, s$, and the sum is over all sequences of $n$ symbols.

By the same definition, a function exhibits the *most randomness* if $H$ is maximized, which occurs when all output sequences are equally likely, or equivalently, if all possible outputs have an equal probability of occurring (Shannon, 1949). From this work, a function is defined as *random* if all outputs have a uniform probability of occurring. Conversely, we define a function as *probabilistic* if the function has an entropy $0 < H < \log_2 n$.

There exist statistical measures of randomness, developed by the National Institute of Standards and Technology (NIST), in the form of a suite consisting of 16 independent tests to measure *how random* a sequence of numbers truly is Random Number Generation and Testing (2014). However, these tests only measure the performance of *random* functions and not *probabilistic* ones such as the circuits presented in this work, although *random* number generation (when $p = 0.5$) is a subset function of these circuits.

Each of the tests are measured on a scale from 0 to 1, where a "passing" mark is considered >0.93 and higher marks indicate a higher quality sequence. For a random output $p = 0.5$ these circuits with thermal noise as the source of randomness passed all but the "Overlapping Template Matching Test" and the "Lempel-Ziv Complexity Test" and even these two tests received high marks >0.80. They also perform consistently better than the software generated, Park–Miller *psuedo-random* numbers used by most algorithms, which failed half the tests in the suite with some failing badly <0.10 (Chakrapani et al., 2006).

## 6. CONCLUSIONS AND FUTURE DIRECTIONS

It was shown in section 3 that to generate an exponentially distributed random variable in software takes a minimum of 261 computational steps with the Park Miller algorithm. And with hardware random number generators shown in previous microprocessor works, only uniformly random numbers were available. Bernoulli trials are generated here in hardware with a single computational step, and an exponentially distributed random number is generated with two computational steps.

Because of the high gain of our amplifiers, the thermal noise distribution used to generate probabilistic distributions with our hardware is extremely sensitive to perturbations such as ambient electrostatic interactions, device variations, and changes in

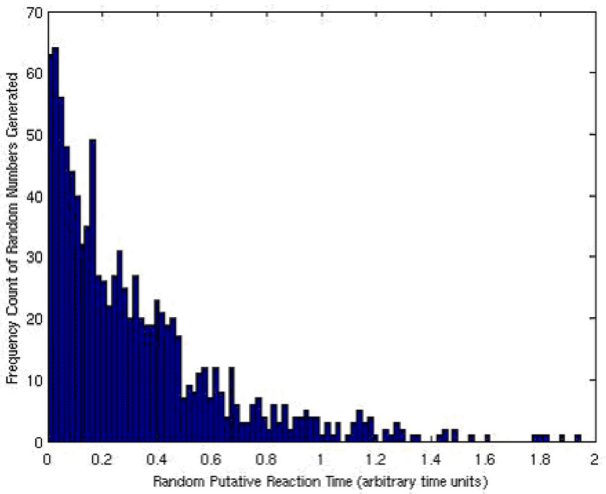



**FIGURE 12 | The histogram (and thus a scaled probability distribution function) of the exponentially distributed Gillespie reaction times, $\tau_i$'s, generated.**

ambient temperature. Environment invariant chaos circuis were compiled and measured to mitigate these concerns. Because the Bernoulli trial circuits presented here can be controlled via a programmable input signal, software calibration can be done to mitigate these concerns as well.

An estimated performance increase of approximately 130X is realized based on measured results to generate exponentially distributed random numbers. With the assumption used in section 3 that generating exponential random variables takes up 98% of the computation time of a single iteration through Gillespie's algorithm, our system could potentially speed up the calculation of the trajectory of this algorithm by approximately 127X.

Further is was shown that these performance increases via hardware generated probabilistic distributions can be applied to *any dynamical system* and possibly have a much wider impact than the field of biological computations.

Such a method to increase computational efficiency by two orders of magnitude is believed to be widely useful in calculating biological, statistical, or quantum mechanical systems. The search for meta-materials new medicines, or any other host of applications could benefit. Future directions for this specific work include attempting to hardware accelerate quantum algorithms and venturing into the world of software defined analog radios.

## ACKNOWLEDGMENTS

This project was funded in part by the National Science Foundation, NSF award ID 0726969 and the Defense Advanced Research Project Agency. The authors would also like to thank the organizers and participants of the 2008 Telluride Neuromorphic Workshop and the Institute for Neuromorphic Engineering. Dr. Marr is currently at Raytheon company and Dr. Hasler is a professor at Georgia Tech where Dr. Marr performed the work. Finally the authors would like to thank Raytheon for providing funding to continue and publish the research.

## REFERENCES

Basu, A., Twigg, C. M., Brink, S., Hasler, P., Petre, C., Ramakrishnan, S., et al. (2008). "Rasp 2.8: a new generation of floating-gate based field programmable analog array," in *Proceedings, Custom Integrated Circuits Conference CICC* (San Jose, CA).

Chakrapani, L., Akgul, B. E. S., Cheemalavagu, S., Korkmaz, P., Palem, K., and Seshasayee, B. (2006). "Ultra-efficient (embedded) SOC architectures based on probabilistic cmos (PCMOS) technology," in *Proceedings of Design Automation and Test in Europe (DATE)* (Munich).

Douglas, R. (2008). *Modeling Development of the Cortex in 3d: From Precursors to Circuits*. Available online at: https://neuromorphs.net/ws2008/

Dudek, P., and Juncu, V. (2005). "An area and power efficient discrete-time chaos generator circuit," in *Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005, IEEE* 2, II–87.

Genov, R., and Cauwenberghs, G. (2001). "Stochastic mixed-signal vlsi architecture for high-dimensional kernel machines," in *Advances in Neural Information Processing Systems* (Vancouver, British Columbia), 1099–1105.

Gibson, M., and Bruck, J. (1998). *An Efficient Algorithm for Generating Trajectories of Stochastic Gene Regulation Reactions*. Technical Report, California Institute of Technology.

Gillespie, D. T. (1976). A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys.* 22, 403–434.

Goldberg, D. H., Cauwenberghs, G., and Andreou, A. G. (2001). "Analog vlsi spiking neural network with address domain probabilistic synapses," in *International Symposium on Circuits and Systems* (Sydney).

Kish, L. B. (2002). End of Moore's law: thermal (noise) death of integration in micro and nano electronics. *Phys. Lett. A* 305, 144–149. doi: 10.1016/S0375-9601(02)01365-8

Kurtz, T. G. (1972). The relationship between stochastic and deterministic models for chemical reactions. *J. Chem. Phys.* 57, 2976–2978. doi: 10.1063/1.1678692

Ohba, R., Matsushita, D., Muraoka, K., Yasuda, S., Tanamoto, T., Uchida, K., et al. (2006). "Si nanocrystal mosfet with silicon nitride tunnel insulator for high-rate random number generator," in *Emerging VLSI Technologies and Architectures* (Karlsruhe).

Oppenheim, I., Shuler, K. E., and Weiss, G. H. (1969). Stochastic and deterministic formulation of chemical rate equations. *J. Chem. Phys.* 50, 460–466. doi: 10.1063/1.1670820

Park, S., and Miller, K. (1998). Random number generators: good ones are hard to find. *Commun. ACM* 31, 10.

Patterson, D., and Hennessy, J. (2004). *Computer Organization and Design, 3rd Edn*. Boston: Morgan Kauffman.

Random Number Generation and Testing. (2014). Available online at: http://csrc.nist.gov/rng/

Shannon, C. (1949). "Communication is the presence of noise," in *Proceedings of the I.R.E.* (New York, NY), 10–21.

Xu, P., Horiuchi, T. K., and Abshire, P. A. (1972). Compact floating-gate true random number generator. *Electron. Lett.* 42, 23. doi: 10.1109/JRPROC.1949.232969

Yang, M., Wang, Y., Wang, J., and Zheng, S. Q. (2002). "Optimized scheduling and mapping of logarithm and arctangent functions on ti tms320c67x processor," in *IEEE International Conference on Aucoustics, Speech, and Signal Processing* (Orlando, FL). doi: 10.1109/ICASSP.2002.5745319

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 01 October 2013; accepted: 04 April 2014; published online: 07 May 2014.*
*Citation: Marr B and Hasler J (2014) Compiling probabilistic, bio-inspired circuits on a field programmable analog array. Front. Neurosci.* ***8****:86. doi: 10.3389/fnins.2014.00086*
*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 02 April 2014
doi: 10.3389/fnins.2014.00054

# An adaptable neuromorphic model of orientation selectivity based on floating gate dynamics

***Priti Gupta* and C. M. Markan***

*VLSI Design Technology Lab, Department of Physics and Computer Science, Dayalbagh Educational Institute, Agra, Uttar Pradesh, India*

***Edited by:***
*Jennifer Hasler, Georgia Insitute of Technology, USA*

***Reviewed by:***
*Shantanu Chakrabartty, Michigan State University, USA*
*Milutin Stanacevic, Stony Brook University, USA*
*Bradley A. Minch, Franklin W. Olin College of Engineering, USA*

****Correspondence:***
*Priti Gupta, VLSI Design Technology Lab, Department of Physics and Computer Science, Faculty of Science, Dayalbagh Educational Institute, Agra-282005, Uttar Pradesh, India*
*e-mail: gupta.priti.84@gmail.com*

The biggest challenge that the neuromorphic community faces today is to build systems that can be considered truly cognitive. Adaptation and self-organization are the two basic principles that underlie any cognitive function that the brain performs. If we can replicate this behavior in hardware, we move a step closer to our goal of having cognitive neuromorphic systems. Adaptive feature selectivity is a mechanism by which nature optimizes resources so as to have greater acuity for more abundant features. Developing neuromorphic feature maps can help design generic machines that can emulate this adaptive behavior. Most neuromorphic models that have attempted to build self-organizing systems, follow the approach of modeling abstract theoretical frameworks in hardware. While this is good from a modeling and analysis perspective, it may not lead to the most efficient hardware. On the other hand, exploiting hardware dynamics to build adaptive systems rather than forcing the hardware to behave like mathematical equations, seems to be a more robust methodology when it comes to developing actual hardware for real world applications. In this paper we use a novel time-staggered Winner Take All circuit, that exploits the adaptation dynamics of floating gate transistors, to model an adaptive cortical cell that demonstrates *Orientation Selectivity*, a well-known biological phenomenon observed in the visual cortex. The cell performs competitive learning, refining its weights in response to input patterns resembling different oriented bars, becoming selective to a particular oriented pattern. Different analysis performed on the cell such as orientation tuning, application of abnormal inputs, response to spatial frequency and periodic patterns reveal close similarity between our cell and its biological counterpart. Embedded in a RC grid, these cells interact diffusively exhibiting cluster formation, making way for adaptively building orientation selective maps in silicon.

**Keywords: feature maps, orientation selectivity, time-staggered WTA, floating gate synapses**

## 1. INTRODUCTION

The past decade has been a landmark decade in the progress of Neuromorphic Engineering. Technological advances have paved the way for large scale neural chips having millions of neurons and synapses (Indiveri et al., 2006; Bartolozzi and Indiveri, 2007; Wijekoon and Dudek, 2008). We now have silicon cochleas and retinas (Chan et al., 2007; Lichtsteiner et al., 2008). A number of groups around the world have built large scale multichip neuromorphic systems for real time sensory processing with programmable network topologies and reusable AER infrastructure (Serrano-Gotarredona et al., 2005; Chicca et al., 2007; Merolla et al., 2007; Schemmel et al., 2008). All these approaches can be broadly classified into analog, digital or hybrid approaches. The analog approach interfaces well with the real world, emulates bio-inspired behavior more closely and is most suited for modeling local neural computations. Digital systems on the other hand efficiently exploit addressing mechanisms to emulate long distance communication in the brain. Therefore, an amalgamation of the digital and analog approaches i.e., the hybrid approach, is most appropriate for implementing large scale neuromorphic systems. The challenge that now lies ahead is to develop truly brain like cognitive systems. Systems that can adapt, self-organize and learn according to cues in their environment (Indiveri et al., 2009; Indiveri and Horiuchi, 2011).

A major step toward building such systems would be to understand the underlying principles that the brain uses to accomplish adaptation. It is well accepted now that very early in development the brain has a generic cortical structure that adapts to the environment by forming neural connections during the critical learning period (Sur and Leamey, 2001; Horng and Sur, 2006). This kind of adaptation leads to the formation of feature maps or interconnectivity patterns between hierarchically organized layers of the cortices. The lower layers extract basic features from the input space so that higher layers can extract more complex features, using the information from the lower layers. Both *Nature* (genetic biases) and *Nurture* (environmental factors) play a crucial role in the formation of these feature maps. Different hardware and software approaches have been explored to model self-organization. Each approach has a set of mechanisms that exploit the available techniques. While models built in software prefer to use mathematical equations, attempting to do the same in hardware can turn out to be extremely cumbersome (Kohonen, 1993, 2006; Martn-del-Bro and Blasco-Alberto, 1995; Hikawa et al., 2007). On the other hand, understanding

the hardware dynamics and then building adaptive algorithms around it seems to be a more robust approach for building real world applications.

To emulate activity dependent adaptation of synaptic connections in electronic devices, we look towards the developing brain for inspiration. In the developing brain, different axons connecting to a post synaptic cell, compete for the maintenance of their synapses. This competition results in synapse refinement leading to the loss of some synapses or synapse elimination (Lichtman, 2009; Misgeld, 2011; Turney and Lichtman, 2012; Carrillo et al., 2013). Temporarily correlated activity prevents this competition whereas uncorrelated activity seems to enhance it (Wyatt and Balice-Gordon, 2003; Personius et al., 2007). Moreover, precise spike timing plays a key role in this process e.g., when activity at two synapses is separated by 20 ms or less, the activity is perceived as synchronous and the elimination is prevented (Favero et al., 2012). Apart from the biological relevance, synapse elimination as a means of honing neural connections is also suitable for implementation in large scale VLSI networks because in analog hardware it is difficult to create new connections but it is possible to stop using some connections. Although some digital approaches work around this by using virtual connections using the Address Event Representation, however, in purely analog designs for ease of management of large scale connections, synapse elimination is best suited. In order to implement synapse pruning we need to have non-volatile adaptable synapses which are best represented by floating gate synapse or memresistors (Zamarreño-Ramos et al., 2011). While memresistor technology is still in development floating gate transistors have gained widespread acceptance due to their capacity to retain charge for very long periods and the ease and accuracy with which they can be programmed during operation (Srinivasan et al., 2005). Floating gate memories are being used for various applications like pattern classification (Chakrabartty and Cauwenberghs, 2007), sensor data logging (Chenling and Chakrabartty, 2012), reducing mismatch (Shuo and Basu, 2011) etc. They have also found extensive application in neuromorphic systems (Diorio et al., 1996; Hsu et al., 2002; Markan et al., 2013). We therefore extend the study of adaptive behavior of floating gate pFETs and demonstrate how this adaptive, competitive and cooperative behavior can be used to design neuromorphic hardware that exhibits orientation selectivity, a widely studied phenomenon observed in the visual cortex.

Prior efforts toward hardware realization of orientation selectivity can be classified into two categories, (1) Ice Cube models, (2) Plastic models. Ice cube models e.g., the model by Choi et al. (2005) assumes prewired feed-forward and lateral connections. Another similar model by Shi et al. (2006) uses DSP and FPGA chips to build a multichip modular architecture. They use Gabor filters to implement orientation selectivity. This approach provides an excellent platform for experimentation with feature maps, however, it falls short when it comes to compactness and power efficiency. Moreover, these models do not capture the developmental aspects of orientation selectivity. Some plastic models that try to capture the developmental aspects include the model by Chicca et al. (2007) that uses a mixed software/hardware approach to simulate a biologically realistic algorithm on a PC that is interfaced with a neuromorphic vision sensor. Another model by Boahen et al. (Taba and Boahen, 2002; Lam et al., 2005) uses activity dependent axon remodeling by using the concept of axonal growth cones and implements virtual connections by re-routing address events. Their design is biologically realistic but hardware intensive since they use an additional latency circuit to decide the wining growth cone. Therefore, what is needed is an approach that is more autonomous in terms of deciding the winner in the competition. Through our approach, that is based on the biologically inspired synapse elimination process, we have attempted to build an analog design that can be used by both analog and hybrid systems. The design has minimum hardware requirements and is capable of self-organized clustering. Our effort in designing a minimal competitive circuit, the time-staggered Winner Take All (ts-WTA) (**Figures 1A–D**) that exploits the adaptation dynamics of floating gate pFETs (Markan et al., 2013) and then using a collective network of these ts-WTA cells to exhibit orientation selectivity (Markan et al., 2007) is a small yet significant effort toward bridging the gap between biological phenomenon and its neuromorphic equivalent. The simulations were performed using Tanner T-Spice v13.0 and Cadence Specter v7.1 with BSIM3 level 49 spice models for 0.35 $\mu$m CMOS process.

Section 2 attempts to highlight the salient features of the ts-WTA circuit and discusses the motivation behind its design. Section 3 describes the development of a framework for multi-dimensional feature selectivity which is then extended to create an orientation selective cortical cell model that learns and eventually recognizes patterns resembling bars of different orientations. In sections 3 and 4, experiments performed on the orientation selective cortical cell, that highlight how close the cortical cell is to its biological counterpart, are discussed. Section 5 describes a framework for diffusive interaction and cluster formation between many orientation selective cells that has implications in orientation selective map formation. Section 6 includes the results and discussion.

## 2. TIME-STAGGERED WINNER TAKE ALL

A novel CMOS time-staggered Winner Take All (ts-WTA) circuit has been described in Markan et al. (2013). The ts-WTA is built with two arms each representing a weighted connection, implemented by means of floating gate pFET "synapses" (**Figure 1A**) (Rahimi et al., 2002). These arms connect at a common source node, $V_s$. Current through a bias pFET, also connected at $V_s$, drives the two arms of the ts-WTA and ensures resource limitation. A buffer device (D) separating $V_s$ from $V_i$ is introduced to ensure that $V_s$ is not influenced directly by the neighboring cells. However, the voltages at $V_s$ and $V_i$ are nearly the same. A feedback mechanism modifies the floating gate voltages of the two floating gate pFET synapses as a function of the activation node voltage $V_i$. The Tunnel (T) and Injection (I) devices, that are a part of the feedback network (**Figure 2**), transform the $V_i$ to appropriate ranges that make tunnel and injection feasible. The initial floating gate voltages of the two synapses are chosen randomly with a small voltage difference $\delta V_{fg}$. Inputs to the cell are applied in the form of pulses of high (6v) and low ($-$1v) voltage represented by 1 and 0, respectively. A {0,0} input means both the synapses are stimulated with $-$1v which is equivalent to saying they are both off . An input {1,1} means that both synapses

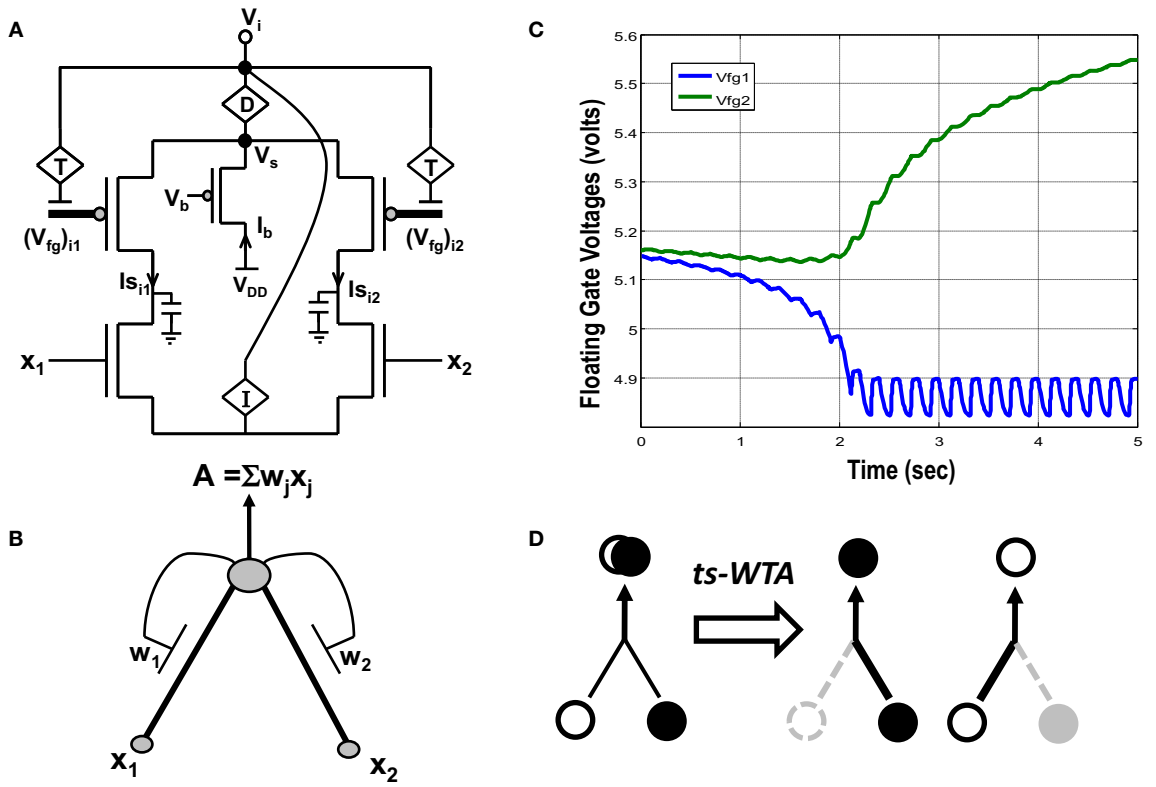



**FIGURE 1 | (A)** Actual circuit of the ts-WTA learning cell and **(B)** its abstract model. In **(A)** $(V_{fg})_{i1}$, $(V_{fg})_{i2}$, and in **(B)** W1, W2 show the floating gate based weighted connections. $x_1$, $x_2$ are inputs and node voltage $V_i$ is activation of the cell which is equivalent to A in **(B)**. **(C)** Shows ts-WTA evolution of floating gate voltages. **(D)** Starting with nearly equal weak connections (left), the cell strengthens stronger of the two connections at the cost of the other (right, shows both possibilities). Here ∘ => connection representing one feature • => connection representing other feature.

are stimulated with a 6v pulse at the same time, which is how conventional WTA circuits receive inputs. The inputs {1,0} and {0,1} mean that the synapses are stimulated alternately or in an uncorrelated manner. The ts-WTA is designed to work on this uncorrelated scheme of inputs. When inputs from the sets {0,1} and {1, 0} are applied at $x_1$ and $x_2$ in a *random-inside-epoch* order (i.e., within an epoch both the synapses are equally stimulated but the order in which they are stimulated is randomized for every epoch) competition between the two arms starts taking place. The below equation expresses the adaptation dynamics of the floating gate voltage $(V_{fg})$ of any branch (synapse) as a function of $V_{fg}$ of the stimulated branch

$$\frac{d\left(V_{fg}\right)_{ij}}{dt} = F_T\left[T\{V_i\}, \left(V_{fg}\right)_{ij}\right] - F_I\left[I\{V_i\}, \left(V_{fg}\right)_{ij}\right] \times x_j \quad (1)$$

The first part of the equation represents tunneling and the second part represents injection feedback. The second part has an additional term $x_j$, which is 1 when the pFET is ON and 0 when it is OFF, taking into consideration that injection works only when the floating gate transistor is ON whereas, tunneling works at all times irrespective of the state of the floating gate transistor. In the first and second parts, $V_i$ is equivalent to $\sum_j f(V_{fg})_{ij} \times x_j$ (which means we can express $V_i$ in terms of floating gate voltages of individual branches under the condition that only one $x_j$ is 1 the other is 0 at any given time). In the first part $T\{V_i\}$ leads to a tunnel voltage $V_{\text{tun}}$ which along with the floating gate voltage $(V_{fg})$ determines the tunneling current $(I_{\text{tunnel}})$ and in the second part $I\{V_i\}$ leads to an injection voltage $V_{\text{inj}}$ which along with $V_{fg}$ determines the injection current $(I_{\text{injection}})$ (please refer to Markan et al., 2013 and Rahimi et al., 2002 for detailed equations). Injection works by lowering the floating gate voltage, $V_{fg}$, thus making the transistor more and more ON whereas tunneling causes the $V_{fg}$ to increase gradually causing the pFET to slowly drift toward the OFF state. On stimulation by uncorrelated inputs over a period of time, injection amplifies the voltage difference between the two floating gates. Tunnel on the other hand helps in setting an upper limit on strength of the active connection, and pruning the strength of the inactive connection. According to Grossberg (1976), Winner Take All action requires that self-excitation of a neuron must be accompanied by global lateral inhibition. This occurs in ts-WTA with self-excitation in the form of injection and global lateral inhibition in the form of tunneling. If over many epochs, the synapse strengthens more than it weakens (there is more injection than tunneling), the floating gate pFET turns more and more ON, but if the synapse weakens more than it strengthens (tunneling is more than injection)

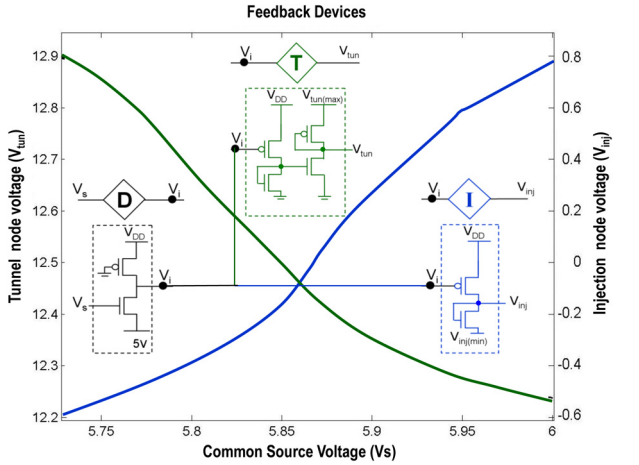



**FIGURE 2 | Shows the circuit level description of the Injection (I), Tunnel (T) and Buffer (D) devices that are a part of the ts-WTA circuit shown in Figure 1A.** The Injection (I) and Tunnel (T) devices modify the voltage at $V_i$ to appropriate ranges that enable injection and tunneling to occur in the floating gate pFETs. The buffer device (D) shields the common source node ($V_s$) from the loading effect of neighboring cells. The graphs show how the tunnel ($V_{tun}$) and injection ($V_{inj}$) voltages vary with the common source voltage. Here $V_{inj(min)}$ is set to −0.65 V, $V_{tun(max)}$ is set to 13.6 V and $V_{DD}$ is 6 v.

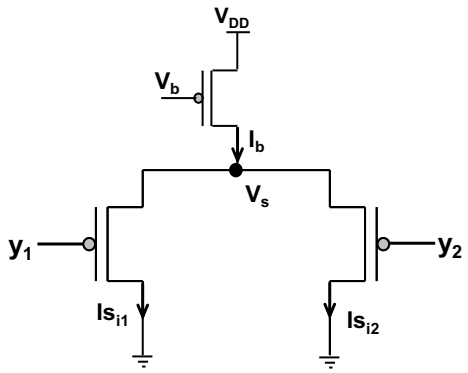



**FIGURE 3 | Shows the Lazzaro's WTA (L-WTA) circuit.** This can be compared with the ts-WTA of **Figure 1A**. In both the circuits, a bias transistor in saturation acts like a current source with constant current $I_b$. This current ensures resource limitation forcing only one of the input arms or "synapse" to survive in the competition. In ts-WTA the inputs $x_1$ and $x_2$ represent voltage pulses of +6 v and −1 v (1 and 0) applied alternately (time-staggered inputs) to both the arms. The inputs $y_1$ and $y_2$ are voltages (equivalent to $x_1w_1$ and $x_2w_2$, respectively) that are applied to the two arms of L-WTA both at the same time. Here $w_1$ and $w_2$ represent the weights of the two floating gate pFET synapses in the ts-WTA. L-WTA performs instantaneous comparison between the two inputs and does not have any memory element. The ts-WTA has floating gate pFET based memory and Tunnel and Injection feedback devices that modify the floating gate voltages as a function of response voltage ($V_i$). This allows it to perform competition based on memory of prior activity unlike the L-WTA that can only perform instantaneous comparison between two inputs that are simultaneously applied.

then after several epochs it reaches a stage of no recovery where the floating gate pFET completely switches OFF. The synapse that strengthens more emerges as the Winner. However, ts-WTA has an additional interesting dimension according to which, if the weaker connection is stimulated more, then that emerges as the winner. Interestingly, this ts-WTA competition can be extended to any two contrasting input synapses (e.g., Left/Right eye in Ocular Dominance, ON/OFF cells in Orientation Selectivity and Lagged/Non-Lagged cells in Direction Selectivity) to perform feature selectivity. It can also be extended to other modalities like auditory, somatosensory etc. Thus the ts-WTA is a very generic cell and can be an essential core around which different feature selectivity models can be built. This is necessary for eventual integration of different feature maps into one universal framework. The ts-WTA has been studied under various stimulation schemes and has been tested for stability under device parameter variations (Markan et al., 2013) and is thus a robust circuit which closely emulates brain like competition and learning and is therefore suitable to build brain like feature maps.

Amongst the various CMOS WTA circuits that have been designed, Lazzaro's WTA(L-WTA) (Lazzaro et al., 1989) has gained widespread acceptance (**Figure 3**). It is an elegant circuit that performs instantaneous comparison between two or more input values and brings about suppression of the outputs associated with lower input values as compared to the highest value giving rise to Winner Take All action. Our ts-WTA is inspired by L-WTA, however, there are significant differences. In both the circuits, a current source restricts the amount of current that can flow in the two competing branches. As a result, the branch that draws more current forces the transistor of the other branch to switch off, thus emerging as a winner. When both inputs are applied at the same time, both ts-WTA and L-WTA behave in much the same way. However, ts-WTA brings in an interesting innovation in the form of long term memory retention using floating gate dynamics. So, in fact, the ts-WTA is a learning WTA cell that is capable of computing a winner based on which input is statistically more significant over many epochs unlike the L-WTA which only computes the winner based on an instantaneous comparison. Another interesting WTA circuit (inspired by L-WTA) that incorporates a sense of time by using floating gate transistors has been developed by Kruger et al. (1997). Their motivation to introduce adaptation is to add a fatigue or refraction time to each cell that wins. Their application is to form saliency maps where there is a need to ensure that the saliency of all inputs is considered and the WTA operation chooses different winners at different times instead of just locking on to the most significant input. Another interesting variant of L-WTA is the one introduced in Indiveri (2001, 2008). In this circuit by using local excitatory feedback and a lateral excitatory coupling mechanism the authors realize distributed hysteresis using which the network is able to lock onto an input with the strongest amplitude and track it as it shifts. They have shown an interesting application of this in adaptive visual tracking sensors (Indiveri et al., 2002). Both these circuits work on the conventional {1,1} or simultaneously applied inputs. They are both ingenious circuits, however, their motivation and design vary significantly from ours.

The true strength of the ts-WTA lies in the way it works on uncorrelated inputs or inputs applied staggered over time.

The inspiration for using time-staggered or uncorrelated inputs comes from the way the brain is designed. In the brain, many pre-synaptic neurons connect to a single post-synaptic neuron through many afferent connections. It is only through correlated or uncorrelated activity between the many pre-synaptic cell afferents that the post-synaptic cell can tell to which pre-synaptic cell the afferents belong. The activity, from all the afferents of one pre-synaptic neuron is perfectly correlated whereas between two different pre-synaptic neurons the activities are uncorrelated and this is the basis on which synapse elimination happens (Stent, 1973). Hence, uncorrelated activity between different pre-synaptic neuron helps the post-synaptic neuron to decide, which connection is relevant and which is not. This selection process happens over a period of time and not instantaneously and therefore L-WTA is not suitable for such selection that involves retaining some information of prior neural activity. One of the most important aspect of neural information processing is feature extraction and formation of feature maps. Formation of feature maps requires that cells with similar feature selectivity cluster together. For this to happen each cell should be able to uniquely convey its feature preference at its output node which requires that each cell has to be identically stimulated by a selected pattern. Then on the basis of the responses of different cells for that pattern, cells that are selective to that pattern can be identified. Similarly, by applying other patterns cells can be marked for feature selectivity toward those patterns. Therefore, learning has to be deferred over an epoch so that all patterns are stimulated once and cells with similar feature preference can cluster together. In an L-WTA this is not possible for two reasons. Firstly, because in L-WTA inputs are applied simultaneously or in the {1,1} manner. This is analogous to applying all patterns at the same time and hence cells cannot be uniquely identified for their feature preference. Secondly, because the L-WTA lacks a mechanism for long term retention of modified weights which is needed for forming clusters. The ts-WTA on the other hand is perfectly suited as a learning cell for developing feature maps in silicon.

It may be apt to mention here that over and above facilitating synapse elimination, time-staggered or uncorrelated inputs play a major role in the formation of feature maps and this has been brought out in many seminal papers in neuroscience. For example Weliky and Katz (1997) reported that by artificially inducing correlated activity in both the eyes of the ferret, they found that the number of cells in the primary visual cortex with clear orientation and direction selectivity was markedly reduced when compared to un-stimulated controls. In a similar experiment on kittens, Stryker and Strickland (1984) found that segregation in ocular dominance columns was promoted when neural activity is synchronized in each eye but not correlated between the eyes. In other similar experiments on cortical feature map development in visual (Elliott and Shadbolt, 1998; Jegelka et al., 2006) as well as auditory cortex (Zhang et al., 2002) it has been reported time and again that spatiotemporal relation between the inputs to both eyes/ears are the key to formation of feature maps. Hence, it comes as a deduction from the above evidences that uncorrelated or "time-staggered" activity is an underlying biological mechanism for the formation of feature maps in the cortex. Therefore, using this inspiration to build artificial feature maps in silicon would help us bridge the gap between actual neural phenomenon and its neuromorphic equivalent.

The use of ts-WTA to build Ocular Dominance (OD) Maps has been described in Markan et al. (2013). In order to build a generic framework for cortical feature map formation in neuromorphic hardware, our ultimate goal, we wanted to extend our model to a larger input space. Orientation Selectivity (OR), a property exhibited by neurons in the visual cortex, is a natural extension to OD. OD is the selective preference cortical neurons show toward inputs from either the left eye or the right eye. The input space in OD is only two dimensional. OR on the other hand, is the selective preference cortical neurons show toward light or dark bars or edges of different orientations. Since orientations can vary anywhere from 0° to 180°, the input space is truly multi-dimensional. The following sections describe how from the basic building block of ts-WTA, we build an adaptable framework for multi-dimensional input features and how we extend it to build an adaptable circuit that is able to learn and eventually respond to different orientations.

## 3. ORIENTATION SELECTIVITY

Cells in the primary visual cortex are known to respond to dark and bright oriented bars. This property of the cortical cells, known as Orientation Selectivity, was first discovered by Hubel and Wiesel (1959). Hubel and Wiesel identified the receptive fields of Simple Cells in the Primary Visual Cortex and then showed bars of different orientations to the eye. Interestingly they observed that a single cell gave maximum response to a bar of only one particular orientation. They also observed that if the bar was in the center of the receptive field, it gave the highest response. In earlier experiments on retinal ganglion cells and lateral geniculate nucleus cells (Kuffler, 1953) it was observed that the receptive fields of these cells are divided into 2 parts (center/surround), one of which is excitatory or "ON," the other inhibitory or "OFF." For an ON/OFF center/surround cell, a spot of light shown on the inside (center) of the receptive field elicits spikes, while light falling on the outside ring (surround) suppresses firing below the baseline rate. Results are opposite for an OFF/ON cell. Hubel and Wiesel were proponents of the theory that receptive fields of cells at one level of the visual system are formed by inputs from cells at a lower level of the visual system, emphasizing that there is a hierarchical arrangement in the cortex, where in the higher layers extract statistically relevant information from the lower layers. Hence, they advanced the theory that small, simple receptive fields could be combined to form large, complex receptive fields. Later theorists also elaborated this simple, hierarchical arrangement by allowing cells at one level of the visual system to be influenced by feedback from higher levels. In their theory of orientation selectivity, Hubel and Wiesel proposed that Simple cells have receptive fields composed of elongated ON and OFF sub-regions (Hubel and Wiesel, 1959, 1962), which seem to originate from single synaptic input from ON and OFF centered lateral geniculate cells. The circularly symmetric receptive fields of neurons in LGN, that excite a cortical cell, are arranged in a row creating elongated receptive

fields see **Figures 4B,C**. These elongated sub-fields are sufficient for generating a weakly tuned orientation response, which is then amplified by local intra-cortical connections. Unlike Ocular Dominance, that seems to develop only after eye opening, orientation selective responses have been observed to be present in primates, cats and ferrets as early as the first recordings can be made (Chapman et al., 1996). However, how the geniculate afferents organize themselves into segregated ON and OFF sub-regions during the prenatal period, in the absence of visual input, is still not clear. Some researchers attribute this development to spontaneous waves of activity that flow in the retina and LGN affecting cortical development (Mooney et al., 1996), and some attribute it to intra-cortical long range connections that exist before birth, forming a scaffold for orientation maps that later mature with visual inputs (Shouval et al., 2000). In order to gauge to what extent, visual experience influences the development of orientation maps, visual cortex of kittens reared in a single striped environment was studied using optical imaging techniques. It was found that even though kittens reared in a striped environment responded to all orientations, however, twice the area of the cortex was devoted to the experienced orientation as compared to the orthogonal one (Sengpiel et al., 1999). This effect is due to an instructive role of visual experience whereby some neurons shift their orientation preferences toward the experienced orientation. Thus, it is now generally accepted that although orientation maps are fairly stable at the time of birth, abnormal visual experience can alter the neuronal responses of a large percentage of cells to the exposed oriented contours. Under normal conditions, the prenatal tuning properties of neurons are retained and get refined with visual stimulus.

A number of models suggesting possible formation of orientation selective cells in cortex have been proposed. These have two main shortcomings. First, they employ a Mexican hat correlation function in the cortex (some use it in the LGN as well Miller, 1994). In the developing cortex, it is highly unlikely that this structure exists (Buzás et al., 2001; Yousef et al., 2001; Roerig and Chen, 2002). Second, competition in these models is brought in through synaptic normalization (multiplicative or subtractive). Normalization has its own associated problems, for linear synaptic weight update multiplicative normalization does not permit positively correlated afferent to segregate, while under subtractive normalization, a synapse either reaches the maximum allowed value or decays to zero (Miller and MacKay, 1994). These shortcomings have brought in the necessity of introducing models that are biologically more plausible (Miller, 1996; Elliott and Shadbolt, 1998). It has been observed that although the horizontal intra-cortical connections are still clustered at birth, the thalamo-cortical connections are well defined (Sur and Leamey, 2001). This indicates that the Orientation selectivity observed at birth could be manifesting out of the relatively well developed thalamo-cortical connections or the receptive fields of the cortical cell. These findings suggest the existence of some common biological mechanisms that could be responsible for the emergence of receptive field structure and thus orientation selectivity in the visual cortex. It has been shown that competition for neurotropic factors and neighborhood cooperation through diffusion of leaking chemicals (that lower the threshold of the neighboring cells and make them fire more readily on receiving same stimulus) are biological phenomenon acting in the brain both before birth and after (Cellerino and Maffei, 1996; Elliott and Shadbolt, 1998; McAllister et al., 1999). Models based on this competitive and cooperative behavior have been able to explain aspects of feature map formation of both orientation selectivity and ocular dominance (Markan, 1996; Bhaumik and Markan, 2000; Bhaumik and Mathur, 2003). Our model is inspired by the three layered model proposed by Bhaumik and Mathur (2003) (see **Figure 4A** for the abstract sketch of the model). However, there are some differences. While their model aims to describe the formation of oriented receptive fields prior to eye opening, our model also takes into account the influence of visual experience or cortical plasticity observed after eye opening. They use competition based on both pre and post synaptic resource limitation and diffusion between ON/ON center and OFF/OFF center cells, requiring precise initial connections between cells. Our resource limitation is only post synaptic and is enforced by limiting current in the bias transistor representing the cortical cell. The diffusion in our model happens between all neighboring cells irrespective of their type.

To build a hardware model of a cortical cell that exhibits orientation selectivity, from the building block of a single ts-WTA circuit, systematic scaling up was required. The next section describes how this scaling up was done and how diffusive interaction between ts-WTA cells was introduced.

### 3.1. BUILDING A FRAMEWORK FOR MULTIDIMENSIONAL FEATURE SELECTIVITY

Any attempt at building self-organizing feature maps in hardware, requires neighborhood interaction to happen in such a way that local clusters are formed autonomously. We showed previously that this can be achieved by means of diffusive coupling between neighboring cells by means of an RC network (Markan et al., 2013). Biologically this happens through leaking chemicals from active neurons and as more recently shown through gap junction coupling (Li et al., 2012; Mrsic-Flogel and Bonhoeffer, 2012). In order to extend our design for feature selectivity over multi-dimensional input space, we took four ts-WTA cells and connected them in a row, with their outputs tied together in a feed-forward manner through MOSFETs (see **Figure 5**). This can be understood as a three-layered model where the first layer is the retina, the second layer is the Lateral Geniculate Nucleus (LGN) and the third layer is the visual cortex. While there is one-to-one mapping between cells in layer 1 and layer 2, there is many-to-one mapping from layer 2 to layer 3 cells, we call these layer 2 cells the receptive field of that layer 3 cortical cell. Therefore, now we have a cortical cell with a $1 \times 4$ receptive field. Individual ts-WTAs are connected to their neighbors with a 10k diffusive resistor ($R_D$). The output of the cortical cell is fed back to the individual ts-WTA cells, through a resistive feedback network ($R_F$), also of 10 k, as can be seen in **Figure 5**. The purpose of these resistances ($R_F$) is to reinforce the initial bias so that the responses of the cells become fine-tuned ensuring that the pattern learnt is one of the applied patterns. The diffusion capacitor

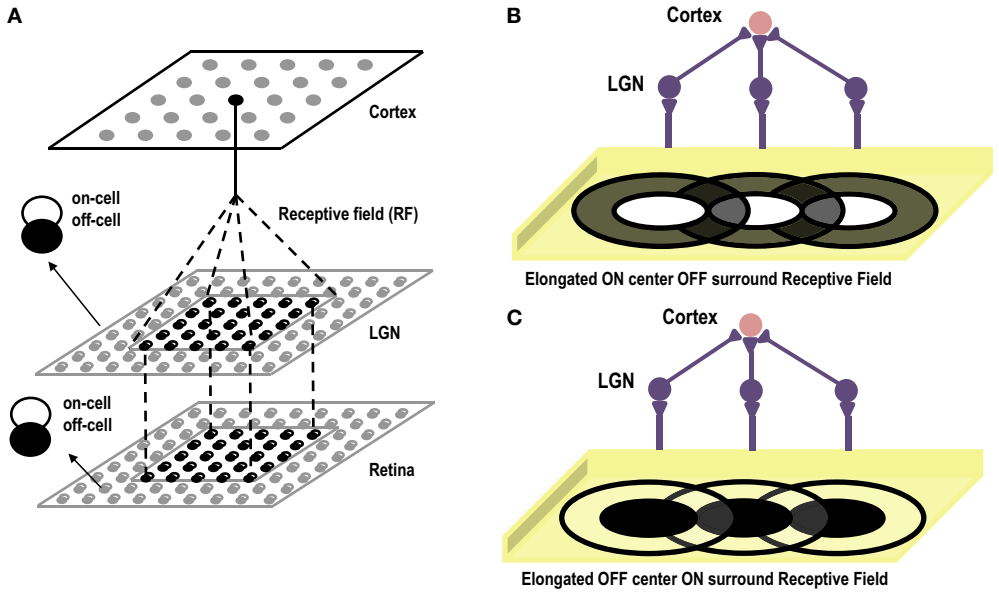



**FIGURE 4 | (A)** Shows the three layer abstract feed-forward model of Orientation Selectivity. The first layer, retina, is the layer that receives inputs. The second layer is the LGN. There is one-to-one mapping between retina and LGN cells. The third layer is the cortex. Many LGN ON/OFF center cells innervate at a single cortical cell forming its receptive field. **(B)** Shows the elongated ON-Centered, OFF-Surround receptive field of a cortical cell (Inspired by Hubel and Wiesel's model of Orientation Selectivity). **(C)** Shows the elongated OFF-Centered, ON-Surround receptive field of a cortical cell.

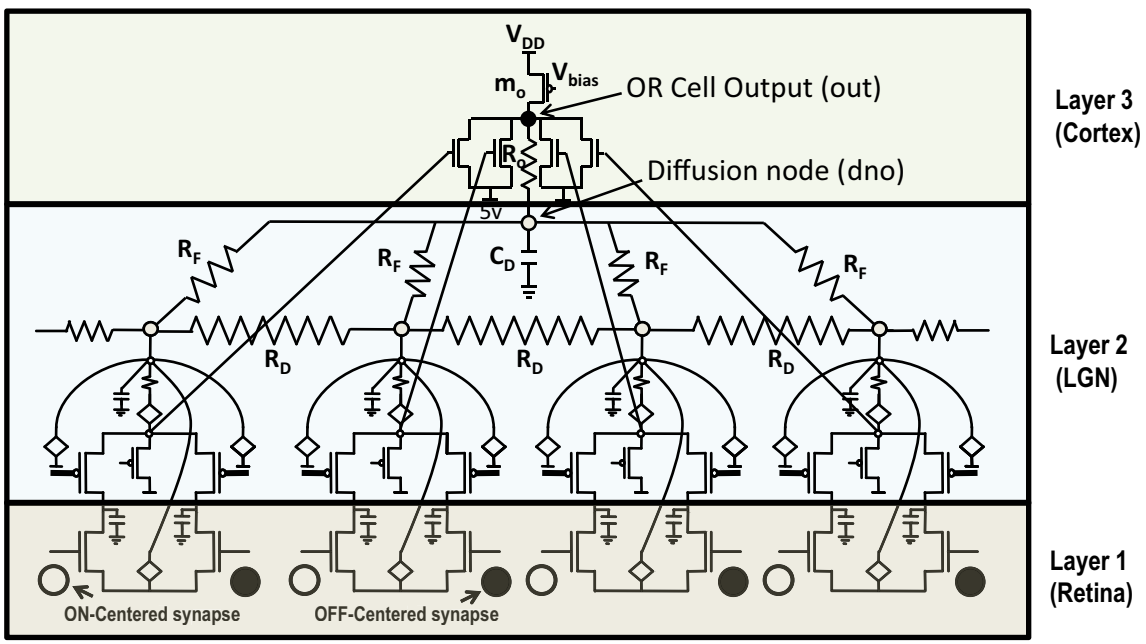



**FIGURE 5 | Shows 4 ts-WTA cells connected in a row by means of diffusive resistors ($R_D$).** The output of each cell ($V_s$) is connected in a feed forward manner using mosfets with their drains connected together at node ***out*** which is the feed forward path conveying the self activation or response of the cell. The activation node of each cell ($V_i$) is connected at the diffusion node, ***dno***, with feedback resistances ($R_F$). This forms the feedback network of the cell. A small resistance $R_o$ connects ***out*** and ***dno*** to keep both these voltages nearly the same. The bias transistor $m_o$ represents the cortical cell. Here $V_{DD}$ is 6 v.

($C_D$) connected at node dno, is of 10 pF. To achieve cluster formation on a larger scale, it is important to achieve cluster formation locally. To ensure formation of local clusters within the set of four ts-WTA cells, the first and fourth ts-WTA of the cortical cell are connected diffusively in a ring fashion (not shown in the figure). This ensures that the receptive field develops into only one of the four patterns (0011), (1100), (0110), (1001), in which 11 and 00 are always clustered. We took 2 such cortical cells with a $1 \times 4$

(ts-WTAs) receptive field. The development of the receptive fields was analyzed in two situations. First, when the cortical cells are in isolation and second, when they are diffusively coupled with each other (**Figures 6.1A,B**).

Both the cortical cells are stimulated with the same random-inside-epoch order of input patterns, however, their initial biases (initial floating gate voltages of LGN cells/layer 2) are different. The initial biases are randomly generated floating gate voltages varying between 5.15 and 5.16 v. We assume that the left branch of each ts-WTA represents an ON-Centered synapse and the right branch represents an OFF-Centered synapse. The inputs patterns are from the set (1100/0011), (1001/0110), (0110/1001), (0011/1100), (1010/0101), (0101/1010). The notation (1100/0011) means that when the ON-Centered synapses (left branches) of the four ts-WTAs of a cortical cell are stimulated by 1100 the OFF-Centered synapses (right branches) are stimulated by 0011 (as described in section 2, 1 here represents a high voltage (+6 v), and 0 represents a low voltage (−1 v) applied for 0.02 s). This is to emulate time-staggered or uncorrelated inputs. Please note that the patterns 0001 and 1000 are omitted from the set because they have unequal number of 0 s and 1 s and thus do not stimulate both the branches equally). When the input patterns are applied in a random-inside-epoch fashion, competition between the two arms of each ts-WTA cell begins. Depending on whether a branch is favored by the initial conditions more or is stimulated more or both, either the ON-Centered or the OFF-Centered branch wins. The resultant receptive field (i.e., the floating gate voltage profile of each branch of the four ts-TWAs) looks like one of the input patterns applied. **Figure 6.2A** represents the 1 × 4 receptive field of cortical cell 1 and **Figure 6.2B** represents the 1 × 4 receptive field of cortical cell 2 when they develop in isolation. When the two cortical cells are isolated, their receptive fields evolve into different patterns. Here cell 1' s receptive field has evolved into 1100 whereas cell 2' s receptive field has evolved into 0011. However, in the second case, when the two cells are diffusively coupled, their receptive fields evolve into similar patterns (1100) (**Figures 6.3A,B**). This happens because the diffusive node (dno) voltage, of the two cells becomes coupled. When the input patterns are applied, if one of the cells has a stronger bias for a particular input pattern the voltage at its node dno becomes high. Since both the cells receive the same random-inside-epoch order of inputs, the other cell also experiences this raised voltage at its node dno for the same pattern. The feedback resistors convey this high response back to the tunnel (T) and injection (I) devices (**Figure 1A**) which modify the floating gate voltages of all the ts-WTA cells reinforcing this pattern on them. Over many epochs, the difference between the ON-Centered and OFF-Centered branches of each ts-WTA cell gets amplified and

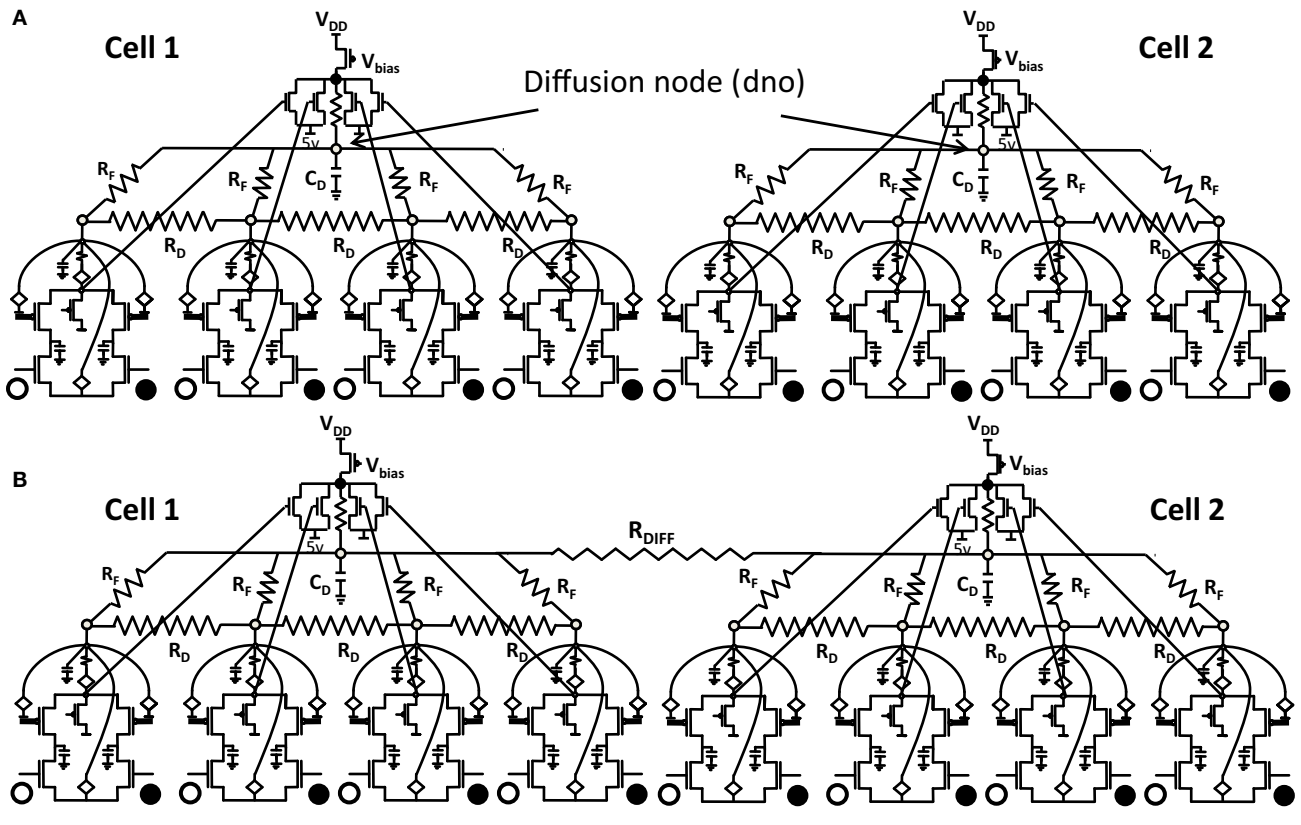



**FIGURE 6.1 | Shows 2 cortical cells Cell 1 and Cell 2 with a 1 × 4 (ts-WTA) receptive field.** In **(A)** the two cells develop independently. In **(B)** the two cells are connected at the diffusion node (dno) by means of a resistance $R_{DIFF}$ for diffusive interaction. **Figures 6.2**, **6.3** show how the receptive fields (floating gate voltages) evolve for the two cells in both the situations.

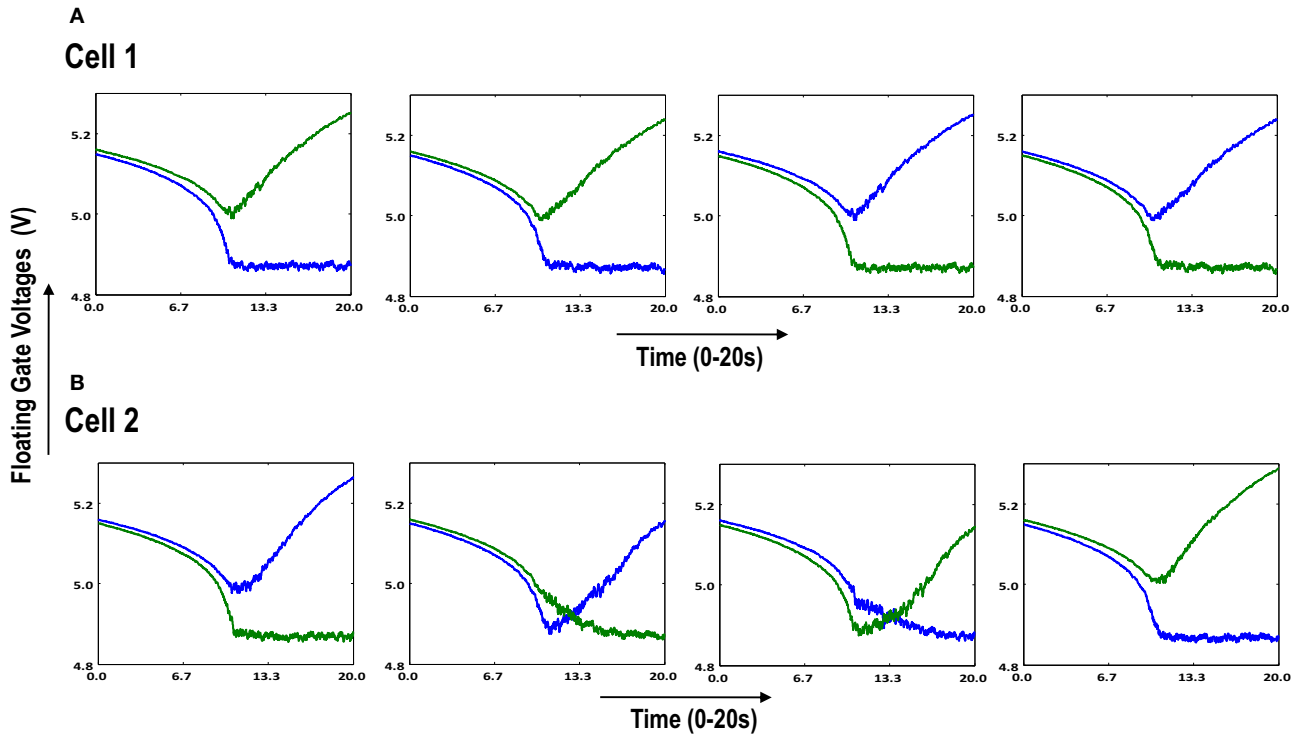



**FIGURE 6.2 | Shows the development of floating gate voltages of the two cortical cells of Figure 6.1A.** Here blue represents the floating gate voltage of the ON-Centered synapse and green represents the floating gate voltage of the OFF-Centered synapse. The cells develop differently according to individual initial biases and inputs. **(A)** Shows the four ts-WTAs of Cell 1. The pattern of receptive field is 1100 and **(B)** Shows the four ts-WTAs of Cell 2. The receptive field has evolved into 0011.

the floating gate voltages get developed for the pattern that evoked the highest response at node dno during the initial few epochs. Towards which pattern the competition tilts occurs depends on the initial biases and the patterns applied and can be changed by changing either. Hence, promising results in the form of cooperation between neighboring cells are visible when two cortical cells are diffusively coupled.

To see neighborhood cooperation and cluster formation on a larger scale we then diffusively connected 10 cortical cells, each with a $1 \times 4$ receptive field, with the tenth cortical cell connected to the first in a ring fashion. By giving all the cells different initial biases but subjecting them to the same sequence of random-inside-epoch patterns, interesting cluster formation was observed (see **Figure 7**). **Figure 7A** shows the development of the 10 cortical cells in isolation whereas **Figure 7B** shows their development under diffusive interaction. In the latter, two clusters of different patterns (or feature preference) are clearly visible. Between two opposite feature preferences (0011 and 1100), there is gradual variation between the feature preferences (1001) [see **Figure 7B** cells 2, 3, and 4 (rows 2, 3, and 4 from the top)]. With this idea of extendibility to multi-dimensional inputs and a framework for neighborhood interaction and clustering in place, we now build the circuit of a cortical cell that is capable of adapting and self-organizing, to become selective to patterns resembling different orientations.

### 3.2. ORIENTATION SELECTIVE CELL MODEL AND SIMULATION

The previous section described the architecture of a cortical cell with a receptive field of $1 \times 4$ (ts-WTA) LGN cells. These cells when connected on an RC grid show diffusive interaction and cluster formation. The Orientation Cell model has a similar three layer topology, with retinal, LGN and cortical cells except that instead of a $1 \times 4$ receptive field, the orientation selective cortical cell has a two dimensional, $9 \times 9$ (ts-WTA), receptive field. However, with some differences in component values to balance out the effect of a larger neighborhood. The values of the diffusion, ($R_D$), and feedback resistances, ($R_F$), are now of 1k ohm each. A $3 \times 3$ simplified subsection of the circuit representing the receptive field of the cortical cell is shown in **Figure 8**. The capacitance connected at the node dno is 10 pF. The feed-forward MOSFETs connecting the common source nodes of the individual ts-WTA cells to the cortical cell (bias transistor $m_o$) ensure that the self-activation of each cell is conveyed appropriately at the OR cell output, however, since there cannot be any current in the reverse direction, the OR cell's output will not affect the common source voltage at each ts-WTA. The purpose of the diffusive and feedback resistances remains the same i.e., to ensure proper neighborhood interaction and to fine tune the cell's response, respectively.

A set of input patterns resembling ON-Centered and OFF-Centered oriented bars of angles 0°, 45°, 90°, and 135° were created (**Figure 9A**). Each pattern comprises of $9 \times 9$ blocks in which a bright block means stimulation with a +6 v pulse given

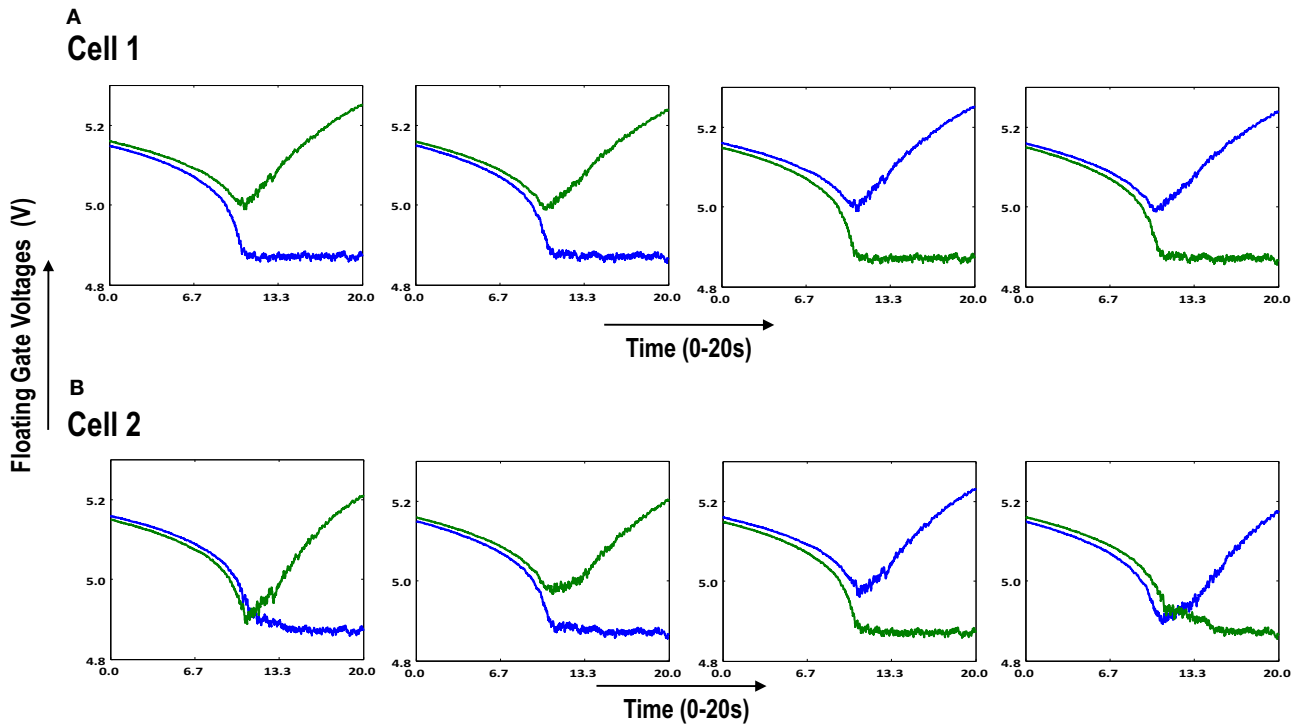



**FIGURE 6.3 | Shows the development of floating gate voltages of the diffusively coupled cortical cells in Figure 6.1B.** Cell 1 which seems to have a stronger bias influences the development of Cell 2 which modifies its original response to become similar to cell 1. **(A)** Shows the unchanged response of cell 1 (1100) and **(B)** shows the response of cell 2 under strong influence of neighborhood (1100).

for 0.02 s and dark block means stimulation by a −1 v pulse given for the same duration. To ensure that the learning is not biased towards the order in which patterns are applied, these bars were applied in a random-inside-epoch manner, however, with two constraints. (1) within each epoch when the left synapses of the 9 × 9 ts-WTA receptive field are stimulated with an ON-Centered oriented bar input, the right synapses are stimulated with the same orientation but with an OFF-Centered oriented bar. This is analogous to applying uncorrelated inputs to each ts-WTA branch and (2) just after that, this order is reversed, meaning, the left synapses are now stimulated with the OFF-Centered oriented bar and the right branches with the ON-Centered oriented bar of the same orientation angle. This is analogous to applying an orientation grating like input pattern that is necessary for orientation map formation. Gratings ensure that all the cells in the 9 × 9 receptive field are stimulated with the same oriented bar. This is necessary for cluster formation since clusters are formed when the cells group together according to similar feature preferences and whether two cells have the same feature preference or not can be known only when they receive the same inputs. Interestingly, the prenatal brain, when external inputs are absent, retinal waves have been identified to play the role of grating like input patterns that help in building a scaffold for orientation selectivity even before birth (Wong, 1999; Akerman et al., 2002). On the onset of simulation, the receptive field of the orientation selective cell i.e., 9 × 9 LGN cells are given random initial biases within 5.15–5.16 v. By applying the eight different input patterns in a random-inside-epoch manner, transient analysis on the circuit is performed for 80 epochs. As the simulation progresses, the synaptic connections from the ON-Centered and OFF-Centered LGN cells to the cortical cell compete and only one of the connections survives, the other gets eliminated (ts-WTA action). The local interaction between LGN cells is both competitive and cooperative. Competitive because of resource limitation in each ts-WTA cell, where only one of the connections (either ON-Centered or OFF-Centered) survives and cooperative by means of diffusive interaction between the neighboring ts-WTA cells, implemented by means of diffusive resistive coupling ($R_D$) of the 9 × 9 ts-WTA cells, in a way similar to the Ocular Dominance model implementation. Details on the feedback mechanism acting on the floating gate pFETs in the individual ts-WTA cells and Ocular Dominance Map formation can be found in Markan et al. (2013). The orientation input pattern for which the voltage at node dno is the highest or a pattern that is statistically more significant gets reinforced through the feedback resistors ($R_F$) and the injection and tunnel feedback mechanisms of each ts-WTA cell (as discussed in the case for a 1 × 4 receptive field) and we say that the cell is selective to that particular orientation. Multiple simulations performed with different random initial biases of LGN cells (floating gate voltages) and different random-inside-epoch order of input patterns result into the cell learning different oriented patterns with equal likelihood of learning any one of the applied eight patterns. A statistical analysis over 100 simulations is presented in **Tables 1A**, **1B**. The results show that each of the eight patterns

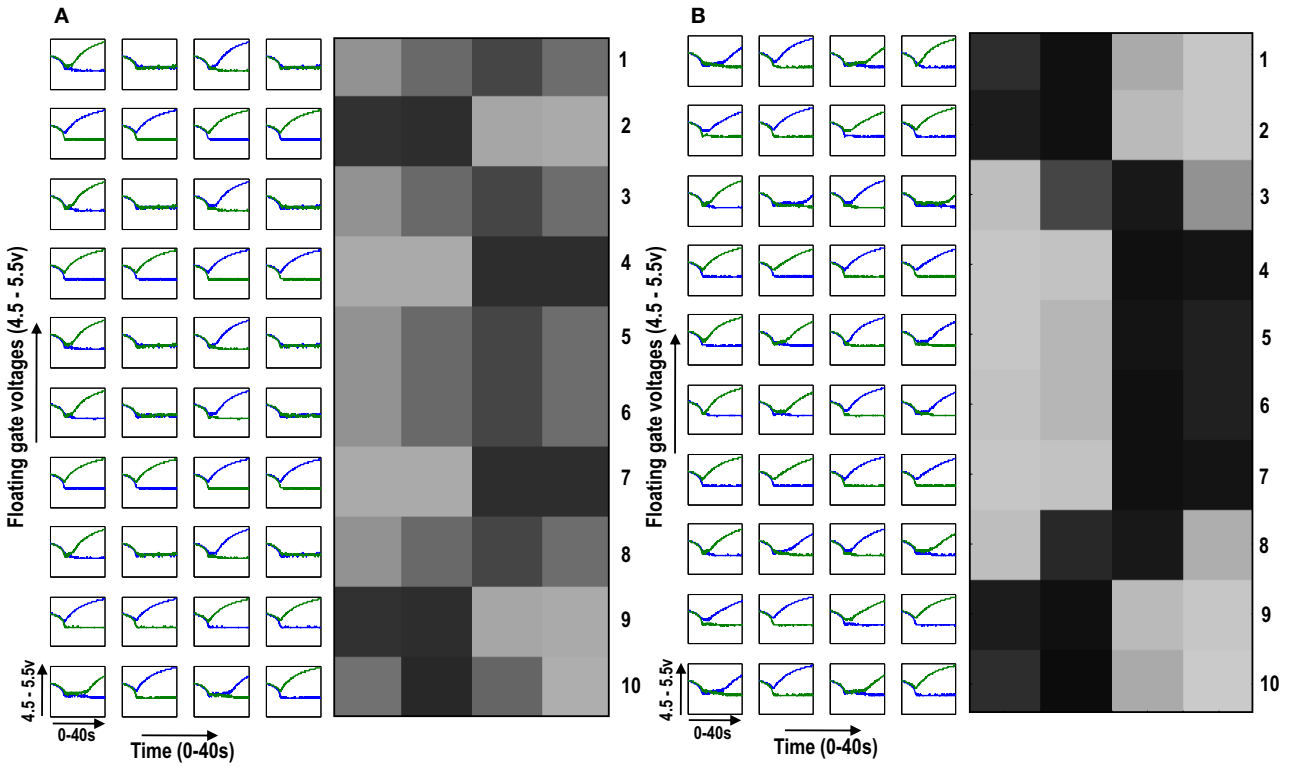



**FIGURE 7 | (A)** Shows the development of floating gate voltages of 10 (1 × 4) ts-WTA cells in isolation. Here Cell 1 is top row, Cell 2 is 2nd row and so on. The black, white and gray squares represent the feature preference of the ts-WTAs. Black represents an OFF-Centered cell, white represents an ON-Centered cell and gray represents an unbiased cell. The cells develop differently according to individual initial biases and inputs. **(B)** Shows the same 10 cells when they interact diffusively. Near neighbor cells begin to cluster developing similar feature preference. Between two opposite patterns (e.g., 1100 and 0011) , there is a gradual variation (1001), see responses of Cells 2, 3, and 4.

is learnt at least 10% of the times. A video of how the receptive field of the orientation cell evolves, starting from initial random biases of LGN cells to an oriented bar pattern, can be found in the supplementary material.

### 3.3. ORIENTATION TUNING AND PERFORMANCE UNDER ABNORMAL STIMULATION

Experiments done on many mammals demonstrate that during the early postnatal periods, the recording over a cortical neuron shows nearly equal response to many orientations or only slight bias toward a particular orientation. If the response of the cell is plotted against different orientation angles, it is a flat curve showing faint selectivity to many different orientations. As the orientation selectivity of the cell develops, as a result of stimulus dependent activity, the tuning curve becomes sharper at a particular orientation (Somers et al., 1995; Dragoi et al., 2000; Seriès et al., 2004). Similar orientation tuning is exhibited by our orientation selective cell. Once the cell has learnt a particular orientation i.e., the floating gate voltages of the cell have matured, the injection and tunnel voltages can be modified in a way that stops further learning, see learning rate parameter in Markan et al. (2013). The cell's response to any orientation can then be obtained by observing the output node voltage (OR cell output node) on the application of that oriented pattern as input. **Figure 9B** shows the orientation tuning curve of our cell at different stages of receptive field development. The development of orientation tuning is clearly visible from the shape of the curve. Initially the cell responds equally to all orientations, depicted by the nearly flat curve, gradually becoming selective to only one, represented by the rising peak at one of the orientations. The *Half Width at Half Height* (HWHH) was computed for each receptive field for the 100 simulations mentioned in the previous subsection. For the receptive fields that were not very finely tuned or they seemed to be close to more than 1 input patterns e.g., receptive field (5,5) in **Table 1A**, the HWHH was computed for each case and the receptive fields were categorized (see **Table 1B**) according to the lower HWHH value. The best HWHH, i.e., the HWHH for a highly tuned receptive field e.g., (1,4) in **Table 1A** is 30° and worst HWHH is 40° for a receptive field similar to (5,5).

Some experimental results also suggest that if on the onset of vision, animals are reared in an abnormal environment such as one with only single stripes, the orientation tuning of a large number of cells, that were initially tuned to different orientations, adjust their tuning to respond to the orientation of the striped environment in which they are reared (Sengpiel et al., 1999; Yoshida et al., 2012) and the cortical space that was initially shared equally by all orientations now becomes exceedingly large for the orientation shown. In other words the orientations shown take up the cortical space of the orientations that were never shown. To test if similar behavior is shown by our orientation selective cell,

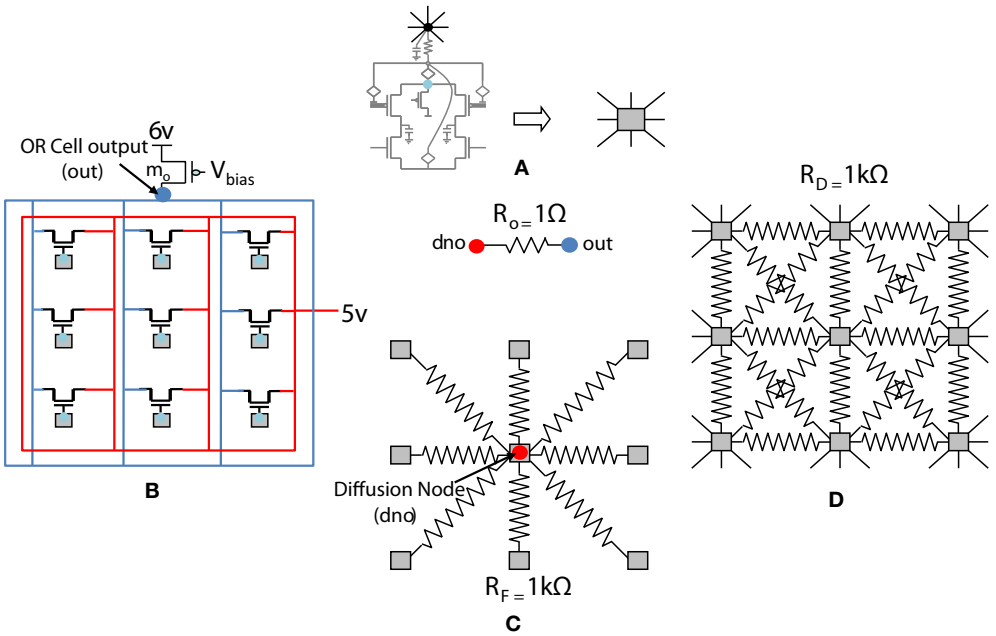



**FIGURE 8 | Simplified and distributed layout of a 3 × 3 portion of the 9 × 9 receptive field of our orientation selective cell. (A)** Shows the symbolic representation of a ts-WTA cell. In subsequent figures, the gray square represents a ts-WTA. **(B)** Is the feed-forward MOSFET network that takes the output of the individual ts-WTAs and feeds them to the OR Cell output. This is a read out node from where self-activation of the cell can be recorded. **(C)** Shows the diffusive resistance network consisting of $R_D$, which connects the ts-WTA cells to all their neighbors. **(D)** Shows the feedback resistive network consisting of $R_F$ that feeds the output of the cell from dno back to the individual ts-WTAs. **Out** and **dno** are connected by $R_o$ which can be replaced by a buffer device discussed in section 5.2. (see **Figure 5** for the lateral view, this is a top view).

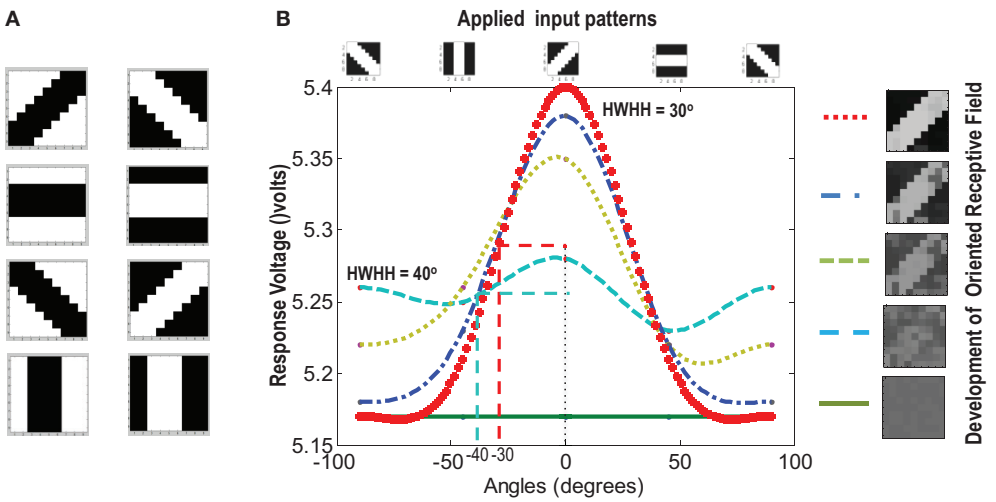
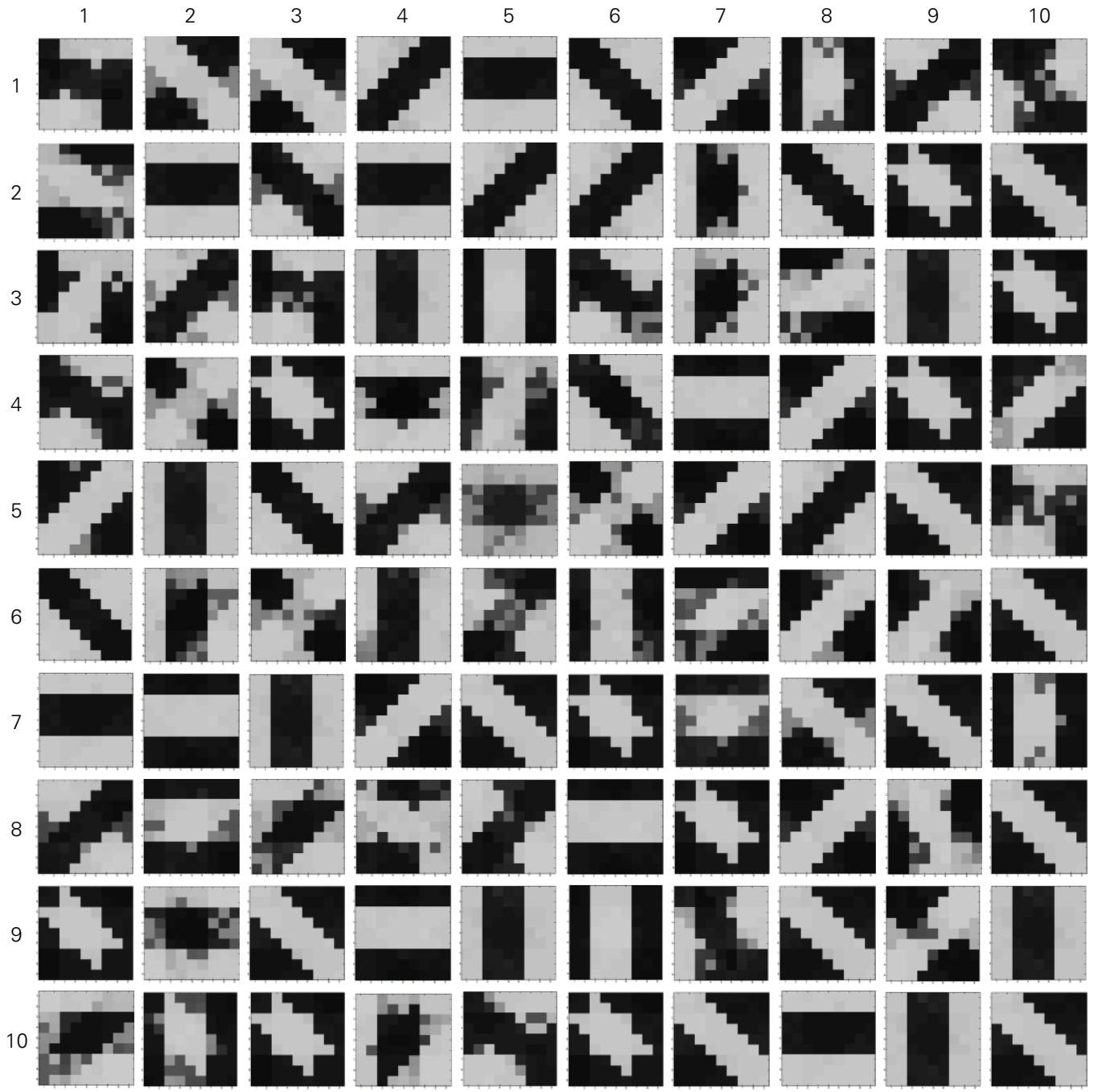



**FIGURE 9 | (A)** Shows the input patterns that are applied to the orientation cell. **(B)** Shows the orientation tuning curve. Initially the response of the cell is low and similar for all input patterns. As the receptive field develops (see on the right, bottom to top), there is increased response toward that specific pattern as can be seen from the sharpening of the tuning curve. The half width at half height (HWHH) parameter for the best and the worst receptive field has been marked. The sharper the tuning, the lower is the value of HWHH.

two sets of experiments were performed. In the first experiment, for 20 different initial conditions, 8 different orientation patterns were applied. It was found that for 20 simulations, the receptive fields developed into one of the eight patterns, with nearly equal probability. Now, for the same set of initial conditions, we applied only six patterns (two horizontal patterns, 1 ON-Centered and one OFF-Centered were omitted). The results are summarized in **Tables 2**, **3**. It was observed that for 20 simulations, the cell

**Table 1A | Results of 100 simulations of the orientation selective cell performed with different random initial biases and different random-inside-epoch inputs.**

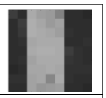



now developed according to the 6 patterns applied with nearly equal probability. Therefore the space that was earlier occupied by eight patterns was now equally distributed amongst six patterns. The cell demonstrated adaptive cortical plasticity by developing receptive fields according to the applied patterns. However, if the initial biases very strongly favor one of the missing patterns, like in **Table 3**, 2nd row 4th column, the receptive field develops according to the initial bias rather than the applied patterns. This kind of adaptive plasticity to accommodate abnormal inputs may not be possible in the model by Bhaumik and Mathur (2003) since their model does not take into account the effect of external stimulation.

### 3.4. ANALYZING THE EFFECT OF NATURE $V_s$ NURTURE

It is known that both Nature (genetic biases) and Nurture (environmental factors) play an important role in feature map

**Table 1B | Analysis of 100 simulations.**

| Orientation Receptive Field |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|
| Appearance (no. of times) in 100 simulations | 10 | 19 | 12 | 11 | 13 | 12 | 13 | 9 |

**The evolved receptive field sometimes resembles two different orientations. In such cases the response towards both the orientations was noted and HWHH was computed in each case. The categorization was done on the basis of the lower HWHH.*

**Table 2 | Summary of 20 simulations of orientation selective cell with all 8 oriented patterns applied as inputs.**

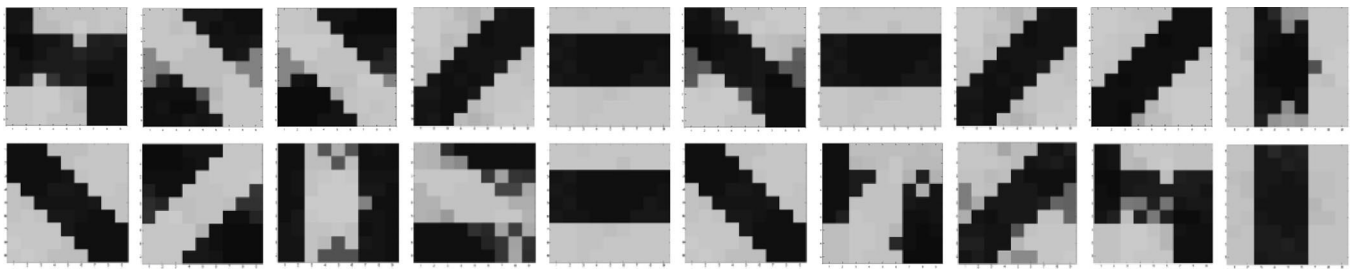


**Table 3 | Summary of 20 simulations of orientation selective cell with horizontal patterns missing.**

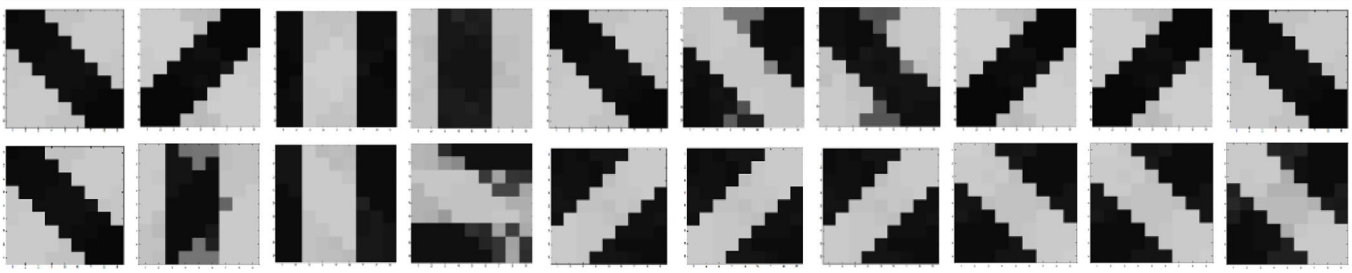
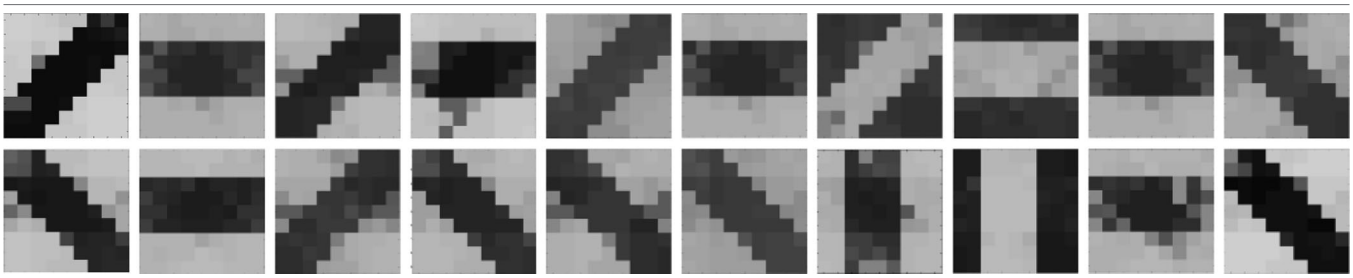
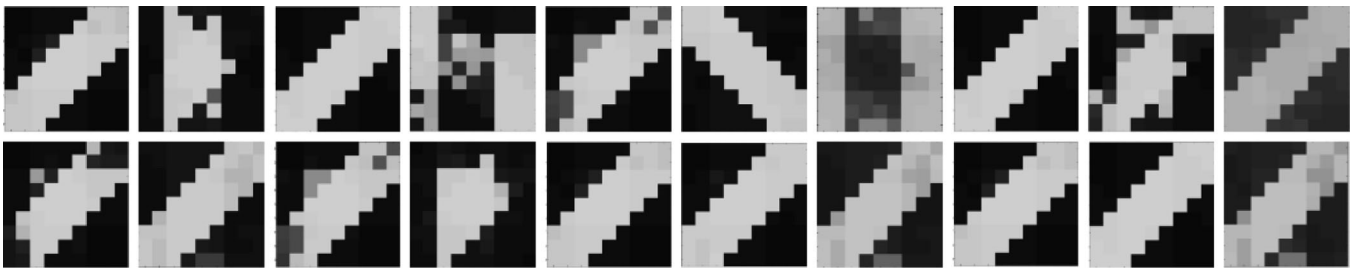


**Table 4A | Summary of 20 simulations of orientation selective cell with same initial conditions but different random-inside-epoch order of input patterns.**



formation. To understand how our orientation selective cell responds to nature (initial biases) vs nurture (pattern stimulation) and to gauge how close it is to biology, two sets of experiments were performed. In the first experiment, repeated simulations were performed by keeping the initial biases over the $9 \times 9$ LGN cells the same, but changing the random-inside-epoch order of input patterns over all the epochs. Statistical analysis over 20 simulations showed that 80% of the times the cell learnt a different oriented pattern, highlighting that stimulus driven activity can override the orientation bent due to the initial floating gate voltages in most of the cells. In the second experiment, the random-inside-epoch order in which inputs are applied was kept constant (creating preference for one of the patterns) over all the simulations but the initial biases were changed every time. It was observed that although 70% of the times the cell developed the same oriented receptive field, but 30% of the times it

**Table 4B | Summary of 20 simulations of orientation selective cell with different initial conditions but same random-inside-epoch order of input patterns.**



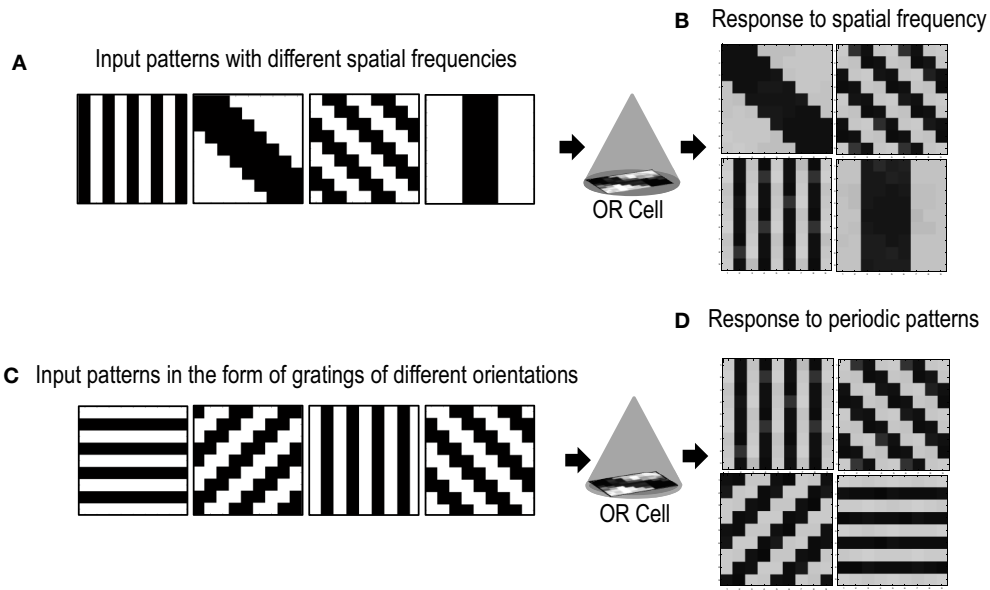



**FIGURE 10 | Shows the response of the Orientation Cell to patterns of different spatial frequencies and periodic patterns of different orientations. (A)** Shows patterns of different spatial frequencies that are applied as inputs to the OR Cell. **(B)** Shows the patterns that the cell learns. Each simulation results in the circuit learning one of the input patterns with equal probability. **(C)** Shows periodic patterns of different orientations that are applied as inputs to the OR cell. **(D)** Shows the periodic patterns that the cell learns.

did learn other patterns. This experiment brings out that it is not just the input patterns applied, but the unique combination of the inputs and the initial biases that decides which oriented pattern the cell would learn or become selective to, bearing close analogy to experimental findings. The results are summarized in **Tables 4A**, **4B**.

## 4. RESPONSE TO SPATIAL FREQUENCY AND PERIODIC PATTERNS

Cells in the primary visual cortex are also known to respond to the spatial frequency of visual inputs (Maffei and Fiorentini, 1973; Tootell et al., 1981; De Valois et al., 1982; Everson et al., 1998). Some cells respond to low spatial frequencies, some to high spatial frequencies, essentially forming spatial low pass, band pass and high pass filters that act on the visual inputs. To test if our cell could also be selective to the spatial frequency of applied inputs, we presented the circuit with patterns of different spatial frequencies (**Figure 10A**). The simulations were performed in the same way as described in section 3.2 except for the new input patterns that have orientations of different spatial frequencies. We took only two spatial frequencies (low and high). Repeated simulations resulted in the cell learning orientations of different spatial frequencies (**Figure 10B**). However, it was observed that the learning time of the cell increased as compared to when all inputs are of the same spatial frequency.

Certain cells in the visual cortex are also known to be selective to periodic patterns (Von der Heydt et al., 1992). These cells respond vigorously to gratings but not so much to bars or edges. Since these cells are not sensitive to the spatial frequencies of the gratings but are only specialized for detection of periodic patterns, they seem to have a role in the perception of texture. In order to test if our circuit could have a similar response to periodic patterns, we presented our circuit with input patterns that resembled gratings of different orientations (see **Figure 10C**). After several epochs of it was observed that the cell's receptive field developed according to one of the grating patterns

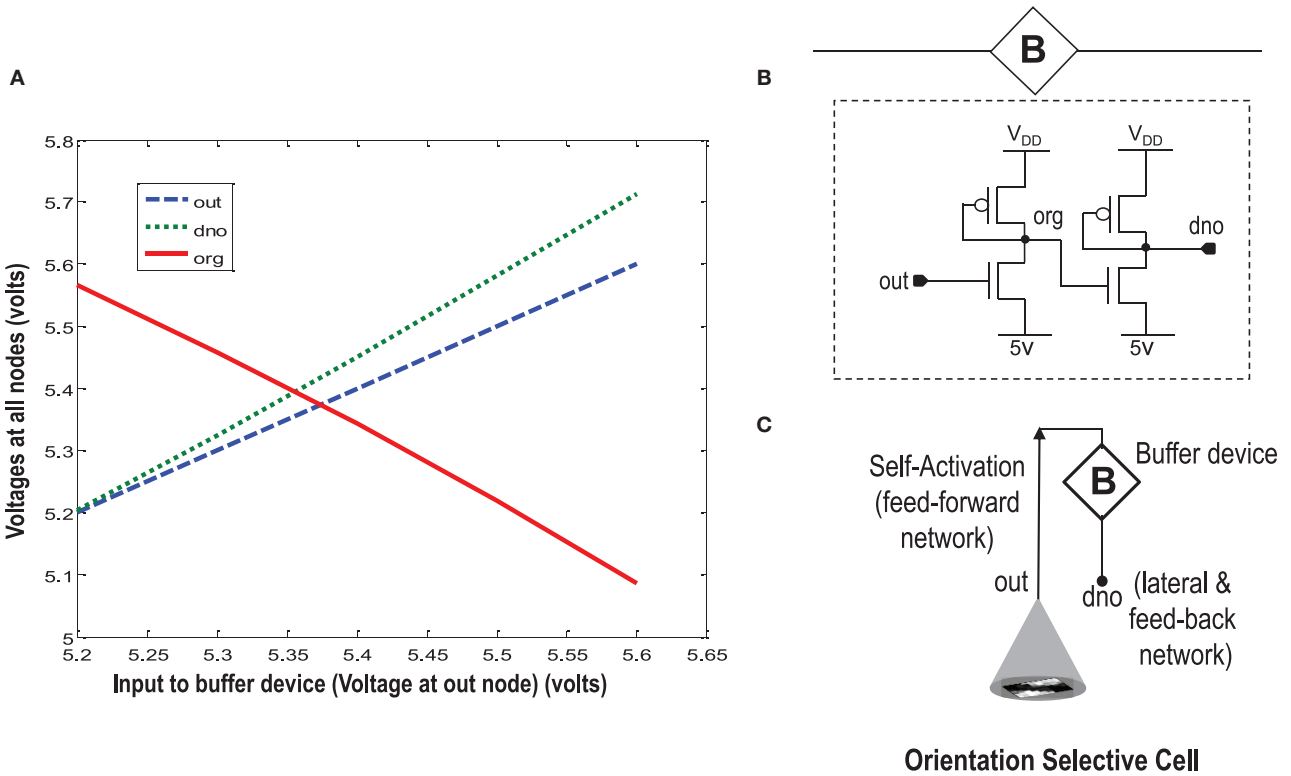



**FIGURE 11 | In order to isolate the OR cell output (out) which conveys the self-activation of the cell, from the diffusion node (dno) at which other orientation cells connect and to prevent loading of node out, a buffer device is created. (A)** Shows the characteristic response of the buffer device. The device is linear, and has a double inverting effect on the voltage at node *out*. The $V_{DD}$ is 6 v. **(B)** Shows a typical design of the buffer device. **(C)** Shows an abstract symbol for orientation selective cell along with the buffer device.

(**Figure 10D**). Repeated simulations with different initial biases and different random-inside-epoch order of inputs resulted in the cell's receptive field evolving into one of the eight grating patterns with equal probability. These experiments show that the cell developed is generic and is extendable to recognizing many different patterns.

## 5. DIFFUSIVE INTERACTION OF CELLS

Feature map formation is based on three important tenets: *continuity*, *diversity* and *global order*. Continuity requires that nearby cells share the same feature preference. Diversity means that there is equal representation of all possible feature preferences and global order implies that there is a periodic organization of different features over the entire cortical surface. Literature sites several mechanisms that coordinate the development of feature selectivity of single cells under neighborhood influence (Grossberg and Olson, 1994). The essence of these mechanisms is that if cells have overlapping receptive fields and they receive similar inputs, then if they can be forced to have similar responses, Hebbian Learning mechanism will ensure that the individual cells' receptive fields develop to form clusters. As discussed earlier, this poses certain requirements on the behavior of the learning cell and the neighborhood function. Firstly, it demands that the learning cell should allow modulation of its feature selectivity under neighborhood influence. Secondly, it demands for a neighborhood function that is capable of generating an appropriate signal that can modulate the development of feature selectivity of a cell in concordance with other cells in the cluster.

Diffusive-Hebbian learning based on the biological phenomenon of reaction-diffusion has been shown to be effective in forming clusters of cells with similar feature preference and has also been used to model Ocular Dominance and Orientation Selectivity Map Formation (Markan, 1996; Krekelberg, 1997; Markan and Bhaumik, 1999; Bhaumik and Markan, 2000; Bhaumik and Mathur, 2003). Biologically, this happens by means of leaking chemicals coming out of an active cell, that lower the threshold of the neighboring cells. Reaction-diffusion can be easily implemented by an RC network as shown in Shi (2009) and Markan et al. (2013). The development of individual cells and cells under diffusive interaction varies significantly. If the cells have different initial biases then in the absence of diffusive coupling they develop into cells with different orientation preferences. On the other hand, the presence of diffusive coupling causes nearby cells to have a similar voltage (at node dno) and hence the injection and tunnel feedback that they receive is also the same. Therefore, if the two cells receive similar inputs, they develop to have similar feature preference. The stronger cell (the cell that generates a higher voltage at node dno) tends to influence the development of the weaker cells around it.

### 5.1. MODIFICATION OF ORIENTATION TUNING UNDER NEIGHBORHOOD INFLUENCE

As discussed earlier, for any map formation, diffusive interaction between cells should happen in such a way that it leads to formation of clusters of cells having similar feature selectivity. This is possible if some of the cells change their feature preference when they are surrounded by strongly biased cells, forming clusters showing gradual variation in orientation selectivity between clusters. This means that some kind of mechanism needs to be present that helps the cell in overcoming its initial orientation bias to develop an orientation preference according to the neighborhood influence. In our orientation cell this is achieved by ensuring two things. (1). Keeping the time constant of the diffusive RC network ($\tau_{\text{Diffusion}}$) much smaller than the time constant of the orientation cell ($\tau_{\text{Reaction}}$) and (2). Limiting the amount of learning in each iteration by applying input patterns for a very short duration (0.02 ms). The first condition ensures that diffusion has precedence over reaction and the strong neighborhood influence is able to modify the individual bias of an orientation cell and the second condition makes sure that the learning in the orientation cell is at a pace that is suitable for diffusion to influence its development i.e., the floating gate voltages are allowed to change by only a small amount in every iteration. This is required because once the difference between the floating gate voltages of the two arms of the ts-WTA becomes large, it cannot be reversed.

It may be noted that the diffusion node (dno) voltage varies between 5.1 and 5.4 volts as the receptive field develops. After development, the response of a developed cell to the pattern that it favors, measured at the diffusion node (dno) is around 5.4 volts. Interestingly, if we apply 5.4 volts to node dno externally, for a pattern of our choice, and do this repeatedly, the circuit begins to develop preference for that orientation instead of its natural bias. Therefore, the receptive field development of the orientation selective cell can be modulated externally by applying appropriate voltage at the dno node of the cell for a particular pattern. This way we force a high response for a pattern of our choice, which causes the feedback mechanism to reinforce the desired pattern on to the individual ts-WTA cells in the $9 \times 9$ receptive field. It was observed that as the floating gate voltages become more developed (developed floating gate voltages mean that the difference between the floating gate voltages of the two synapses of the individual ts-WTA cells has become large) it becomes difficult to modulate the orientation preference of the cell. For fully developed floating gate voltages, i.e., strong orientation preference, modulation does not happen at all, and the cells preserve their original response as expected. Details of how the floating

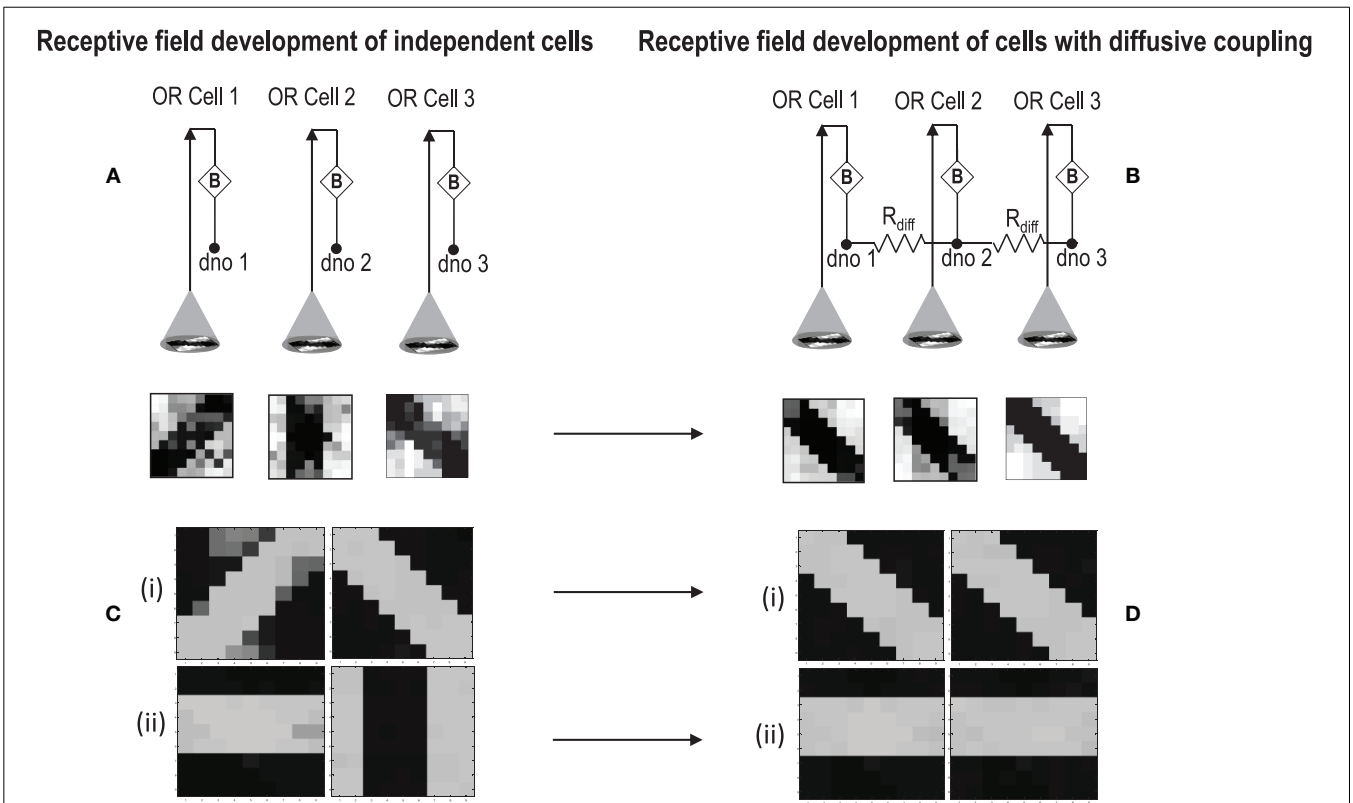



**FIGURE 12 | (A)** Shows the independent development of receptive fields of three orientation selective cells with different initial biases and same random inside epoch order of inputs. **(B)** Shows the development of the same three cells with the initial conditions and order of inputs same as **(A)**, but with diffusive interaction between neighbors. All the cells develop similar feature preference. **(C)** Two more example of cells developing independently under the same random inside epoch order of inputs but different initial biases. **(D)** Shows the development of the same cells as **(C)** under diffusive coupling. Diffusion causes the cells to develop the same feature preference in each case.

gate voltages vary during unlearning and the influence of injection and tunnel voltages are examined critically in Markan et al. (2013).

### 5.2. BUFFER DEVICE FOR DIFFUSIVE COUPLING

When more than one orientation cells are connected with each other diffusively using resistances at the diffusion node (dno), the increased current at the node dno tends to undesirably load the output node or OR cell output (out) (**Figures 5**, **8**). Since the OR cell output (out) node conveys the self-activation of each cell, this value should not get altered. In order to avoid this loading effect, we designed a buffer device (B) that shields the orientation cell output (activation) from the excessive current coming to the node dno of each orientation cell from other diffusively coupled cells. This device ensures that the self-activation (feedforward network) of the orientation cell driving the voltage at node out can influence the voltage at node dno, that drives the feedback network, but node dno cannot influence the voltage at node out directly. This buffer device is essentially a linear device that inverts the voltage at the OR cell output (out) twice and feeds it to the dno node (see **Figures 8**, **11**). This way current only flows in one direction, i.e., out of OR cell output node and not into it. A typical design of the buffer device is shown in **Figure 11B**, however, any other device performing the same function can be used as well.

### 5.3. SIMULATION OF DIFFUSIVE INTERACTION BETWEEN CELLS

In order to test if our orientation cell fulfills the premise laid down for diffusive interaction between cells, we performed multiple simulations with orientation cells having different initial biases but similar random inside epoch order of inputs, and we let them develop under two conditions, (1) independently, i.e., without any diffusive interaction and, (2) with diffusive interaction. As discussed previously, the voltage at node dno affects the feedback that regulates the response of the cell. If we connect two orientation cells at the diffusive node (dno) by means of a resistance, then on receiving similar inputs, the cell with the higher voltage at node dno, starts to influence the response of the other cell by making the injection and tunnel feedback mechanisms of both the cells similar, thus enforcing the same pattern on each of the cells. By changing the value of the diffusion resistance (by increasing the resistance we reduce diffusion constant and by reducing its value we increase the diffusion) we can modify the extent of interaction we want between the cells. Several experiments were performed with different diffusion constants, different biases and different inputs. Each time for moderate($300 > R_{\text{diff}} > 100$ Ohms) and high values of diffusion constant($100 > R_{\text{diff}} > 0$ Ohms), it was found that the response of the two cells became similar. To which side the orientation preference tilts is dependent on which cell has a stronger bias. The simulations were done for two and three cells connected in a row. **Figure 12** shows some of the interesting results. Irrespective of the way the cells develop independently, whether one is ON-Centered and other OFF-Centered, whether their orientation preferences are totally opposite of each other i.e., 135° and 45°, with diffusion, they become selective to the same orientation. It is important to note that the lateral diffusive network and the feedback network are only important as long as the learning is taking place and the receptive field of the cells are developing. Once the receptive fields have evolved, the lateral connectivity i.e., RC diffusive network and the cell's feedback network become ineffective and the cell work's in a feed forward mode where in on applying a set of inputs, the cell responds according to its developed orientation preference. The power dissipation also varies according to the learning profile of the cell e.g., between two orientation cells connected by a 100 ohm resistance, the current through the diffusive resistor is maximum (∼150 μA) during learning but reduces drastically (∼10 μA) once the learning is over (**Figure 13**). The power can be reduced by shifting the whole resistance regime of the cell to larger values but keeping the necessary ratio between ($\tau_{\text{Diffusion}}$) and ($\tau_{\text{Reaction}}$) intact.

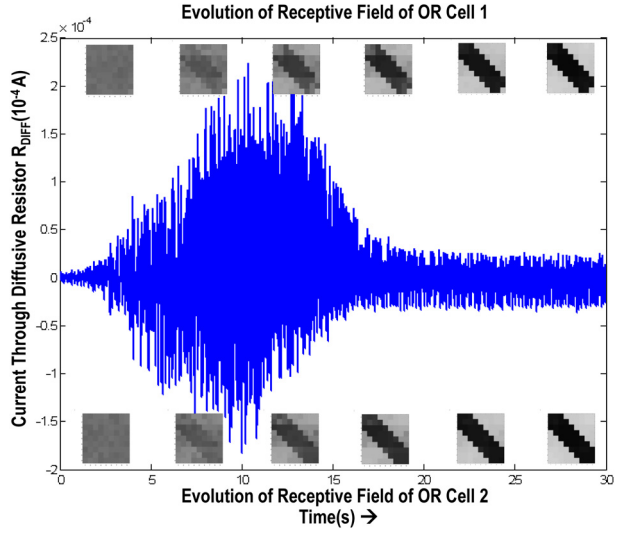



**FIGURE 13 | Shows the variation of current in the resistance, $R_{\text{diff}}$, connecting two orientation selective cells.** The top boxes show the evolution of receptive field of orientation cell 1 and the bottom boxes show the evolution of receptive field of orientation cell 2. The current is high (∼150 μA) during the learning phase. Once the orientation has been learnt, or the floating gate voltages have matured, the current reduces and remains constant thereafter.

## 6. RESULTS AND DISCUSSION

Time-staggered or uncorrelated inputs have been shown to be essential for feature map formation (Stryker and Strickland, 1984; Weliky and Katz, 1997; Buffeli et al., 2002; Zhang et al., 2002). The time-staggered Winner Takes All algorithm, based on uncorrelated inputs, has previously been shown to be biologically more realistic and a mechanism underlying formation of Ocular Dominance Maps (Markan et al., 2013). This paper introduces the design of a cortical cell that is built using ts-WTA cells comprising of ON/OFF Centered synapses forming a three layered structure similar to the visual sensory system in the brain. On application of patterns resembling different orientations, the floating gate dynamics, the diffusive interaction and the feedback regime act in a way that the cell is able develop orientation selectivity. Repeated simulations show that the orientation selectivity develops according to two major factors, initial biases(nature) and the inputs applied(nurture) and that there is an equal likelihood

of the circuit becoming selective to any of the eight patterns applied. Embedded in a RC grid, these orientation selective cells are able to modify their feature preference under strong neighborhood influence to form clusters of cells with similar feature preference. The cell also responds to periodic patterns and spatial frequency just like experimentally observed cells of the visual cortex. This is a significant step toward developing neuromorphic equivalents of biological phenomenon that could have diverse applications in artificial vision systems.

Diffusive hebbian learning based on reaction-diffusion and competition for neurotropic factors (Markan, 1996; Markan and Bhaumik, 1999; Bhaumik and Mathur, 2003), has strong biological support as basis to explain local computation and organization in the brain. It is now well known that the developing cortex is a generic neural structure that gets compartmentalized for processing different sensory inputs through an adaptive learning process. It therefore becomes important to explore the basic learning paradigms that are active in the brain, which are able to extract statistically relevant information from the sensory input space and map it onto the cortex, so that such principles can be applied in artificial systems. In this sense, the model developed is very generic and can be applied to inputs from any sensory modality such as olfaction, gustatory, somatosensory and auditory. Some preliminary work also demonstrates the applicability of the model to abstract pattern recognition. In the brain no sensory system works in isolation. Rather, it is a combination of sensory inputs to different sensory modalities that the brain responds best to. Eventual integration of features maps, corresponding to different sensory systems, onto a common platform could act as a database for higher cognitive algorithms to work on. The work presented in this paper is a small yet significant step toward the goal of building truly cognitive neuromorphic systems because it presents a novel approach towards incorporating adaptability and learning in artificial systems by modeling the developmental aspects of feature selectivity and feature map formation in the brain. While reaction diffusion has been able to address local range, non-axonal interactions in the brain and explain how cortical feature maps evolve to a large extent, more recent research has highlighted the role of gap junctions in lateral information processing in the brain (Hameroff, 2010; Ebner and Hameroff, 2011; Gupta and Markan, 2013). Experiments have revealed that sibling neurons connected by gap junctions develop to have the same feature preference (Li et al., 2012; Mrsic-Flogel and Bonhoeffer, 2012). Since gap junctions can form networks of neurons spanning large areas of the cortex, understanding how they function, could give us new insights into multi-modal information processing in the brain. It seems interesting to explore gap junctions and see how similar behavior can be emulated in hardware.

## ACKNOWLEDGMENTS

This work was funded by research grants to C. M. Markan, (III.6(74)/99-ST(PRU)) under SERC Robotics and Manufacturing PAC, and (SR/CSI/22/2008-12) under Cognitive Science Research Initiative, Department of Science and Technology, Govt. of India. The authors wish to acknowledge the funding sources and the Department of Physics and Computer Science, Dayalbagh Educational Institute, Agra, India for the support.

## SUPPLEMENTARY MATERIAL

A video showing the development of orientation receptive field has been made available as a part of the online supplementary data. A document on the Monte-Carlo Analysis of the cell under device parameter variations has also been provided in the supplementary section. The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2014.00054/abstract

## REFERENCES

Akerman, C. J., Smyth, D., and Thompson, I. D. (2002). Visual experience before eye-opening and the development of the retinogeniculate pathway. *Neuron* 36, 869–879. doi: 10.1016/S0896-6273(02)01010-3

Bartolozzi, C., and Indiveri, G. (2007). Synaptic dynamics in analog VLSI. *Neural Comput.* 19, 2581–2603. doi: 10.1162/neco.2007.19.10.2581

Bhaumik, B., and Markan, C. M. (2000). "Orientation map: a reaction diffusion based model," in *Proceedings of IJCNN '2000* (Como, Italy).

Bhaumik, B., and Mathur, M. (2003). A cooperation and competition based simple cell receptive field model and study of feed-forward linear and nonlinear contributions to orientation selectivity. *J. Comput. Neurosci.* 14, 211–227. doi: 10.1023/A:1021911019241

Buffeli, M., Busetto, G., Cangiano, L., and Cangiano, A. (2002). Perinatal switch from synchronous to asynchronous activity of motoneurons: link with synapse elimination. *Proc. Natl. Acad. Sci. U.S.A.* 99, 13200–13205. doi: 10.1073/pnas.202471199

Buzás, P., Eysel, U. T., Adorján, P., and Kisvárday, Z. F. (2001). Axonal topography of cortical basket cells in relation to orientation, direction, and ocular dominance maps. *J. Comp. Neurol.* 437, 259–285. doi: 10.1002/cne.1282

Carrillo, J., Nishiyama, N., and Nishiyama, H. (2013). Dendritic translocation establishes the winner in cerebellar climbing fiber synapse elimination. *J. Neurosci.* 33, 7641–7653. doi: 10.1523/JNEUROSCI.4561-12.2013

Cellerino, A., and Maffei, L. (1996). The action of neurotrophins in the development and plasticity of the visual cortex. *Prog. Neurobiol.* 49, 53–71. doi: 10.1016/S0301-0082(96)00008-1

Chakrabartty, S., and Cauwenberghs, G. (2007). Sub-microwatt analog VLSI trainable pattern classifier. *IEEE J. Solid State Circ.* 42, 1169–1179. doi: 10.1109/JSSC.2007.894803

Chan, V., Liu, S. C., and van Schaik, A. (2007). AER EAR: a matched silicon cochlea pair with address event representation interface. *IEEE Trans. Circ. Syst. I Reg. Pap.* 54, 48–59. doi: 10.1109/TCSI.2006.887979

Chapman, B., Stryker, M. P., and Bonhoeffer, T. (1996). Development of orientation preference maps in ferret primary visual cortex. *J. Neurosci.* 16, 6443–6453.

Chenling, H., and Chakrabartty, S. (2012). An asynchronous analog self-powered CMOS sensor-data-logger with a 13.56 MHz RF programming interface. *IEEE J. Solid-State Circ.* 47, 1–14. doi: 10.1109/JSSC.2011.2172159

Chicca, E., Whatley, A. M., Lichtsteiner, P., Dante, V., Delbruck, T., Del Giudice, P., et al. (2007). A multichip pulse-based neuromorphic infrastructure and its application to a model of orientation selectivity. *IEEE Trans. Circ. Syst. I Reg. Pap.* 54, 981–993. doi: 10.1109/TCSI.2007.893509

Choi, T. Y. W., Merolla, P. A., Arthur, J. V., Boahen, K. W., and Shi, B. E. (2005). Neuromorphic implementation of orientation hypercolumns. *IEEE Trans. Circ. Syst. I*, 52, 1049–1060. doi: 10.1109/TCSI.2005.849136

De Valois, R. L., Albrecht, D. G., and Thorell, L. G. (1982). Spatial frequency selectivity of cells in macaque visual cortex. *Vis. Res.* 22, 545–559. doi: 10.1016/0042-6989(82)90112-2

Diorio, C., Hasler, P., Minch, B., and Mead, C. (1996). A single transistor silicon synapse. *IEEE Trans. Electron Devices* 43, 1972–1980. doi: 10.1109/16.543035

Dragoi, V., Sharma, J., and Sur, M. (2000). Adaptation-induced plasticity of orientation tuning in adult visual cortex. *Neuron* 28, 287–298. doi: 10.1016/S0896-6273(00)00103-3

Ebner, M., and Hameroff, S. (2011). Lateral information processing by spiking neurons: a theoretical model of the neural correlate of consciousness. *Comput. Intell. Neurosci.* 2011:11. doi: 10.1155/2011/xya247879

Elliott, T., and Shadbolt, N. R. (1998). Competition for neurotrophic factors: ocular dominance columns. *J. Neurosci.* 18, 5850–5858.

Everson, R. M., Prashanth, A. K., Gabbay, M., Knight, B. W., Sirovich, L., and Kaplan, E. (1998). Representation of spatial frequency and orientation in the visual cortex. *Proc. Natl. Acad. Sci. U.S.A.* 95, 8334–8338. doi: 10.1073/pnas.95.14.8334

Favero, M., Busetto, G., and Cangiano, A. (2012). Spike timing plays a key role in synapse elimination at the neuromuscular junction. *Proc. Natl. Acad. Sci. U.S.A.* 109, E1667–E1675. doi: 10.1073/pnas.1201147109

Grossberg, S. (1976). Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. *Biol. Cybern.* 23, 121–134. doi: 10.1007/BF00344744

Grossberg, S., and Olson, S. J. (1994). Rules for the cortical map of ocular dominance and orientation columns. *Neural Netw.* 7, 883–894. doi: 10.1016/S0893-6080(05)80150-9

Gupta, P., and Markan, C. M. (2013). Exploring a quantum-Hebbian approach towards learning and cognition. *NeuroQuantology* 11, 416–425. doi: 10.14704/nq.2013.11.3.669

Hameroff, S. (2010). The conscious pilot-dendritic synchrony moves through the brain to mediate consciousness. *J. Biol. Phys.* 36, 71–93. doi: 10.1007/s10867-009-9148-x

Hikawa, H., Harada, K., and Hirabayashi, T. (2007). Hardware feedback self-organizing map and its application to mobile robot location identification. *JACIII* 11, 937–945.

Horng, S. H., and Sur, M. (2006). Visual activity and cortical rewiring: activity-dependent plasticity of cortical networks. *Prog. Brain Res.* 157, 3–381. doi: 10.1016/S0079-6123(06)57001-3

Hsu, D., Figueroa, M., and Diorio, C. (2002). Competitive learning with floating-gate circuits. *IEEE Trans. Neural Netw.* 13, 732–744. doi: 10.1109/TNN.2002.1000139

Hubel, D. H., and Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.* 148, 574–591.

Hubel, D. H., and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol.* 160, 106.

Indiveri, G. (2001). A current-mode hysteretic winner-take-all network, with excitatory and inhibitory coupling. *Analog Integr. Circ. Signal Process.* 28, 279–291. doi: 10.1023/A:1011208127849

Indiveri, G. (2008). Neuromorphic VLSI models of selective attention: from single chip vision sensors to multi-chip systems. *Sensors* 8, 5352–5375. doi: 10.3390/s8095352

Indiveri, G., Chicca, E., and Douglas, R. (2006). A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE Trans. Neural Netw.* 17, 211–221. doi: 10.1109/TNN.2005.860850

Indiveri, G., Chicca, E., and Douglas, R. J. (2009). Artificial cognitive systems: from VLSI networks of spiking neurons to neuromorphic cognition. *Cogn. Comput.* 1, 119–127. doi: 10.1007/s12559-008-9003-6

Indiveri, G., and Horiuchi, T. K. (2011). Frontiers in neuromorphic engineering. *Front. Neurosci.* 5:118. doi: 10.3389/fnins.2011.00118

Indiveri, G., Oswald, P., and Kramer, J. (2002). An adaptive visual tracking sensor with a hysteretic winner-take-all network. *IEEE Int. Symp. Circ. Syst.* 2, 324–327. doi: 10.1109/ISCAS.2002.1010990

Jegelka, S., Bednar, J. A., and Miikkulainen, R. (2006). Prenatal development of ocular dominance and orientation maps in a self-organizing model of V1. *Neurocomputing* 69, 1291–1296. doi: 10.1016/j.neucom.2005.12.094

Kohonen, T. (1993). Physiological interpretation of the self organizing map algorithm. *Neural Netw.* 6, 895–905. doi: 10.1016/S0893-6080(09)80001-4

Kohonen, T. (2006). Self-organizing neural projections. *Neural Netw.* 19, 723–733. doi: 10.1016/j.neunet.2006.05.001

Krekelberg, B. (1997). *Modelling cortical self-organization by volume learning*. London: Doctoral dissertation.

Kruger, W. F., Hasler, P., Minch, B. A., and Koch, C. (1997). An adaptive WTA using floating gate technology. *Adv. Neural Inform. Processing Syst.* 720–726.

Kuffler, S. W. (1953). Discharge patterns and functional organization of mammalian retina. *J. Neurophysiol.* 16, 37–68.

Lam, S. Y. M., Shi, B. E., and Boahen, K. (2005). Self-organized cortical map formation by guiding connections. *IEEE Int. Symp. Circ. Syst.* 5, 5230–5233.

Lazzaro, J., Ryckebusch, S., Mahowald, M. A., and Mead, C. A. (1989). *Winner-Take-All Networks of O (n) Complexity NIPS 1*. San Mateo, CA: Morgan Kaufman Publishers.

Li, Y., Lu, H., Cheng, P. L., Ge, S., Xu, H., Shi, S. H., et al. (2012). Clonally related visual cortical neurons show similar stimulus feature selectivity. *Nature* 486, 118–121. doi: 10.1038/nature11110

Lichtman, J. W. (2009). It's lonely at the top: winning climbing fibers ascend dendrites solo. *Neuron* 63, 6–8. doi: 10.1016/j.neuron.2009.07.001

Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128× 128 120 dB 15 $\mu$s latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circ.* 43, 566–576. doi: 10.1109/JSSC.2007.914337

Maffei, L., and Fiorentini, A. (1973). The visual cortex as a spatial frequency analyser. *Vis. Res.* 13, 1255–1267. doi: 10.1016/0042-6989(73)90201-0

Markan, C. M. (1996). *Sequential development of orientation and ocular dominance maps: reaction diffusion approach*. Ph.D. thesis, Department of Electrical Engineering, (Delhi: IIT).

Markan, C. M., and Bhaumik, B. (1999). "A diffusive Hebbian model for cortical orientation maps formation," in *Proceedings of IJCNN '99* (Washington, DC). doi: 10.1109/IJCNN.1999.831482

Markan, C. M., Gupta, P., and Bansal, M. (2007). "Neuromorphic building blocks for adaptable cortical feature maps," *IFIP International Conference on VLSI*, 15–17 Oct 2007 (Atlanta, GA).

Markan, C. M., Gupta, P., and Bansal, M. (2013). An adaptive neuromorphic model of Ocular Dominance map using floating gate "synapse." *Neural Netw.* 45, 117–133. doi: 10.1016/j.neunet.2013.04.004

Martn-del-Bro, B., and Blasco-Alberto, J. (1995). "Hardware-oriented models for VLSI implementation of self-organizing maps," in *From Natural to Artificial Neural Computation*, eds J. Mira and F. Sandoval (Berlin: Springer), 712–719. doi: 10.1007/3-540-59497-3_242

McAllister, A. K., Katz, L. C., and Lo, D. C. (1999). Neurotrophins and synaptic plasticity. *Annu. Rev. Neurosci.* 22, 295–318. doi: 10.1146/annurev.neuro.22.1.295

Merolla, P. A., Arthur, J. V., Shi, B. E., and Boahen, K. A. (2007). Expandable networks for neuromorphic chips. *IEEE Trans. Circ. Syst. I Reg. Pap.* 54, 301–311. doi: 10.1109/TCSI.2006.887474

Miller, K. D. (1994). A model for the development of simple cell receptive fields and the ordered arrangement of orientation columns through activity-dependent competition between ON and OFF-center inputs. *J. Neurosci.* 14, 409–409.

Miller, K. D. (1996). "Receptive fields and maps in the visual cortex: models of ocular dominance and orientation columns," in *Models of neural networks III* (New York, NY: Springer), 55–78. doi: 10.1007/978-1-4612-0723-8_2

Miller, K. D., and MacKay, D. J. (1994). The role of constraints in Hebbian learning. *Neural Comput.* 6, 100–126. doi: 10.1162/neco.1994.6.1.100

Misgeld, T. (2011). Lost in elimination: mechanisms of axonal loss. *e-Neuroforum* 2, 21–34. doi: 10.1007/s13295-011-0017-2

Mooney, R., Penn, A. A., Gallego, R., and Shatz, C. J. (1996). Thalamic relay of spontaneous retinal activity prior to vision. *Neuron* 17, 863–874. doi: 10.1016/S0896-6273(00)80218-4

Mrsic-Flogel, T. D., and Bonhoeffer, T. (2012). Neuroscience: sibling neurons bond to share sensations. *Nature* 486, 41–42. doi: 10.1038/486041a

Personius, K. E., Chang, Q., Mentis, G. Z., O'Donovan, M. J., and Balice-Gordon, R. J. (2007). Reduced gap junctional coupling leads to uncorrelated motor neuron firing and precocious neuromuscular synapse elimination. *Proc. Natl. Acad. Sci. U.S.A.* 104, 11808–11813. doi: 10.1073/pnas.0703357104

Rahimi, K., Diorio, C., Hernandez, C., and Brockhausen, M. D. (2002). "A simulation model for floating-gate MOS synapse transistors," in *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, Vol. 2 (Phoenix-Scottsdale, AZ: IEEE), II–532. doi: 10.1109/ISCAS.2002.1011042

Roerig, B., and Chen, B. (2002). Relationships of local inhibitory and excitatory circuits to orientation preference maps in ferret visual cortex. *Cereb. Cortex* 12, 187–198. doi: 10.1093/cercor/12.2.187

Schemmel, J., Fieres, J., and Meier, K. (2008). "Wafer-scale integration of analog neural networks," in *IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)* (Hong Kong: IEEE), 431–438. doi: 10.1109/IJCNN.2008.4633828

Sengpiel, F., Stawinski, P., and Bonhoeffer, T. (1999). Influence of experience on orientation maps in cat visual cortex. *Nat. Neurosci.* 2, 727–732. doi: 10.1038/11192

Seriès, P., Latham, P. E., and Pouget, A. (2004). Tuning curve sharpening for orientation selectivity: coding efficiency and the impact of correlations. *Nat. Neurosci.* 7, 1129–1135. doi: 10.1038/nn1321

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gomez-Rodriguez, F., et al. (2005). "AER building blocks for multi-layer multi-chip neuromorphic vision systems," in *NIPS* (Vancouver, BC: Vancouver).

Shi, B. E. (2009). The effect of mismatch in currentversus voltagemode resistive grids. *Int. J. Circ. Theor. Appl.* 37, 53–65. doi: 10.1002/cta.494

Shi, B. E., Tsang, E. K. S., Lam, S. Y., and Meng, Y. (2006). "Expandable hardware for computing cortical feature maps," in *Proceedings 2006 IEEE International Symposium on Circuits and Systems, 2006. ISCAS 2006* (Island of Kos: IEEE). doi: 10.1109/ISCAS.2006.1693407

Shouval, H. Z., Goldberg, D. H., Jones, J. P., Beckerman, M., and Cooper, L. N. (2000). Structured long-range connections can provide a scaffold for orientation maps. *J. Neurosci.* 20, 1119–1128.

Shuo, S., and Basu, A. (2011). "Analysis and reduction of mismatch in silicon neurons," in *Proceedings of IEEE Biomedical Circuits and Systems Conference* (San Diego, CA), 257–260.

Somers, D. C., Nelson, S. B., and Sur, M. (1995). An emergent model of orientation selectivity in cat visual cortical simple cells. *J. Neurosci.* 15, 5448–5465.

Srinivasan, V., Graham, D. W., and Hasler, P. (2005). "Floating-gates transistors for precision analog circuit design: an overview," in *48th Midwest Symposium on Circuits and Systems, 2005* (Covington, KY: IEEE), 71–74. doi: 10.1109/MWSCAS.2005.1594042

Stent, G. S. (1973). A physiological mechanism for Hebb's postulate of learning. *Proc. Natl. Acad. Sci. U.S.A.* 70, 997–1001. doi: 10.1073/pnas.70.4.997

Stryker, M. P., and Strickland, S. L. (1984). Physiological segregation of ocular dominance columns depends on the pattern of afferent electrical activity. *Invest. Opthalmol. Vis. Sci.* 25, 278.

Sur, M., and Leamey, C. A. (2001). Development and plasticity of cortical areas and networks. *Nat. Rev. Neurosci.* 2, 251–262. doi: 10.1038/35067562

Taba, B., and Boahen, K. (2002). Topographic map formation by silicon growth cones. *Proc. NIPS* 1139, 1146.

Tootell, R. B., Silverman, M. S., and De Valois, R. L. (1981). Spatial frequency columns in primary visual cortex. *Science* 214, 813–815. doi: 10.1126/science.7292014

Turney, S. G., and Lichtman, J. W. (2012). Reversing the outcome of synapse elimination at developing neuromuscular junctions *in vivo*: evidence for synaptic competition and its mechanism. *PLoS Biol.* 10:e1001352. doi: 10.1371/journal.pbio.1001352

Von der Heydt, R., Peterhans, E., and Dursteler, M. R. (1992). Periodic-pattern-selective cells in monkey visual cortex. *J. Neurosci.* 12, 1416–1434.

Weliky, M., and Katz, L. (1997). Disruption of orientation tuning visual cortex by artificially correlated neuronal activity. *Nature* 386, 680–685. doi: 10.1038/386680a0

Wijekoon, J., and Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behaviour. *Neural Netw.* 21, 524–534. doi: 10.1016/j.neunet.2007.12.037

Wong, R. O. (1999). Retinal waves and visual system development. *Annu. Rev. Neurosci.* 22, 29–47. doi: 10.1146/annurev.neuro.22.1.29

Wyatt, R. M., and Balice-Gordon, R. J. (2003). Activity-dependent elimination of neuromuscular synapses. *J. Neurocytol.* 32, 777–794. doi: 10.1023/B:NEUR.0000020623.62043.33

Yoshida, T., Ozawa, K., and Tanaka, S. (2012). Sensitivity profile for orientation selectivity in the visual cortex of goggle-reared mice. *PLoS ONE* 7:e40630. doi: 10.1371/journal.pone.0040630

Yousef, T., Tóth, E., Rausch, M., Eysel, U. T., and Kisvárday, Z. F. (2001). Topography of orientation centre connections in the primary visual cortex of the cat. *Neuroreport* 12, 1693–1699. doi: 10.1097/00001756-200106130-00035

Zamarreño-Ramos, C., Camuñas-Mesa, L. A., Perez-Carrasco, J. A., Masquelier, T., Serrano-Gotarredona, T., and Linares-Barranco, B. (2011). On spike-timing-dependent-plasticity, memristive devices, and building a self-Learning visual cortex. *Front. Neurosci.* 5:26. doi: 10.3389/fnins.2011.00026

Zhang, L., Bao, S., and Merzenich, M. (2002). Disruption of primary auditory cortex by synchronous auditory inputs during critical period. *Proc. Natl. Acad. Sci. U.S.A.* 99, 2309–2314. doi: 10.1073/pnas.261707398

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 31 August 2013; accepted: 09 March 2014; published online: 02 April 2014.*
*Citation: Gupta P and Markan CM (2014) An adaptable neuromorphic model of orientation selectivity based on floating gate dynamics. Front. Neurosci.* ***8****:54. doi: 10.3389/fnins.2014.00054*
*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 18 March 2014
doi: 10.3389/fnins.2014.00051

# A mixed-signal implementation of a polychronous spiking neural network with delay adaptation

***Runchun M. Wang*, Tara J. Hamilton, Jonathan C. Tapson and André van Schaik***

*Bioelectronics and Neuroscience, The MARCS Institute, University of Western Sydney, Sydney, NSW, Australia*

***Edited by:***
*Jennifer Hasler, Georgia Institute of Technology, USA*

***Reviewed by:***
*Christian G. Mayr, Dresden University of Technology, Germany*
*Arindam Basu, Nanyang Technological University, Singapore*

****Correspondence:***
*Runchun M. Wang, Bioelectronics and Neuroscience, The MARCS Institute, University of Western Sydney, Locked Bag 1797, Penrith, NSW 2751, Australia*
*e-mail: mark.wang@uws.edu.au*

We present a mixed-signal implementation of a re-configurable polychronous spiking neural network capable of storing and recalling spatio-temporal patterns. The proposed neural network contains one neuron array and one axon array. Spike Timing Dependent Delay Plasticity is used to fine-tune delays and add dynamics to the network. In our mixed-signal implementation, the neurons and axons have been implemented as both analog and digital circuits. The system thus consists of one FPGA, containing the digital neuron array and the digital axon array, and one analog IC containing the analog neuron array and the analog axon array. The system can be easily configured to use different combinations of each. We present and discuss the experimental results of all combinations of the analog and digital axon arrays and the analog and digital neuron arrays. The test results show that the proposed neural network is capable of successfully recalling more than 85% of stored patterns using both analog and digital circuits.

**Keywords: mixed-signal implementation, polychronous spiking neural network, analog implementation, multiplexed neuron array, neuromorphic engineering**

## INTRODUCTION

Increasing evidence has been found that the mammalian neural system uses spatio-temporal coding in at least some of its operations (Van Rullen and Thorpe, 2001; Masuda and Aihara, 2003), largely due to this coding's potential to reduce energy consumption (Levy and Baxter, 1996). An artificial network that can learn and recall spatial and temporally encoded spike information will have significant benefits in terms of modeling these biological systems.

A polychronous spiking neural network is a candidate for implementing a memory for spatio-temporal patterns. Polychronization is the process in which spikes travel down axons with specific delays to arrive at a common target neuron simultaneously and cause it to fire, despite the source neurons firing asynchronously (Izhikevich, 2006). This time-locked relation between the firing of different neurons is the key feature of spatio-temporal patterns. Neural networks based on this principle are referred to as "polychronous" neural networks and are capable of storing and recalling quite complicate spatio-temporal patterns. **Figure 1** shows an example of a spatio-temporal pattern involving five neurons. The threshold voltage of each neuron is set so that it will fire if two pre-synaptic spikes arrive simultaneously. Whenever a neuron fires, its spike is transmitted to all connected neurons via its axonal connections, each of which has its own independent delay. These spikes will then generate post-synaptic currents at the connected neurons. The example pattern starts when neuron 1 fires at time 0 and neuron 5 fires at time T1. The spikes from both neurons will arrive at neuron 3 at time T1+T2, and together they will induce neuron 3 to fire at time T1+T2. In the same manner, the spikes from neuron 5 and neuron 3 arrive at neuron 2 simultaneously at time T1+T2+T3 and will cause neuron 2 to fire. This process will continue as long as at least two spikes arrive simultaneously at a neuron in the network.

Izhikevich (2006) calls these spatio-temporal patterns groups, and concludes that "spiking networks with delays have more groups than neurons" after presenting a network developed based on this polychronous principle. The groups in Izhikevich's network emerge in a randomly connected network of spiking neurons with axonal delays, following persistent stimulation and Spike Timing Dependent Plasticity (STDP) (Gerstner et al., 1996). However, one of the open problems of the theoretical model is to find patterns (groups): "Our algorithm for finding polychronous groups considers various triplets firing with various spiking patterns and determines the groups that are initiated by the patterns. Because of the combinatorial explosion, it is extremely inefficient" (Izhikevich, 2006). The method used by Izhikevich will take months of simulation time just to find these spatio-temporal patterns. Moreover, the polychronous groups emerge randomly and the same stimulus is not likely to result in the same polychronous groups every time. This makes the Izhikevich polychronous network unsuitable for practical applications such as pattern recognition. Finally this model is not efficient for hardware implementations, which we will discuss in detail in section Discussion.

To solve the problems presented above, we have proposed a digital implementation of a reconfigurable polychronous spiking neural network that can, in real time, learn specific patterns, and retrieve them (Wang et al., 2013b). Furthermore, our proposed polychronous neural network can use all the available hardware resources to store patterns. Test results show that the proposed neural network is capable of successfully recalling more than 95% of all spikes for 96% of the stored patterns. Unlike biological neural networks, the digital implementation is totally free

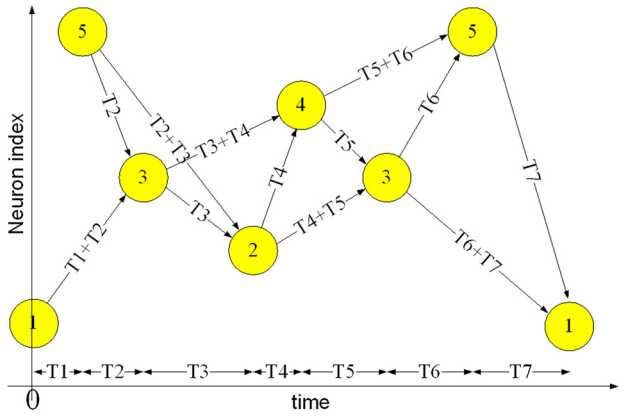



**FIGURE 1 | Example of a spatio-temporal pattern.** The neurons fire asynchronously while their spikes arrive at the destination neurons synchronously, after traveling along axons with appropriate delays. This time-locked relation is the key feature of the spatio-temporal patterns.

of mismatch and noise. Therefore, we also designed an analog implementation, which is naturally subject to process variation and device mismatch, and which more closely emulates the analog computation in biological neurons.

Mixed-signal implementations of spiking neural networks benefit from many of the advantages of both analog and digital implementations. Analog implementations can realize biological behaviors of neurons in a very efficient manner, whereas digital implementations can provide the re-configurability needed for rapid prototyping of spiking neural networks. As a result, mixed-signal implementations offer an attractive neural network and many designs have been proposed for such systems (Goldberg et al., 2001; Gao and Hammerstrom, 2007; Mirhassani et al., 2007; Vogelstein et al., 2007; Harkin et al., 2008, 2009; Schemmel et al., 2008; Saighi et al., 2010; Yu and Cauwenberghs, 2010; Zaveri and Hammerstrom, 2011; Minkovich et al., 2012).

These proposed systems tend to employ programmable devices such as FPGAs and ASICs to route the spikes between analog computation modules. Some programmable platforms using floating gates (Basu et al., 2010; Brink et al., 2013). Furthermore, most of these systems use DACs to configure the analog modules to emulate different biological behaviors. Implementations of spiking neural networks with time-multiplexed analog circuits are described in Mirhassani et al. (2007), Yu and Cauwenberghs (2010), Minkovich et al. (2012) and a version that uses nanotechnology is described in Gao and Hammerstrom (2007), Zaveri and Hammerstrom (2011).

Here, we report on a mixed-signal platform, which combines both our analog and digital implementations and provides test results. Section Proposed Polychronous Network gives an overview of the proposed polychronous neural network. Section Design Choice presents the design choices that have been made for the neuromorphic implementation of the proposed polychronous network. The analog building blocks of the polychronous network (i.e., the neurons, axons, and other analog components) are detailed in section Analogue Implementation. Section Mixed-signal Implementation presents the proposed mixed-signal implementation, which includes the multiplexed analog neuron array and the interface between the asynchronous communication of the analog array and the (synchronous) FPGA. Measured results and a comparison to the fully digital implementation are given in section Results. In Section Discussion we discuss the performance of the different implementations and the key elements that influence the capacity and scaling of electronic realizations of polychronous networks and we conclude in section Conclusions.

## MATERIALS AND METHODS

### PROPOSED POLYCHRONOUS NETWORK

#### *Training and recalling patterns*

Two procedures are needed to use our proposed polychronous network to memorize and recall spatio-temporal patterns. The first is a training procedure in which the connection delay values of the axon paths between neurons are configured in order to meet the required timing relations of a given pattern. The second is a recall procedure, needed to retrieve a pattern that has been stored in the neural network through training. A pattern can be recalled by presenting the first few spikes of the pattern to the network, after which the network will complete the pattern if it is recognized. For example, to recall the example pattern shown above, neuron 1 needs to fire at time 0 and neuron 5 needs to fire at time T1. Together they will cause neuron 3 to fire and the remainder of the pattern will be induced by the network. The network is also capable of recalling parts of patterns that start somewhere in the middle, e.g., neuron 2 firing at time T1+T2+T3 and neuron 4 firing at time T1+T2+T3+T4 will retrieve the remainder of the example pattern.

The goal of the training procedure is to assign appropriate connection delays to axons in the polychronous neural network so that it is able to recall a specific pattern. We propose two mechanisms, which are *delay programming* and *delay adaptation*, to implement this function. Delay programming relies on a connection storing the delay value between a spike from its input neuron and a spike from its output neuron when both are induced to fire by some external training signal. It is not a biologically plausible method, but it is efficient in training and reduces testing time in scenarios where the result will not be affected by the training method. We therefore commonly use it to initialize a network.

Inspired by STDP, we developed a *delay adaptation* method, Spike Timing Dependent Delay Plasticity (STDDP), to fine-tune the delays during the training phase. We decrease the delay value of one axon by a small amount if the destination neuron fires (generating the post-synaptic spike) before the pre-synaptic spike arrives (at the synapse of the destination neuron), and we increase the delay in the opposite case. This procedure is repeated until the pre-synaptic spike arrives at the synapse simultaneously with the post-synaptic spike being generated. In the training phase, delay adaptation causes the connections to attain the desired delays through repeated presentation of the desired spatio-temporal patterns. The *delay programming* method can be regarded as a special case of the *delay adaptation* method in which the delay adaption is completed in just a single step and the delay is never altered subsequently. With the *delay adaptation* method, every time a pattern is recalled the delay values in the pattern will be updated,

allowing the learned delays to be modified over time. Hardware implementations of non-polychronous networks that also adapt axonal delays can be found in (Hussain et al., 2012, in press).

### *Neural network structure*

The structure of the proposed neural network is shown in **Figure 2**. It contains two functional parts: a "neuron array" and an "axon array." The neurons and the axons communicate with each other via Address-Event Representation (AER) buses (Boahen, 2000). Each neuron in the neuron array is identical in structure and has a unique AER address. The axon modules in the axon array are also identical in structure, and have both a unique physical address (their position in the array) and configurable input and output addresses, to place an axon between two neurons. The axon modules generate pre-synaptic spikes, which are received by the neurons. The neurons will then generate post-synaptic spikes if more than a certain number of pre-synaptic spikes arrive simultaneously. To decrease the likelihood of cross-talk between patterns, i.e., that a coincidence detecting neuron would be set off by a random coincidence, we used coincidence detectors with four inputs and a threshold of three spikes (Wang et al., 2013b).

The post-synaptic spikes are sent to the axon modules in the axon array. The axon array propagates these post-synaptic spikes with axonal-specific delay values and generates pre-synaptic spikes at the end of the axons. In the proposed neural network, the communication between any two neurons must be conducted via the axon modules in order to implement the polychronous network. This axon array, with reconfigurable input and output addresses, is capable of achieving much higher resource utilization than the method we have used previously (Wang et al., 2011), which generated spatio-temporal patterns based on fixed connectivity between neurons. That approach always resulted in networks where some axons remained unused. Our current approach is to generate delay paths *de novo*, so that only connections that actually appear in the training patterns will be created, by configuring the appropriate input and output addresses for each axon. Additionally we configured the system such that there can be any number of axonal delay paths between any two neurons in the network. In other words, several axons can have identical input and output addresses, placing them between the same two neurons. They would still be able to have different delay values, so that a spike originating from the input neuron would arrive at the output neuron multiple times after different delays, emulating the case where a neuron makes multiple synapses with another neuron.

The axon module (see **Figure 3**) has five address registers, one ramp generator, and four identical axonal delay paths. The address registers are used to store the input address and the four output addresses for the axonal delay paths. To place one axon module between neurons, we need to configure its address registers. At the beginning of the training, axon module[0] (see **Figure 2**) is enabled and all the other axon modules are disabled. When the first post-synaptic spike in a training pattern arrives, axon module[0] will latch the address of this spike as its input address and enable axon module[1]. The output addresses will be configured after the input address is configured. As there are four output addresses, one for each of the destination neurons, it will take four iterations for one axon module to finish the configuration of its output addresses (using the addresses of the next four sequential post-synaptic spikes in the training pattern after its input address is configured).

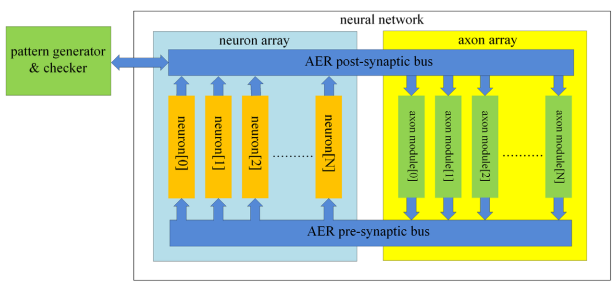



**FIGURE 2 | Structure of the proposed polychronous neural network.** The neuron array generates post-synaptic spikes and then sends them to the axon array, which propagates these post-synaptic spikes, with programmable axonal delays, and generates the pre-synaptic spikes at the end of the axons. These pre-synaptic spikes are sent to the neuron array to cause the neurons to fire. The connectivity and delay of all the axons in the axon array are configurable.

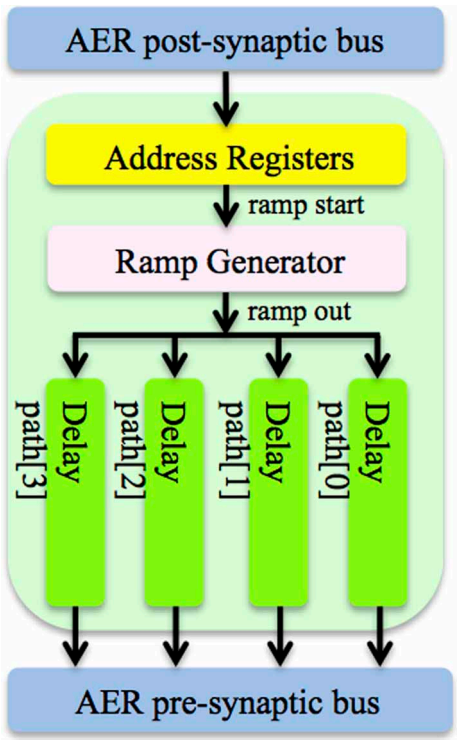



**FIGURE 3 | Structure of the axon module.** The axon module receives post-synaptic spikes generated by the neuron in the neuron array via the AER post-synaptic bus. The axon module propagates these spikes with axonal-specific programmable delays and generates pre-synaptic spikes at the end of the axons. The address registers are used to store the input address and the four output addresses for the axonal delay paths.

Delay programming is carried out in the same way as the address configuration. When the first post-synaptic spike arrives at axon module[0], it will start a ramp generator, which will send its value (ramp_out) to the four axonal delay paths. The delay of each axonal delay path is programmed when the output addresses are being configured (i.e., when the next four sequential post-synaptic spikes from the training pattern arrive). After delay programming, when a post-synaptic spike arrives and its address matches the input address of one axon module, it will start the ramp generator again. The axonal delay path will compare the output of the ramp generator with the programmed delay. A pre-synaptic spike will be generated when the output of the ramp generator exceeds the programmed delay with an address as stored in the output address register. The delays can also be configured using delay adaptation rather than delay programming. In this case the axonal delay is increased or decreased based on the delay between pre-synaptic spike and post-synaptic spike by using one of the three strategies: exact correction of the delay error in one step, correction of the error by a fixed amount each time, or correction by an amount proportional to the error. We have implemented all three strategies in the digital axon module. The first method is identical to just using the delay programming method. The second method, which uses a small fixed step, is very slow and produces similar results to the third method with a coefficient of 0.5. The digital axon presented here uses the third strategy. Slightly differently, the delay of the analog axon is programmed in an initial phase followed by a number of iterations of delay adaptation with a fixed update step, which was the simplest method to implement. An analog implementation that implements all three strategies would be too large for practical implementation on silicon.

## DESIGN CHOICE

### *Topology*

**Figure 4** shows the topology of the proposed mixed-signal platform. It consists of one FPGA and one analog chip containing an analog neuron array and an analog axon array. The FPGA contains the digital axon array, the digital neuron array, a pattern generator and checker module for training and testing, and a router. The function of the router is to remap the addresses of the spikes between the digital implementation and the analog implementation; but in practice the router also needs to synchronize the spikes from the analog circuits before it can remap the addresses for these spikes. This is due to the analog circuits operating asynchronously and therefore without a clock, whereas the router is a fully digital design, which does require a clock. The spikes from the analog circuit therefore have to be synchronized to the clock domain in which the router works. We will present the design of an interface circuit for synchronization, followed by a circuit to implement the address remapping in section Synchronization Interface Circuit.

The system contains two types of implementations for the axon array and two for the neuron array, resulting in four potential combinations, which are presented below:

1. *A digital axon array and a digital neuron array*: This is simply the default FPGA implementation.

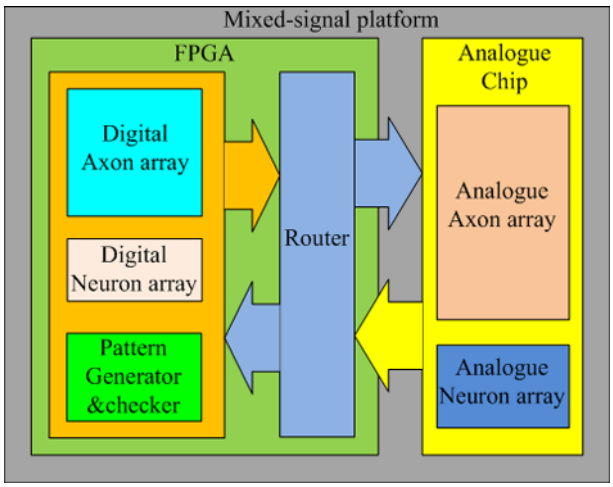



**FIGURE 4 | Topology of the mixed-signal platform.** The FPGA contains the digital axon and neuron array, a router to control the destinations of spikes on the bus, and a pattern generator and checker for testing purposes. A separate IC contains the analog implementations of the axon and neuron arrays.

2. *Digital axon array and analog neuron array*: In this configuration, the router is required to re-map the addresses of the spikes transmitted between the analog neuron array and the digital axon array.
3. *Analog axon array and digital neuron array:* In this configuration, the router is also required to re-map the addresses of the spikes transmitted between the digital neuron array and the analog neuron array.
4. *Analog axon array and analog neuron array*: Despite having only analog implementations, the router is still required to transmit spikes between the analog axon array and the analog neuron array, as the addresses still require remapping. This is done to multiplex the analog neurons, so that inactive neurons in the network are not using hardware resources. This increases the size of the analog neuron array significantly. We will present the details of this approach in section Mixed-signal Implementation.

The neurons in the neuron array work as coincidence detectors that detect how many pre-synaptic spikes have arrived simultaneously. The FPGA implementation of these neurons uses four timers and one comparator (see Wang et al., 2013b). The analog version of these neurons is implemented using simple Leaky Integrate and Fire (LIF) neurons, which will be described in detail in section Analog Neuron Array. Since no complicated biological behaviors, such as spike rate adaptation or bursting, are required for the neurons in a polychronous network, we chose to implement LIF neurons, instead of more complex neuron models, e.g., the Izhikevich neuron model (Izhikevich, 2003) and the Mihalas-Niebur neuron model (Mihala and Niebur, 2009), to keep the size of the neuron circuit to a minimum.

For the axon module, the FPGA implementation uses a counter to implement the ramp generator, and registers to store

the delay values. In the analog implementation, the ramp generator is implemented with a circuit that starts charging a MOS capacitor after receiving a spike on the AER bus. The axonal delay is generated by comparing a programmable voltage, stored on a capacitor, with the output signal of the ramp generator. The design and implementation of the ramp generator and the delay path can be found in (Wang et al., 2013a).

### *AER bus*

There are two different AER buses in the proposed neural network: the AER post-synaptic bus and the AER pre-synaptic bus. The first is used to transmit post-synaptic spikes generated by the neurons to the axon modules. The second is used to transmit pre-synaptic spikes generated by the axon modules to the neurons (see **Figure 3**). The AER bus and protocol used in this system differs slightly from the standard AER bus and protocol (Boahen, 2000). We do not use handshaking, so we have omitted the request and acknowledge signals. Instead we use "active" lines to tell the receiver (neurons or axon modules) that a spike has been placed on the bus. Each neuron receives input from four neurons via four axons in our network. The pre-synaptic bus therefore uses four active lines, one for each synapse of the neuron. A further difference in our AER implementation is that there is no arbiter to deal with collisions when two addresses are placed on the bus simultaneously. We will address this issue in detail in section Discussion.

In our digital implementation, a single minimum-width binary address is used to reduce hardware costs, as the wiring for the bus will entail more resources than the implementation of the encoders/decoders in large scale FPGA designs (Harkin et al., 2008). This structure, however, doesn't satisfy our analog implementation, in which a full encoder/decoder will cost more area than the analog neuron itself in a 0.6 μm technology (typically each bit needs one XOR gate with 16 transistors in a full decoder). The AER buses in the analog neuron array use active lines and a 3/8-bit (three out of eight) address for which the encoding and decoding can be efficiently implemented in aVLSI, as will be shown in section Analog Neuron Array. The number of different addresses, $C$, for this code are given by the binomial coefficient:

$$C_M^N = \frac{M!}{N!(M-N)!} \tag{1}$$

where $M$ is the width of the bus and $N$ is the number of bits that are HIGH in each address. In our implementation, $M$ and $N$ are set to 8 and 3, respectively, so that 56 addresses exist, which suffices for the size of our implementation. Both pre- and post-synaptic buses use this 3/8 bit code. The post-synaptic bus uses one active line in addition to the address to indicate an address has been placed on the bus, while the pre-synaptic bus uses four active lines—one for each of the four synapses an axon can target.

The addresses of the AER buses in the analog axon array are encoded in a format of 4 out of 9 high bits, yielding 126 addresses—one for each neuron. Increasing the bus width would allow more neurons at the cost of additional area for the bus and the decoder. The choice of 4/9 for this bus is a trade-off between performance and the cost of silicon.

## ANALOG IMPLEMENTATION

### *Analog neuron array*

The proposed LIF neuron comprises four identical charge-and-discharge synapses, one for each active line on the pre-synaptic bus. The structure of the synapse was first proposed by Arthur and Boahen (2004). **Figure 5A** shows the schematic of the charge-and-discharge synapse, which will generate a post-synaptic current for every incoming pre-synaptic spike. This synapse comprises a reset circuit (N1-N4), a MOS capacitor ($C_{syn}$, ∼100 fF), a voltage-to-current conversion circuit (P1-P2) and a DC current source ($I_{exp}$, set to 12 pA).

The 3/8 high bits of the pre-synaptic address are connected to N1-N3. On arrival of a pre-synaptic spike with these three bits HIGH and the appropriate active line high, N1-N4 will conduct and pull $V_{syn}$ down to ground. After that, $V_{syn}$ will be pulled up to $V_{dd}$ by $I_{exp}$. The voltage-to-current conversion circuit will transduce $V_{syn}$ into $I_{syn}$, the post-synaptic current, which will decay exponentially, due to the linearly increasing $V_{syn}$. To reduce power consumption, P1, a diode connected pMOS transistor, is added to limit the gate-source voltage of P2. $I_{syn}$ will be injected into the soma for integration. All four synapses of a LIF neuron are identical, using the same 3/8 bit address, but are connected to different active lines.

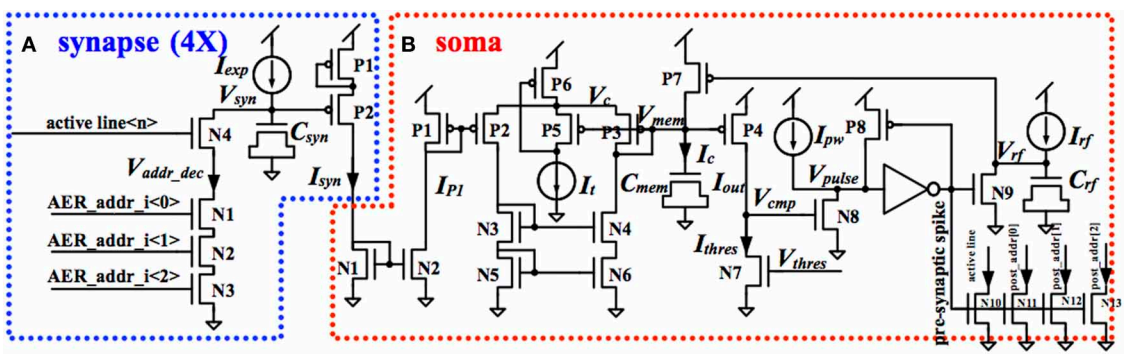



**FIGURE 5 | Circuit diagram of the analog synapse (A) and soma (B).**

**Figure 5B** shows the schematic of the soma. The post-synaptic currents from four synapses are sent to a current mirror (N1-N2) for summing. The current mirror will convey $I_{syn}$, the sum of the post-synaptic currents, to $I_{P1}$, which is the input current of a first-order low-pass filter. Furthermore, by changing the width/length ratio of N1 or N2, the input current to the low pass filter can be easily scaled or amplified.

The low-pass filter, which was first proposed in Python and Enz (2001), is the basic building block of the soma. In our previous work (Wang et al., 2011), we have shown that its output current $I_{out}$ has the following equation:

$$\tau_{mem}\frac{dI_{out}}{dt} + I_{out} = I_{P1} \tag{2}$$

where the time constant of the implementation is given by:

$$\tau_{mem} = \frac{nU_T C_{mem}}{I_t} \tag{3}$$

where $U_T$ is the thermal voltage, $n$ is the weak inversion slope factor, and $I_t$ is a DC current source (set to 1 nA ). More details can be found in Wang et al. (2011).

To generate the post-synaptic spike, the output current of this low-pass filter $I_{out}$ is compared with a constant current $I_{thres}$ introduced by N7. The value of $I_{thres}$ is set by $V_{thres}$ to a value such that three pre-synaptic spikes arriving within 1 ms will make $I_{out}$ strong enough to pull $V_{cmp}$ up to $V_{dd}$. When $V_{cmp}$ exceeds the threshold of N8, N8 will conduct and pull $V_{pulse}$ down to ground. $V_{pulse}$ is sent to an inverter to generate the post-synaptic spike. It is HIGH when $V_{pulse}$ is lower than the threshold of the inverter.

The refractory period is implemented by a circuit composed of N9, P7, a MOS capacitor ($C_{rf}$, ∼100 fF) and a DC current source ($I_{rf}$, set to 12 pA). When the post-synaptic spike is HIGH, N9 will conduct and pull $V_{rf}$ down to ground. After that, $V_{rf}$ will be pulled up to $V_{dd}$ by $I_{rf}$. P7 will conduct and pull $V_{mem}$ up to $V_{dd}$ when $V_{rf}$ is lower than the threshold of P7. The time when $V_{mem}$ is at $V_{dd}$ is the refractory period, during which the low-pass filter will not do any integration. Since this refractory time is active when $V_{rf}$ is lower than the threshold of P7, the refractory time is thus controlled by the size of $C_{rf}$, the capacitor, and $I_{rf}$, the charging current.

When $V_{mem}$ is pulled up to $V_{dd}$ and $I_{out}$ is reset to 0, $V_{cmp}$ will be pulled down to ground by $I_{thres}$. N8 will stop conducting when $V_{cmp}$ is low and $V_{pulse}$ will then be pulled up to $V_{dd}$ by a constant current $I_{pw}$. The post-synaptic spike, which is the inverted signal of $V_{pulse}$, will then be reset. A feedback circuit (P8) will pull $V_{pulse}$ up to $V_{dd}$ quickly once $V_{pulse}$ exceeds the threshold voltage of the inverter, to reduce power consumption. The pulse width of the post-synaptic spike, which is the time when $V_{pulse}$ is lower than the threshold of the inverter, is controlled by $I_{pw}$, which is used to pull $V_{pulse}$ up.

An address encoder (N10-N13, using four minimum-sized nMOS transistors to drive the active line and 3/8-bit address of the AER post-synaptic bus), will convert the voltage-mode post-synaptic spike into a current-mode spike. The current-mode spike will be sent to the AER post-synaptic bus. As the AER post-synaptic bus needs to be driven in parallel by all the analog LIF neurons, an implementation with voltage-mode spikes would need a high fan-in OR gate or an arbiter, which would take up a significant amount of area in the layout. Furthermore, using voltage-mode spikes for on-chip routing will take up significant area as each spike needs one wire, whereas the current-mode spikes can share one bus, e.g., one wire can be shared by the active lines from all the 50 neurons.

As a trade-off between fabrication cost and the size of the neuron array, we chose to implement 50 analog LIF neurons in the analog neuron array, which led to the choice of the 3/8-bit address format. The layout of the analog LIF neuron is as compact as possible and all signals are routed across the neuron. In this way, the placement of the neurons in an array is quite straightforward; the neurons are placed in one row.

All transistors are 2.4 μm wide and 3.6 μm long (P8, N3, N4, and N8 is 0.6 μm long, N1 is 4.5 μm wide and P7 is 4.8 μm wide and 0.6 μm long). The inverter I1 use transistors are 2.4 μm wide and 0.6 μm long. The MOS capacitor values are: $C_{mem} = 15 \times 24\,\mu m$ (∼0.6 pF) and $C_{rfc} = 3.6 \times 2.4\,\mu m$ (∼0.02 pF). In the layout of the neuron array, for each neuron, we just need to connect the three transistors that form the address decoder (N1-N3) in the current synapse (see **Figure 5A**), to three bits in the address of the AER pre-synaptic bus according to the unique 3/8-bit address of that neuron. An active line on the AER pre-synaptic bus is connected to N4 of a current synapse. Each of the four current synapses will have its own active line on the AER pre-synaptic bus. Similarly, for each neuron, we just need to connect the four transistors, which compose the address encoder (N10-N13) in **Figure 5B**, to the active line and to the three high bits in the address on the current-mode AER post-synaptic bus according to the unique 3/8-bit address of that neuron. In this way, the layout of the neuron array will remain compact as no extra routing of the AER buses is needed.

### *Analog axon array*

The structure of the analog axon module is shown in **Figure 3**. It comprises three parts: a ramp generator, four axonal delay paths and an AER interface circuit. The AER interface circuit carries out the function of the address configuration, the address decoding and the address encoding. The ramp generator will start when receiving a spike on the AER bus. The details of the design and implementation of the ramp generator and the delay path can be found in Wang et al. (2013a).

The analog axon array contains 100 identical analog axon modules connected serially. Due to the size of the axon module, we cannot place these 100 axon modules physically in one row (it would be 20 mm long) but instead the array is folded to create a 10×10 2-D array, as shown in **Figure 6**. As in the layout of the neuron module all the AER buses, control signals, and bias currents are routed horizontally across the axon module so that neighboring neurons in a row are simply connected by placing them next to each other. The horizontal buses in each row are connected to two vertical buses placed on both sides of the axon array for interconnection. As for the neuron array, the spikes generated by the axon modules are all current-mode spikes within the

chip and they are converted to voltage-mode spikes for off-chip transmission.

## MIXED-SIGNAL IMPLEMENTATION

### *Multiplexed analog neuron array*

The motivation for developing a multiplexed analog neuron array is to increase the size of the analog neuron array without increasing the cost of the system significantly. A polychronous neural network composed of a neuron array with 50 neurons will suffer from severe cross-talk between patterns, which occurs when a neuron belonging to one pattern fires accidently as a result of pre-synaptic spikes from other patterns or another part of the same pattern. The effect of cross-talk depends on the overlap (correlation) of the patterns and can be regarded as noise. The more overlap there is, the higher the possibility that a pattern plus some noise spikes will also set off a different pattern. Also, the more input connections a neuron has, i.e., the more patterns this neuron is a member of, the more likely this neuron is to get three simultaneous inputs as a result of noise. In severe cases of cross-talk, all neurons in the network will fire continuously in an uncontrolled manner. To mitigate this problem, we need to increase the sparsity of the neural network, i.e., decrease the number of patterns to which each neuron is sensitive. This can be achieved by increasing the size of the neuron array, as the patterns generated by the pattern generator are evenly distributed over the whole network. The conventional approach to increase the size of the analog neuron array is to simply add more physical neurons. As expected, hardware costs increase linearly in relation to the size of the neuron array if all the neurons are to be implemented physically.

Inspired by the multiplexed neuron array used in the digital implementation (Wang et al., 2013b), we propose a similar approach to implement a multiplexed analog neuron array. We can use the fact that in a typical polychronous network, only a small percentage (less than 5%) of the neurons are active at any given time, and only those active neurons need to be physically implemented.

The structure of the multiplexed analog neuron array is shown in **Figure 7**. It consists of two sub-blocks: a physical neuron array and a controller. They communicate with each other via two internal AER buses: the AER physical pre-synaptic bus and the AER physical post-synaptic bus. The controller receives pre-synaptic spikes from the axon array and assigns them to the physical neurons for the generation of post-synaptic spikes, which will be sent to the axon array. From the point of view of the axon array, the multiplexed neuron array appears as a neuron array with 4k neurons. The addresses of the spikes between the controller (a single minimum-width binary address) and the analog neuron array (the 3/8-bit address format) need to be remapped by the router, which will also synchronize the spikes from the analog circuits. For simplicity, in the following description, we assume the controller is connected to the analog neuron array without synchronization and address remapping.

The controller dynamically assigns analog neurons to each incoming pre-synaptic spike. The analog neurons are used to detect how many pre-synaptic spikes have arrived within 1 ms of each other. When a spike arrives from the axon array and an analog neuron has already been assigned for that spike's address, the spike will be sent to that neuron. The address of this incoming spike will have been latched in a register linked to that analog neuron. If no neuron has been assigned for the arriving address, the spike will be sent to an unassigned neuron, which will then be labeled as assigned by the controller, by latching the address of the spike. The controller will also start a timer linked to that analog neuron. Once the timer of that neuron has expired (after 1 ms), the neuron will be freed and labeled as unassigned by the controller. When a post-synaptic spike is generated by an analog neuron, the controller will send it to the axon array with

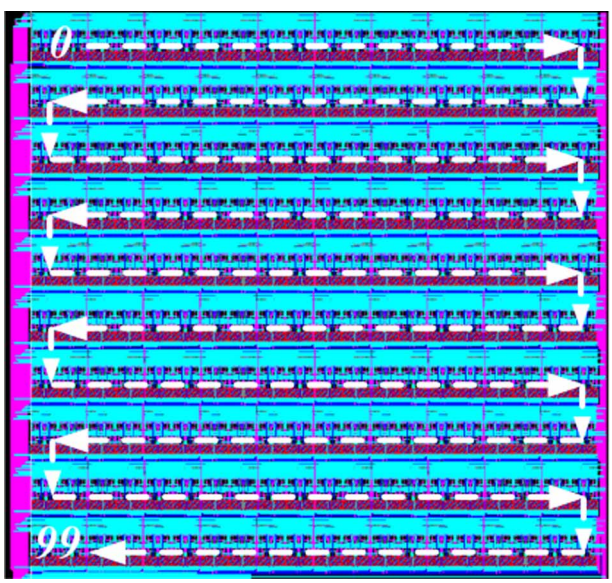



**FIGURE 6 | Layout of the axon array.** Arrows show how the axons modules are placed in a 1-D array.

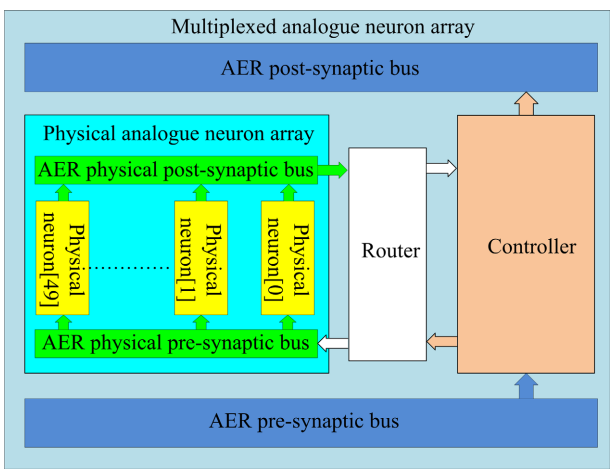



**FIGURE 7 | Structure of the multiplexed analog neuron array.** The controller and router map virtual addresses from the AER busses to physical addresses on the analog neuron array, so that only active neurons in the network are using hardware resources.

the address that is stored in its register. More details about the controller can be found in Wang et al. (2013b).

Based on this structure, a neuron array with 4k virtual analog neurons can be achieved using only 50 physical neurons. This multiplexed analog neuron array is thus 80 times more efficient in silicon area on the analog side. It does, however, require a controller implemented on an FPGA. This does not increase the cost of the system significantly as the FPGA is needed anyway to carry out other tasks such as pattern generation, address remapping and other miscellaneous tasks. Furthermore, this mixed-signal implementation offers a much higher degree of extensibility as the LIF neurons used in this implementation could easily be replaced with other neuron models if desired.

### *Synchronization interface circuit*

To use the asynchronous analog circuits with the FPGA, synchronization with its clock domain is needed. In digital circuit design, a general method used to do this is to use two (or more) serially connected flip-flops to sample the input (Weste and Harris, 2005). This scheme works well for 1-bit signals but it does not extend to catering for parallel signals, such as the address bus and data bus, due to potential timing skew on these buses that could cause each bit in the bus to arrive at a slightly different time. This can lead to race conditions and hazard problems, and can ultimately lead to the wrong address being sampled (Weste and Harris, 2005).

In our design, this timing skew comes from two sources. The first is the analog circuit that converts the current-mode spikes to voltage-mode spikes. Due to process variation and parasitic capacitors between the wires and transistors, the conversion for each line of the bus will take a slightly different amount of time. For the very same reasons, the pulse width of each active line and each bit in the address will also be slightly different. The second source of timing skew is caused by the propagation delay of the signals along the tracks of the Printed Circuit Board on their way to the FPGA.

**Figure 8** illustrates a waveform of a post-synaptic spike from an analog LIF neuron (the waveform from the analog axon is quite similar). In the figure, the timing skew can clearly be seen as each bit in the bus arrives at a slightly different time. Besides the timing skew, there is also an additional problem in the form of glitches, which are brief digital pulses, up to tens of nanoseconds long. They are caused by the coupling capacitance between the wires and transistors. These glitches, in spite of their short period, are still likely to be sampled by the digital circuit (running at 50 MHz) and ultimately may lead to the wrong addresses being sampled.

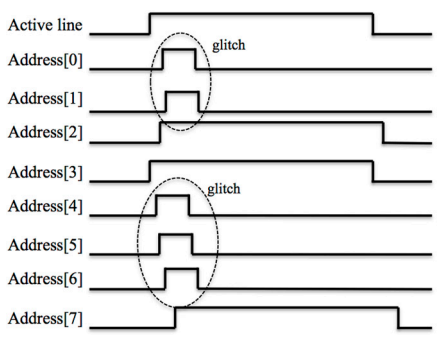



**FIGURE 8 | Waveform of a spike from an analog neuron on the post-synaptic AER bus showing timing skew and glitches.**

One common method to minimize the timing skew caused by transistor mismatch is to use clocked flip-flops (Weste and Harris, 2005) to generate these spikes. We have not used this method because it would increase the design overhead of circuit and introduce another problem, namely that of synchronizing the clock signal of the chip and the FPGA. The timing skew caused by propagation delays on the PCB is usually minimized by carefully tuning the length of the tracks on the PCB. We have not used that method either as it would significantly increase the effort and cost of manufacturing the PCB.

In digital designs, the general way to sample an asynchronous parallel bus is to use a handshake protocol to guarantee that the receiver will only sample the data when the data is stable (Weste and Harris, 2005). In other words, the sender needs to inform the receiver when to sample the data. The drawback of this method is that it requires extra logic circuits on both the sender and the receiver. In cases where there is more than one sender on the bus trying to send data, some form of arbitration is required, further increasing the circuit complexity and the cost of hardware resources.

Instead of the above methods, we chose to synchronize the spikes from the analog implementations by using an interface circuit to carry out the synchronization in three steps without requiring a handshake protocol. For illustration, we will use the AER bus of the analog neuron array in the following explanation. The interface circuit handles the AER bus of the analog axon array in the same way.

The first step is to synchronize each active line and each bit of the address of the incoming spike in the conventional manner by using a circuit composed of a serial connected flip-flop for each of them (four in total). The output values of the flip-flops for the address and active lines are referred to as the synchronized address and the synchronized active line, respectively. The address of the post-synaptic spike is encoded in the 3/8-bit format, which means that any address that does not have exactly three out of eight bits active is invalid.

The second step is then to latch the synchronized address and active line only when a valid address is present, i.e., when exactly three bits are HIGH, and store it in a register. We have implemented this register as a $32\times9$ bit FIFO, using eight bits for the address and one bit for the active line. We use a counter to determine how many bits are HIGH in the synchronized address and we can distinguish two situations that need an action when a valid address is detected:

1. The arrival of a spike with a valid address when the address at the previous clock cycle was invalid. In this condition, the value of the counter in current clock cycle is three, whilst the value of the counter at previous clock cycle was not equal to three. The address of the spike is latched in the FIFO
2. The arrival of a spike with a valid address that is different from a valid address at the previous clock cycle. In this case, the value of the counter in the current clock cycle and previous

clock cycle are both equal to three, whereas the value of the synchronized address in current clock cycle is not equal to the value at the previous clock cycle. The new address is stored in the FIFO.

In all other cases, including when a valid address is detected that is the same as in the previous clock cycle, the data on the bus is ignored. In this way, the asynchronous spikes from the analog neuron array are synchronized and stored in the FIFO. The third step is to generate spikes with a fixed pulse width (four clock cycles) by reading the FIFO. If the FIFO is empty, all the synchronized pre-synaptic spikes have been read out and no new spikes will be generated.

The interface circuit for the spikes from the analog axon array operates in the same way with the exception that a third condition needs to be handled:

3. The arrival of a spike with a valid address that is the same as the last one that arrived, but on a different synapse. In this case, the value of the counter in current clock cycle and previous clock cycle are both four (4/9-bit format) and the value of the synchronized address in both cycles is the same, but the value of the synchronized active lines is different. The new address and active line are stored in a 32×13 bit FIFO (nine bits for the address and four bits for the active lines).

The interface circuit effectively eliminate the problems of timing skew and glitches on the bus. It is also capable of sampling the asynchronous spikes from the analog circuits with a high temporal accuracy, as shown by the results that will be presented in section Performance of the Interface Circuit. For spikes that need to be sent to the analog chip, we use the conventional means of synchronizing them to the system clock by using flip-flops on the FPGA to minimize the timing skew on the address lines (Weste and Harris, 2005).

#### *Address remapping*

Address remapping is the second function of the router. The controller can be configured for multiplexed analog neuron arrays or multiplexed digital neuron arrays. When it is configured for a multiplexed analog neuron array, the router needs to carry out the remapping for the addresses of spikes traveling between the controller and the analog neuron array. To use the analog axon array, the router needs to carry out the address remapping for the spikes traveling between the analog axon array and the controller regardless of whether it is configured for multiplexed analog or digital neuron array.

The router was implemented using four look-up tables, one for each of the four address remapping possibilities. For spikes from the analog axon/neuron array, the router synchronizes them using the interface circuit first. These synchronized spikes are then compared to the look-up tables in order to convert their addresses to the corresponding binary-encoded addresses. These spikes are then sent to the controller for processing. Spikes generated by the controller are also compared against the look-up tables to convert their addresses to either 3/8-bit or 4/9-bit addresses. After being converted, these spikes are sent to the analog axon/neuron array.

## RESULTS

The proposed polychronous neural network is designed to train and recall patterns rather than to randomly react to some spatio-temporal patterns (groups) that have emerged in the network, as is the case in Izhikevich (2006). Performance in our network is therefore measured as the rate of success in recalling the trained patterns. The advantage of our approach is that the network can be used as a memory that can learn spatio-temporal patterns. Furthermore this approach optimizes the use of the available hardware, so that in our approach all available neurons and axons in the hardware arrays can be used, while in the original polychronous network some neurons and many connections are not part of any pattern and thus never used. The disadvantage of our approach is that overlap between patterns (cross-talk) has to be limited and it is not possible to store near identical patterns.

There are four possible combinations of analog or digital axons and neurons. The fully digital (FPGA) combination implements the proposed neural network faithfully with hardly any effect of noise and process variations. The measurements form this combination therefore present the optimal performance of our polychronous neural network model. The results of all the other three combinations will be compared with the results of the fully digital implementation in the sections Digital Axon Array and Analog Neuron Array to Analog Axon Array and Analog Neuron Array. Section Performance of the Interface Circuit first discusses the performance of the interface circuit described in section Synchronization Interface Circuit.

### PERFORMANCE OF THE INTERFACE CIRCUIT

Testing the interface circuit is the first step in testing the whole system. To obtain a direct measurement of the ability of the interface circuit to synchronize and latch addresses correctly, we use the FPGA to send a pre-synaptic spike to an analog neuron to induce it to fire. The interface circuit is then used to synchronize and latch the spike from the analog neuron with the FPGA's clock. We then compare the address of this latched post-synaptic spike with the expected address, as determined by which neuron the FPGA induced to fire. If their addresses match, this means the interface circuit works correctly.

Sometimes the interface circuit samples the same address twice. This is caused by glitches that can cause a valid address to become briefly invalid, when more than three address lines are high, before returning to the valid address as the glitches subside. This double sampling could be solved by adding an internal timer to the interface circuit to guarantee that an address could only be sampled once within a short period (say 1 $\mu$s). However, we have not employed this method as the second spike sampled will only cause a small offset ($<1\,\mu$s) in the axonal delay, which starts on the arrival of a post-synaptic spike. This offset will not affect the performance of the proposed polychronous neural network at all.

**Figure 9** shows the results of the tests. All 50 addresses (one for each analog neuron) were tested 128 times (with an interval time of 5 ms to guarantee there will be one post-synaptic spike each time). This test was then repeated 10 times. In each of the 10 runs, for approximately 75% of the time the correct address was sampled once while for the remainder of the cases, the correct

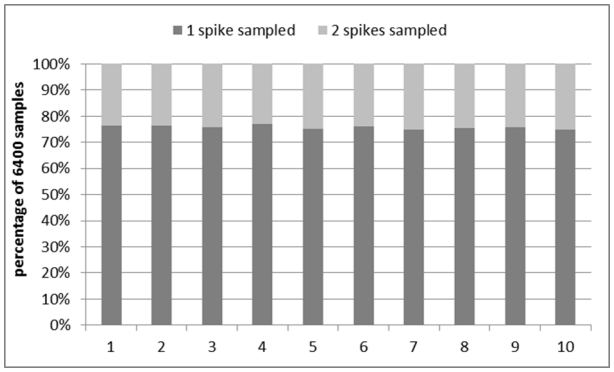



**FIGURE 9 | Performance of the interface circuit.** Dark gray: valid address sampled once; Light gray: valid address sampled twice in succession.

address was sampled twice in succession. No wrong addresses were sampled in these tests.

## DIGITAL AXON ARRAY AND ANALOG NEURON ARRAY

### *Delay programming*

In the setup for the delay programming tests, a single axon array was used in the neural network, yielding 4k axon modules with 16k (16384) axonal delay paths (connections). Note that unlike in Izhikevich (2006), no connections are shared between two patterns, so that the number of available connections directly determines the maximum number of inter-spike intervals that can be programmed into our network. Each axon module contains four axonal delay paths (see **Figure 3**), and for each spike in the polychronous pattern, 4 delay paths are needed from the four previous spikes in the pattern. Thus, the number of the inter-spike intervals that our neural network can store is simply equal to the number of axon modules. If, for instance, the patterns to be stored each contain 50 inter-spike intervals, the maximum number of such patterns that can be stored in the neural network is 82 (4k/51).

The patterns are trained only once when using delay programming. There is also only one recall test as there is no adaptation, and the result of a recall will be the same each time. For each configuration of the neural network, 10 test runs were conducted. The pattern generator & checker module generates spatio-temporal patterns for training and for testing whether the patterns can be recalled successfully. We tested neuron array sizes ranging from 128 to 4k neurons and test results are shown in **Figure 10A**. For the configurations consisting of 128 and 256 neurons (not shown in **Figure 10A**) and trained with 82 patterns having 51 spikes each, the neural network enters an all firing state in which all the neurons fire simultaneously, showing that a network of this size using analog neurons cannot cope with that number of patterns. In the digital implementation, this only happens for configurations consisting of 128 neurons, while a network with 256 neurons achieves an average success rate about 80%. To achieve a similar success rate when using analog LIF neurons, the network needs at least 512 neurons. Furthermore, the results for a network with 1k and 2k analog neurons are also slightly worse than their digital counterparts. Only the result for

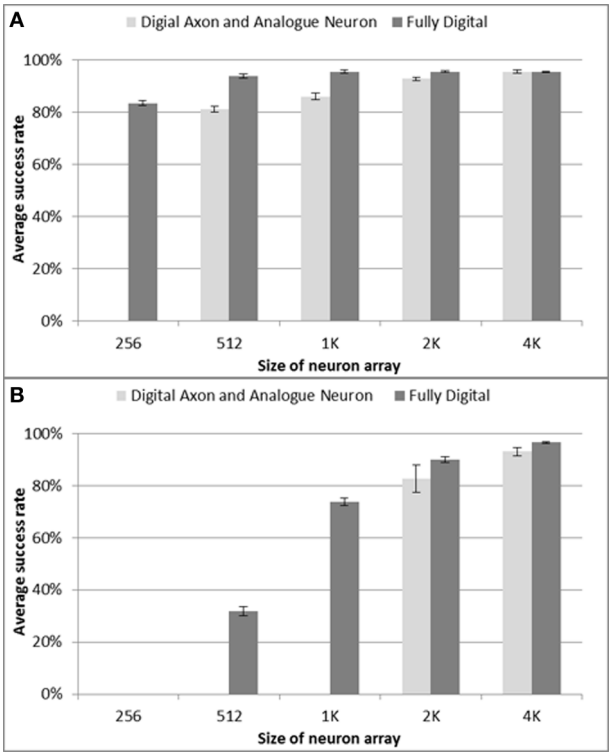



**FIGURE 10 | Percentage of stored patterns successfully recalled for different neuron array sizes. (A)** delay programming and **(B)** delay adaptation, respectively. The results for the fully digital implementation are added for comparison purpose. Error bars are standard errors of the mean.

4k analog neurons matches the digital implementation. As an aside, this proves that the proposed interface circuit is capable of sampling the asynchronous spikes from the analog circuits correctly, because otherwise the performance would be much worse than in the digital implementation.

The results indicate that the effects of cross-talk are more serious when using the multiplexed analog neuron array, so that a network with analog neurons performs worse than one with digital neurons when the size of the network is small. Due to process variation and device mismatch, the analog neurons cannot be perfectly tuned to all generate a post-synaptic spike only when at least 3 out of 4 pre-synaptic spikes arrive within 1 ms. In other words, the analog neuron is not as precise a coincidence detector as the digital neuron. Moreover, due to the parasitic capacitances on chip, the analog LIF neuron will sometimes generate spikes by accident, e.g., the firing of one neuron will trigger its neighboring neuron to fire, which increases cross-talk. Increasing the size of the network increases the sparsity (i.e., decreases the number of patterns to which a neuron belongs Wang et al., 2013b), and the difference in the performance between the analog neurons and the digital neurons will become negligible for larger networks.

### *Delay adaptation*

In the tests for the delay-adaptation mode, each pattern was trained five times and recalled one time. The strategy used

adapted the delay by half the time difference between the pre- and post-synaptic spikes each time a neuron fired. The same settings used in the delay programming scenario were used for these tests, but all delays were initialized with random values. We again tested neuron array sizes from 128 to 4k neurons and the test results are shown in **Figure 10B**. For the networks with a size smaller than 2k neurons, only a few patterns can be recalled successfully and their results are therefore not included in **Figure 10B**. The results in **Figure 10** also show the performance drops more in delay adaptation mode than in the delay programming mode when compared with the digital implementation. This is again the result of the larger sensitivity to cross-talk in the analog neuron array.

#### *Effect of noise*

In this set of tests, random noise was injected into the network. The Poisson rate of the noise, generated by a LFSR, was varied from 2 to 128 spikes per second. This firing rate represents the number of additional spikes, i.e., not belonging to any of the trained patterns, presented to the network in a one second window. As each spike is generated by a randomly chosen neuron, the spike rate measures the total noise input, not the firing rate of individual neurons.

All other settings were kept the same as in the delay-programming mode and the delay-adaptation mode with a neuron array consisting of 4k neurons. In both modes, no noise was added during the first training time. **Figure 11** shows the result, which proves that the system is fairly robust to noise when the sparsity of the neural network is large.

#### *Capacity for storing spatio-temporal patterns*

To test the capacity for storing spatio-temporal patterns when using the multiplexed analog neuron array, it was configured with 4k neurons and 80k axon modules. Delay programming and delay adaptation were both used with a pattern length of 51 spikes. For a pattern length of 51 spikes, we tested storing and recalling 1000 and 1200 patterns. Ten test runs were conducted. The system works well for the 1000 pattern case. **Figure 12** shows the results for 1000 patterns and the successful recall rate is about 95% on average which is quite close to the result of the fully digital implementation (Wang et al., 2013b). With 1200 patterns the recall no longer works as the effect of cross-talk becomes too severe, indicating that once cross-talk reaches a critical level, it quickly becomes catastrophic. Two reasons caused this performance drop. The first reason is that the mixed-signal system suffers more noise compared to the fully digital implementation, the successful rate of which is 95% for 1200 patterns. The second reason is that the theoretical maximum firing rate of the pre-synaptic spikes that the multiplexed analog neuron array can handle is only $50/128 \approx 40\%$ of the maximum firing rate that the digital one can handle, as the number of the physical neurons is only 50, whereas the digital implementation has 128 physical neurons.

### ANALOG AXON ARRAY AND DIGITAL NEURON ARRAY

Unlike the results presented in section Digital axon array and Analog Neuron Array, the testing scenarios for the combination of analog axon array and digital neuron array will focus on the percentage of spikes in a pattern that have been recalled successfully. This is because the capacity of the analog axon array is much smaller than that of the digital axon array, which means that only a few patterns can be stored in this network, so that the percentage of patterns recalled is a much less accurate measure of performance. Furthermore, the dynamics caused by process variation and device mismatch causes variations in the number of spikes that are correctly recalled in each pattern.

For this test, we only had access to one analog axon array with 100 analog axon modules, each with 4 axonal delay paths. The maximum accessible address of the 4/9-bit bus on the analog axon array is 126, which means the maximum size of the digital neuron array that can be used is 126 neurons. As the experimental results in Wang et al. (2013b) show, a neural network consisting of only 126 neurons will be affected seriously by cross-talk. To measure the performance of the analog axon array without the effect of this cross-talk, we used specially generated random patterns with no overlap (correlation) for testing.

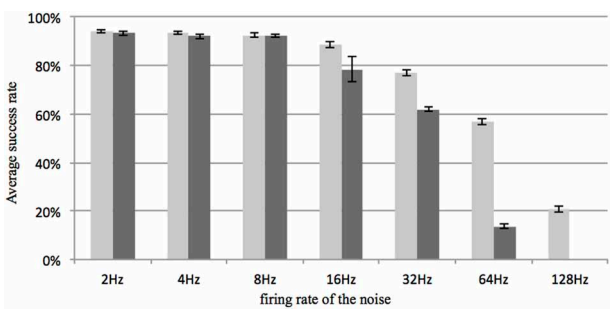



**FIGURE 11 | Recall percentage for various Poisson rates of the noise generator.** The firing rate represents the total number of additional random spikes per second in the network. For comparison, the firing rate of a stored pattern is about 100 spikes per second (50 events in about 500 ms). Light gray: delay programming; Dark gray: delay adaptation. Error bars are standard errors of the mean.

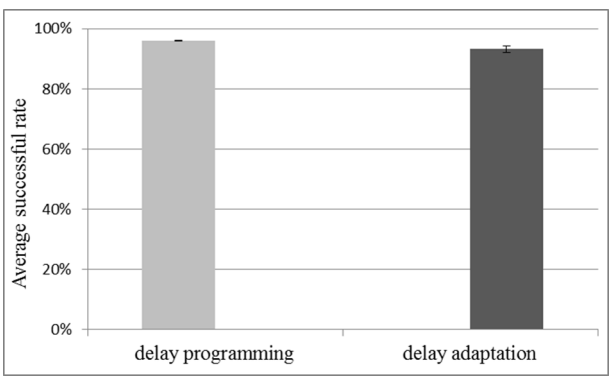



**FIGURE 12 | Result for capacity testing with 1000 stored patterns of 51 spikes each.** The network consists of 4k neurons and 80k axon modules. Both methods of delay configuration resulted in approximately 95% of the stored patterns being successfully recalled. Error bars are standard errors of the mean.

Delay programming and delay adaptation were both used with pattern lengths of 20, 25, 33, and 50 spikes. The patterns were trained with a single presentation in the delay programming mode and for 20 presentations in the delay adaptation mode. As there are 400 axons in the analog axon array, for the pattern length of 20, 25, 33, 50 spikes, the maximum number of such patterns that can be stored in the neural network is five, four, three, and two, respectively. For each pattern length, 127 test runs were conducted.

**Figure 13** shows, for each pattern stored in the neural network, what percentage of spikes were recalled correctly. As discussed in section Analog Axon Array, the delay of the analog axon is programmed in an initial phase followed by a number of iterations of delay adaptation with a fixed delay update step. This is to reduce the errors in delay that result from the initial delay programming step. **Figure 13** shows that after 20 iterations of delay adaptation, the percentage of the spikes in the patterns that have been correctly recalled has been slightly increased for the patterns with 50 spikes. For the other pattern lengths, the improvement is negligible. The average percentage of spikes in each pattern correctly recorded across four pattern lengths (over 127 test runs) using delay programming is 86.2% and using delay adaptation is 87%.

Compared to the test results presented in Wang et al. (2013b), which uses the fully digital implementations, the combination of analog axon array and digital neuron array has an 8% drop in performance, which is mainly because the analog axon cannot be as precisely programmed and tuned as the digital axon. As the experimental results of one axon module presented in (Wang et al., 2013a) show, the offset between the actual programmed and the desired value is about 10%, after delay programming. When the ramp generator's voltage is latched by the analog memory (for delay programming), there is always a slight deviation (∼10 mV) between the programmed voltage and the desired voltage, as a combined result of charge injection (Liu et al., 2002) and the inaccuracy of the ramp generator itself. The ramp generator will not charge at exactly the same speed each time due to noise in the charging current. The analog axon will therefore propagate each incoming pre-synaptic event with an offset compared to the desired axonal delay. After delay adaptation, this error can be reduced to less than 300 μs throughout the working range of a single axonal delay path (Wang et al., 2013a), but due to process variation and device mismatch, it is impossible to tune all axonal delay paths with such accuracy. This offset, when large enough, will destroy the time-locked relations that are the basis of polychronous spiking neural networks. We will discuss possible solutions for this issue in section Analog vs. Digital Implementations. Another factor in the drop in performance is the fact that the analog axon will sometimes generate spikes due to on-chip parasitic coupling between axons, so that the firing of one axonal delay path can trigger its neighboring paths to fire by accident.

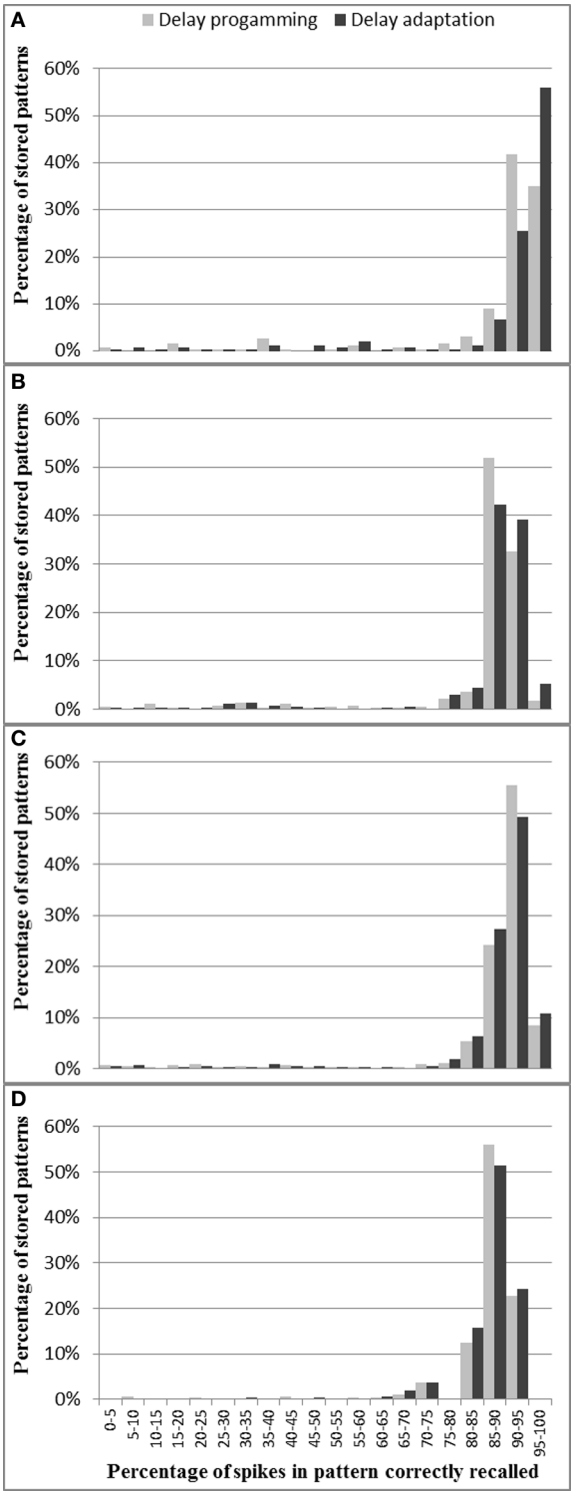



**FIGURE 13 | Percentage of spikes in pattern correctly recalled for different pattern lengths: (A) 50 spikes, (B) 33 spikes, (C) 25 spikes, and (D) 20 spikes.** These results are from the combination of analog axons and digital neurons. For most patterns across all four pattern lengths, more than 85% of spikes are recalled successfully.

## ANALOG AXON ARRAY AND ANALOG NEURON ARRAY

In this section, we will present the experimental results of the combination with an analog axon array and an analog neuron

array. For the same reasons as presented in the previous section, the testing scenarios will also focus on the percentage of spikes in a pattern that have been recalled successfully, and the setup for testing is the same as described in the previous section.

**Figure 14** shows for each pattern stored in the analog axon array how many spikes were recalled correctly. **Figure 14** shows that more than 70% of the spikes are correctly recalled for nearly all the patterns across three pattern lengths (20, 25, and 33 spikes) in both delay programming mode and delay adaptation mode. For the longest patterns (50 spikes) the probability of correctly recalling the full pattern is significantly lower, with only 57.4% of the spikes successfully recalled on average, as mismatch and noise are more likely to destroy the time-locked relations, resulting in the final part of the pattern not being recalled. **Figure 14** also shows that for these longest patterns, 20 iterations of delay adaptation improve the percentage of the spikes in the patterns that have been correctly recalled to 64.7%. The average percentages of spikes in pattern correctly recorded across four pattern lengths (20, 25, 33, and 50 spikes) using delay programming are 77.2, 78, 72.8, and 57.4%, respectively. After 20 iterations of delay adaptation, these numbers have been improved to 78.1, 78.6, 73.9, 64.7%, respectively. Compared to the results presented in section Analog Axon Array and Digital Neuron array for the analog axon array and digital neuron array, the fully analog combination has an overall Performance drop of about 14%. Compared to the test results presented in section Digital Axon Array and Analog Neuron Array for the digital axon array and analog neuron array, the performance drop increases to about 20%.

These drops are the results of two major factors. The first one is that the analog axon and neuron arrays both generate spurious spikes due to on-chip parasitic coupling. The second factor is that the analog axon fails to perfectly produce the time-locked relations as the digital axon does. Both factors play a larger role the longer the pattern is (in terms of number of spikes). Together, these effects causes the combination of the analog axon and analog neuron array to have the lowest performance of the four combinations.

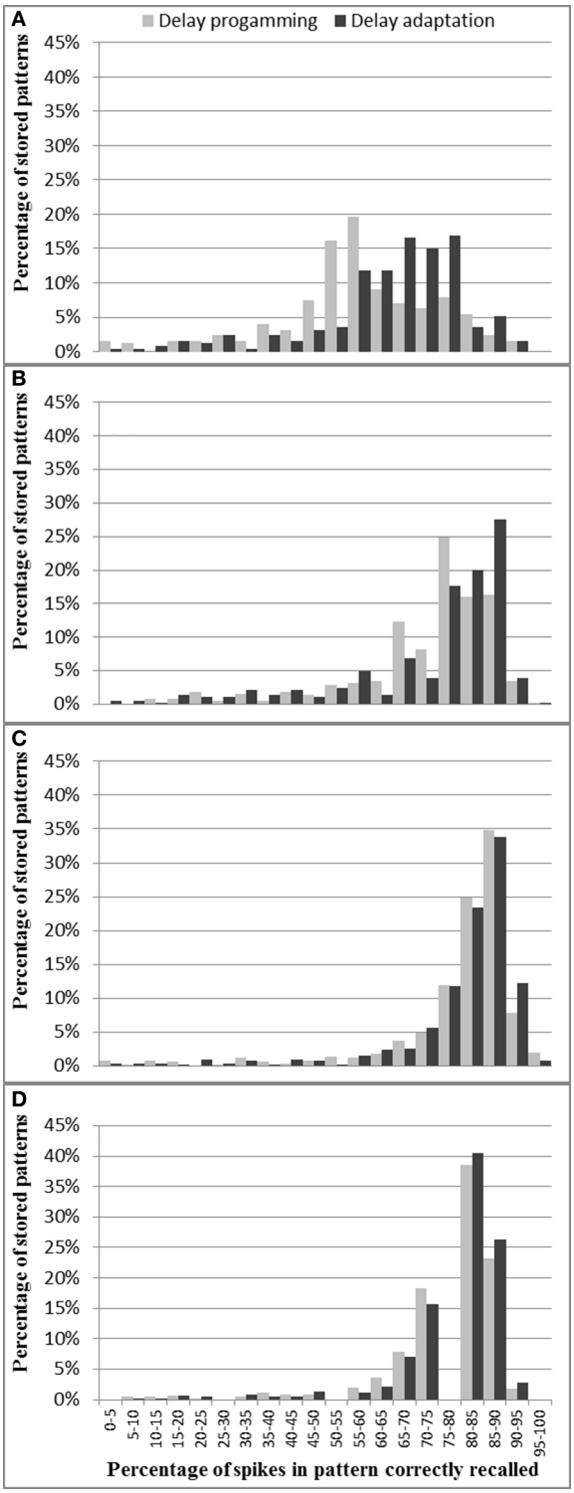



**FIGURE 14 | Percentage of spikes in each pattern correctly recalled for different pattern lengths: (A) 50 spikes, (B) 33 spikes, (C) 25 spikes, and (D) 20 spikes.** These results are from the full analog system.

## DISCUSSION

### PERFORMANCE COMPARISON

#### *Efficiency of the implementation*

In Izhikevich (2006), the polychronous network is created with random delays, and STDP is used to prune the connections. Patterns are not stored or programmed into the network, but rather, random patterns emerge. A single connection between neurons could be active in a number of patterns, while other connections will become totally inactive. In our implementation, patterns can be directly programmed into the network and all connections are used when the maximum number of patterns has been programmed into the network. We aimed to avoid inactive connections, since hardware would still be dedicated to these inactive connections, but never used.

A drawback of a polychronous neural network is that a common sequence of four spikes in multiple patterns would initiate all patterns that have this sequence when it occurred. To distinguish between two patterns with identical sub-sequences, it will be necessary to set up the network so that continuous input is needed from the input pattern to keep the pattern going, for example by setting the threshold to 5 simultaneous input spikes (4 from the previous neurons in the pattern and 1 from the input). Such a system would then only follow a pattern if it had been

previously learned, and if it corresponded with the input pattern. One of the two potential patterns (with identical starts) would die out once the input signal identified which of the two patterns is being presented.

The probability of overlap between patterns can be reduced by setting a higher threshold at each neuron and connecting it to more of the previous neurons in the pattern. The number of patterns a network can store decreases linearly with the number of neurons each neuron is connected to, so this would come at the cost of a decreased storage capacity.

#### *Analog vs. digital implementations*

The experimental results show that, on average, the fully digital implementation has the best performance. For comparison, the combination of the digital axon array and the analog neuron array achieves a similar performance when the network is sparse. The combination of the analog axon array and digital neuron array has a considerable performance drop, even when care has been taken to remove all cross-talk from the spatio-temporal patterns. Finally, the combination of the analog axon and neuron array has the worst performance out of the four combinations. The fully digital implementation has the strongest time-locked relation, whereas the fully analog implementation has the weakest, due to the offset between the actual programmed and the desired delay during programming; and the analog implementation is further hampered by noise and spurious spikes. As a result, we may conclude that the most important requirement of a hardware implementation of a polychronous network is to provide a strong time-locked relation.

For the analog axon, as presented in section Analogue Axon Array and Digital Neuron Array, the error is introduced when the ramp generator is writing its output voltage to the analog memory (for delay programming) as a combined result of the charge injection and the inaccuracy of the ramp generator. As the results presented in Wang et al. (2013a) show, the offset will still be about 300 $\mu$s even after adaptation. One possible solution is to use analog-to-digital conversions and then store these digital values in digital memories (Horio et al., 1990; Cauwenberghs, 1996). This method has a major advantage in that data can be stored in non-volatile digital memory. The drawback is also quite obvious. It requires at least one analog-to-digital converter (ADC) for storage and usually one digital-to-analog converter (DAC) for read out. This problem will become critical when massive storage is required as each analog cell will either have its own ADC or share one ADC, which will increase the complexity of the circuit. Other factors, such as the accuracy and the bandwidth of the converters, will lead to the requirement for a high precision ADC. The second possible solution is to use floating-gate devices, which employ programmable elements that that could be used to store the analog values in a non-volatile memory (Basu et al., 2010; Brink et al., 2013; Hasler and Marr, 2013). This feature is a promising alternative for the implementation of our polychronous spiking neural network. On the other hand, the time-multiplexed digital axon achieves an excellent balance between hardware cost and performance and therefore is the preferred choice when using FPGAs. As for a custom design, this design choice needs to be carefully investigated because the cost will be highly process dependent.

While it is common cause in neuromorphic engineering that analog circuits provide superior simulation of biological neurons as a result of their continuous and noisy representation of signals, these results show that in this application the analog implementation is consistently poorer in performance and scalability than the digital implementation, which emphasizes that practitioners should recognize that the use of analog circuits comes at a significant cost and should not necessarily be an automatic choice in all applications.

#### *Comparison with other solutions*

For the analog implementation of the axonal delay, a similar approach was implemented by charging a capacitor using a transistor operating in sub-threshold (Dowrick et al., 2013), so that the duration of the delay can be programmed by adjusting the gate voltage of the charging transistor. However, their implementation is not able to learn delays, as the value of the gate voltage was assigned externally and the authors have not addressed the issues of obtaining and maintaining this voltage. In contrast, our circuit is capable of learning and storing the axonal delay between two spikes. In (Sheik et al., 2012, 2013), the authors show how slow dynamics of analog synapses, combined with the variability of neuromorphic analog circuits, can be used to generate a range of temporal delays. Again, this work is used to generate the desired delay rather than learn the delay.

For the digital implementation of the (axonal) delay, another approach is to use a look-up table for the axonal delay values and use a delay sorter directly before the neurons (Scholze et al., 2011). The delay sorter records the arrival time of a spike and will re-emit the spike when the axonal delay time found in the look-up-table is reached. Our polychronous network generates delay paths *de novo*, so that only connections that actually appear in the training patterns will be created. Each axon module of our polychronous network not only propagates the post-synaptic spike with a programmable axonal delay but also transmits the pre-synaptic spike to the destination neuron (using address remapping by configuring the input and output addresses). An implementation with a look-up table would need the axon module to store the address of the desired axonal delay from the look-up-table, and would need to receive the notification from the look-up-table when that axonal delay is reached. Address-remapping would then have to be carried out by the axon module through the configuration of its input and output addresses. An implementation using look-up tables would therefore be more complex and larger than our proposed implementation.

## SCALING

The performance of the proposed polychronous network (the number of storable patterns) will scale linearly with the number of axons as long as the average number of connections per neuron is kept below 1/4 of the number of neurons in the network to ensure that cross-talk is not much of an issue (Wang et al., 2013b). In other words, the number of neurons needs to be increased proportionally to the number of axons to maintain performance.

The fully digital implementation of the polychronous neural network is a scalable design. The number of time-multiplexed

axons implemented by one physical axon will increase linearly with the amount of available on-chip SRAM, as long as the multiplexing rate keeps the time resolution of the system within the biological time scale, which is generally less than 1 ms. The number of physical axons (i.e., the ones that could be activated simultaneously) will increase linearly with the number of available Slice LUTs, which is indeed the bottleneck for large-scale FPGA designs. The total number of virtual axons therefore scales linearly with the quantity of both the available on-chip SRAM and Slice LUTs. The number of physical neurons also scales directly with the number of the available Slice LUTs. Finally, the timing requirement will become quite critical when the utilization becomes high, e.g., 90% of the LUTs on an FPGA, due to the difficulties in routing. A good balance between the number of the physical axons, the multiplexing rate and the number of physical neurons is therefore the key to the implementation of a large-scale polychronous network with a good time resolution and a high utilization of the available hardware recourses on FPGA.

The analog implementation is nowhere near as scalable as the digital implementation, since it can only be scaled up by implementing more physical copies of the neurons and axons. However, the introduction of the multiplexed analog neuron array, making use of the fact that only a few neurons are active at any given time in a polychronous network, allows the number of virtual neurons to be about 80 times larger than the number of physical neurons. In systems that need slow dynamics or memory of past events, i.e., using neurons with longer time constants than we have used here, the multiplex rate would go down and we would need more physical neurons.

### LESSONS LEARNED

Some lessons have been learnt from the implementation of this mixed-signal platform and these are discussed below.

Virtualization, i.e., the mapping of a larger address space onto a smaller number of physical components through multiplexing these components, is one of the key ideas for implementing large-scale spiking neural networks, because physical components are costly. Virtualization, when simulating neural networks, is supported by biological observations that only 1% of neurons in our brains are active on average at any moment (Johansson and Lansner, 2007), which means it is not necessary to implement all neurons physically on silicon.

A mixed-signal system appears to be a powerful tool for real-time emulation of large-scale neural networks as it can use analog circuits for computation while keeping the flexibility of using programmable devices such as FPGA. As the on-chip topology of the analog circuits is generally fixed after fabrication, it is better to implement the whole system in an FPGA for prototyping and optimization before fabricating the analog circuits.

For the sake of multiplexing analog building blocks such as neurons and axons in a neuromorphic system, these circuits must be designed as standardized building blocks with a standard protocol for communication (such as AER) with programmable devices. Furthermore, for the maximum utilization of a fixed sized analog chip, it is best to reduce the on-chip routing as much as possible as the routing can be carried out off-chip by FPGAs with more flexibility and extensibility.

Our polychronous network stores spatiotemporal patterns. A certain amount of jitter can be tolerated in the initial spikes when recalling a stored pattern, which is controlled by setting a time window for coincidence detection in the FPGA implementation, and by the neuronal time constant in the analog implementation. If the patterns are to be generated by a neuromorphic sensor, then care needs to be taken that the sensor reliably produces (near) identical spatiotemporal patterns for identical input signals.

## CONCLUSIONS

We have presented a mixed-signal implementation of a polychronous spiking neural network composed of both an analog implementation and a digital implementation of the axon array and the neuron array. A multiplexed analog neuron array with 4k analog neurons was achieved by multiplexing 50 physical analog neurons. Compared to conventional time-multiplexing systems that operate serially and have to store and retrieve analog variables, our scheme operates in parallel, and does not require analog storage. A novel interface circuit for synchronizing the spikes from the analog circuits has also been presented. The proposed interface circuit effectively eliminates the problems of timing skew and glitches on the bus and is capable of sampling the asynchronous spikes from the analog circuits correctly. The test results using the four possible configurations of analog or digital components have been compared and discussed. We compared our mixed-signal implementation with our fully digital implementation and addressed the key factor that most influences the performance of the neural network—that of generating accurate time locked relations. The proposed implementation can be linearly scaled up with the quantity of available hardware resources, although the digital implementations are significantly easier to scale than the analog equivalents, owing to the generic FPGA platforms used.

## ACKNOWLEDGMENTS

This work has been supported by the Australian Research Council Grant DP0881219.

## REFERENCES

Arthur, J. V., and Boahen, K. (2004). "Recurrently connected silicon neurons with active dendrites for one-shot learning," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541) (IEEE)* (Vancouver, BC), 1699–1704. doi: 10.1109/IJCNN.2004.1380858

Basu, A., Ramakrishnan, S., Petre, C., Koziol, S., Brink, S., and Hasler, P. E. (2010). Neural dynamics in reconfigurable silicon. *IEEE Trans. Biomed. Circuits Syst.* 4, 311–319. doi: 10.1109/TBCAS.2010.2055157

Boahen, K. (2000). Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* 47, 416–434. doi: 10.1109/82.842110

Brink, S., Nease, S., Hasler, P., Ramakrishnan, S., Wunderlich, R., Basu, A., et al. (2013). A learning-enabled neuron array IC based upon transistor channel models of biological phenomena. *IEEE Trans. Biomed. Circuits Syst.* 7, 71–81. doi: 10.1109/TBCAS.2012.2197858

Cauwenberghs, G. (1996). "Analog VLSI long-term dynamic storage," in *1996 IEEE International Symposium on Circuits and Systems. Circuits and Systems Connecting the World. ISCAS 96 (IEEE)* (Atlanta, GA), 334–337. doi: 10.1109/ISCAS.1996.541601

Dowrick, T., Hall, S., and McDaid, L. (2013). A simple programmable axonal delay scheme for spiking neural networks. *Neurocomputing* 108, 79–83. doi: 10.1016/j.neucom.2012.12.004

Gao, C., and Hammerstrom, D. (2007). Cortical models onto CMOL and CMOS—architectures and performance/price. *IEEE Trans. Circuits Syst. I Regul. Pap.* 54, 2502–2515. doi: 10.1109/TCSI.2007.907830

Gerstner, W., Kempter, R., van Hemmen, J. L., and Wagner, H. (1996). A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383, 76–81. doi: 10.1038/383076a0

Goldberg, D., Cauwenberghs, G., and Andreou, A. (2001). Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons. *Neural Netw.* 14, 781–793. doi: 10.1016/S0893-6080(01)00057-0

Harkin, J., Morgan, F., Hall, S., Dudek, P., Dowrick, T., and McDaid, L. (2008). "Reconfigurable platforms and the challenges for large-scale implementations of spiking neural networks," in *2008 International Conference on Field Programmable Logic and Applications (IEEE)* (Heidelberg), 483–486. doi: 10.1109/FPL.2008.4629989

Harkin, J., Morgan, F., McDaid, L., Hall, S., McGinley, B., and Cawley, S. (2009). A reconfigurable and biologically inspired paradigm for computation using network-on-chip and spiking neural networks. *Int. J. Reconfig. Comput.* 2009, 1–13. doi: 10.1155/2009/908740

Hasler, J., and Marr, B. (2013). Finding a roadmap to achieve large neuromorphic hardware systems. *Front. Neurosci.* 7:118. doi: 10.3389/fnins.2013.00118

Horio, Y., Ymamamoto, M., and Nakamura, S. (1990). Active analog memories for neuro-computing. *IEEE Int. Symp. Circuits Syst.* 4, 2986–2989. doi: 10.1109/ISCAS.1990.112638

Hussain, S., Basu, A., Wang, R., and Hamilton, T. (in press). Delay learning architectures for memory and classification. *Neurocomputing* 1–27.

Hussain, S., Basu, A., Wang, M., and Hamilton, T. J. (2012). "DELTRON: neuromorphic architectures for delay based learning," in *2012 IEEE Asia Pacific Conference on Circuits and Systems (IEEE)* (Kaohsiung), 304–307. doi: 10.1109/APCCAS.2012.6419032

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M. (2006). Polychronization: computation with spikes. *Neural Comput.* 18, 245–282. doi: 10.1162/089976606775093882

Johansson, C., and Lansner, A. (2007). Towards cortex sized artificial neural systems. *Neural Netw.* 20, 48–61. doi: 10.1016/j.neunet.2006.05.029

Levy, W. B., and Baxter, R. A. (1996). Energy efficient neural codes. *Neural Comput.* 8, 531–543. doi: 10.1162/neco.1996.8.3.531

Liu, S., Kramer, J., Indiveri, G., Delbrück, T., and Douglas, R. (2002). *Analog VLSI: Circuits and Principles*. Cambridge, MA: MIT Press.

Masuda, N., and Aihara, K. (2003). Duality of rate coding and temporal coding in multilayered feedforward networks. *Neural Comput.* 15, 103–125. doi: 10.1162/089976603321043711

Mihalaş, S., and Niebur, E. (2009). A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. *Neural Comput.* 21, 704–718. doi: 10.1162/neco.2008.12-07-680

Minkovich, K., Srinivasa, N., Cruz-Albrecht, J. M., Cho, Y., and Nogin, A. (2012). Programming time-multiplexed reconfigurable hardware using a scalable neuromorphic compiler. *IEEE Trans. Neural Netw. Learn. Syst.* 23, 889–901. doi: 10.1109/TNNLS.2012.2191795

Mirhassani, M., Ahmadi, M., and Miller, W. C. (2007). A feed-forward time-multiplexed neural network with mixed-signal neuron–synapse arrays. *Microelectron. Eng.* 84, 300–307. doi: 10.1016/j.mee.2006.02.014

Python, D., and Enz, C. C. (2001). A micropower class-AB CMOS log-domain filter for DECT applications. *IEEE J. Solid State Circuits* 36, 1067–1075. doi: 10.1109/4.933462

Saighi, S., Levi, T., Belhadj, B., Malot, O., and Tomas, J. (2010). "Hardware system for biologically realistic, plastic, and real-time spiking neural network simulations," in *2010 International Joint Conference on Neural Networks* (Barcelona), 1–7. doi: 10.1109/IJCNN.2010.5596979

Schemmel, J., Fieres, J., and Meier, K. (2008). "Wafer-scale integration of analog neural networks," in *2008 International Joint Conference on Neural Networks (IEEE World Congr. Comput. Intell.)* (Hong Kong), 431–438. doi: 10.1109/IJCNN.2008.4633828

Scholze, S., Schiefer, S., Partzsch, J., Hartmann, S., Mayr, C. G., Höppner, S., et al. (2011). VLSI implementation of a 2.8 Gevent/s packet-based AER interface with routing and event sorting functionality. *Front. Neurosci.* 5:117. doi: 10.3389/fnins.2011.00117

Sheik, S., Chicca, E., and Indiveri, G. (2012). "Exploiting device mismatch in neuromorphic VLSI systems to implement axonal delays," in *2012 International Joint Conference on Neural Networks* (Brisbane, QLD), 1–6. doi: 10.1109/IJCNN.2012.6252636

Sheik, S., Pfeiffer, M., Stefanini, F., and Indiveri, G. (2013). "Spatio-temporal spike pattern classification in neuromorphic systems," in *Biomimetic and Biohybrid Systems,* eds N. F. Lepora, A. Mura, H. G. Krapp, P. F. M. J. Verschure, and T. J. Prescott (Heidelberg: Springer), 262–273. doi: 10.1007/978-3-642-39802-5_23

Van Rullen, R., and Thorpe, S. J. (2001). Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Comput.* 13, 1255–1283. doi: 10.1162/08997660152002852

Vogelstein, R. J., Mallik, U., Vogelstein, J. T., and Cauwenberghs, G. (2007). Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Trans. Neural Netw.* 18, 253–265. doi: 10.1109/TNN.2006.883007

Wang, R., Cohen, G., Hamilton, T. J., Tapson, J., and Van Schaik, A. (2013a). "An improved aVLSI axon with programmable delay using spike timing dependent delay plasticity," in *2013 IEEE International Symposium of Circuits and Systems (ISCAS) (IEEE)* (Beijing), 2–5.

Wang, R., Cohen, G., Stiefel, K. M., Hamilton, T. J., Tapson, J., and van Schaik, A. (2013b). An FPGA implementation of a polychronous spiking neural network with delay adaptation. *Front. Neurosci.* 7:14. doi: 10.3389/fnins.2013.00014

Wang, R., Tapson, J., Hamilton, T. J., and van Schaik, A. (2011). "An analogue VLSI implementation of polychronous spiking neural networks," in *2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing (IEEE)* (Adelaide, SA), 97–102. doi: 10.1109/ISSNIP.2011.6146572

Weste, N., and Harris, D. (2005). *CMOS VLSI Design: a Circuits and Systems Perspective, 3rd Edn.* Boston, MA: Addison-Wesley.

Yu, T., and Cauwenberghs, G. (2010). "Log-domain time-multiplexed realization of dynamical conductance-based synapses," in *Proceedings of 2010 IEEE International Symposium on Circuits System* (Paris), 2558–2561. doi: 10.1109/ISCAS.2010.5537114

Zaveri, M. S., and Hammerstrom, D. (2011). Performance/price estimates for cortex-scale hardware: a design space exploration. *Neural Netw.* 24, 291–304. doi: 10.1016/j.neunet.2010.12.003

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 25 September 2013; accepted: 26 February 2014; published online: 18 March 2014.*

*Citation: Wang RM, Hamilton TJ, Tapson JC and van Schaik A (2014) A mixed-signal implementation of a polychronous spiking neural network with delay adaptation. Front. Neurosci.* ***8****:51. doi: 10.3389/fnins.2014.00051*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 21 November 2013
doi: 10.3389/fnins.2013.00215

# Real-time biomimetic Central Pattern Generators in an FPGA for hybrid experiments

***Matthieu Ambroise[1], Timothée Levi[1]*, Sébastien Joucla[2], Blaise Yvert[2] and Sylvain Saïghi[1]***

[1] *Laboratoire IMS, UMR Centre National de la Recherche Scientifique, University of Bordeaux, Talence, France*
[2] *Laboratoire INCIA (Institute for Cognitive and Integrative Neuroscience), UMR Centre National de la Recherche Scientifique, University of Bordeaux, Talence, France*

***Edited by:***
*André Van Schaik, The University of Western Sydney, Australia*

***Reviewed by:***
*Jorg Conradt, Technische Universität München, Germany*
*Runchun M. Wang, University of Western Sydney, Australia*
*M. Anthony Lewis, Qualcomm, QTI, USA*

****Correspondence:***
*Timothée Levi, Laboratoire IMS, UMR Centre National De La Recherche Scientifique 5218, Université de Bordeaux, 351 Cours de la Libération, 33405 Talence, France*
*e-mail: timothee.levi@ims-bordeaux.fr*

This investigation of the leech heartbeat neural network system led to the development of a low resources, real-time, biomimetic digital hardware for use in hybrid experiments. The leech heartbeat neural network is one of the simplest central pattern generators (CPG). In biology, CPG provide the rhythmic bursts of spikes that form the basis for all muscle contraction orders (heartbeat) and locomotion (walking, running, etc.). The leech neural network system was previously investigated and this CPG formalized in the Hodgkin–Huxley neural model (HH), the most complex devised to date. However, the resources required for a neural model are proportional to its complexity. In response to this issue, this article describes a biomimetic implementation of a network of 240 CPGs in an FPGA (Field Programmable Gate Array), using a simple model (Izhikevich) and proposes a new synapse model: activity-dependent depression synapse. The network implementation architecture operates on a single computation core. This digital system works in real-time, requires few resources, and has the same bursting activity behavior as the complex model. The implementation of this CPG was initially validated by comparing it with a simulation of the complex model. Its activity was then matched with pharmacological data from the rat spinal cord activity. This digital system opens the way for future hybrid experiments and represents an important step toward hybridization of biological tissue and artificial neural networks. This CPG network is also likely to be useful for mimicking the locomotion activity of various animals and developing hybrid experiments for neuroprosthesis development.

**Keywords: central pattern generator, biomimetic, neuron model, spiking neural networks, digital hardware, FPGA**

## INTRODUCTION

Millions of people worldwide are affected by neurological disorders which disrupt connections between brain and body, causing paralysis or affecting cognitive capabilities. The number is likely to increase over the next few years and current assistive technology is still limited. In recent decades, extensive research has been devoted to Brain-Machine Interfaces (BMIs) and neuroprosthesis in general (Hochberg et al., 2006, 2012; Nicolelis and Lebedev, 2009), working toward effective treatment for these disabilities. The development of these devices has had and, hopefully, will continue to have a profound social impact on these patients' quality of life. These prostheses are designed on the basis of our knowledge of interactions with neuronal cell assemblies, taking into account the intrinsic spontaneous activity of neuronal networks and understanding how to stimulate them into a desired state or produce a specific behavior. The long-term goal of replacing damaged neural networks with artificial devices also requires the development of neural network models that match the recorded electrophysiological patterns and are capable of producing the correct stimulation patterns to restore the desired function. The hardware set-up used to interface the biological component is a Spiking Neural Network (SNN) system implementing biologically realistic neural network models, ranging from the electrophysiological properties of a single neuron to large-scale neural networks.

Our study describes the development of a neuromorphic hardware device containing a network of real-time biomimetic Central Pattern Generators (CPG). The main goal of this research is to create artificial CPGs that will be connected to *ex vivo* spinal cord of rats and guinea pigs, thus achieving one main objective of the Brainbow European project (Brainbow, 2012) toward hybridization. Hardware-based SNN systems were developed for hybrid experiments with biological neurons and the description of those pioneer platforms was reported in the literature (Jung et al., 2001; Le Masson et al., 2002; Vogelstein et al., 2006). The Brainbow project will go further by using a large-scale neural network instead of few neurons to substitute the functions of a biological sub-network. The final goal is the development of a new generation of neuro-prostheses capable to restore the lost communication between neuronal circuitries.

Locomotion is one of the most basic abilities of animals. Neurobiologists have established that locomotion results from the activity of half-center oscillators that provides alternating bursts. The first half-center oscillator was proposed by Brown (1914). Pools of interneurons control flexor and extensor motor neurons with reciprocal inhibitory connections. Most rhythmic

movements are programmed by central pattern-generating networks consisting of neural oscillators (Marder and Bucher, 2001; Ijspeert, 2008). CPGs are neural networks capable of producing rhythmic patterned outputs without rhythmic sensory or central input. CPGs underlie the production of most rhythmic motor patterns and have been extensively studied as models of neural network function (Hooper, 2000). Half-center oscillators control swimming in xenopus, salamander (Ijspeert et al., 2007), and lamprey (Cohen et al., 1992), as well as leech heartbeat (Cymbalyuk et al., 2002), as described in numerous publications. One key article on modeling the leech heartbeat system is Hill et al. (2001), where the Hodgkin–Huxley formalism is used to reproduce the CPG.

The main novelty of this research was to implement the leech heartbeat system neural network with minimum resources while maintaining its biomimetic activity. Indeed, the final application is a hybrid experiment that requires spike detection, spike sorting, and micro-electrode stimulation. All of these modules are implemented in the same digital board. To achieve this, the Hill et al. (2001) model and results were reproduced using a simpler model (Izhikevich, 2004), implemented in an FPGA (Field Programmable Gate Array) board. This digital board made it possible to design a frugal, real-time network of several CPGs (in this case, a network of 240 CPGs implemented on a Spartan6 FPGA board). For instance, this CPG network is capable of mimicking the activity of a salamander, which requires 40 CPGs (Ijspeert, 2001), or developing hybrid experiments (Le Masson et al., 2002) for neuroprosthesis development (Brainbow, 2012).

The first part of this article describes the biological leech heartbeat system, based on one segmental CPG. The next section focuses on choosing a frugal neuron model to match the same biological behavior. The following section explains the topology of a single neuron and its implementation in the hardware, followed by its extension to a neuron computation core for increasing the size of the neural network. The next stage was to develop a new synaptic model reproducing activity-dependent depression phenomena to fit the biological activity of a leech heartbeat. The architecture of this digital system is then described in full, including the various blocks. Finally, the system was used to design a CPG network, validated by comparing our measurements with *ex vivo* rat spinal cord locomotion results following pharmacological stimulation.

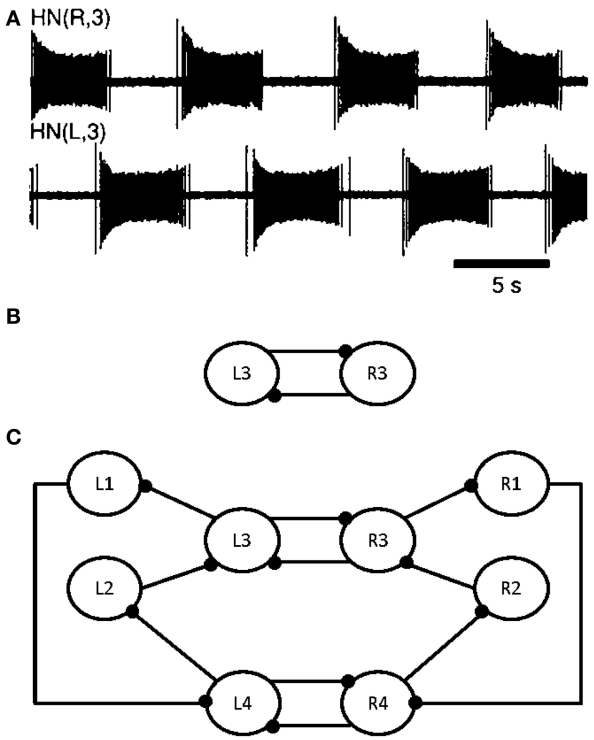



**FIGURE 1 | Electrical activity of the leech heartbeat system and diagram of the CPG.** Neuron cell bodies are represented by circles. Axons and neurite processes are represented by lines. Inhibitory chemical synapses are represented by small filled dots. **(A)** Electrical activity of two heart interneurons recorded extracellularly from a chain of ganglia (Hill et al., 2001). **(B)** A diagram of the elemental oscillator in the leech heartbeat system. **(C)** A diagram of the segmental oscillator in the leech heartbeat system, including two elemental oscillators, L3/R3 and L4/R4, and two pairs of coordination neurons, L1/R1 and L2/R2.

## MATERIALS AND METHODS

### DESCRIPTION OF THE LEECH BIOLOGICAL HEARTBEAT SYSTEM

All leech heartbeat studies agree that the CPG (**Figure 1C**) responsible for this activity (**Figure 1A**) requires few neurons, making it an ideal candidate system for elucidating the various biomechanisms governing CPG behavior.

Modeling studies indicate that the burst duration of a leech heart interneuron in an elemental oscillator is regulated by the interneuron itself and by the opposite interneuron (see L3 and R3 in **Figure 1B**) (Calabrese, 1995; Nadim et al., 1995; Olsen et al., 1995; Hill et al., 2001; Jezzini et al., 2004; Norris et al., 2007). **Figure 1A** shows the electrical activity in the leech heartbeat system from extracellular recordings. The pair of neurons is known as an elemental oscillator (**Figure 1B**), i.e., the smallest unit that capable of producing robust oscillations under normal conditions. These neurons oscillate in alternation with a period of about 10–12 s (Krahl and Zerbst-Boroffka, 1983; Calabrese et al., 1989; Olsen and Calabrese, 1996) demonstrated that the synaptic connections among interneurons and from interneurons to motor neurons were inhibitory. The synaptic interaction between reciprocally inhibitory heart interneurons consists of a graded component in addition to spike-mediated synaptic transmissions (Angstadt and Calabrese, 1991). This kind of synapse is really difficult to implement in hardware as it contains sigmoid functions, differential equations, memory of last spikes, and so on. A description of our synapse model reproducing the same behavior is included below.

Nadim et al. (1995) and Olsen et al. (1995) developed a biophysical model of a pair of reciprocally inhibitory interneurons in the leech heartbeat system. This model included synaptic ionic currents based on voltage-clamp data. Synaptic transmissions between the interneurons consist of spike-mediated and graded synaptic currents. The Hill et al. (2001) model was derived from a previous two-cell, elemental oscillator model (Nadim et al., 1995) by incorporating intrinsic and synaptic current modifications based on the results of a realistic waveform voltage-clamp

study (Olsen and Calabrese, 1996). This new, segmental oscillator model behaves more similarly to biological systems. **Figure 1C** shows a model of the system. The real-time digital segmental oscillator model design will be based on this architecture. The next part will describe the system modeling the leech heartbeat with the goal of implementing it in hardware. The leech heartbeat CPG was chosen for the long duration of the burst.

## SYSTEM MODELING FOR HARDWARE IMPLEMENTATION

### *State of art*

Some previous studies used silicon neurons (Indiveri et al., 2011) to simulate the leech heartbeat system (Simoni et al., 2004; Simoni and DeWeerth, 2007). Sorensen et al. (2004) created a hybrid system of a heart interneuron and a silicon neuron. The silicon neuron provides real-time operation and implements a version of the Hodgkin–Huxley formalism (Hodgkin and Huxley, 1952). However, due to the complexity of the model, it was only possible to use a small number of silicon neurons and, therefore, only one CPG. This study describes the same results using a large CPG network (240 CPGs on a Spartan6 FPGA board), in preparation for future hybrid experiments with different CPGs. For instance, in the salamander model (Ijspeert, 2001), the body CPG consists of 40 interconnected segmental networks.

When a silicon neuron and heart interneuron are connected with reciprocal inhibitory synapses of appropriate strength, they form a hybrid elemental oscillator that produces oscillations remarkably similar to those seen in the living system. Olypher et al. (2006) described the control of burst duration in heart interneurons using a hybrid system, where a living, pharmacologically-isolated, heart interneuron was connected with artificial synapses to a model heart interneuron running in real-time (software). Using an FPGA board will make it possible to operate in real time using a large number of neurons, together with customized systems for various applications (hybrid experiments).

A few studies (Torres-Huitzil and Girau, 2008; Rice et al., 2009; Serrano-Gotarredona et al., 2009; Barron-Zambrano et al., 2010; Barron-Zambrano and Torres-Huitzil, 2013) reported on CPG in FPGA for robotic applications. These studies used simple neuron-models and were more bio-inspired than biomimetic. Guerrero-Riberas et al. (2006) implemented a network of LIF neurons with synapses and plasticity, but not in biological time, so it was impossible to perform hybrid experiments. While multi-legged robots need CPG to move or coordinate their movements, they implement an Amari–Hopfield CPG (Amari, 1972) or basic CPGs (Van Der Pol, 1928), modeled as nonlinear oscillators. Those models provide sinusoidal oscillations that are not biorealistic. The ultimate goal of these studies is to create a robot that mimics biological behavior but these systems cannot be used for hybrid experiments. Analog hardware has also been implemented (Linares-Barranco et al., 1993; Still and Tilden, 1998; Lewis et al., 2001; Nakada, 2003; Still et al., 2006; Lee et al., 2007; Wijekoon and Dudek, 2008). However, it is very difficult to tune analog circuits due to parameter mismatch. For these works, they either design bio-inspired oscillators for creating CPG or implement few biomimetic neurons.

### *Choice and presentation of the Izhikevich model*

In designing a SNN, the first step is the choice of a biologically realistic model. Indeed, a mathematical model based differential equations is capable of reproducing a behavior quite similar to that of a biological cell. The choice of model was based on two criteria: the family of neurons able to be reproduced and the number of equations. These criteria were used to compare several models, including the Leaky Integrate and Fire model (LIF) (Indiveri, 2007), the Hodgkin–Huxley model (HH), and the Izhikevich model (IZH).

Hill et al. (2001) used the HH to reproduce the leech heartbeat system with eight neurons (**Figure 1C**). From the equations defined in this paper, it was established that the eight neurons in the heartbeat leech behaved like regular spiking ones (RS). Indeed, this model was composed of nine voltage-dependent currents with different calcium conductances.

The HH model reproduces all types of neurons with good accuracy (spike timing and shape). Its main drawbacks are the large number of parameters and the equations required. In the heartbeat network, the main focus is on excitatory neurons, like RS. The HH model required 32 parameters for an RS and 26 for a fast-spiking neuron (FS) (Grassia et al., 2011). Furthermore, simulating an RS neuron required four ionic channels (dynamics of potassium and sodium ions, leak current, and slow potassium). In contrast, LIF only involves two equations but is only capable of simulating a few types of neurons.

The IZH represents a good solution, as it is based on two equations and is capable of reproducing many different families of neurons by changing four parameters. Furthermore, according to Izhikevich (2004), this model is resource-frugal, a key advantage when the aim is to design a large CPG network embedded in the same board as other modules required for hybrid experiments (spike detection, spike sorting, stimulation, etc.).

The IZH model depends on four parameters, which make it possible to reproduce the spiking and bursting behavior of specific types of cortical neurons. From a mathematical standpoint, the model is described by a two-dimensional system of ordinary differential equations (Izhikevich, 2003):

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I_{\mathrm{Izh}} \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

with the after-spike resetting conditions:

$$\text{if} \quad v \geq 30\,\mathrm{mV} \Rightarrow \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{3}$$

In equation (3), $v$ is the membrane potential of the neuron, $u$ is a membrane recovery variable, which takes into account the activation of potassium and inactivation of sodium channels, and $I_{\mathrm{Izh}}$ describes the input current from other neurons.

The IZH model was chosen to emulate the behavior of the excitatory cells for its simplicity and its capacity to implement various families of neurons. The next step was to determine the

network system topology. The next section describes the design of one neuron and its extension to a neuron computation core, then the different synapse models implemented, and, finally, the topology of the network.

## SYSTEM TOPOLOGY

### *Topology of one neuron core: architecture and implementation*

In order to make the Izhikevich neural network more biomimetic, the $I_{Izh}$ current from equation (1) was split into three: $I_{bias}$, $I_{exc}$, and $I_{inh}$. $I_{bias}$ is the biasing current, $I_{exc}$ is the positive contribution due to excitatory synapses, and $I_{inh}$ is the negative contribution of inhibitory synapses. Those currents will be detailed in Synapse Model. As suggested in Cassidy and Andreou (2008), equation (1) was multiplied by 0.78125 to make it easier to implement on a digital board. These modifications gave (4), where the $u$ coefficient is still 1 thanks to $I_{bias}$ current.

$$\begin{aligned}\frac{dv}{dt} &= {}^{1}\!/_{32}\, v^2 + 4v + 109.375 - u + I_{bias} + I_{exc} + I_{inh} \\ \frac{du}{dt} &= a \cdot (bv - u)\end{aligned} \tag{4}$$

Moreover, $\frac{dv}{dt} = \frac{v[n+1]-v[n]}{\Delta t}$ and, as the time step of the IZH model is equal to one millisecond ($\Delta t = 1$):

$$\begin{aligned}v[n+1] &= {}^{1}\!/_{32}\, v[n]^2 + 5v[n] + 109.375 - u[n] \\ &\quad + I_{bias}[n] + I_{exc}[n] + I_{inh}[n] \\ u[n+1] &= u[n] + a.(b.v[n] - u[n])\end{aligned} \tag{5}$$

One neuron was implemented on the FPGA board according to these equations and specifications. This neuron was then extended into a neuron computation core that updated the $u$ and $v$ values of all neurons in the network. Consequently, the neuron implementation became a neuron computation core. For instance, around 2000 independent neurons could be implemented on our digital board. In this system, the type of neuron is defined by the four Izhikevich parameters: a, b, c, and d from equations (2) and (3). Moreover, the state of a neuron is defined by values u and v, and the three current values. Those 9 values were saved in a RAM for use in the next millisecond in the step computation. By extension, the same process can be used for every neuron in the network.

Each $u$ and $v$ computation step is run in parallel, using two pipelines based on the architecture presented in [9]. The topology is presented in **Figure 2**. All parameters from equations (2), (3), and (4), as well as the $u$ and $v$ values used in the computation are synchronized in one cycle before going through the pipelines (not shown in **Figure 2**).

To resume, each neuron is represented by one "v" and "u" value, four Izhikevich coefficients (a, b, c, and d), and three currents ($I_{bias}$, $I_{exc}$, and $I_{inh}$).

$I_{exc}$, $I_{inh}$, and $I_{bias}$ are added in two cycles at the beginning of the "$v$" pipeline, while the "$u$" pipeline is still inactive (steps 1 and 2). The current sum is added to the constant *109.375* and at the same time as the first multiplication (step 3). By multiplexing operands, the same multiplier is used for the following multiplications in different computation cycles. In step 4, $v^2$ is obtained by another multiplication. A simple two-bit shift makes it possible to obtain *4v* and add it to *v*. At the same time, $u$ is used in two subtractions. Step 5 consists of a 5-bit shift to obtain *(1/32)*$v^2$, an addition, and the last multiplication. In step 6, the computation of both $u$ and $v$ is completed. In the next step, the $v$ value is tested against the threshold to determine whether the neuron has emitted a spike or not. This test gives the next $u$ and $v$ values for this neuron to be stored in the RAM.

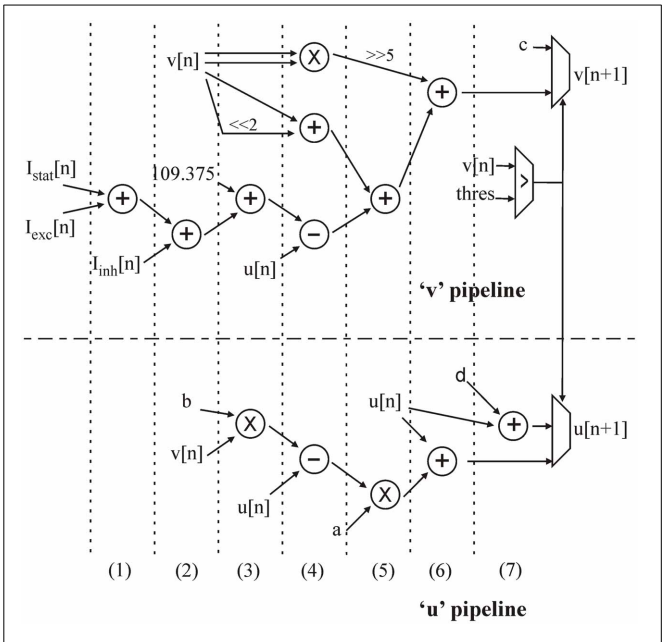



**FIGURE 2 | Architecture of the "u" and "v" pipelines in the neural computation core.** The computation cycles are separated by dotted lines.

An RS neuron with $a = 0.002$, $b = 0.2$, $c = -65$, and $d = 8$ was used to implement the CPG.

Once the neuron computation core was implemented, the synaptic model was chosen and implemented.

### *Synapse model*

A network is defined by a group of neurons and a group of synapses. Once the neuron model had been chosen, it was obviously necessary to choose a synapse model. Like the neuron model, this model had to be biomimetic but frugal in its use of resources. In biology, synapses are described as links between neurons that transmit different types of synaptic currents to each other to either excite or inhibit neuron activity. In our implementation, a synaptic weight ($W_{syn}$) was added to the synaptic current. When $W_{syn}$ was positive, it was added to $I_{exc}$ (excitatory synaptic current) and when $W_{syn}$ was negative, it was added to $I_{inh}$ (inhibitory synaptic current).

Thanks to AMPA and GABA effects, all synaptic current excitations or inhibitions, respectively decay exponentially (Ben-Ari et al., 1997). AMPA is an excitatory neurotransmitter that depolarizes the neuron membrane whereas GABA is an inhibitory

neurotransmitter with a hyperpolarizing effect. Depolarization or hyperpolarization are represented by a positive or negative contribution on the synaptic current.

The synaptic current $I_{syn}$ was implemented with a time constant $\tau_{syn}$ for the exponential decay, as follows:

$$I_{syn}(t) = -\tau_{syn} \cdot I'_{syn}(t) = -\tau_{syn}.\frac{I_{syn}(t+T) - I_{syn}(t)}{T} \quad (6)$$

$$I_{syn}(t+T) = \left(1 - \frac{T}{\tau_{syn}}\right) \cdot I_{syn}(t) \quad (7)$$

When computation step T equals one millisecond and $\tau_{syn}$ is in ms:

$$I_{syn}(t+1) = \left(1 - \frac{1}{\tau_{syn}}\right) \cdot I_{syn}(t) \quad (8)$$

$$I_{syn}[n+1] = I_{syn}[n] - \frac{1}{\tau_{syn}} \cdot I_{syn}[n] \quad (9)$$

Adding the synaptic weight to the synaptic current, the new equation is:

$$I_{syn}[n+1] = I_{syn}[n] - \frac{1}{\tau_{syn}} \cdot I_{syn}[n] + W_{syn}[n] \quad (10)$$

The synaptic computation core implementation is based on the same principle as the neuron computation core. However, this model is not adequate to fit biological data. It was, therefore, decided to implement an activity-dependent depression, where the new synaptic weight, $W_s$, was dependent on $W_{syn}$.

#### *Activity-dependent depression*

As the synaptic behavior described in Hill et al. (2001) requires too many resources to be implemented on FPGA, the method chosen to fit overall biological behavior was activity-dependent depression (Tabak et al., 2000). Activity-dependent depression of synapses is another biological phenomenon consisting of reducing a synaptic weight after a spike. In biology, each synapse contribution is provided by a synaptic vesicle. These vesicles contain ions that empty out at each spike and then regenerate, following an exponential rule. According to Matsuoka (1987), four methods provide a stable rhythm within a network (regulation of stimulus intensity, change in input, alteration of stimuli, and change in synaptic weight). The phenomenon, known as activity-dependent depression changes the synaptic weight depending on the activity of the network.

This phenomenon has been reported in neurobiology literature but no model had been devised. This paper proposes a model of this activity-dependent depression that was implemented in digital hardware to improving our CPG network.

As previously explained, each time a neuron emits a spike; the synapse adds a synaptic weight ($W_{syn}$) to the synaptic current. At the same time, the factor ($\delta_{syn}$) indicating the level of depression on a synaptic weight increases. Furthermore, $\delta_{syn}$ regulates $W_{syn}$. The value of $\delta_{syn}$ is between 0 and 1. Consequently, when $\delta_{syn}$ equals zero there is no depression on $W_{syn}$ and when $\delta_{syn}$ equals one there is maximum depression on $W_{syn}$ and the synapse is exhausted.

$W_s$ was used instead of $W_{syn}$ as the synaptic weight for each synapse. Then, according to the activity-dependent depression effect, when there is a spike, $W_s$ is added to the synaptic current:

$$W_s[n] = W_{syn} - \delta_{syn}[n] \cdot W_{syn} \quad (11)$$

The other effect of activity-dependent depression is to increase $\delta_{syn}$ after each spike, thanks to the percentage dissipation ($P$).

$$\delta_{syn}[n+1] = \delta_{syn}[n] + P \cdot (1 - \delta_{syn}[n]) \quad (12)$$

The regeneration or reloading of synaptic vesicles is represented by $\delta_{syn}$ decreasing to zero. Thus, $\delta_{syn}$ decays exponentially when no spike is emitted. So, using the method described in Synapse Model:

$$\delta_{syn}[n+1] = \delta_{syn}[n] - \frac{1}{\tau_{reg}} \cdot \delta_{syn}[n] \quad (13)$$

To summarize, all synapses are now represented by (12), (13), (14) and:

$$I_{syn}[n+1] = I_{syn}[n] - \frac{1}{\tau_{syn}} \cdot I_{syn}[n] + W_s[n] \quad (14)$$

The main parameters are: synaptic weight, $W_{syn}$; level of depression, $\delta_{syn}$; and percentage dissipation, $P$. All these parameters are stored in the RAM on the digital board. Furthermore, this computation required greater precision due to the sensitivity of the parameters. The 26-bit signed fixed representation chosen had 1-bit for the sign, 9-bits for the whole numbers, and 16 for the decimals.

Once the neuron and synapse models had been designed, it was possible to develop the neural network topology.

#### *Network topology*

***Three elementary blocks.*** The architecture was based on three main blocks: the neuron implemented (or neuron computation core), a synapse, and the RAM. The connectivity between those blocks is shown in **Figure 3**.

So far, the neuron computation core can update the state ("u" and "v" variables) of each neuron. In the digital network, the role of the synapse is to update all synaptic currents and weights related to the activity of all neurons, so the synapse block exhibits two behaviors (spiking or not). These two behaviors are summarized in **Table 1.**

The IZH model has a time step of one millisecond, so the other computation was synchronized with this time step. The new values of u and v, the exponential decay of $I_{syn}$, and the new values of each synaptic current are computed in the same millisecond.

Moreover, a biological neural network is composed of *Nn* neurons and *Ns* synapses. To define which neuron is connected to which and with which kind of synapse (excitatory or inhibitory), the network is described using two matrixes: connectivity and synaptic weight (see **Figure 3**). To save RAM, both matrixes are implemented as sparse matrixes with *Nn* lines. The *i*th line in the

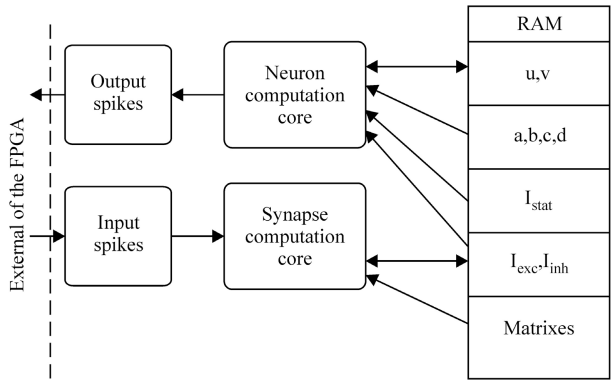



**FIGURE 3 | Global architecture of the spiking neural network.**

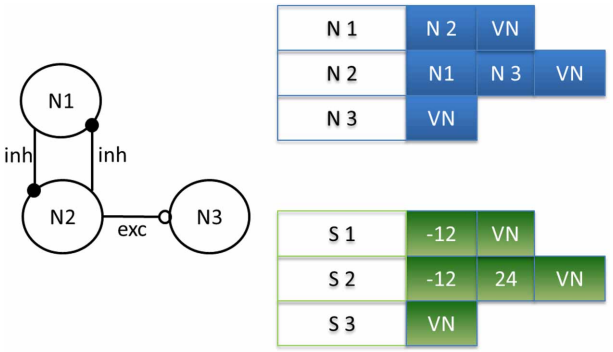



**FIGURE 4 | Example of matrix design depending on the neural network.** Neuron 2 (N2) is connected by an inhibitory synapse to neuron 1 (N1) and by an excitatory synapse to neuron 3 (N3). Then, on line 2 in the connectivity matrix, N2 is connected to N1, N3, and a virtual neuron (VN), indicating the end of the connection. In the synaptic weight matrix, the synapses for neuron 2 (S2) have a negative weight for the inhibitory synapse and a positive weight for the excitatory synapse. Note the correspondence of its position in both matrixes.

**Table 1 | Description of the equations for synaptic currents and activity-dependent depression.**

| When a spike is emitted | When no spike is emitted |
|---|---|
| Synaptic current | Synaptic current |
| $I_{exc}[n+1] = I_{exc}[n] + W_s[n]$ or $I_{inh}[n+1] = I_{inh}[n] + W_s[n]$ | $I_{exc}[n+1] = I_{exc}[n]$ and $I_{inh}[n+1] = I_{inh}[n]$ |
| Activity-dependent depression | Activity-dependent depression |
| $\delta_{syn}[n+1] = \delta_{syn}[n] + P(1 - \delta_{syn}[n])$ | $\delta_{syn}[n+1] = \delta_{syn}[n] - \frac{\delta_{syn}[n]}{\tau_{reg}}$ |

connectivity matrix corresponds to the connectivity of presynaptic neuron $N_i$ to the other neurons. The synapses are identified by the postsynaptic neuron addresses. For example, the connection to neuron $N_j$ is identified by the number $j$ on the $i$th line. In the worst case, each neuron is connected to itself and all the others, giving $Nn$ columns. Each matrix line ends with a virtual neuron (address $Nn + 1$). This implementation is not optimum for the worst case, but the gain is significant for biologically plausible networks, where the total number of synapses is at least four times smaller. Marom and Shahaf (2002) and Garofalo et al. (2009) estimated the average connectivity level of neural networks at their mature phase each neuron is mono-synaptically connected to 10–30% of all the other neurons.

There is a direct link between the matrixes: the synaptic weight matrix is the same size as the connectivity matrix, i.e., the same number of lines and columns, with the virtual neurons in the same position (**Figure 4**). The connectivity between two neurons described by the coordinates ($k$, $l$) in the connectivity matrix has the weight shown in box ($k$, $l$) in the synaptic weight matrix. A third matrix based on the same principle completes the system: the percentage efficiency matrix, which gives the percentage dissipation, $P$, of each synapse in a network, as defined in the previous section on activity-dependent depression. We will describe now the state machine of the neural network.

***Network machine states.*** The synaptic current is computed in three successive steps:

- EXT state: for closed-loop experiments, we implement this state in which external feedback can interact with the artificial neural network. This first state consists of using the synaptic block to update the synaptic current. In this case, presynaptic spikes are external events (see **Figure 3**), such as stimulation from biological neurons in the case of neuroprosthesis. This state makes it possible to stimulate each neuron.
- NEUR state: during this step, the neuron membrane ("$u$" and "$v$" from **Figure 2**) and all exponential decay values are computed in parallel.
- SYN state: the last step consists of updating the synaptic current to reflect the presynaptic spikes computed in the NEUR state. These updated current values are used in the EXT state during the next cycle.

The EXT, NEUR, and SYN states must be completed within a one millisecond time step. If the computation of all three states is completed in less than 1 ms, an IDLE state is implemented until the end of the cycle. Moreover, the blocks (neuron computation core and synapse computation core) described in **Figure 3** are multiplexed in time to reduce the implementation area in large-scale neural networks.

Our architecture has two main limits: the number of available cycles (Nc) in one millisecond and the size of the RAM used to save all parameters. Two equations derived from these limits determine the maximum size of the implementable neural network, in terms of number of neuron (Nn) and synapses (Ns).

In the EXT state, all synaptic currents are updated in 10 cycles for each neuron, i.e., 10· Nn cycles. Each neuron requires 11 cycles to compute the NEUR state, i.e., 11· Nn cycles. The synaptic current update during state SYN requires 10 cycles per synapse, i.e.,

10· Ns. **Figure 2** describes 7 cycles for the neuron computation core, but 4 more cycles are required to read and save the various parameters in the RAM.

This leads to the following equation for computing the maximum number of neurons that may be implemented, depending on the number of cycles available:

$$10 \cdot Nn + 11 \cdot Nn + 10 \cdot Ns \leq Nc \quad (15)$$

Having built all the component parts of this real-time, biomimetic digital system, it was possible to validate it by several experiments, presented in the following section.

## RESULTS

A CPG is defined by the number of neurons and the families of neurons and synapses. The leech heartbeat neural network was simulated by an appropriate CPG configuration.

Hill et al. (2001) presented an elemental oscillator, based on two excitatory neurons linked by inhibitory synapses. A segmental oscillator may consist of 4–10 neurons. A two-neuron network (elemental oscillator from **Figure 1B**) was chosen to validate our topology, followed by an eight-neuron neural network (segmental oscillator from **Figure 1C**). The activity of our system was then compared with that of an *ex vivo* rat spinal cord, stimulated with pharmacological solutions. It was also demonstrated that the period of bursting activity could be modified depending on one parameter. This will be useful in future closed-loop hybrid experiments.

Biological CPGs provide specifications concerning their behavior. Indeed, their activity is characterized by periodic long bursts (lasting many seconds). Each burst begins by a quick rise in spike frequency to a maximum and ends with a low final spike frequency.

### COMPARISON OF BIOLOGICAL/DIGITAL ELEMENTAL OSCILLATOR

The first example of a CPG was the elemental oscillator (with only two neurons). To reproduce activity accurately, it was necessary to obtain the following values: $\tau_{ampa}$ (time constant of the inhibitory synaptic current exponential decay), $\tau_{reg}$ (time constant of the recovery of synaptic vesicles), and $P$ (percentage dissipation). These values will be the same for each synapse. The following values were chosen to match biological behavior: $\tau_{current} = 100$ ms and $\tau_{reg} = 4444$ ms (so $1/\tau_{current} = 0.01$ and $1/\tau_{reg} = 0.0002$). The $I_{bias}$ current was equal to 8 for both neurons. The synaptic weights are $-5.1$ and the percentages of dissipation are 1.49.

This model was validated by comparing its implementation with the complex model in Hill et al. (2001) (see **Figure 5**).

In this case, the activity of one neuron inhibits the second neuron. Due to activity-dependent depression and the GABA effect, the inhibition ends and lets the second neuron fire again. In both cases (biological modeling system and digital system), the bursting activity was similar in terms of period and duty cycle, thus validating the simplified elemental oscillator with the complex one. The next step was to validate the segmental oscillator and compare its implementation with biological data.

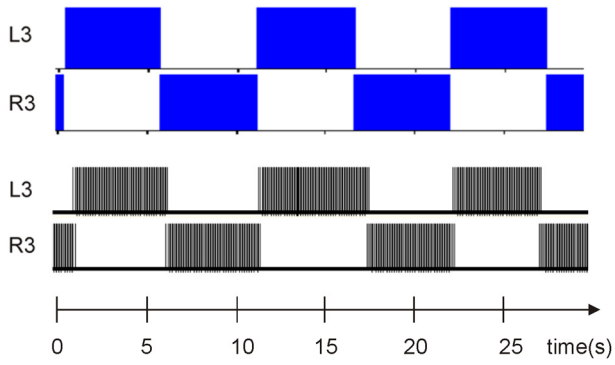



**FIGURE 5 | Comparison between elemental oscillator (Figure 1B) bursting activity in the complex model simulated by scilab, as described in Hill et al. (2001) and the elemental oscillator presented above thanks to a logic analyzer.** The time scale is the same.

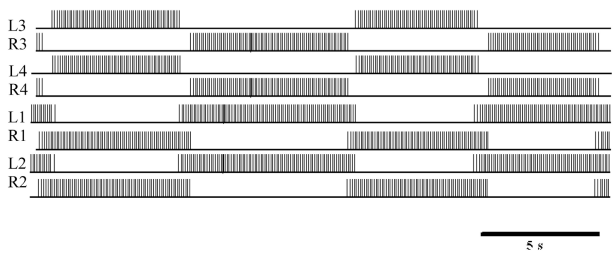



**FIGURE 6 | Logic analyzer measurements of the digital eight-neuron CPG.** L3 and R3 show the activity of the first oscillator. L4 and R4 show the activity of the second oscillator. L1/R1 and L2/R2 are the coordination neurons.

### COMPARISON OF BIOLOGICAL/DIGITAL SEGMENTAL OSCILLATOR

Keeping the time constant, the biological behavior of the eight-neuron network was duplicated using the following parameters. This time, an eight-neuron CPG was implemented using the same values for $\tau_{current}$ and $\tau_{reg}$ than as those used for the elemental oscillator. The use of 8 neurons made it possible to maintain the period without variation (see **Table 2**) by slowing down the two pairs of oscillators with coordination neurons (De Schutter, 2000).

In **Figures 1C**, **6**, L3/R3 and L4/R4 correspond to the two elemental oscillators and are coupled to the L1/R1 and L2/R2 coordination neurons. The connectivity between each neuron is following **Figure 1C**. The synaptic weights are $-7$ and the percentage of dissipation is 2.65.

The mean period, duty cycle, and variations in spike frequency depending on their position in the burst were measured to quantify the overlap of bursting activity (**Table 2**). The mean period of this digital implementation was similar to biological values. Note that the segmental oscillator exhibited less variation than the elemental system, thanks to its coordination neurons.

Also, in general, the spike frequency of our implementation was similar to that of the biological system. Due to our synapse

**Table 2 | Comparison of burst characteristics in the two digital implementations and the biological system.**

| | Biological system (Hill et al., 2001) | Elemental oscillator (digital) | Segmental oscillator (digital) |
|---|---|---|---|
| Mean period | 10–12 s | 12.6 ± 1.4 s | 11.2 ± 1 s |
| Mean duty cycle | 57.2 ± 2.9% | 54.7 ± 6% | 46.1 ± 6% |
| Mean spike frequency | 11.9 ± 2.1 Hz | 12.1 ± 1 Hz | 11.2 ± 1 Hz |
| Initial spike frequency | 4.3 ± 0.7 Hz | 8.5 ± 0.2 Hz | 8.6 ± 0.4 Hz |
| Peak spike frequency | 17.5 ± 3.2 Hz | 13 ± 0 Hz | 12.5 ± 0 Hz |
| Final spike frequency | 5.8 ± 1.0 Hz | 8.1 ± 0.2 Hz | 9.3 ± 3 Hz |

model, the frequency reached a maximum in each spike burst but remained on a plateau instead of decreasing to the minimum frequency immediately. In the biological system, the behavior described is due to the enhancement and attenuation of variations in conductance. However, the IZH model does not include conductance, so it cannot be as biomimetic as the HH model. This highlights a weak point of the implementation presented here, but even the HH model, Hill et al. (2001) was unable to mimic this biological variation in spike frequency in a single burst. One discrepancy between the model and the biological system is that the initial and final spike frequencies of a burst were consistently lower in the biological system. In both implementations, the most inconvenient drawback was the variation in the duty cycle, explained by the stability of the IZH model. One perspective of this work to ensure stability is described in the discussion section.

These experiments validated the implementation of our elemental and segmental oscillators. This table also confirms that designing a biomimetic system was a good choice. Indeed, the variations of the duty cycle and the period for the bursting activity could not be reproduced by bio-inspired oscillators. The next step was to identify one parameter that would modify the bursting activity period, which would be useful in closed-loop applications.

### VARIATION IN THE MEAN PERIOD DEPENDING ON ONE PARAMETER

A CPG is defined here by the number of neurons and the type synapses involved, the static currents of each neuron, the percentage dissipation, and the synaptic efficiency time constant.

Changing the synaptic efficiency time constant $\tau_{reg}$ modifies the period of each spike burst (**Table 3**). The variation in $\tau_{reg}$ affects the period and duration of each burst, as well as the duty cycle and the variability of these parameters: the greater the value of $1/\tau_{reg}$, the longer the mean period of bursting activity.

The possibility of modifying the period using a single parameter is very useful and was applied in a closed-loop hybrid experiment concerning locomotion behaviors.

**Table 3 | Variation in the mean period depending on the $\tau_{reg}$ parameter.**

| $1/\tau_{reg}(ms^{-1})$ | Mean period (s) |
|---|---|
| 0.09 | 4.4 ± 1.6 |
| 0.15 | 7.2 ± 1.2 |
| 0.20 | 11.2 ± 1 |
| 0.22 | 12.9 ± 1.1 |

**Table 4 | Resources required for one CPG on a Spartan 6 digital board.**

| Resources | Total available | Used for one CPG | Used for 240 CPGs |
|---|---|---|---|
| Slice FF's | 184304 | 1,093 (0.6%) | 1,459 (0.8%) |
| Slice LUT | 92152 | 1,037 (1.2%) | 1,756 (1.9%) |
| DSP48A1 | 180 | 10 (5.6%) | 10 (5.6%) |
| RAMB16BWER | 268 | 1 | 42 (756 kb) |
| Total RAM | 4824 kb | 9 kb (0.2%) | 765 kb (16%) |

### FPGA RESOURCES

Originally, a CPG consisted of 8 neurons and 12 synapses but 2 additional synapses per CPG were required to create a network of CPG, by connecting CPG to another one. Thus, each CPG consisted of 8 neurons and 14 synapses. In terms of cycles and available memory, this implementation was capable of running 240 CPGs on a Spartan 6 digital board [see equation (15) and **Table 4**]. The power consumption of one CPG is 8 mW and for CPGs is 20 mW. We could reduce it in the future by designing a custom ASIC. For neuroprosthesis application, the power consumption should be lower than 80 mW/cm2 chronic heat dissipation level considered to prevent tissue damage (Zumsteg et al., 2005).

### COMPARISON WITH *EX-VIVO* RAT SPINAL CORD RESULTS USING PHARMACOLOGICAL STIMULATION

The final validation of this system consisted of comparing the CPG output with *ex vivo* physiological data obtained from the spinal cord of newborn rat [postnatal day (P)1–2]. Bursting locomotor-like activity was induced by bath-application of aCSF (artificial cerebrospinal fluid) mixed with N-methyl-DL-aspartate (NMA; 10 μM), serotonin (5HT; 5 μM), and dopamine (DA; 50 μm) (all purchased from Sigma-Aldrich, France).

For the elemental oscillator. Neuron N1 (corresponds to neuron L3 in **Figure 1B**) is connected to neuron N2 (corresponds to neuron R3 in **Figure 1B**) by an inhibitory synapse with a synaptic weight of −7 and a percentage dissipation of 12%. The $I_{bias}$ current is equal to 7 for both neurons.

**Figure 7** shows that the digital system fits the biological recordings of the newborn rat spinal cord. The period and duty cycle of the bursting activity are the same, confirming that the digital system was suitable for hybrid experiments. Instead of using pharmacological stimulation, the digital board will be used in the near future to create a hybrid experiment involving the *ex vivo* spinal cord and the digital CPGs. A closed-loop is also possible

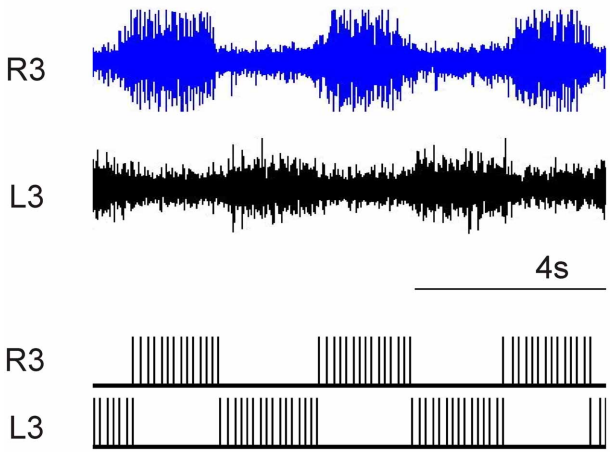



**FIGURE 7 | Comparison of pharmacological *in-vitro* spinal cord with digital CPG.**

thanks to the possibility of changing the mean period of bursting activity by modifying a single parameter ($\tau_{reg}$).

## DISCUSSION

One key step in designing a neuroprosthesis is to produce a large, resource-frugal biomimetic SNN. A biologically realistic CPG (i.e., the leech heartbeat system neural network) was implemented with a minimum resource cost in terms of neuron model, while maintaining its biomimetic activity, as shown in the Results. The first step was to model the biological leech heartbeat system using a single, segmental CPG. The next stage was to choose an efficient neuron model that required few resources for its digital implementation but remained biorealistic enough to match the behavior of biological cells. The topology and hardware implementation of a single neuron were then extended to form a neuron computation core built into a large-scale neural network: 240 CPGs on a Spartan6 FPGA board. Furthermore, the new synaptic model proposed reproduced the activity-dependent depression phenomenon, which had only previously been described in biology literature. The architecture of the entire real-time systemwas described in detail. Finally, the system was validated by several experiments comparing both elemental and segmental oscillators with biological data, and comparing the segmental oscillator with *ex vivo* rat spinal cord stimulated by pharmacological solutions.

The short-term prospect of this work is to improve the stability of the system using another neuron model. Currently our work is focused on the quartic model (Touboul, 2009), which is more stable than the Izhikevich one and also requires few resources. As described in **Table 2**, this system is subject to variations in duty cycle and mean period, likely to be reduced by using the new model. However, these variations also exist in biology, so it is necessary to study the actual effect of these variations in the biological system to determine whether they should be eliminated or not.

In the medium term, this system will be included in a hybrid experiment using an *ex vivo* rat spinal cord. The experiment board includes several modules, including an MEA (Micro-Electrode Array) and spike detection block, to detect and record neural activity in the spinal cord. All these modules, together with the CPG network, will be implemented in the same FPGA. Our neurophysiologist colleagues will identify the best spinal cord sites to stimulate and record bursting activity. These sites will be hybridized to the output of the artificial CPG described in this paper and, in turn, its activity will drive the various ventral root outputs of the spinal cord into full locomotor-like activity. These future experiments aim to demonstrate that hybrid artificial/biological networks provide possible solutions for rehabilitating lost central nervous system function.

Our CPG network could be also used to study the locomotion of different animals. Indeed, according to Ijspeert (2001), the locomotion activity of a salamander requires 40 CPGs, so the 240 CPGs implemented on the Spartan 6 digital board would be suitable for studying more complex locomotion. Our system will be used in a closed-loop system with different sensors and actuators.

## ACKNOWLEDGMENTS

This work is supported by the European Union's Seventh Framework Programme (ICT-FET FP7/2007-2013, FET Young Explorers scheme) under grant agreement n° 284772 BRAIN BOW (www.brainbowproject.eu).

## REFERENCES

Amari, S. (1972). Characteristic of the random nets of analog neuron-like elements. *IEEE Trans. Syst.Man Cybern.* 2, 643–657. doi: 10.1109/TSMC.1972.4309193

Angstadt, J. D., and Calabrese, R. L. (1991). Calcium currents and graded synaptic transmission between heart interneurons of the leech. *J. Neurosci.* 11, 746–759.

Barron-Zambrano, J. H., and Torres-Huitzil, C. (2013). FPGA implementation of a configurable neuromorphic CPG-based locomotion controller. *Neural Netw.* 45, 50–61. doi: 10.1016/j.neunet.2013.04.005

Barron-Zambrano, J. H., Torres-Huitzil, C., and Girau, B. (2010). FPGA-based circuit for central pattern generator in quadruped locomotion. *Aust. J. Intell. Inform. Process. Syst.* 12, 24–29.

Ben-Ari, Y., Khazipov, R., Leinekugel, X., Caillard, O., and Gaiarsa, J. L. (1997). GABAA, NMDA and AMPA receptors: a developmentally regulated 'ménage à trois'. *Trends Neurosci.* 20, 523–529. doi: 10.1016/S0166-2236(97)01147-8

Brainbow. (2012). *Brainbow Project European Union's Seventh Framework Programme (ICT-FET FP7/2007-2013, FET Young Explorers scheme) Under Grant Agreement n° 284772*. Available online at: www.brainbowproject.eu

Brown, T. (1914). On the nature of the fundamental activity of the nervous centres; together with an analysis of the conditioning of rhythmic activity in progression and a theory of the evolution of function in the nervous system. *J. Physiol.* 48, 18–46.

Calabrese, R. L. (1995). "Half-center oscillators underlying rhythmic movements," in *The Handbook of Brain Theory And Neural Networks*, ed M. A. Arbib (Cambridge, MA: MIT Press), 444–447.

Calabrese, R. L., Angstadt, J., and Arbas, E. (1989). A neural oscillator based on reciprocal inhibition. *Perspect. Neural Syst. Behav.* 10, 33–50.

Cassidy, A., and Andreou, A. G. (2008). "Dynamical digital silicon neurons," in *IEEE Biomedical Circuits and Systems Conference, BioCAS 2008,* 289–292. doi: 10.1109/BIOCAS.2008.4696931

Cohen, A. H., Ermentrout, G. B., Kiemel, T., Kopel, N., Sigvardt, K. A., and Williams, T. L. (1992). Modelling of intersegmental coordination in the lamprey central pattern generator for locomotion. *Trends Neurosci.* 15, 434–438. doi: 10.1016/0166-2236(92)90006-T

Cymbalyuk, G. S., Gaudry, Q., Masino, M. A., and Calabrese, R. L. (2002). Bursting in leech heart interneurons: cell-autonomous and network-based mechanisms. *J. Neurosci.* 22, 10580–10592.

De Schutter, E. (ed.) (2000). *Computational Neuroscience: Realistic Modeling for Experimentalists*. Boca Raton, FL: CRC Press. doi: 10.1201/9781420039290

Garofalo, M., Nieus, T., Massobrio, P., and Martinoia, S. (2009). Evaluation of the performance of information theory-based methods and cross-correlation to estimate the functional connectivity in cortical networks. *PLoS ONE* 4:e6482. doi: 10.1371/journal.pone.0006482

Grassia, F., Buhry, L., Levi, T., Tomas, J., Destexhe, A., and Saighi, S. (2011). Tunable neuromimetic integrated system for emulating cortical neuron models. *Front. Neurosci.* 5:134. doi: 10.3389/fnins.2011.00134

Guerrero-Riberas, R., Morrison, A., Diesmann, M., and Pearce, T. (2006). Programmable logic construction kits for hyper-real-time neuronal modeling. *Neural Comput.* 18, 2651–2679. doi: 10.1162/neco.2006.18.11.2651

Hill, A. A., Lu, J., Masino, M. A., Olsen, O. H., and Calabrese, R. L. (2001). A model of a segmental oscillator in the leech heartbeat neuronal network. *J. Comput. Neurosci.* 10, 281–302. doi: 10.1023/A:1011216131638

Hochberg, L. R., Bacher, D., Jarosiewicz, B., Masse, N. Y., Simeral, J. D., Vogel, J., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature* 485, 372–375. doi: 10.1038/nature11076

Hochberg, L. R., Serruya, M. D., Friehs, G. M., Mukand, J. A., Saleh, M., Caplan, A. H., et al. (2006). Neuronal ensemble control of prosthetic devices by a human with tetraplegia. *Nature* 442, 164–171. doi: 10.1038/nature04970

Hodgkin, A. L., and Huxley, A. F. (1952). A quantitative description of membrane current and its applications to conduction and excitation in nerve. *J. Physiol.* 117, 500–544.

Hooper, S. (2000). Central pattern generators. *Curr. Biol.* 10, 176–177. doi: 10.1016/S0960-9822(00)00367-5

Ijspeert, A. (2001). A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *J. Biol. Cybern.* 84, 331–348. doi: 10.1007/s004220000211

Ijspeert, A. (2008). Central pattern generators for locomotion control in animals and robots: a review. *J. Neural Netw.* 21, 642–653. doi: 10.1016/j.neunet.2008.03.014

Ijspeert, A., Crespi, A., Ryczko, D., and Cabelguen, J. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science* 315, 1416–1420. doi: 10.1126/science.1138353

Indiveri, G. (2007). Synaptic plasticity and spike-based computation in VLSI networks of integrate-and-fire neurons. *Neural Inform. Process. Lett. Rev.* 11, 135–146.

Indiveri, G., Linares-Barranco, B., Hamilton, T., Van Schaik, A., Etienne-Cummings, R., and Delbruck, T. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons. *IEEE Trans. Neural Netw.* 15, 1063–1070. doi: 10.1109/TNN.2004.832719

Jezzini, S., Hill, A. A., Kuzyk, P., and Calabrese, R. L. (2004). Detailed model of intersegmental coordination in the timing network of the leech heartbeat central pattern generator. *J. Neurophysiol.* 91, 958–977. doi: 10.1152/jn.00656.2003

Jung, R., Brauer, E. J., and Abbas, J. J. (2001) Real-time interaction between a neuromorphic electronic circuit and the spinal cord. *IEEE Trans. Neural Syst. Rehabil. Eng.* 9, 319–326. doi: 10.1109/7333.948461

Krahl, B., and Zerbst-Boroffka, I. (1983). Blood pressure in the leech. *J. Exp. Biol.* 107, 163–168.

Le Masson, G., Renaud-Le Masson, S., Debay, D., and Bal, T. (2002). Feedback inhibition controls spike transfer in hybrid thalamic circuits. *Nature* 417, 854–858. doi: 10.1038/nature00825

Lee, Y. J., Lee, J., Kim, K. K., Kim, Y. B., and Ayers, J. (2007). Low power cmos electronic central pattern generator design for a biomimetic underwater robot. *Neurocomputing* 71, 284–296. doi: 10.1016/j.neucom.2006.12.013

Lewis, M. A., Hartmann, M. J., Etienne-Cummings, R., and Cohen, A. H. (2001). Control of a robot leg with an adaptive VLSI CPG chip. *Neurocomputing* 38–40, 1409–1421. doi: 10.1016/S0925-2312(01)00506-9

Linares-Barranco, B., Sánchez-Sinencio, E., Rodríguez-Vázquez, A., and Huertas, J. L. (1993). "CMOS Analog Neural Network Systems Based on Oscillatory Neurons," in *Silicon Implementation of Pulse Coded Neural Networks*, eds M. Zaghloul, J. Meador, and R. Newcomb (Boston: Kluwer Academic Publishers), 199–247.

Marder, E., and Bucher, D. (2001). Central pattern generators and the control of rhythmic movements. *Curr. Biol.* 11, 986–996. doi: 10.1016/S0960-9822(01)00581-4

Marom, S., and Shahaf, G. (2002) Development, learning and memory in large random networks of cortical neurons: lessons beyond anatomy. *Q. Rev. Biophys.* 35: 63–87. doi: 10.1017/S0033583501003742

Matsuoka, K. (1987). Mechanism of frequency and pattern control in the neural rhythm generators. *Biol. Cybern.* 56, 345–353. doi: 10.1007/BF00319514

Nadim, F., Olsen, O. H., De Schutter, E., and Calabrese, R. L. (1995). Modeling the leech heartbeat elemental oscillator. *J. Comput. Neurosci.* 2, 215–235. doi: 10.1007/BF00961435

Nakada, K. (2003). An analog cmos central pattern generator for interlimb coordination in quadruped locomotion. *IEEE Tran. Neural Netw.* 14, 1356–1365. doi: 10.1109/TNN.2003.816381

Nicolelis, M. A. L., and Lebedev, M. A. (2009). Principles of neural ensemble physiology underlying the operation of brain-machine interfaces. *Nat. Rev. Neurosci.* 10, 530–540. doi: 10.1038/nrn2653

Norris, B., Weaver, A., Wenning, A., Garcia, P., and Calabrese, R. L. (2007). A central pattern generator producing alternative outputs: phase relations of leech hear motor neurons with respect of premotor synaptic input. *J. Neurophysiol.* 98, 2983–2991. doi: 10.1152/jn.00407.2007

Olsen, O. H., and Calabrese, R. L. (1996). Activation of intrinsic and synaptic currents in leech heart interneurons by realistic waveforms. *J. Neurosci.* 16, 4958–4970.

Olsen, O. H., Nadim, F., and Calabrese, R. L. (1995). Modeling the leech heartbeat elemental oscillator: II. Exploring the parameter space. *J. Comput. Neurosci.* 2, 237–257. doi: 10.1007/BF00961436

Olypher, A., Cymbalyuk, G., and Calabrese, R. L. (2006). Hybrid systems analysis of the control of burst duration by low-voltage-activated calcium current in Leech heart interneurons. *J. Neurophysiol.* 96, 2857–2867. doi: 10.1152/jn.00582.2006

Rice, K. L., Bhuiyan, P. A., Taha, T. M., Vutsinas, C. N., and Smith, M. C. (2009). "FPGA Implementation of Izhikevich Spiking Neural Network for Character Recognition," in *International Conference on Reconfigurable Computing and FPGAs*, (Cancun), 451–456.

Serrano-Gotarredona, R., Oster, M., Lichtsteiner, P., Linares-Barranco, A., Paz-Vicente, R., Gómez-Rodríguez, F., et al. (2009). CAVIAR: A 45k-Neuron, 5M-Synapse, 12G-connects/sec AER Hardware Sensory-Processing-Learning-Actuating System for High Speed Visual Object Recognition and Tracking. *IEEE Trans. Neural Netw.* 20, 1417–1438. doi: 10.1109/TNN.2009.2023653

Simoni, M., and DeWeerth, S. (2007). Sensory feedback in a half-center oscillator model. *IEEE Trans. Biomed. Eng.* 54, 193–204. doi: 10.1109/TBME.2006.886868

Simoni, M., Cymbalyuk, G., Sorensen, M., R. Calabrese, R. L., and DeWeerth, S. (2004). A multi-conductance silicon neuron with biologically matched conductances. *IEEE Trans. Biomed. Eng.* 51, 342–354. doi: 10.1109/TBME.2003.820390

Sorensen, M., DeWeerth, S., Cymbalyuk, G., and Calabrese, R. L. (2004). Using a hybrid neural system to reveal regulation of neuronal network activity by an intrinsic current. *J. Neurosci.* 24, 5427–5438. doi: 10.1523/JNEUROSCI.4449-03.2004

Still, S., and Tilden, M. W. (1998). "Controller for a four legged walking machine," in *Neuromorphic Systems: Engineering Silicon from Neurobiology*, eds L. S. Smith and A. Hamilton, (World Scientific Publishing Co Pte Ltd), 138–148 doi: 10.1142/9789812816535_0012

Still, S., Hepp, K., and Douglas, R. J. (2006). Neuromorphic walking gait control. *IEEE Trans. Neural Netw.* 17, 496–508. doi: 10.1109/TNN.2005.863454

Tabak, J., Senn, W., O'Donovan, M., and Rinzel, J. (2000). Modeling of spontaneous activity in developing spinal cord using activity-dependent depression in an excitatory network. *J. Neurosci.* 20, 3041–3056.

Torres-Huitzil, C., and Girau, B. (2008). Implementation of central pattern generator in an FPGA-based embedded system. *18th International Conference on Artificial Neural Networks,* Vol. 5164, 179–187. doi:10.1007/978-3-540-87559-8_19

Touboul, J. (2009). Importance of the cutoff value in the quadratic adaptive integrate-and-fire model. *Neural Comput.* 21, 2114–2122. doi: 10.1162/neco.2009.09-08-853

Van Der Pol, B. (1928). The heartbeat considered as a relaxation oscillation, and an electrical model of the heart. *Philos. Mag.* 6, 763–775.

Vogelstein, R. J., Tenore, F., Etienne-Cummings, R., Lewis, M. A., and Cohen, A. H. (2006). Dynamic control of the central pattern generator for locomotion. *Biol. Cybern.* 95, 555–566. doi: 10.1007/s00422-006-0119-z

Wijekoon, J., and Dudek, P. (2008). Compact silicon neuron circuit with spiking and bursting behavior. *Neural Netw.* 21, 524–534. doi: 10.1016/j.neunet.2007.12.037

Zumsteg, Z., Kemere, C., O'Driscoll, S., Santhanam, G., Ahmed, R. E., Shenoy, K. V., et al. (2005). Power feasibility of implantable digital spike sorting circuits for neural prosthetic systems. *IEEE Trans. Neural Syst. Rehabil. Eng.* 13, 272–279. doi: 10.1109/TNSRE.2005.854307

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 06 August 2013; accepted: 29 October 2013; published online: 21 November 2013.*

*Citation: Ambroise M, Levi T, Joucla S, Yvert B and Saïghi S (2013) Real-time biomimetic Central Pattern Generators in an FPGA for hybrid experiments. Front. Neurosci. **7**:215. doi: 10.3389/fnins.2013.00215*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

METHODS ARTICLE
published: 22 January 2014
doi: 10.3389/fnins.2013.00276

# Dynamic neural fields as a step toward cognitive neuromorphic architectures

***Yulia Sandamirskaya****

*Chair for Theory of Cognitive Systems, Institute for Neural Computation, Ruhr-University Bochum, Bochum, Germany*

***Edited by:***
*André Van Schaik, The University of Western Sydney, Australia*

***Reviewed by:***
*Dylan R. Muir, University of Basel, Switzerland*
*Jonathan Binas, University of Zurich and ETH Zurich, Switzerland*

****Correspondence:***
*Yulia Sandamirskaya, Chair for Theory of Cognitive Systems, Institute for Neural Computation, Ruhr-University Bochum, 150, 44780 Bochum, Germany*
*e-mail: yulia.sandamirskaya@ini.rub.de*

Dynamic Field Theory (DFT) is an established framework for modeling embodied cognition. In DFT, elementary cognitive functions such as memory formation, formation of grounded representations, attentional processes, decision making, adaptation, and learning emerge from neuronal dynamics. The basic computational element of this framework is a Dynamic Neural Field (DNF). Under constraints on the time-scale of the dynamics, the DNF is computationally equivalent to a soft winner-take-all (WTA) network, which is considered one of the basic computational units in neuronal processing. Recently, it has been shown how a WTA network may be implemented in neuromorphic hardware, such as analog Very Large Scale Integration (VLSI) device. This paper leverages the relationship between DFT and soft WTA networks to systematically revise and integrate established DFT mechanisms that have previously been spread among different architectures. In addition, I also identify some novel computational and architectural mechanisms of DFT which may be implemented in neuromorphic VLSI devices using WTA networks as an intermediate computational layer. These specific mechanisms include the stabilization of working memory, the coupling of sensory systems to motor dynamics, intentionality, and autonomous learning. I further demonstrate how all these elements may be integrated into a unified architecture to generate behavior and autonomous learning.

**Keywords: dynamic neural fields, cognitive neuromorphic architecture, soft winner-take-all, autonomous learning, neural dynamics**

## 1. INTRODUCTION

Organisms, such as animals and humans, are remarkable in their ability to generate behavior in complex and changing environments. Their neural systems solve challenging problems of perception and movement generation in the real world with a flexibility, adaptability, and robustness that surpasses the capabilities of any technical system available today. The question of how biological neural systems cope with the complexity and dynamics of real-world environments and achieve their behavioral goals, does not have a simple answer. Processes such as memory formation, attention, adaptation, and learning all play crucial roles in the biological solution to the problem of behavior generation in real-world environments. Understanding how these processes are realized by the neural networks of biological brains is at the core of understanding biological cognition and building cognitive artifacts that successfully contend with real world constraints.

The field of neuromorphic engineering may contribute to the ambitious goal of understanding these cognitive processes by offering platforms in which neural models may be implemented in hardware using the VLSI (Very Large Scale Integration) technology. The analog neuromorphic hardware shares several properties with biological neural networks such as the presence of the inherent noise, the potential mismatch of computing elements, constraints on connectivity, and a limited number of learning mechanisms. Apart from these shared constraints, artificial and biological neural networks also maintain the advantages of pervasive parallel computation, redundant systems to handle sensory and motor noise, and low power consumption. Success in the implementation of cognitive models on neuromorphic hardware may lead to breakthroughs both in understanding the neural basis of human cognition and in the development of performant technical systems (robots) acting in real-world environments.

VLSI technology allows one to implement large neural networks in hardware by configuring the VLSI device to simulate the dynamics and connectivity of a network of spiking neurons. Such networks may be efficiently configured, connected to sensors and motors, and operate in real time (Mead and Ismail, 1989; Indiveri et al., 2009, 2011). However, a challenging question remains: how to develop these neuromorphic systems beyond simple feed-forward reactive architectures toward architectures capable of cognitive behavior?

Soft winner-take-all (WTA) connectivity has been recently proposed as an important milestone on the way toward such functional cognitive neuromorphic systems (Indiveri et al., 2009; Rutishauser and Douglas, 2009). Soft WTA networks are computational elements that are hypothesized to play a central role in cortical processing (Douglas and Martin, 2004; Rutishauser and Douglas, 2009). Recently, a wide variety of WTA networks of spiking neurons have been implemented in hardware (Indiveri et al., 2001; Abrahamsen et al., 2004; Oster and Liu, 2004; Indiveri et al., 2009). These initial architectures have made use of WTA connectivity to enable the effective processing of sensory information (Liu and Delbruck, 2010) and the implementation of finite state machines (Neftci et al., 2013). Soft WTAs introduce

a *cognitive* layer to the neuromorphic hardware systems, which enables reliable processing on unreliable elements (Neftci et al., 2013). The WTA networks contribute to making neuromorphic systems more cognitive, because they stabilize localized attractor patterns in neural networks. These stable attractors organize the dynamics of the neural system in a macroscopical way and enable the coupling of the network to sensors and motors despite noise, fluctuations, and neural mismatch. WTA connectivity therefore introduces macroscopic neural dynamic states which may persist long enough to interact with other parts of the neural-dynamic architecture, thus moving neuromorphic systems beyond mere reactive behavior.

However, there are still open questions on the way toward cognitive processing with hardware WTAs. The first question concerns representational power: How can we add contents to the state in a WTA network and link this network state to perceptual or motor variables? How can the system represent associations and concepts such as "a red ball on the table" or "a hand moving toward an object" in this framework? The second line of open questions concerns movement generation and the motor behavior: How should the system represent and control movements in this framework? How should it decide when to initiate or terminate a movement? Finally, questions regarding learning also arise: How may a system learn WTA connectivity of its neural network? How may the system learn the connections between WTA networks in a complex architecture? Such questions are often addressed in the fields of psychophysics, cognitive science, and artificial intelligence, but the proposed models and solutions are often not compatible with neural implementations. Here, I propose that Dynamic Field Theory (DFT) is a framework which may make such cognitive models feasible for neuromorphic implementation because it formulates the principles of cognitive representations and processes in a language compatible with neuromorphic soft WTA architectures. Identifying the computational and architectural principles underlying these cognitive models may facilitate the development of large-scale neuromorphic cognitive systems.

DFT is a mathematical and conceptual framework which was developed to model embodied human cognition (Schoner, 2008). DFT is an established framework in modeling many aspects of human cognition and development including visual and spatial working memory, object and scene representation, sequence generation, and spatial language (Johnson et al., 2008). DFT cognitive models have been used to control robots and demonstrate that the developed architectures can function autonomously in the real-world (Erlhagen and Bicho, 2006; Sandamirskaya et al., 2013). DFT builds on Dynamic Neural Fields (DNFs), which, as I will discuss in the Methods section, are analogous to soft WTAs in their dynamics and lateral connectivity within networks (Neftci et al., 2010). Accordingly, their dynamical and structural principles may be applied to the design of neuromorphic WTA architectures.

In this paper, I discuss computational and architectural principles recently developed in DFT that may be applied to WTA neuromorphic networks. These principles can increase the representational power and autonomy of such networks, and thus contribute to the greater scalability and robustness of neuromorphic architectures. In particular, these principles enable the coupling of DNFs of differing dimensionality, the coupling of the architectures to sensors and motors, cognitive control over behavior, and autonomous learning. On a simple exemplar architecture, I demonstrate how these principles enable autonomous behavior and learning in a neural-dynamic system coupled to real-world sensors and motors. I also discuss the possibility of implementing DNF architectures in neuromorphic hardware.

## 2. MATERIALS AND METHODS

### 2.1. DYNAMIC NEURAL FIELDS: BASIC DYNAMICS AND INSTABILITIES

A DNF is a mathematical description of activation dynamics of a neuronal population in response to certain parameters of the agent's behavioral state. The behavioral parameters, such as a perceptual feature, location, or motor control variable, span dimension(s), over which the DNFs are defined (Schoner, 2008). The dynamics of DNF may be mathematically formalized as a differential equation, Equations (1–3), which was first analyzed by Amari (1977), and used to model neuronal dynamics on a population level (Wilson and Cowan, 1973; Grossberg, 1988; Ermentrout, 1998).

$$\tau \dot{u}(x,t) = -u(x,t) + h + \int f\big(u(x',t)\big)\omega(x-x')dx' + S(x,t), \quad (1)$$

$$\omega(x-x') = c_{exc}\exp\left[-\frac{(x-x')^2}{2\sigma_{exc}^2}\right] - c_{inh}\exp\left[-\frac{(x-x')^2}{2\sigma_{inh}^2}\right], \quad (2)$$

$$f\big(u(x,t)\big) = \frac{1}{1+\exp[-\beta u(x,t)]}. \quad (3)$$

In Equation (1), $u(x, t)$ is the activation of the DNF over dimension $x$, to which the underlying neuronal population is responsive. $h$ is a negative resting level and $S(x, t)$ is an external input driving the DNF. The lateral interactions in DFT are shaped by a symmetrical homogeneous interaction kernel, Equation (2), with a short-range excitation and a long-range inhibition (Ellias and Grossberg, 1975); $\sigma_{exc}$, $\sigma_{inh}$, $c_{exc}$, and $c_{inh}$ are the width and the amplitude of the excitatory and the inhibitory parts of the interaction kernel respectively. The sigmoidal non-linearity, Equation (3), shapes the output of the DNF in such a way, that only sufficiently activated field locations contribute to neural interactions; $\beta$ determines the slope of the sigmoid.

An example of how a DNF may be linked to the activity of a neuronal population is shown in **Figure 1**: First, each neuron in the population contributes its tuning curve in respect to the behavioral parameter of interest as a (virtual) input to the DNF. The tuning curve is determined as a dependence of the mean firing rate or the action potential of the neuron on the value of the behavioral parameter (**Figure 1A**). Second, the tuning curves of the neurons in the population are summed, each weighted by the current activation level (e.g., mean firing rate) of the respective neuron. The resulting Distribution of Population Activity [DPA, introduced by Bastian et al. (2003) to derive a DNF description of neuronal data on movement preparation in studies of reaching movements in monkeys] represents the overall activity of the selected neuronal population in response to a given stimulus or state of the behaving neural

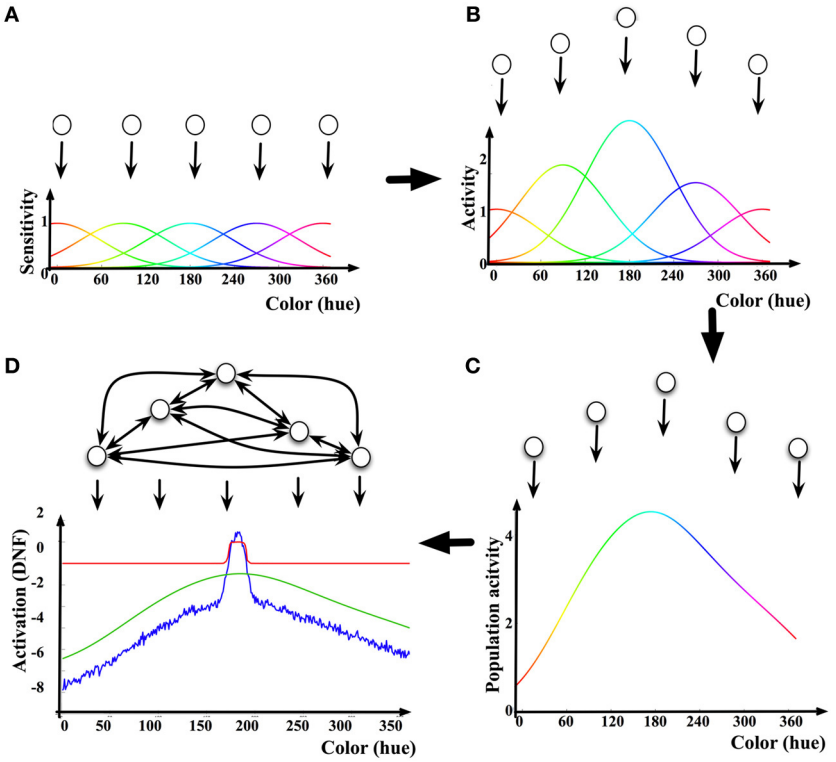



**FIGURE 1 | Illustration of the relationship between neuronal activity and a DNF. (A)** Five exemplar "neurons" (neuronal populations) and their tuning curves in the color dimension. **(B)** The tuning curves are scaled by the mean firing rate (activation) of the neurons. **(C)** By summing the scaled tuning curves, the Dynamic Population Activity [DPA, Bastian et al. (2003)] curve in response to a given color stimulus is constructed. **(D)** The DNF dynamics adds lateral interactions between neurons according to Equation (1). The activation of the DNF is shown as a blue line, the red line shows the output (sigmoided activation) of the DNF, the green line is the DPA [same as in **(C)**].

system (**Figures 1B,C**). Finally, the neurons in the population are assumed to be interconnected so that the nearby (in the behavioral space) locations exert excitatory influence on each other, and the far-off locations inhibit each other ("on-center, off-surround" connectivity Ellias and Grossberg, 1975). The resulting activation function $u(x, t)$, is activation of the DNF. A sigmoidal non-linearity $f(u(x, t))$, shapes the output of the DNF, which impacts on the DNF itself through the lateral connections and on the other parts of the neural architecture connected to this DNF.

The pattern of lateral connectivity of DNFs results in existence of a localized-bump solution in their dynamics (**Figure 1D**), which is at the core of the properties of DNFs to exert elementary cognitive functions, discussed further. In the realm of modeling human cognition, activity peaks bridge the low-level, graded sensory-motor representations to categorical, symbol-like representations. The localized (and stabilized, i.e., sustainable over macroscopical time intervals) representation facilitates perception, action generation, and learning.

The connectivity pattern within DNF also makes it a soft WTA architecture. Indeed, a WTA-connected network may be formalized in terms of two neuronal populations, an excitatory and an inhibitory one (Rutishauser and Douglas, 2009):

$$\tau\dot{x}_i = -x_i + f\left(I_i + \alpha x_i - \beta_1 x_N - T_i\right) \quad (4)$$

$$\tau\dot{x_N} = -x_N + f\left(\beta_2 \sum_{j=1}^{N-1} x_j - T_N\right). \quad (5)$$

In Equations (4, 5), the excitatory population of nodes (neurons) $x_i$ has an attractor dynamics driven by the external input, $I_i$, the resting level potential, $T_i$, the self-excitatory term with strength $\alpha$, and the inhibitory term with strength $\beta_1$. The inhibition is shared by all excitatory nodes and is provided by the inhibitory neuron, $x_N$, which also follows an attractor dynamics, driven by activity in the excitatory population and the resting level $T_N$.

In these equations, the excitation constant, $\alpha$, is analogous to the excitatory part of the interaction kernel of a DNF, $c_{exc}$ in Equation (2), and the strength of the coupling of the inhibitory population onto the excitatory population, $\beta_1$, corresponds to the inhibitory part of the interaction kernel with the strength $c_{inh}$. In the DNF equation, the inhibition is coupled into the field's dynamics without delay, which is present in the WTA network of Equations (4, 5).

In several studies on development of working memory and spatial cognition in infants and toddlers, a more general DNF equation is used, in which a separate inhibitory layer is introduced [e.g., Johnson et al. (2006, 2008)]. Separate inhibitory layer

leads to a delay in the inhibitory interaction among neural field's locations, which allows to model fine-grained effects in competition among items in the working memory depending on timing of their presentation. The separate inhibitory layer is also used to create a shared inhibition among perceptual and working memory neural fields, which plays a critical role in a change detection process.

When DNF architectures are created to generate behavior in an embodied agent, the DFT postulates that only attractor states impact on the behavior of the controlled agent and thus the dynamics of DNFs is typically tuned to relax as fast as possible to the attractor state. Since this holds for the separate inhibitory layer, the presence of the delay in the inhibitory dynamics is negligible in robotic DNF architectures. For this reason, when DNFs are used to control robots, only single-layer dynamics are used, where inhibition and excitation are integrated in a single equation. Since WTA dynamics in Equations (4,5) is a more general formulation than DNFs, discussed in this paper, the equivalence between these two mathematical structures requires a constraint on the timing constant of the inhibitory population, which needs to be faster than the timing constant of the excitatory population, which in its turn is faster than the dynamics of sensor inputs to the field.

The stable localized activity peak solution of the DNF dynamics is the DNF variant of soft-WTA behavior. Intuitively, the short-range excitatory interactions stabilize the peak solution against decay and the long-range inhibitory interactions stabilize peaks against spread by diffusion. The sites of the DNF, which have above zero activity, are the "winners" of the DNF dynamics. The sigmoidal non-linearity increases stability of the localized peak. The important contribution of DFT to understanding the dynamics of soft WTA networks is the characterization of stable states and instabilities between them based on the analysis of Equation (1) (Amari, 1977; Schoner, 2008; Sandamirskaya et al., 2013):

- The *detection instability* separates a quiescent state of the DNF from an active state. In the quiescent state, the inputs are not strong enough to collectively drive the DNF over the activation threshold. The DNF produces no output in this state, it is invisible for the down-stream structures, driven by the DNF. To the contrary, when inputs are strong enough to drive the field over the activation threshold in one or several locations, an activity peak emerges in the field, which provides input to the down-stream structures, or the motor system.
- The DNF's inputs may drive the field over the threshold at several locations. In this case, the field may build several activation peaks or it may select and amplify activity at one location only, depending on the spread of the lateral inhibition. In the latter case, a *selection instability* separates an inactive state from an activated state of the DNF dynamics.
- If the lateral interactions are strong enough, a peak in the DNF may be sustained even if the input, which initiated the peak, ceases. This *working memory instability* separates the state of the field with no activation from the state, in which an external inhibiting input is needed to deactivate the field.
- A negative external input or a decrease of the excitatory input may lead to an extinction of the activity peak. This causes a *reverse detection instability*, or forgetting instability, which separates an active state from the quiescent state.

The localized-peak stable states and instabilities between them form the basis for more complex DNF architectures, just as WTA networks form the basis for state-based spiking network architectures. In the following, I present additional components in the DFT, which may be translated into VLSI WTA networks and enhance their scalability and autonomy.

### 2.2. COUPLING DYNAMIC NEURAL FIELDS TO SENSORY SYSTEMS

**Figure 2** shows a small DNF architecture, which exemplifies the coupling structures in DFT: coupling the DNFs to each other, to sensors, and to motors. Here, I will introduce the principles behind these coupling structures, while referring to the figure for a concrete example. Overall, the simple system in the **Figure 2** performs saliency computations based on color- or spatial cues by means of neuronal dynamics (DNF or WTA computation) and will be a building block, used in the example, presented in Section 3.

In **Figure 2**, a two-dimensional perceptual color-space DNF receives input from the robotic camera. Camera input to this DNF is constructed in the following way. The raw hue value of every pixel corresponds to the vertical location in the DNF, the location of the pixel on the horizontal axis of the image to the horizontal location in DNF, and the saturation value of the pixel to the intensity value of the sensory input. Thus, the input to the perceptual DNF is an unsegmented stream of color-space associations. If the input is strong enough to pass the activation threshold and is localized in space, a peak of suprathreshold activity evolves, which represents the perceived object. In **Figure 2A**, the camera input is not sufficient to activate the perceptual DNF—only subthreshold hills of activity represent the four salient objects in the visual scene. However, when the perceptual DNF receives an additional input, which specifies the color of the target object and which overlaps with one of the subthreshold hills, an activity peak evolves in the perceptual DNF and signals the selection of an object of interest (**Figure 2B**). The additional input arrives from another—color—DNF, which is coupled to the perceptual DNF, as described in Section 2.3.

Another example of coupling a sensor to the DNF is shown in **Figure 3**. Here, a neuromorphic embedded Dynamic Vision Sensor [eDVS, Conradt et al. (2009)] drives the perceptual DNF. In the eDVS, each pixel sends an event when it sensed luminance changes. Consequently, the sensor naturally detects moving objects. If the object of interest is not moving too fast relative to the motor capabilities of the agent, the perceptual DNF may be used to stabilize the representation of the instantaneous position of the moving object in order to use this position to parametrize the motor action (e.g., to direct the agent's gaze toward the object). If the object is moving too fast for the behaving system, a predictive mechanism needs to be built into the DNF's dynamics (Erlhagen and Bicho, 2006).

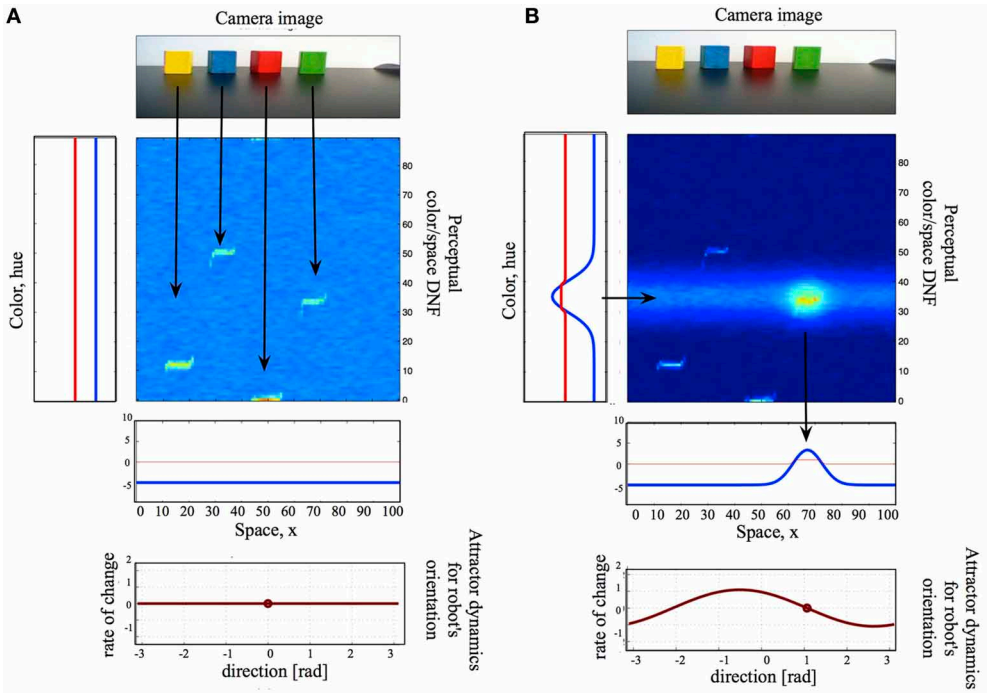



**FIGURE 2 | A small DNF architecture, which consists of a two-dimensional color-space DNF (center), one-dimensional color- and space- DNFs, coupled to the perceptual DNF, a camera input (top), and an attractor motor dynamics (bottom). (A)** The camera input alone is not sufficient to activate the perceptual DNF, the system is quiescent and produces neither output nor behavior. **(B)** A color cue creates an activity peak in the color DNF over the hue value of the named color. This activity peak is projected onto the 2D perceptual DNF as a subthreshold activity ridge, which overlaps with the camera input for the green object. The resulting activity peak in the 2D DNF provides input to the spatial DNF, which, in its turn, sets an attractor for the motor dynamics. The latter drives the motor system of the agent, initiating an overt action.

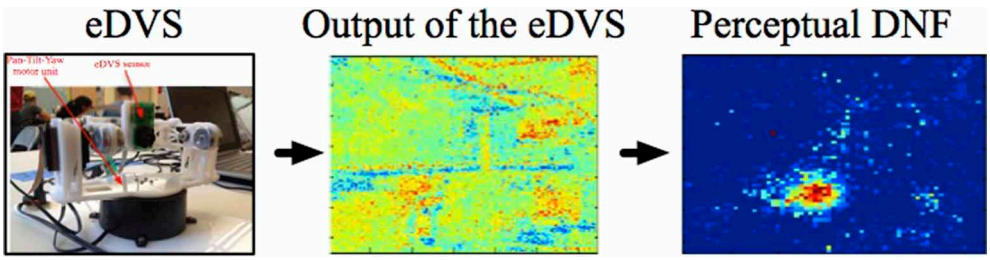



**FIGURE 3 | The neuromorphic Dynamic Vision Sensor [eDVS, Conradt et al. (2009)] on a pan-tilt unit, the output of the eDVS, integrated over a time window of 100 ms, and the instantaneous output of the perceptual DNF.** The perceptual DNF enhances the perceptual input in a selected region (which reached the activation threshold first), and inhibits all other locations in the visual array, performing an elementary object segregation operation.

### 2.3. DYNAMIC NEURAL FIELDS OF HIGHER DIMENSIONALITY AND COUPLINGS

A single DNF describes activation of a neuronal population, which is sensitive to a particular behavioral parameter. Activity of any behaving agent, however, is characterized by many such parameters from different sensory-motor modalities. In DFT, there are two ways to represent such multimodality of a system: multidimensional DNFs and coupled DNFs.

The multidimensional DNFs are sensitive to combinations of two or several behavioral parameters. The perceptual color-space field in **Figure 2** is an example of a two-dimensional DNF, which may be activated by combinations of color and locations in space. Such multidimensional DNFs have typically low dimensionality.

Two DNFs of the same or different dimensionality may be coupled with weighted connections, according to Equation (7) (Zibner et al., 2011).

$$\tau \dot{u}_1(x,t) = -u_1(x,t) + h + \int f\big(u_1(x',t)\big)\omega(x-x')dx' + S(x,t), \quad (6)$$

$$\tau \dot{u}_2(y,t) = -u_2(y,t) + h + \int f\big(u_2(y',t)\big)\omega(y-y')dy' + W(x,y) \times f(u_1(x,t)). \quad (7)$$

Here, $u_1(x, t)$ and $u_2(y, t)$ are two DNFs, defined over two different behavioral spaces, $x$ and $y$. The first DNF provides an additive input to the second DNF through the (adaptable) connection weights matrix, $W(x, y)$, which maps the dimensions of the space $x$ onto dimensions of the space $y$.

For example, the one-dimensional color DNF in **Figure 2** represents distributions in the color (hue) dimension. This DNF projects its activation onto the two-dimensional color-space DNF. In particular, since the two DNFs share one dimension (color), the output of the one-dimensional DNF is copied along the not shared dimension (space) of the two-dimensional DNF. This typically results in a ridge-shaped input to the two-dimensional DNF (stamming from the Gaussian shape of the activity peak in the one-dimensional DNF). If this ridge overlaps with a localized subthreshold input in the two-dimensional DNF, an activity peak evolves over the cued (in this case, by color) location (Zibner et al., 2011).

Further, the localized output of the two-dimensional perceptual DNF in **Figure 2** is in its turn projected on a one-dimensional spatial DNF, which represents locations on the horizontal axis of the image plane. This projection may be either a sum or a maximum of the DNF's output in the dimension, not shared between the two DNFs (here, color). An example of an adaptive coupling between DNFs of the same dimensionality is presented in Section 2.6.2.

In terms of WTA network, coupling between two DNFs is equivalent (under constraints, stated in Section 2.1) to two WTA networks, one of which receives output from the other one as an external input, which is mapped through synaptic connections.

### 2.4. COUPLING THE DNF TO ATTRACTOR MOTOR DYNAMICS

In order to close the behavioral loop, DNF architectures have to be coupled to the motor system of a behaving agent. The control of motor actions may be expressed mathematically as an attractor dynamics, where the neural system sets attractors for motor variables, such as position, velocity, or force of the effector. Deviations from the attractor due to an external or an internal perturbation are then actively corrected by the neural controller in the motor system. Such motor attractor dynamics have been probed in control of mobile robots (Bicho and Schoner, 1997) and multi degrees of freedom actuators (Schaal et al., 2003; Iossifidis and Schöner, 2004; Reimann et al., 2011), and also used to model human motor control (Latash et al., 2007).

In order to couple the DNF dynamics to the attractor dynamics for motor control, the space-code representation of the DNF (in terms of locations of activity peaks) has to be mapped onto the rate-code representation of the motor dynamics (in terms of the value of the control variable). **Figure 2** (bottom) and Equation (8–9) show how the space-code of a DNF may be translated into the rate-code of attractor dynamics through a weighted projection to the rate-coding neural node. The weights (or gain field, $\lambda(x)$) of this projection may be subject to learning (or adaptation) (see Section 3).

$$\tau\dot{u}(x,t) = -u(x,t) + h + \int f\big(u(x',t)\big)\omega(x-x')dx' + S(x,t), \quad (8)$$

$$\tau\dot{\phi}(t) = -\phi\int f(u(x,t))dx + \int \lambda(x) f(u(x,t))dx. \quad (9)$$

Here, $u(x, t)$ is a one-dimensional motor DNF, which represents the target values of the motor variable using space coding. $\phi$ is the motor variable, which controls movement of the robot (e.g., velocity, position, force of a motor, or the target elongation of a muscle). This variable follows an attractor dynamics, Equation (9) with an attractor defined by the position of the activity peak in the DNF, $u(x, t)$. This attractor is only turned on when an activity peak is present in the motor DNF. The typical choice for $\lambda(x)$ is $\lambda(x) = cx$, but generally, this factor is subject to a learning (gain adaptation) process (see Section 2.6.3).

In a WTA architecture, the motor variable $\phi$ [Equation (9)] may be implemented as a neural population without lateral connections, which receives input from the a motor WTA [that is analogous to the motor DNF in Equation(8)] through a set of synaptic connections, $\lambda(x)$. This input is summed by the motor variable population. The critical difference of this dynamics to the DNF (or WTA) dynamics is that the motor command is defined by the activity of the population rather than the location of an activity peak in the population (Bicho et al., 2000).

### 2.5. AUTONOMY AND COGNITIVE CONTROL IN DFT

Critically, in order to close the behavioral loop, the cognitive control of the neural architecture is necessary. In particular, the agent that has access to several perceptual and motor modalities has to decide at each point in time, which perceptual input to use to control the motor system and which effector of the motor system to use to achieve a behavioral goal. This problem was addressed recently in DFT in terms of modeling executive control in human cognition (Buss and Spencer, 2012) and in the behavioral organization in robotics (Richter et al., 2012).

The crucial element that gives a neural architecture the desired autonomy of executive control is based on the principle of intentionality (Searle, 1983; Sandamirskaya and Schoner, 2010a). In practice, this principle amounts to a structural extension of DNFs, so that every behavioral state of the system has two components—a representation of an intention, which eventually drives the motor system of the agent, and a representation of the condition-of-satisfaction (CoS), which is activated by the sensory input when the action is finished and which inhibits the respective intention. The CoS DNF is biased, or preshaped, by the intention DNF to be sensitive to particular sensory input, characteristics for the action outcome. This coupling from the intention to the CoS DNF carries a predictive component of the intentional behavior, which may be shaped in a learning process (Luciw et al., 2013). Together, the intention and the CoS comprise an elementary behavior (EB, Richter et al., 2012), which generally has the dynamics of Equations (10).

$$\begin{aligned}\tau\dot{u}_{int}(x,t) &= -u_{int}(x,t) + h + \int f\big(u_{int}(x',t)\big)\omega(x-x')dx' \\ &\quad + S_1(x,t) - c_1\int f(u_{CoS}(y,t))dy, \quad (10)\\ \tau\dot{u}_{CoS}(y,t) &= -u_{CoS}(y,t) + h + \int f\big(u_{CoS}(y',t)\big)\omega(y-y')dy' \\ &\quad + S_2(y,t) + c_2 W(x,y) f(u_{int}(x,t))\end{aligned}$$

Here, $u_{int}(x, t)$ is a DNF which represents possible intentions of the agent. These intentions may be motor or perceptual goals,

which the agent aims to achieve through contact with the environment. For instance, "locate a red object" is a typical perceptual intention, "turn 30 degrees to the left" is an example of a motor intention. $x$ is a perceptual or motor variable, which characterizes the particular intention; $S_1(x, t)$ is an external input which activates the intention. This input may be sensory (condition of initiation) or motivational (task input) (Sandamirskaya et al., 2011). $u_{CoS}(y, t)$ is the condition-of-satisfaction DNF, which receives a localized input from the intention DNF through a neuronal mapping $W(x, y)$ (as introduced in Section 2.3). This input makes the CoS DNF sensitive to a particular part of the sensory input, $S_2(y, t)$, which is characteristic for the termination conditions of the intended perceptual or motor act. The mapping $W(x, y)$ may be learned (Luciw et al., 2013). When the CoS DNF is activated, it inhibits the intention DNF by shifting its resting level below the threshold of the forgetting instability.

The DNF structure of an elementary behavior (EB) further stabilizes the behavioral state of the neural system. Thus, the intentional state of the system is kept active as long as needed to achieve the behavioral goal. The CoS autonomously detects that the intended action is successfully accomplished and inhibits the intention of the EB. Extinction of the previously stabilized intention gives way to the next EB to be activated. With this dynamics, the exact duration of an upcoming action does not need to be represented in advance (and action durations may vary to a large degree in real-world environments). The intentional state will be kept active until the CoS signals that the motor action has reached its goal. This neural-dynamic mechanism of intentionality enables autonomous activation and deactivation of different modalities of a larger neuronal architecture.

Since the intention and the CoS are interconnected DNFs, their WTA implementation may be achieved as described in Section 2.3.

## 2.6. LEARNING IN DFT

The following learning mechanisms are available in the DFT framework.

### 2.6.1. Memory trace of previous activity

The most basic learning mechanism in DFT is the memory trace formation, also called preshape. The memory trace changes the subsequent dynamics of a DNF and thus is considered an elementary form of learning. In neural terms, the memory trace amounts to local increase in excitability of neurons, which may be counterbalanced with homeostatic processes.

Formally, the preshape is an additional layer over the same dimensions as the associated DNF. The preshape layer receives input from the DNF, which is integrated into the preshape dynamics as an attractor that is approached with a time-constant $\tau_l/\lambda_{build}$, Equation (11). This build-up constant is slower than the time-constant of the DNF dynamics. When there is no activity in the DNF, the preshape decays with an even slower time-constant, $\tau_l/\lambda_{decay}$ in Equation (11).

$$\tau_l \dot{P}(x,t) = \lambda_{build}\Big(-P(x,t) + f\big(u(x,t)\big)\Big) f\big(u(x,t)\big) - \lambda_{decay} P(x,t)\Big(1 - f\big(u(x,t)\big)\Big). \quad (11)$$

Here, $P(x, t)$ is the strength of the memory trace at site $x$ of the DNF with activity $u(x, t)$ and output $f\big(u(x, t)\big)$, $\lambda_{build}$ and $\lambda_{decay}$ are the rates of build-up and decay of the memory trace. The build-up of the memory trace is active on sites with a high positive output $f\big(u(x, t)\big)$, the decay is active on the sites with a low output. The memory trace $P(x, t)$ is an additive input to the DNF dynamics.

The memory trace formation can be used to account for one-shot learning of object categories (Faubel and Schöner, 2009), representation of visual scenes (Zibner et al., 2011), or action sequences (Sandamirskaya and Schoner, 2010b).

In a neuromorphic WTA implementation, the memory trace, or preshape, may be interpreted as the strength of synaptic connections from the DNF (or WTA), $u(x, t)$, to a "memory" population. This "memory" population activates the preshape by transmitting its activation through the learned synaptic connections, $P(x, t)$. Learning of the synaptic connections amounts to attractor dynamics [as in the first parenthesis of Equation (11)], in which the pattern of synaptic connections approaches the pattern of the DNF's (WTA's) output. This learning dynamics may also be implemented as a simple Hebbian rule: the synaptic weights which connect active sites of the DNF (WTA) with the memory population are strengthened. Another possible interpretation of the preshape as a change in the resting levels of individual nodes in the DNF (WTA) is harder to implement in neuromorphic WTA networks.

### 2.6.2. Learning mappings and associations

When the memory trace dynamics is defined within a structure with a higher dimensionality than the involved DNFs, the preshape dynamics leads to learning of mappings and associations. The dynamics of an associating map is similar to the memory trace dynamics, Equation (12).

$$\tau \dot{W}(x,y,t) = \epsilon(t)\Big(-W(x,y,t) + f(u_1(x,t)) \times f(u_2(y,t))\Big). \quad (12)$$

The weights function, W(x, y, t), which couples the DNFs $u_1(x, t)$ and $u_2(y, t)$ in Equation (12), as well as in Equations (4, 5), has an attractor at the intersection between positive outputs of the DNFs $u_1$ and $u_2$. The intersection is computed as a sum between the output of $u_1$, expanded along the dimensions of the $u_2$, and the output of the $u_2$, expanded in the dimensions of the $u_1$, augmented with a sigmoidal threshold function (this neural-dynamic operation is denoted by the $\times$ symbol). The shunting term $\epsilon(t)$ limits learning to time intervals when a rewarding situation is perceived, as exemplified in the architecture in Section 3.

This learning mechanism is equivalent to a (reward-gated) Hebbian learning rule: the cites of the DNFs $u_1$ and $u_2$ become coupled more strongly if they happen to be active simultaneously when learning is facilitated by the (rewarding) signal $\epsilon(t)$. Through the DNF dynamics, which builds localized activity peaks in the functionally relevant states, the learning dynamics has the properties of the adaptive resonance networks (ART, Carpenter et al., 1991), which emphasize the need for localization of the learning processes in time and in space.

#### 2.6.3. Adaptation

Adaptation [Equation (13)] is considered a learning process, which amounts to an unnormalized change of the coupling weights (gains) in a desired direction. A typical example is learning in the transition from the DNF's space-code to the rate-code of motor dynamics.

$$\tau\dot{\lambda}(x,t) = \epsilon(t) f(u(x,t)) \qquad (13)$$
$$\epsilon(t) = \text{error} \times \text{time window}$$

Here, $\lambda(x,t)$ is a matrix of weights, or gains, defined over the dimension of the DNF, $u(x,t)$, which is coupled to the motor dynamics, as in Equation (9). The gain changes in proportion to the output of the driving DNF, $u(x,t)$, in a learning window, defined by the term $\epsilon(t)$. The learning window is non-zero in a short time window when an intended action within EB, to which the DNF $u(x,t)$ belongs, is finished (the respective $u_{CoS}$ is active), but activity in the intention DNF is not yet extinguished. The *error* is determined in a DNF system, which compares the outcome of an action with the intended value of the motor variable and determines the direction of change of the weights in $\lambda(x,t)$.

Now that all neural-dynamic structures developed within DFT are presented, which may be implemented in hardware neuronal networks through the WTA architecture, I will introduce an exemplar robotic architecture, which integrates these mechanisms in a neural-dynamic system, which generates behavior and learns autonomously.

## 3. AN EXAMPLE OF AN ADAPTIVE ARCHITECTURE IN DFT

### 3.1. THE SCENARIO AND SETUP

The simple, but functioning in a closed loop learning architecture presented in this section employs several of the principles, presented above, such as categorization properties of DNFs, coupling between DNFs of different dimensionality, coupling to sensors and motors, autonomous action initiation and termination, as well as learning.

The robot, used to demonstrate the closed-loop behavior of a neuromorohic agent, consists of an eDVS camera and a pan-tilt unit. The eDVS camera has 128x128 event-based pixels, each sending a signal when a luminance change is detected. The pan-tilt unit consists of two servo motors, which take position signals in the range 0–2000 and are controlled to take the corresponding pose with a small latency. The task for this robot is to direct its gaze at a small blinking circle, which is moved around on a computer screen in front of the robot. A successful looking movement leads to the blinking circle settled in the central portion of the robot's camera array.

In order to accomplish this task, the robot, similarly to an animal, needs to detect the target in the visual input and in particular, estimate and represent its location relative to the center of the field of view of the robot. Next, according to the current location of the target, the system needs to select a motor command, which will bring the target into the center of the field of view. Thus, the system needs to select the desired values for pan and tilt, which will be sent to the servo motors.

This simple task embraces the following fundamental problems. First, the mapping between the target location and the required motor command is a priori unknown. The system needs to calibrate itself autonomously. In particular, the system needs to learn a mapping between the position of the input in the camera array and the motor command, which will bring the target in the center of the visual field. The second fundamental problem revealed in this setting is that when the camera moves, the perceived location of the target on the surface of the sensor changes, and the system needs a mechanism to keep the initial location of the target in memory in order to learn the mapping between the visually perceived locations and the motor commands. The third problem is the autonomy of the looking behavior: the systems needs a mechanism to update the target representation after both successful and unsuccessful looking actions.

**Figure 4** shows the scheme of the DNF architecture, which demonstrates how all these problems may be addressed in a closed-loop system. Next, I will present the dynamical structures, which constitute the architecture.

### 3.2. THE NEURAL-DYNAMICS ARCHITECTURE

#### 3.2.1. Perceptual DNF

The perceptual DNF is coupled to the eDVS, as described in Section 2.2 and effectively performs a low-pass filter operation on the camera input in time and in space. This DNF builds peaks of activation at locations, where events are concentrated in time and in space in the visual array of the robot.

#### 3.2.2. Visual intention DNF

This DNF builds sustained activity peaks that represent the target locations (**Figure 5**). The peaks are sustained even if the input, which initiated them ceases or moves. Thus, even during or after a gaze movement, the representation of the current target is stably represented. This allows, on the one hand, the robust coupling to the motor system (the attractor, set for the motor system, is guaranteed to be kept constant for the time of the movement). On the other hand, this memory system enables learning, since the representation of the previous target is still active when a rewarding input is perceived after a successful gaze.

#### 3.2.3. Motor intention DNF

The visual intention DNF represents the target of the current gaze action in sensory, here visual, coordinates. The movement generation system takes attractors in the motor coordinates, however (here, the desired pan and tilt). The motor intention DNF is

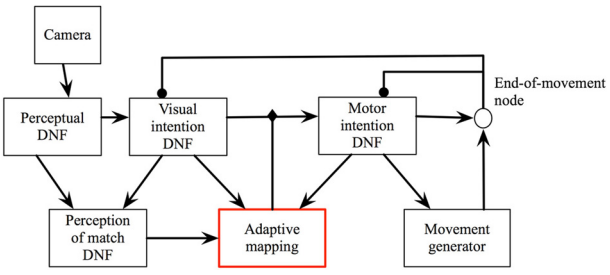



**FIGURE 4 | The DFT architecture for looking.** See main text for details.

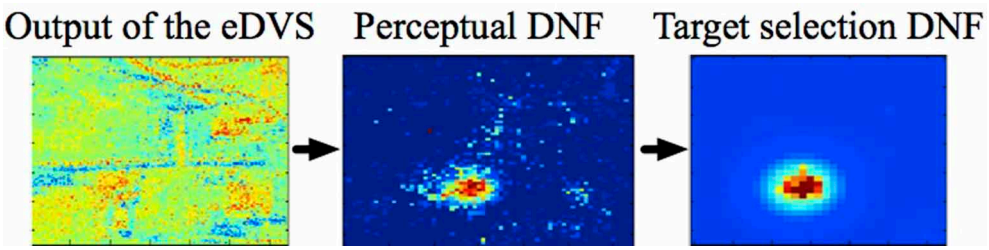



**FIGURE 5 | The cascade from the visual input to perceptual DNF to the visual intention (target) DNF segregates and stabilizes the selected region in the input stream.**

defined over the motor coordinates and an activity peak in this DNF creates an attractor for the motor dynamics and initiates a gaze movement.

#### 3.2.4. Condition of satisfaction node

The CoS DNF in this architecture is a zero-dimensional CoS node, since it monitors directly the state of the motor system, which is characterized by two single-valued variables, pan and tilt. The CoS node is activated when the motor action is accomplished [Equation (14)].

$$\tau\dot{v}_{cos}(t) = -v_{cos}(t) + h + c_{exc}f(v_{cos}(t)) + c\int f(u_{mot}(y,t))dy + c_a f_{diff}, \quad (14)$$

where $v_{cos}(t)$ is the activation of the CoS node for either the pan or the tilt movement (the CoS of the overall movement is a thresholded sum of the two CoSs). The CoS node is activated if (1) there is activity in the motor intention DNF, $u_{mot}$, and (2) the detector $f_{diff} = f(0.5 - |\xi_{pan} - \dot{pan}|)$ signals that the state variable for the pan or the tilt dynamics reaches the respective attractor, $\xi$. $c$ and $c_a$ are scaling constants for these two contributions, $c_{exc}$ is the strength of self-excitation of the CoS node.

The activated CoS node inhibits both the motor and the visual intention DNFs below activation threshold. The otherwise self-sustained activity peaks in these DNFs cease, which causes the CoS node loose its activation as well. The intention DNFs are released from inhibition and regain their initial resting levels, allowing the sensory input to induce a stabilized representation of the next target.

#### 3.2.5. The transformation array

The transformation between the visual and the motor coordinates, needed to achieve a particular behavioral goal, e.g., center the target object in the visual field, is a priori unknown. In the DFT architecture presented here, this transformation is represented by a randomly initialized coupling matrix, which implements a potential all-to-all connectivity between the two DNFs. Thus, an active visual intention DNF initially induces a peak at a random location in the motor DNF. The lateral interactions in the motor DNF ensure that a peak may be built, although the connection matrix is random (and sparse) in the beginning of the learning process.

In the transformation array, a learning dynamics is implemented [Equation (12)]. The learning window, $\lambda(t)$ is defined by the activity in the visual match DNF, which signals when the visual input falls onto the central part of the camera array.

#### 3.2.6. The visual match DNF

The visual match DNF receives a preshape in the center of the field when the visual intention DNF is active. This preshaping input is equivalent to an expectation to perceive the target in the visual field, which biases the visual match DNF to be sensitive to the respective sensory input. The connectivity which enables this predicting coupling is assumed to be given here, but could potentially emerge in a developmental process [e.g., similar to Luciw et al. (2013)].

$$\tau\dot{u}_{match}(x,t) = -u_{match}(x,t) + h + \int f(u_{match}(x',t))w(x-x')dx' + f(u_{perc}(x,t)) + c_G(x,t)\int f(u_{vis}(x,t))dx, \quad (15)$$

In Equation (15), the visual match DNF, $u_{match}(x,t)$ is defined over the same (visual, 2D here) coordinates as the perceptual DNF, $u_{perc}$, and the visual intention DNF, $u_{vis}$, and receives a one-to-one input from the perceptual DNF, as well as a Gaussian-shaped input, $c_G(x,t)$ if there's activity in the visual intention DNF. When the visual match DNF is active, it drives learning in the transformation array, according to Equations (16, 12).

$$\epsilon(t) = \int f(u_{match}(x,t))dx. \quad (16)$$

### 3.3. THE DYNAMICS OF THE ARCHITECTURE

**Figure 6** show the DNF architecture for looking at work.

When salient visual input is perceived by the eDVS sensor, one or several activity peaks emerge in the perceptual DNF (**Figure 6**, left), the most salient of these peaks (i.e., the one that reached the activation threshold first) drives the visual intention DNF (**Figure 6**, middle) and induces a self-sustained activity peak in this DNF. The peak in the visual intention DNF is sustained even when the camera starts moving and the visual input shifts, representing the instantaneous goal of the upcoming camera movement.

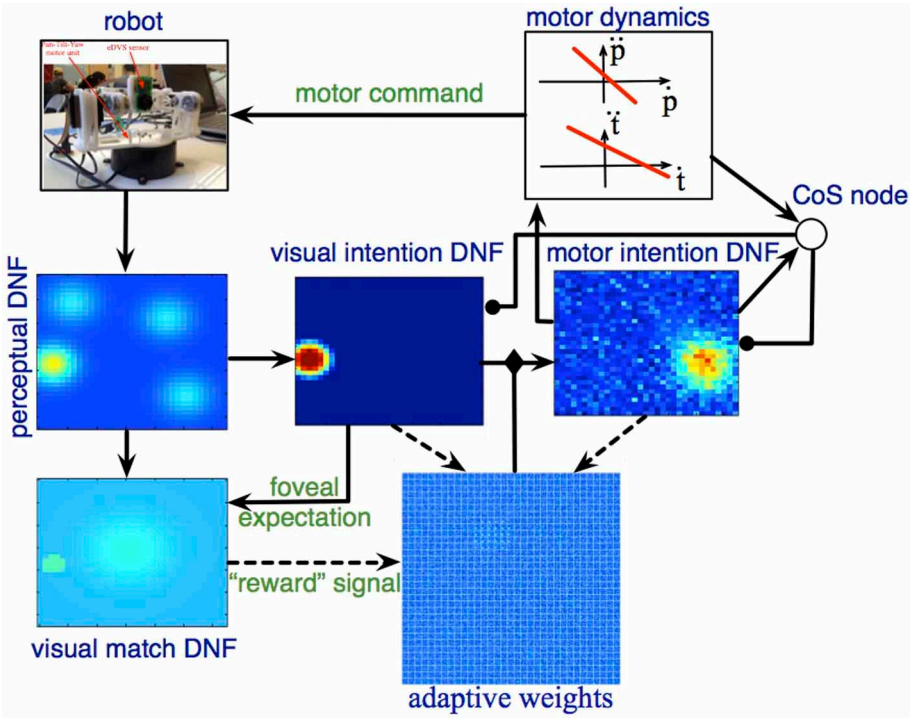



**FIGURE 6 | The DFT architecture for autonomous looking and learning the sensorimotor map.** The robotic camera provides input to the perceptual DNF, which performs initial segregation of object-like regions in the visual stream. The visual intention DNF selects and stabilizes the spatial representation (in visual coordinates) of a single target for the upcoming looking action. Through adaptive weights, the visual intention DNF provides input to the motor intention DNF, which generates attractors for the motor dynamics. Motor dynamics signals completion of the looking act through the CoS node, which inhibits the intention DNFs. If the looking action brings the target object into the foveal (central) region of the field of view, the adaptive weights are updated according to the current (decaying) activation in the visual and motor intention DNFs.

The visual intention DNF induces an activity peak in the motor intention DNF through the coupling weights, which are random in the beginning of the learning process. A localized activity peak emerges in the motor intention DNF, formed by the lateral interactions in this field. The motor intention peak sets an attractor for the dynamics of the pan and the tilt control variables, which drive the robotic pan-tilt unit. When the control variables are close to the attractor, the CoS node is activated and inhibits the visual and the motor intention DNFs. Activity in the motor intention DNF ceases in a forgetting instability, which leads to the CoS node to loose its activation as well. The inhibitory influence on the intention DNFs is released and the visual intention DNF may build a new activity peak from the perceptual input.

When the camera movement is finished (event, detected by the CoS node), if the input falls onto the central part of the visual array, the visual match DNF is activated and triggers the learning process in the adaptive weights. In particular, the weights are strengthened between the currently active positions in the visual intention DNF and the currently active positions in the motor intention DNF, which correspond to the just-been-active intentions. When the CoS node inhibits the intention DNFs, learning stops and a new gazing action is initiated.

**Figure 7** shows the activity of the motor variables during the gaze movements in the learning process and **Figure 8** shows the 2D projections of the 4D transformation matrix, learned over several hundred gaze movements to different target locations (Sandamirskaya and Conradt, 2013).

## 4. DISCUSSION

### 4.1. GENERAL DISCUSSION

The principles of DFT presented in this paper set a possible roadmap for the development of neuromorphic architectures capable of cognitive behavior. As modeling framework, DFT is remarkable in its capacity to address issues of embodiment, autonomy, and learning using neural dynamics throughout. In this paper, I have reviewed the DFT mechanisms that provide for the creation of stabilized sensory representations, learned associations, coupled sensory-motor representations, intentionality, and autonomous behavior and learning. In an exemplar architecture, I demonstrated how the computational and architectural principles of DFT come together in a neural-dynamic architecture, that coupled a neuromorphic sensor to motors and autonomously generated looking behavior while learning in a closed behavioral loop. The categorization properties of DNFs achieve the stabilization of the visual input against sensory noise, while the memory mechanisms allow the relevant representations to be kept active long enough to parameterize and initiate motor actions and also drive the learning process after a successful movement. Adaptive couplings between DNFs together with a mechanism that enables autonomous activation and deactivation of intentions

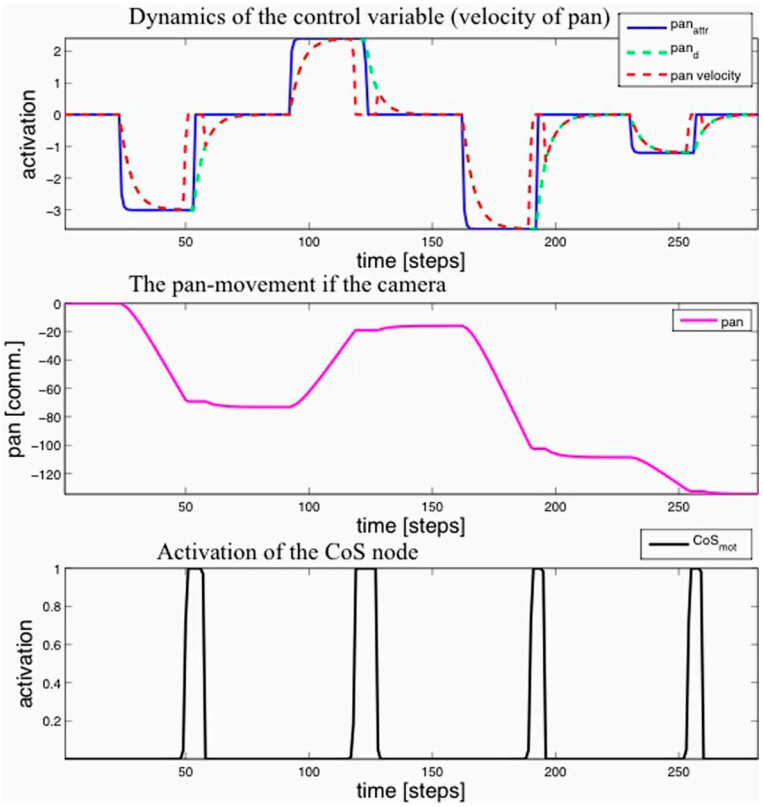



**FIGURE 7 | Top:** Time-course of the activation of the motor variable (velocity of the pan joint) during four steps of the learning procedure. **Middle**: The value of the pan variable. **Bottom**: Activation of the CoS node.

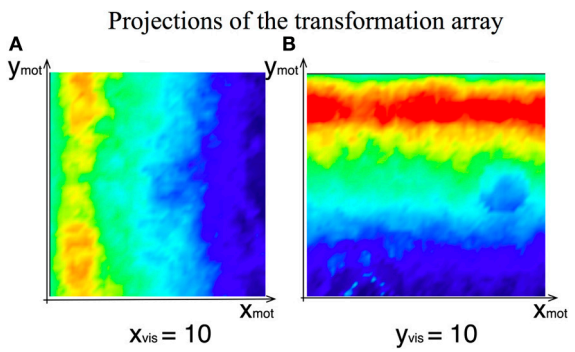



**FIGURE 8 | Two exemplar projections of the learned 4D transformation array between the visual and the motor intention DNFs of the agent.** **(A)** Weights' strength at the given visual-intention DNF horizontal position as function of motor intention field coordinates (overlayed projections along $y_{mot}$). **(B)** Weights' strength at the given visual-intention DNF vertical position.

make for an architecture in which autonomous learning accompanies behavior.

In order to "translate" the language of behavior-based attractor dynamics of DFT to spiking networks implemented in VLSI, several possibilities have been reported recently. One solution (Neftci et al., 2013) constitutes a method to set parameters of the neuromorphic hardware in relation to parameters of a more abstract WTA layer. By measuring the activity of hardware units, the parameter mappings are calibrated in an automated procedure. Another way to translate DNF dynamics to spiking networks is to use the vector-encoding of a dynamical system in the neural-dynamic framework of Eliasmith (2005). This framework allows one to implement the attractor dynamics of DNFs in terms of a network of spiking units, which in its turn may define the parametrization for a VLSI neuromorphic network.

These powerful tools allow one to translate between levels of description and can be used to implement different models of cognition in order to facilitate the development of behaving, neuromorphic cognitive systems. DFT is one of the frameworks that defines the principles and constraints critical to this goal. There are of course several other frameworks that may be used for this purpose, each with its own advantages and limitations. Thus, the probabilistic framework allows one to use noisy and incomplete sensory information to infer hidden states of the environment and weigh alternative actions, which may bring the agent closer to its goals. Such a Bayesian framework has been applied both in the field of modeling human cognition [e.g., Griffiths et al. (2008)] and in robotics (Thrun et al., 2005). However, this framework has two limitations with respect to modeling human cognition. First, the probabilistic models focus on the functional or behavioral

aspects of cognition and not the neuronal mechanisms underlying cognitive processing. They often require normalization procedures which are not trivial to implement neurally. Second, the probabilistic models often need an external mechanism to make inferences on the probability distributions and do not account for the process of decision making. Thus, the Bayesian architectures may achieve powerful performance and may be used to account for empirical data on human cognition, but they do not provide a process model of cognitive functions or offer a mechanism of how these functions are achieved or realized neurally. On the contrary, in neuronal modeling, the developed architectures are anchored in neuronal data and focus on the mechanisms and processes behind cognition. However, their functional implementations (i.e., embodiment) are typically limited and fail to address important problems such as representational coupling, autonomy, and development. DFT aims at bridging the two approaches to understanding cognitive processing—the functional (behavioral) and the mechanistic (neuronal)—and thus naturally fits the goal of providing for a tool to implement neuromorphic cognition. The scaling of DFT toward higher cognitive functions, such as concept representation, language, and complex action sequencing is currently under way.

This paper aims to reveal the formalized DFT principles and concepts developed in embodied cognition and autonomous robotics in such a way that they may be integrated into the language of spiking neural networks in VLSI hardware through the structure of WTA networks. DNF may be considered a functional description of the soft WTA networks. The successful implementation of soft WTA networks in VLSI devices to date opens the way to employing the architectural elements of DFT in spiking hardware architectures. These structural elements as summarized here are (1) coupling between fields of different dimensionality, (2) coupling to sensors through space-coding, (3) coupling to rate-coded motor dynamics, (4) application of principles of autonomy (intentionality), and (5) autonomous neural-dynamic learning. Some of the DFT principles, such as categorization and memory formation, are already probed in VLSI WTA networks, resulting in a framework of state-based computing in spiking networks. In addition, this paper formalizes mechanisms that allow for autonomous transition between stable states through the introduction of elementary behavior structures, namely the intention and the conditions-of-satisfaction. This formalization also enables autonomous learning and the robust coupling of WTAs to each other, to sensors, and to motor dynamics.

The DFT approach considers cognitive systems from a behavioral perspective while neuromorphic hardware system development aims at understanding the neuronal mechanisms underlying cognition. The fact that these two approaches converge to a mathematically equivalent object—a DNF or a soft WTA—as an elementary computational unit in the development of cognitive neuromorphic systems is a strong argument for the fundamental character of this computational element. Here, I aimed at establishing a common ground for future collaborative projects that can facilitate progress in both fields. The VLSI networks could scale up to produce cognitive autonomous behavior and the DFT framework could gain access to a neural implementation which is not only more efficient and biologically grounded, but also open to empirical links between the behavioral and neuronal dynamics. Bringing principles of DFT onto VLSI chips will, on the one hand, allow one to model human cognition and make predictions under both neuronal and behavioral constraints. On the other hand, the cooperation between the two fields could foster the development of powerful technical cognitive systems based on a parallel, low-power implementation with VLSI.

## ACKNOWLEDGMENTS

The author gratefully acknowledges support from the organizers of the Telluride Cognitive Neuromorphic Engineering worksop 2012 and the Capo Caccia Neuromorphic Cognition 2013 workshop, as well as of Prof. J. Conradt for providing the hardware setup.

## FUNDING

The project was funded by the DFG SPP "Autonomous learning" within the Priority program 1527.

## REFERENCES

Abrahamsen, J. P., Hafliger, P., and Lande, T. S. (2004). "A time domain winner-take-all network of integrate-and-fire neurons," in *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems, 2004. ISCAS'04*. (Vancouver), Vol. 5, V-361.

Amari, S. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol. Cyber*. 27, 77–87. doi: 10.1007/BF00337259

Bastian, A., Schoner, G., and Riehle, A. (2003). Preshaping and continuous evolution of motor cortical representations during movement preparation. *Eur. J. Neurosci.* 18, 2047–2058. doi: 10.1046/j.1460-9568.2003.02906.x

Bicho, E., Mallet, P., and Schöner, G. (2000). Target representation on an autonomous vehicle with low-level sensors. *I. J. Robotic Res.* 19, 424–447. doi: 10.1177/02783640022066950

Bicho, E., and Schoner, G. (1997). The dynamic approach to autonomous robotics demonstrated on a low-level vehicle platform. *Robot. Auton. Syst*. 21, 23–35. doi: 10.1016/S0921-8890(97)00004-3

Buss, A. T., and Spencer, J. P. (2012). When seeing is knowing: the role of visual cues in the dissociation between childrens rule knowledge and rule use. *J. Exp. Child Psychol*. 111, 561–569. doi: 10.1016/j.jecp.2011.11.005

Carpenter, G. A., Grossberg, S., and Rosen, D. B. (1991). Art 2-a: an adaptive resonance algorithm for rapid category learning and recognition. *Neural Netw*. 4, 493–504. doi: 10.1016/0893-6080(91)90045-7

Conradt, J., Berner, R., Cook, M., and Delbruck, T. (2009). "An embedded aer dynamic vision sensor for low-latency pole balancing," in *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*. (Kyoto), 780–785.

Douglas, R. J., and Martin, K. A. C. (2004). Neural circuits of the neocortex. *Ann. Rev. Neurosci*. 27, 419–451. doi: 10.1146/annurev.neuro.27.070203.144152

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. *Neural Comput*. 7, 1276–1314. doi: 10.1162/0899766053630332

Ellias, S. A., and Grossberg, S. (1975). Pattern formation, contrast control, and oscillations in the short term memory of shunting on-center off-surround networks. *Biol. Cyber*. 20, 69–98. doi: 10.1007/BF00327046

Erlhagen, W., and Bicho, E. (2006). The dynamic neural field approach to cognitive robotics. *J. Neural Eng*. 3, R36–R54. doi: 10.1088/1741-2560/3/3/R02

Ermentrout, B. (1998). Neural networks as spatio-temporal pattern-forming systems. *Rep. Prog. Phys*. 61, 353–430. doi: 10.1088/0034-4885/61/4/002

Faubel, C., and Schöner, G. (2009). "A neuro-dynamic architecture for one shot learning of objects that uses both bottom-up recognition and top-down prediction," in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*. (St. Louis, MO: IEEE Press).

Griffiths, T. L., Kemp, C., and Tenenbaum, J. B. (2008). "Bayesian models of cognition," in *Cambridge Handbook of Computational Cognitive Modeling* (Cambridge: Cambridge University Press), 59–100.

Grossberg, S. (1988). Nonlinear neural networks: principles, mechanisms, and architectures. *Neural Netw*. 1, 17–61. doi: 10.1016/0893-6080(88)90021-4

Indiveri, G., Chicca, E., and Douglas, R. J. (2009). Artificial cognitive systems: from vlsi networks of spiking neurons to neuromorphic cognition. *Cogn. Comput*. 1, 119–127. doi: 10.1007/s12559-008-9003-6

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci*. 5:73. doi: 10.3389/fnins.2011.00073

Indiveri, G., Murer, R., and Kramer, J. (2001). Active vision using an analog vlsi model of selective attention. *IEEE Trans. Cir. Syst. II* 48, 492–500. doi: 10.1109/82.938359

Iossifidis, I., and Schöner, G. (2004). "Autonomous reaching and obstacle avoidance with the anthropomorphic arm of a robotic assistant using the attractor dynamics approach," in *Proceedings of the IEEE 2004 International Conference on Robotics and Automation*. (New Orleans, LA).

Johnson, J. S., Spencer, J. P., and Schöner, G. (2006). "A dynamic neural field theory of multi-item visual working memory and change detection," in *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci 2006)* (Vancouver, BC), 399–404.

Johnson, J. S., Spencer, J. P., and Schoner, G. (2008). Moving to higher ground: the dynamic field theory and the dynamics of visual cognition. *New Ideas Psychol*. 26, 227–251. doi: 10.1016/j.newideapsych.2007.07.007

Latash, M. L., Scholz, J. P., and Schoner, G. (2007). Toward a new theory of motor synergies. *Motor Control* 11, 276–308.

Liu, S. C., and Delbruck, T. (2010). Neuromorphic sensory systems. *Curr. Opin. Neurobiol*. 20, 288–295. doi: 10.1016/j.conb.2010.03.007

Luciw, M., Kazerounian, S., Lakhmann, K., Richter, M., and Sandamirskaya, Y. (2013). "Learning the perceptual conditions of satisfaction of elementary behaviors," in *Robotics: Science and Systems (RSS), Workshop "Active Learning in Robotics: Exploration, Curiosity, and Interaction."* (Berlin).

Mead, C., and Ismail, M. (1989). *Analog VLSI Implementation of Neural Systems*. Norwell, MA: Springer.

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. J. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proc. Natl. Acad. Sci. U.S.A*. 110, E3468–E3476. doi: 10.1073/pnas.1212083110

Neftci, E., Chicca, E., Cook, M., Indiveri, G., and Douglas, R. (2010). "State-dependent sensory processing in networks of vlsi spiking neurons," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. (Paris), 2789–2792.

Oster, M., and Liu, S. C. (2004). "A winner-take-all spiking network with spiking inputs," in *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, ICECS* 2004. (Tel-Aviv), 203–206.

Reimann, H., Iossifidis, I., and Schoner, G. (2011). "Autonomous movement generation for manipulators with multiple simultaneous constraints using the attractor dynamics approach," in *Proceedings of the IEEE International Conference on Robotics and Automation ICRA* (Shanghai).

Richter, M., Sandamirskaya, Y., and Schoner, G. (2012). "A robotic architecture for action selection and behavioral organization inspired by human cognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Algarve), IROS.

Rutishauser, U., and Douglas, R. J. (2009). State-dependent computation using coupled recurrent networks. *Neural Comput*. 21, 478–509. doi: 10.1162/neco.2008.03-08-734

Sandamirskaya, Y., and Conradt, J. (2013). "Learning sensorimotortransformations with dynamic neural fields," in *International Conference on Artificial Neural Networks (ICANN)*. (Sofia).

Sandamirskaya, Y., Richter, M., and Schoner, G. (2011). "A neural-dynamic architecture for behavioral organization of an embodied agent," in *IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL EPIROB 2011)*. (Frankfurt).

Sandamirskaya, Y., and Schoner, G. (2010a). An embodied account of serial order: how instabilities drive sequence generation. *Neural Netw*. 23, 1164–1179. doi: 10.1016/j.neunet.2010.07.012

Sandamirskaya, Y., and Schoner, G. (2010b). An embodied account of serial order: how instabilities drive sequence generation. *Neural Netw*. 23, 1164–1179. doi: 10.1016/j.neunet.2010.07.012

Sandamirskaya, Y., Zibner, S., Schneegans, S., and Schoner, G. (2013). Using dynamic field theory to extend the embodiment stance toward higher cognition. *New Ideas Psychol*. 30, 1–18. doi: 10.1016/j.newideapsych.2013.01.002

Schaal, S., Ijspeert, A., and Billard, A. (2003). Computational approaches to motor learning by imitation. *Philos. Trans. R. Soc. Lond. B* 358, 537–547. doi: 10.1098/rstb.2002.1258

Schoner, G. (2008). "Dynamical systems approaches to cognition," in *Cambridge Handbook of Computational Cognitive Modeling*, ed R. Sun (Cambridge: Cambridge University Press), 101–126.

Searle, J. R. (1983). *Intentionality—An Essay in the Philosophy of Mind*. Cambridge: Cambridge University Press.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. Vol. 1. Cambridge: MIT press.

Wilson, H. R., and Cowan, J. D. (1973). A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik* 13, 55–80. doi: 10.1007/BF00288786

Zibner, S. K. U., Faubel, C., Iossifidis, I., and Schöner, G. (2011). Dynamic neural fields as building blocks for a cortex-inspired architecture of robotic scene representation. *IEEE Trans. Auton. Ment. Dev*. 3, 74–91. doi: 10.1109/TAMD.2011.2109714

**Conflict of Interest Statement:** The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 07 October 2013; accepted: 25 December 2013; published online: 22 January 2014.*

*Citation: Sandamirskaya Y (2014) Dynamic neural fields as a step toward cognitive neuromorphic architectures. Front. Neurosci. 7:276. doi: 10.3389/fnins.2013.00276*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 17 January 2014
doi: 10.3389/fnins.2013.00278

# A robust sound perception model suitable for neuromorphic implementation

**Martin Coath[1,2]*, Sadique Sheik[3], Elisabetta Chicca[4], Giacomo Indiveri[3], Susan L. Denham[1,2] and Thomas Wennekers[1,5]**

[1] Cognition Institute, Plymouth University, Plymouth, UK
[2] Faculty of Health and Human Sciences, School of Psychology, Plymouth University, Plymouth, UK
[3] Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland
[4] Faculty of Technology, Cognitive Interaction Technology – Center of Excellence, Bielefeld University, Bielefeld, Germany
[5] Faculty of Science and Environment, School of Computing and Mathematics, Plymouth University, Plymouth, UK

**Edited by:**
*André Van Schaik, The University of Western Sydney, Australia*

**Reviewed by:**
*John Harris, University of Florida, USA*
*Dylan R. Muir, University of Basel, Switzerland*

***Correspondence:**
*Martin Coath, Cognition Institute, Plymouth University, A222, Portland Square, Drake Circus, Plymouth, Devon PL48AA, UK*
*e-mail: mcoath@plymouth.ac.uk*

We have recently demonstrated the emergence of dynamic feature sensitivity through exposure to formative stimuli in a real-time neuromorphic system implementing a hybrid analog/digital network of spiking neurons. This network, inspired by models of auditory processing in mammals, includes several mutually connected layers with distance-dependent transmission delays and learning in the form of spike timing dependent plasticity, which effects stimulus-driven changes in the network connectivity. Here we present results that demonstrate that the network is robust to a range of variations in the stimulus pattern, such as are found in naturalistic stimuli and neural responses. This robustness is a property critical to the development of realistic, electronic neuromorphic systems. We analyze the variability of the response of the network to "noisy" stimuli which allows us to characterize the acuity in information-theoretic terms. This provides an objective basis for the quantitative comparison of networks, their connectivity patterns, and learning strategies, which can inform future design decisions. We also show, using stimuli derived from speech samples, that the principles are robust to other challenges, such as variable presentation rate, that would have to be met by systems deployed in the real world. Finally we demonstrate the potential applicability of the approach to real sounds.

**Keywords: auditory, modeling, plasticity, information, VLSI, neuromorphic**

## 1. INTRODUCTION

Neurons in sensory cortex are highly adaptive, and are sensitive to an organism's sensory environment. This is particularly true during early life and an epoch known as the "critical period" (Zhang et al., 2001; Insanally et al., 2009). For many organisms sounds of ecological importance, such as communication calls, are characterized by time-varying spectra. Understanding how to build auditory processing systems that can cope with time-varying spectra is important. However, most neuromorphic auditory models to date have focused on distinguishing mainly static patterns, under the assumption that dynamic patterns can be learned as sequences of static ones.

One strategy for devices that implement artificial sensory systems is to emulate biological principles. Developing this approach holds out the hope that we might be able to build devices that approach the efficiency and robustness of biological systems and, in doing so, new insights in to neural processing might be gained. If, as is widely believed, the perception of complex sensory stimuli *in vivo* is based upon the population response of spiking neurons that are tuned to stimulus features then important questions arise, including "what are these features, and how do they come in to existence?" The situation for artificial auditory perception is complicated by the fact that the way in which sounds are represented in mammalian auditory cortex is not well understood, and neither are the neural mechanisms underlying the learning of dynamic sound features.

Neural mechanisms thought to underlie, for example, sensitivity to frequency sweeps include differential latency between excitatory inputs (Razak and Fuzessery, 2008), or excitatory and inhibitory inputs (Razak and Fuzessery, 2010), and asymmetric inhibition (Zhang et al., 2003; Razak and Fuzessery, 2009), all of which have been shown to correlate with sweep direction and/or rate preference. However, these studies have focussed primarily on local neural mechanisms (Ye et al., 2010) whereas anatomical studies of the auditory system reveal widespread lateral connections and nested recurrent loops, and in many cases feedback connections outnumbering feed-forward ones (Friston, 2005).

We have demonstrated previously that it is possible to address the problem of sensitivity to dynamic stimuli, including but not limited to frequency modulated (FM) sweeps, with a biophysically plausible model of auditory processing (Coath et al., 2010). We have validated the model with a real-time physical system implemented using neuromorphic electronic circuits (Sheik et al., 2011). However, neither of these studies has investigated the robustness of the system to stimuli that exhibit variation, either in spike pattern, or presentation rate, or to the order of similar

stimuli when sets of stimuli are presented continuously. In addition, the spectro-temporal patterns used as stimuli in these earlier studies are not derived from, or related to, those found in natural sounds such as speech, or other communication calls of animals. All of these considerations are important if the principles involved are to be implemented in artificial sensory systems that can be deployed in realistic environments.

In the present paper we provide evidence that the approach first presented in Sheik et al. (2011) is suitable for "real-world" deployment in that we extend the hardware results to an investigation of responses to "noisy" stimuli. We also present results from a software simulation that replicates the hardware as closely as possible using stimuli derived from speech and presented continuously at different rates. Robustness to both of these types of stimulus variation is a necessary condition for any practical system. Finally we predict the results from networks with comparable architectures trained on real world stimuli. This approach is useful in that it provides guidelines that can be used to inform the design of more complex neuromorphic processing systems that could be implemented in the future.

## 2. METHODS

### 2.1. NETWORK

#### *2.1.1. Schematic*

A schematic representation of the network, as implemented in both hardware and software, is shown in **Figure 1**. The horizontal axis in the figure represents the tonotopic arrangement of the auditory system, divided in to a number of frequency channels representing positions on the basilar membrane. The pattern of spiking in the A neurons thus represents the output of an artificial cochlea (Chan et al., 2007). Three channels only are shown in **Figure 1**, the central channel is labeled and the two flanking channels are shown "dimmed" to illustrate how the neurons within each channel are laterally connected to other channels. The hardware implementation and the software simulation both use 32 tonotopic channels. Where real stimuli are processed (see section 2.1.5) the cochlea uses a linear gammatone filter bank, followed by half wave rectification and low pass filtering to simulate the phase locking characteristics of auditory nerve firing, and center frequencies ranging from 50 to 8000 Hz equally spaced on the Equivalent Rectangular Bandwidth scale (Glasberg and Moore, 1990).

The input neuron A, at each tonotopic position projects to a $B_1$ and a $B_2$ neuron in the same channel *via* excitatory synapses. The output of the network is taken to be the activity of the $B_2$ neurons. This activity is derived from the input, but controlled by excitatory and inhibitory projections from $B_1$ neurons. However, the excitatory $B_1 \rightarrow B_2$ projections originate *only* from other tonotopic channels, these connections exhibit distance dependent propagation delays, and terminate with plastic synapses which are the loci of Spike Timing Dependent Plasticity (STDP) (see section 2.1.2). Each $B_1$ neuron is connected to a number of $B_2$ neurons *via* these delayed connections that have a fan out of 14 neurons on either side. The learning rule implemented at the synapses associated with these connections (shown as filled triangles in **Figure 1**) ensures that the $B_2$ neurons are active only if there are coincidences between spikes within the channel and delayed spikes from other channels; it is this feature that allows the network to learn dynamic spectro-temporal patterns. The units marked C represent the delays in the $B_1 \rightarrow B_2$ connections which are implemented differently in hardware and software, see sections 2.1.3 and 2.1.4.

#### *2.1.2. Spike timing dependent plasticity*

Plasticity in both the hardware and software networks is implemented in each of the $B_1 \rightarrow B_2$ synapses in the form of an

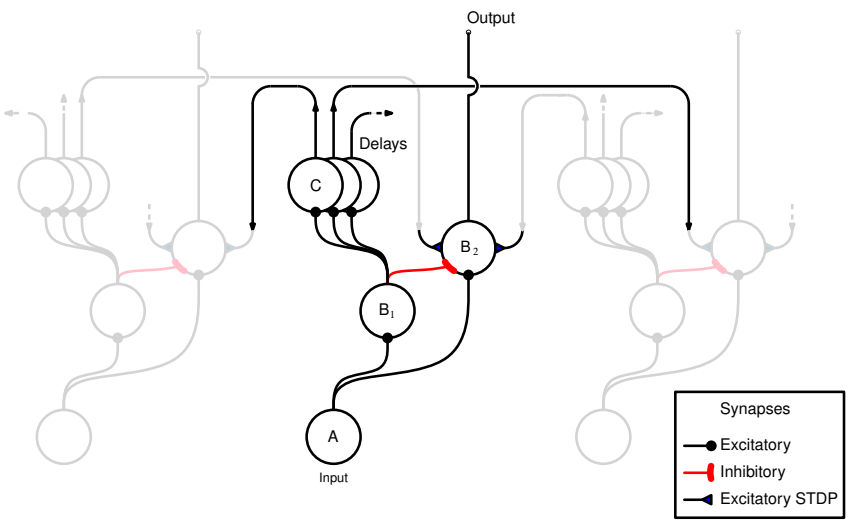



**FIGURE 1 | Schematic representation of the network as implemented in hardware and software.** Neurons are arranged in groups representing positions on the tonotopic axis. Three channels only are shown, and of these only the central channel is labeled for clarity. The two flanking channels are shown dimmed to illustrate how the channels interconnected by delayed excitatory projections that terminate in plastic synapses. The two populations of B neurons receive input from the same A neurons within the same channel. $B_2$ neurons are excited by $B_1$ neurons from outside their own channel.

STDP-like model of synaptic plasticity described fully in Brader et al. (2007). In the absence of activation, the synaptic weight, or efficacy, drifts toward one of two stable values, 0 or 1; and although it can take on other values, it is bounded by these two values and stays constant at one of them unless further learning events occur. This has the advantage of preventing instabilities in the adaptation, such as the unbounded growth of connection strengths.

### 2.1.3. Hardware implementation

The first set of results presented in section 3.1 were obtained using a hybrid analog /digital hardware implementation of the network model which consists of a real-time, multi-chip set-up as described in Sheik et al. (2011). Three multi-neuron spiking chips and an Address Event Representation (AER) mapper (Fasnacht and Indiveri, 2011) are used connected in a serial loop. The multi-neuron chips were fabricated using a standard AMS 0.35 μm CMOS process.

The hardware does not directly support propagation delays between neurons. To overcome this limitation, long synaptic and neuronal time constants are exploited, which due to the variability in hardware have a range of values (Sheik et al., 2012). Given that the weights associated with the synapses of a neuron are strong enough to produce a single output spike, the time difference between the pre-synaptic spike and the post-synaptic spike is considered equivalent to a propagation/transmission delay. Therefore, every projection in the model that requires a delay is passed through an additional neuron, referred to as a delay neuron. The delay neurons are labeled C in **Figure 1**.

***2.1.3.1. Frequency modulated stimuli.*** Trials with the hardware network were conducted with stimuli representing Frequency Modulated (FM) sweeps. These were prepared off-line by injecting current in to integrate and fire neurons. A current pulse of duration 5.5 ms is used in each channel in turn to generate the burst of input spikes representing the activity of the A neurons (see **Figure 1**) when presented with a frequency modulated stimulus. In order to evaluate the robustness of the network response to stimulus variation, or noise, an additional noisy current signal is added to this injection current used to generate the input spikes as illustrated in **Figure 2**. Noise is generated from an Ornstein Uhlenbeck (OU) process with a zero mean using the forward Euler method (Bibbona et al., 2008). We define the noise level, σ as the ratio between the standard deviation of the OU process and the magnitude of the actual noise free current signal used to generate the spikes.

***2.1.3.2. FM sweep trials and analysis.*** Trials for the hardware and software versions of the network consisted of two parts; first the *exposure* phase, using the exposure stimulus (ES), followed by a *probe* phase using a number of different probe stimuli (PS) presented many times. During the exposure phase the learning rule forces the weight, or efficacy, of each $B_1 \rightarrow B_2$ plastic synapses to either one or zero; this effects a pattern of stimulus driven connectivity. The selection by the learning rule of only a few high efficacy connections is the origin of the difference in response characteristics between the $B_1$ and the $B_2$ neurons (Sheik et al., 2011).

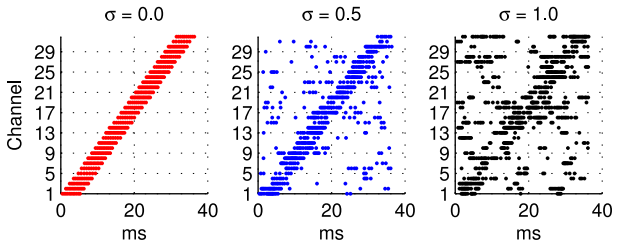



**FIGURE 2 | Illustration of sample spike patterns used as probe stimuli in the hardware experiments described in section 2.1.3 to investigate the robustness of the network response to variability in the probe stimulus.** In all cases these patterns of spikes are prepared off-line using a model integrate and fire neuron, a 5.5 ms current pulse, and a noise current that extends over the whole stimulus period. From left to right the ratio between the noisy current the current pulse used to generate the spike in a channel increases from 0 to 1. The range illustrated is greater than that used in the experiments where the highest level of noise is $\sigma = 0.45$.

The method adopted in the first set of experiments (section 3.1) using the hardware implementation of the network and FM sweep stimuli was the same as that described in Sheik et al. (2011). We reset the neurons to their resting state at the beginning of each ES and the plastic synapses to their "low" state, that is with effectively null synaptic efficacy. Input patterns were presented 30 times over a period of 3 seconds during which time the network "learns." We then measured the response of the exposed network by probing with a set of PS that consisted of linear frequency sweeps with different velocities; during each of these presentations the number of spikes in the $B_2$ neurons was recorded. Stimuli representing each of the 10 sweep rates were presented 100 times for each noise level during the probe phase. These results were used to determine the Stimulus Specific Information (SSI) as described below.

***2.1.3.3. Stimulus Specific Information.*** As artificial sensory systems become increasingly complex it will become increasingly important to make principled decisions about their design. In the majority of cases choices will have to be made where the detailed neurobiological data is incomplete or difficult to interpret. This inevitably leads to a requirement to quantify the performance of the network (for comparison with *in vivo* data, and to guide choices of architecture, learning rule, etc.) where no clear guidance is available from physiology.

A measure that has been used to characterize neuronal acuity is the Stimulus Specific Information (SSI) which is a formalization of the intuitive view that a stimulus is well encoded if it produces an unambiguous response; that is a response that is associated with a unique, or very small number of, stimuli. Where this is true the stimulus is readily identified when one of these responses, or a response in the correct range, appears (Butts and Goldman, 2006). This characterization has the advantage of not being dependant on the design or performance of a classifier. The specific

information of a response $i_{sp}(r)$ given a set of stimuli $\Theta$ can be written:

$$i_{sp}(r) = -\sum_{\Theta} p(\Theta)\log_2 p(\Theta) + \sum_{\Theta} p\left(\Theta|r\right)\log_2 p\left(\Theta|r\right)$$

where it is defined in terms of the entropy of the stimulus ensemble, and that of the stimulus distribution conditional on a particular response. This makes the $i_{sp}(r)$ a measure of the reduction in uncertainty about the stimulus $\Theta$ gained by measuring particular response $r$. Thus the value of the $i_{sp}(r)$ is high for unambiguous responses and low for ambiguous responses. The SSI is simply the average specific information of the responses that occur when a particular stimulus, $\Theta$, is present:

$$i_{\mathrm{SSI}}(\Theta) = \sum_{r} p\left(r|\Theta\right) i_{sp}(r)$$

We show that the performance of the network can be characterized by the SSI which combines features of the tuning curve, where information is encoded in the rate of response, and of the Fisher Information where the high-slope regions of the tuning curve are the most informative (Butts and Goldman, 2006).

***2.1.3.4. Receiver Operating Characteristics.*** A Receiver Operating Characteristic (ROC) can be used as a measure of performance of classifiers (Fawcett, 2006). ROC graphs have their origin in signal detection theory but are also popular in other fields, including the evaluation and comparison of machine learning algorithms. The output from the network can be interpreted as a binary classifier if we designate the Exposure Stimulus as the target for identification by setting a detection threshold. The ROC is then a graph of the False Positive Rate (FPR) against the True Positive Rate (TPR) for all values of the detection threshold. The FPR is simply the ratio between the number of stimuli of the target class correctly identified (True Positives, TP) and the total number of stimuli identified as belonging to this class (Positives, P):

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}}$$

Likewise, the FPR is the ratio between the the number of stimuli incorrectly identified as belonging to the target class (False Positives, FP) and the total number of stimuli identified as not belonging to this class (Negatives, N):

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}}$$

The ROC curve is a two-dimensional visualization of the system's potential as a classifier. We also make use of a common method to reduce this to a single scalar value, that is to calculate the area under the ROC curve, abbreviated AUC; this is achieved by adding the area of successive trapezoids (Fawcett, 2006). The Area Under Curve (AUC) is used to quantify the relative overall ability of the network to discriminate between the two classes of stimuli; that is those that match the class of the Exposure Stimulus and those that do not. This method has been widely used in the characterization of classifiers and is believed to perform very well (Fawcett, 2006). In all cases the AUC will be between 0.5, representing a network that will not function as a classifier, and 1.0 which represents a perfect classifier at all thresholds. Although useful, unlike the SSI this ignores the information present in the response concerning any of the other six classes.

#### *2.1.4. Software implementation*

A second set of results, presented in section 3.2, were obtained using a network implemented in custom "C" code closely based on the hardware implementation. The learning rule implemented is also the same as in the hardware implementation (see section 2.1.2). In these software simulations of the hardware implementation the lateral, or $B_1$ to $B_2$, projections exhibit distance dependent delays that cover the same range of values as the hardware network, however these delays were implemented in a queued data structure whereas in the hardware these delays are implemented by exploiting variability of time constants that result from the fabrication of the chip (Sheik et al., 2011, 2012). Beside this difference the software model was designed to be close to the hardware implementation in order to allow for reliable predictions of the hardware's learning and recognition capabilities. Because the hardware operates in real biological time, use of an emulated software version allowed us to run a large number of tests, which would have been impossible in hardware.

***2.1.4.1. Stimuli derived from speech.*** The stimuli used in these experiments using the software network were derived from speech and represent the formant tracks of a set of English words. Formants are peaks in the frequency response of sounds caused by resonances in the vocal tract. These peaks are the characteristics that identify vowels and in most cases the two first formants are enough to disambiguate a vowel. This approach was chosen as it results in stimuli that increase the complexity and realism from the single, and double, FM sweeps used in the first set of experiments.

A vocabulary of seven words was chosen: *And, Of, Yes, One, Two, Three, Four* and three examples of each were recorded using a male speaker (the first author). Seven words were chosen because they exhibit a variety of vowel sounds, and hence their formant tracks exhibit a range of spectrotemporal correlations, also they are monosyllabic and (almost) free of diphthongs. The formant tracks of these words exhibit spectrotemporal correlations, for example changes in frequency over time and maxima at two different spectral positions at the same time, that we have shown can be learned by the network—there is more on the mechanism of this learning in section 3.1.

The first and second formant tracks of these seven classes were extracted using LPC which yields position (frequency) and magnitude parameters for formants (Ellis, 2005). The results are shown in **Figure 3** in which parts of the stimulus indicated with a thicker line (in blue) are those with an LPC magnitude of greater than 15% of the maximum value indicating the position of the vowel. The thin line sections (in gray) correspond to the parts of the sound files that were silent or contained consonants. **Figure 4** shows how the three examples of each word have formant tracks

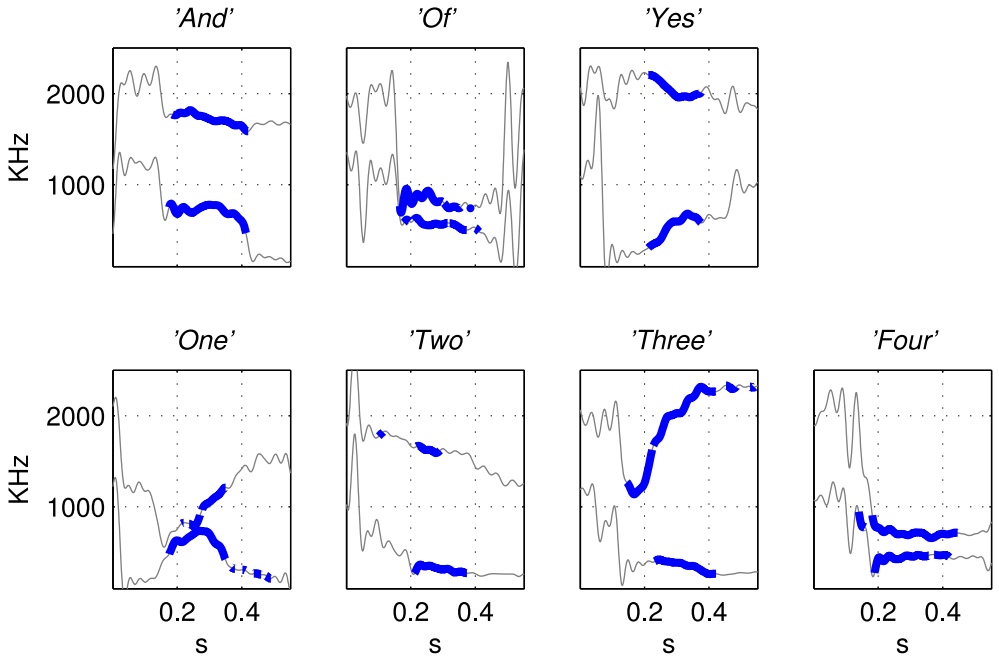



**FIGURE 3 | Illustration of the derivation of simplified stimuli consisting of the first and second formant tracks for the seven words extracted using Linear Predictive Coding.** The words were *"And", "Of", "Yes", "One", "Two", "Three", "Four"* as labeled in titles of subfigures. The thin line segments (in gray) are the parts of the sound files that were silent or contained consonants. Formant tracks of vowels, shown in thicker blue line segments, were smoothed and down-sampled to produce the patterns of current injection that were a highly simplified representation of the speech stimuli, see **Figure 5**.

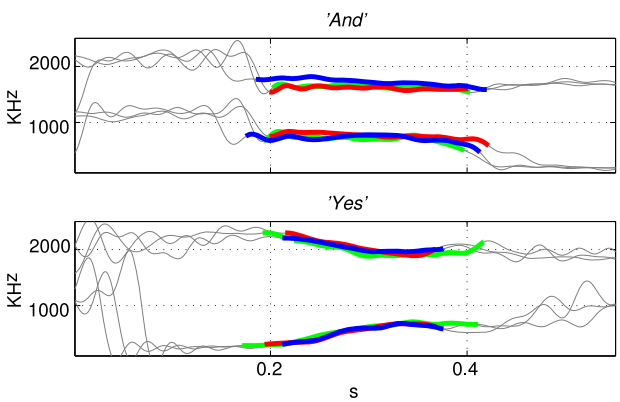



**FIGURE 4 | Formant tracks extracted using LPC for three examples of two different stimuli "And" and "Yes."** Sections of the individual stimuli corresponding to vowels are indicated by thicker red, green, and blue segmemts this Figure clearly shows that there is some variation in the formant tracks among sets of stimuli of the same class even when recorded from a single speaker.

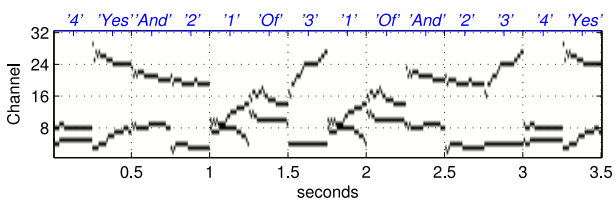



**FIGURE 5 | An example of a stimulus sequence, or "sentence," used as a Probe Stimulus (PS) for the second set of experiments in software simulations.** The stimulus is a concatenation of simplified formant tracks drawn from the set of words illustrated in **Figure 3**. The labels on the upper abcissa (in blue) show the stimulus class. Each "word" is arranged to be 250 ms long, hence the presentation rate in the trials referred to as "normal" is 4 stimuli per second, see section 2.1.4.

that are comparable. For clarity only two of the seven words are shown in **Figure 4** and the extracted formant tracks highlighted using thicker colored lines as in **Figure 3**.

The formant tracks were then smoothed and down-sampled to produce the patterns of current injection that were a simplified representation of the stimulus, see **Figure 5**. These patterns of current injection derived from the formant tracks are stored as 32 × 25 binary patterns used as inputs to the network simulation, with each of the 32 rows representing a frequency channel and each of the 25 columns representing temporal bins of 10 ms. Thus with each monosyllabic word occupying 250 ms presentation at the 'normal' or 100% presentation rate is 4 stimuli per second, a realistic rate for speech. We use the same stimuli presented at other rates (60, 150, 200% of the normal rate of 4 stimuli per second) to investigate robustness to time warping, see section 2.1.4.

Random concatenations of the 21 stimuli produced simplified, formant based representations of nonsense sentences of the type *"three and one of four two four yes and one of two three yes one"* etc,

an example of which is shown in **Figure 5**. The sentences were arranged to contain equal numbers of each stimulus and were presented during the exposure phase without gaps.

***2.1.4.2. Formant track trials and analysis.*** In the exposure phase of the second set of experiments reported in section 3.2 the network was exposed to 20 repetitions (5 s) of all three examples of a single utterance; during this time the learning was switched on. This was followed by the probe phase where all stimuli were presented 50 times in randomized order, without gaps, and with the learning switched off. The output spikes from the $B_2$ were counted for each stimulus and the total number of spikes recorded. This allows the SSI to be calculated for the speech-derived stimuli in the same way as for the FM sweeps in the hardware results—using the methods detailed in section 2.1.3. Sample results are shown in **Figure 9**.

In addition to the SSI it is possible, because these experiments can be interpreted as a set of keyword spotting trials based on spiking rate, to characterize the network as a binary classifier. The output spikes from the $B_2$ neurons were counted during each stimulus, and the total number of spikes recorded; from these data we can construct the ROC and hence the AUC of the responses of the network.

### *2.1.5. Learning predictions*

The third set of results in section 3.3 deals with analytical predictions of what the network, either hardware or software, would learn in ideal circumstances if exposed to an arbitrary stimulus. These analytical predictions of what pattern of learning would result from exposure to a particular stimulus are based on the principle, mentioned in section 2, that the function of the $B_2$ neurons is to learn correlations between activity at different times at different tonotopic channels. Calculating the strength of these correlations should therefore give us an approximation of the connectivity pattern that would result from exposure to any arbitrary stimulus.

We calculate the strength of the correlation, and hence the predicted strength of connectivity, between two network channels $x$ and $y$ after exposure. This can be written $C_{x,y}$ and is calculated as the sum of the products of the stimulus activity $A$, over all times $t$, over all pairs of frequency channels $x$, $y$, taking in to account the time difference caused by the delays in the lateral connections $\Delta_t$, and the time difference between the pre- and post-synaptic spikes that is required by the STDP rule $\epsilon$. The STDP rule also penalizes any activity in $y$ that precedes activity in $x$ thus the pattern of connectivity can be approximated by:

$$C(x,y) = \sum_t \left[ \left( A_{x,t} \cdot A_{y,t+\Delta_t+\epsilon} \right) - \left( A_{x,t} \cdot A_{y,t+\Delta_t-\epsilon} \right) \right]$$

The value of $\Delta_t$ is a function of the channel separation between $x$ and $y$, and the time taken for the activity to propagate between adjacent channels $v$:

$$\Delta_t = |x - y| \cdot v$$

It is important to note that the range of effective values of $v$ is extremely limited in the current hardware due to the implementation of the delays using the variability of time constants that result from the fabrication of the chip (Sheik et al., 2011, 2012). However, although this limitation is taken in to account in the software model results, future hardware designs need not exhibit these limitations if the delays are implemented differently. It is partly to explore these possibilities that results in section 3.3 include examples that employ a wide range of values for $v$.

A simple example of how correlation in the stimulus leads to potentiation of a small set of synapses is illustrated in **Figure 6**. The left subfigure shows activity in a channel followed by activity in another channel some time later, represented by two dots. The propagation of activity through the lateral connections has a fixed offset and a velocity; represented by horizontal and sloping broken gray lines respectively. The right subfigure shows that the synapses connecting neurons in two channels are potentiated if they lie on the broken gray line representing the propagation.

In a second more complex example shown in **Figure 7** the two labeled dots are in exactly the same position as **Figure 6** for comparison. In this case however the stimulus consists of two tones, both rising in frequency but with different rates and starting times. The network can learn the fact that there are two sweep velocities present at the same time as indicated by the predicted connectivity pattern. Because the sweeps are linear the potentiated synapses in red and blue are parallel to the diagonal in the weight matrix. The black synapses are potentiated by the apparent 'up' velocities between pairs of points of different colors as they diverge. Note, there will be no corresponding apparent "down" correlations (below the diagonal) until the sweeps are further apart because of the fixed propagation offset.

## 3. RESULTS

### 3.1. FM SWEEPS

The first set of results were obtained by recording spikes from silicon neurons in a hardware implementation of the network shown in **Figure 1**. Using the spikes recorded form the $B_2$ neurons it

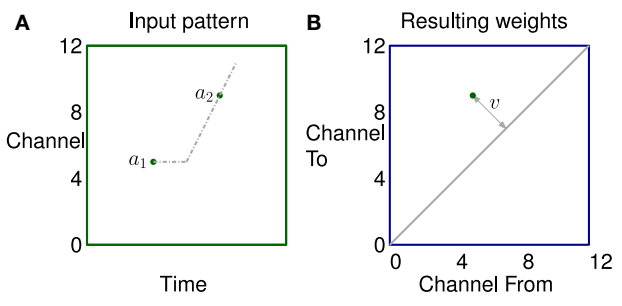



**FIGURE 6 | The learning of a simple correlation in the network. (A)** Activity in a channel followed by activity in another channel some time later, represented by the dots $a_1$ and $a_2$ which are in channels 5 and 9 in this example. The propagation of activity through the lateral connections has a fixed offset and a velocity each represented by broken gray lines in **(A)**. **(B)** Plastic synapses connecting the neuron excited by $a_1$ to the neuron excited by $a_2$ are potentiated if they are on the broken gray line representing the propagation of activity. These synapses are at position (5, 9) on the weight matrix shown by a dot. The distance from the diagonal $v$ is proportional to the apparent sweep velocity from $a_1$ to $a_2$.

is possible to calculate the SSI with respect to all the FM Probe Stimuli (PS) after using each these as Exposure Stimuli (ES). These results are shown in **Figure 8** which summarizes the SSI for all Exposure-Probe stimulus combinations at four noise levels. **Figure 8** shows that the maximum of the SSI occurs often, but not always, at the sweep rate representing the ES. This is in contrast to what we would expect if we were measuring tuning curves. The SSI measures the reduction in uncertainty, or informativeness, provided by the response which is not necessarily at the same place as the response maximum.

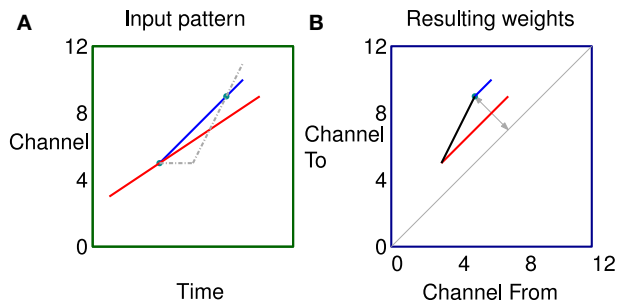



**FIGURE 7 | Correlations in a more complex stimulus. (A)** Stimulus consisting of two rising tones, the two dots are in exactly the same position as **Figure 6** for comparison. **(B)** The network can learn the fact that there are two sweep velocities present at the same time as indicated by the colors. Because the sweeps are linear the potentiated synapses representing the individual sweeps (red and blue) are parallel to the diagonal in the weight matrix. The black synapses are potentiated by the apparent "up" velocities between pairs of points of different colors as they diverge.

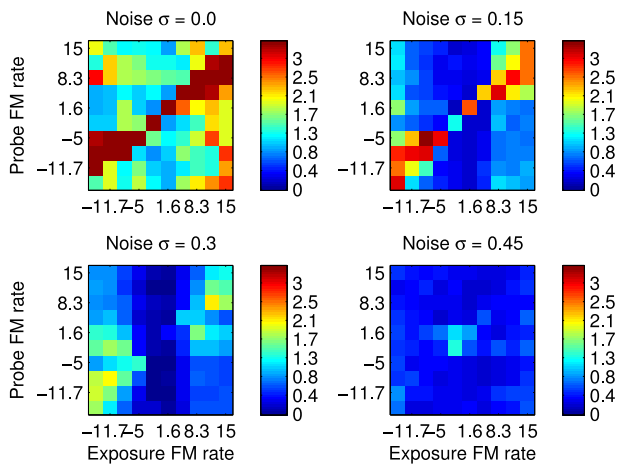



**FIGURE 8 | Robustness to variation in the stimulus using hardware implementation and synthetic FM sweeps.** These plots illustrate the Stimulus Specific Information for trained network and FM sweeps using noisy stimuli. Subfigures represent increasing values of added noise σ that causes the spike pattern to added to and disrupted from the simple sweep produced by current injection in to successive channels as illustrated in **Figure 2**. Color scale is in bits. The maximum value is $\log_2(10) \approx 3.32$ as there are 10 classes of stimuli, interpreting each FM rate as a separate class.

### 3.2. FORMANT TRACKS

The second set of results comes from the software version of the network using the simplified formant track stimuli. These results are collected in the same way as for the results in section 3.1. **Figure 9** shows the SSI for two of the seven classes of Exposure Stimuli; "*Two*" and "*Four.*" The SSI values are shown for the no noise condition ($\sigma = 0.00$ in blue), and for the noisiest condition ($\sigma = 0.45$ in red). The maximum value for the SSI is $\log_2(7) \approx 2.80$ there being 7 classes of stimulus. The maximum SSI is approached in the no noise condition for the ES class in both cases; it is however also clear that there is information in the network response concerning all classes, not only for the class of the Exposure Stimuli. These results are representative of those obtained with all other ES classes.

The next results, shown in **Figure 10**, are ROC curves for trials using one of the seven stimulus classes for training; *"And."* Unlike the SSI results these figures can be obtained only by designating the Exposure Stimuli as belonging to the class to be detected by the network after training; that is treating the network as a binary classifier. Two presentation rates (Rate = 100 and 200%) combined with two noise levels ($\sigma = 0.0$ and 0.45) are shown such as to generate four conditions including the best and worst cases. Other results for this class are intermediate and this pattern is repeated for all other ES classes. Full summary results from the ROC curves presented as Area Under Curve (AUC) for all presentation rates and noise levels are shown, for four representative classes, in **Table 1**. Results for the remaining three classes are comparable.

### 3.3. PREDICTED PATTERNS OF LEARNING

The third and final set of results shows the predicted pattern of connectivity that would result from the exposure of an idealized network to spectrographic representations derived from real sounds. These results are derived from the analytical approach described in section 2.1.5.

First the approach is validated using sound files designed to mimic the simple patterns used in the other experiments previously reported. **Figure 11** shows the predicted connectivity pattern derived from a spectrographic representation of a sound file, alongside a previously reported result from Sheik et al. (2011) using a synthetic stimulus pattern in the hardware

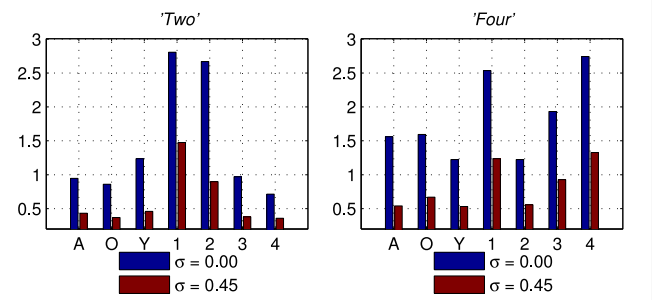



**FIGURE 9 | Stimulus Specific Information results for two of the seven classes of stimuli, *"Two"* and *"Four."*** In Blue is the no noise ($\sigma = 0.00$) condition and in Red the noisiest ($\sigma = 0.45$) condition for comparison.

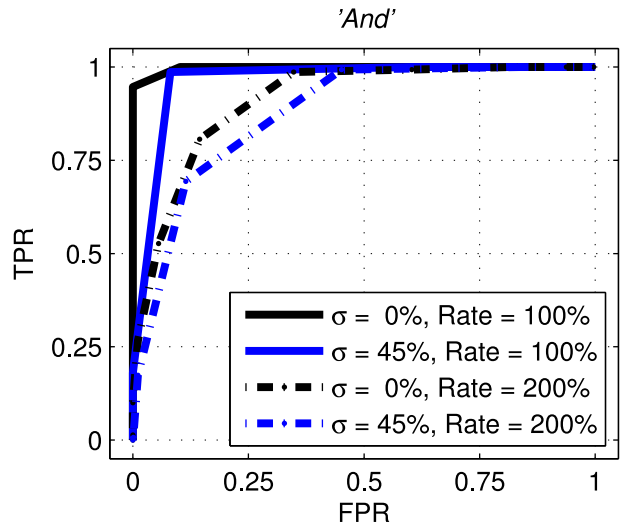



**FIGURE 10 | Representative Receiver Operating curves (ROC) for the network with exposure stimulus *"And."*** ROC curves plot the False Positive Rate (FPR or *fall-out*) against the True Positive Rate (TPR–sometimes called *recall*) for all detection thresholds. For clarity only four conditions are shown with two representing the best and worst case. The solid black line represents the best case with no added noise ($\sigma = 0.00$) and with the probe stimuli presented at the normal rate (Rate = 100%). The broken blue line represents the worst case across all conditions with the added noise at 45% ($\sigma = 0.45$) and the presentation rate at twice the normal rate (Rate = 200%).

**Table 1 | Combined table showing Area Under Curve (AUC) results for all noise conditions and all presentation rates for four of the Exposure Stimuli, *"And","Of","Yes","Four."***

| | Rate | | | |
|---|---|---|---|---|
| | **60%** | **100%** | **150%** | **200%** |
| **"And"** | | | | |
| **σ =** | | | | |
| 0.00% | 0.97 | 0.97 | 0.95 | 0.95 |
| 0.15% | 0.94 | 0.95 | 0.96 | 0.93 |
| 0.35% | 0.92 | 0.92 | 0.92 | 0.90 |
| 0.45% | 0.87 | 0.86 | 0.85 | 0.85 |
| **"Of"** | | | | |
| **σ =** | | | | |
| 0.00% | 0.88 | 0.93 | 0.86 | 0.84 |
| 0.15% | 0.88 | 0.88 | 0.87 | 0.84 |
| 0.35% | 0.85 | 0.84 | 0.85 | 0.80 |
| 0.45% | 0.78 | 0.76 | 0.81 | 0.78 |
| **"Yes"** | | | | |
| **σ =** | | | | |
| 0.00% | 0.95 | 0.98 | 0.96 | 0.95 |
| 0.15% | 0.95 | 0.95 | 0.95 | 0.94 |
| 0.35% | 0.92 | 0.92 | 0.91 | 0.90 |
| 0.45% | 0.88 | 0.85 | 0.86 | 0.81 |
| **"Four"** | | | | |
| **σ =** | | | | |
| 0.00% | 0.96 | 0.97 | 0.97 | 0.95 |
| 0.15% | 0.95 | 0.96 | 0.96 | 0.92 |
| 0.35% | 0.90 | 0.94 | 0.93 | 0.88 |
| 0.45% | 0.86 | 0.87 | 0.85 | 0.82 |

*Example ROC curves for the "And" stimulus can be seen in **Figure 10**. Other ROC and AUC results are comparable in all seven classes.*

implementation. A range of simple patterns give comparable results in hardware and software.

An example of this approach using a recording of a biological communication call is shown in **Figure 13**. The example chosen is a recording of a call from a Weddle Seal; the cochleagraphic representation of this call can be seen in **Figure 12**. These results show the predicted connection patterns that would result from training a network similar to that used in the hardware and simulation experiments. However the results require a wider range of propagation rates between channels than can be achieved with the current hardware.

Four results are illustrated in **Figure 13**, each using a different value of $\nu$, the time taken for activity to propagate between adjacent channels. **Figure 13A** shows the result with the lowest value for $\nu$; note the emphasis on connections below the diagonal indicating down-sweeps and the distance from the diagonal to the lower left maximum of the connectivity represents the "chirp" FM rate of the successive downward sweeps in the seal call. In contrast to A the predicted connectivity in C results from an apparent up sweep. This apparent "up" activity in fact represents correlations between successive down-sweeps, that is the relationship between the maxima of each down-sweep (at low frequency) and the majority of the succeeding down-sweep at higher frequency. B contains features visible in A and C and so best characterizes the stimulus, while the longest value for $\nu$ in **Figure 13D** captures few, if any, of the dynamic features of the stimulus.

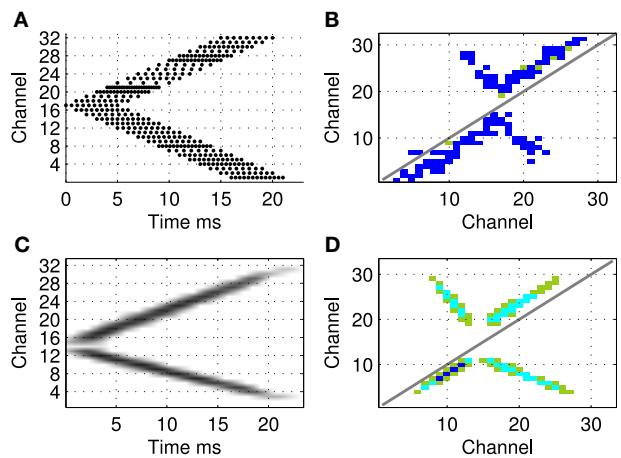



**FIGURE 11 | Comparison between hardware result using a synthetic stimulus pattern (A,B) and learning prediction using a real sound file (C,D).** Top row shows raster of synthetic exposure stimulus **(A)** and resulting network connectivity after exposure **(B)** for hardware network—these figures are taken from Sheik et al. (2011). Bottom row shows spectrogram of comparable sound file **(C)** and the analytically predicted pattern of connectivity **(D)** based on correlations in the stimulus representation as described in section 2.1.5.

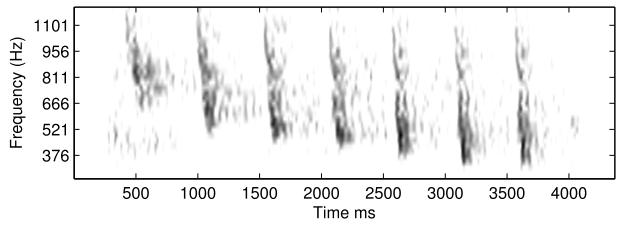



**FIGURE 12 | An example of a natural communication call, in this case a Weddle Seal, shown here as a spectrogram.** This pattern was used to derive the predicted learning patterns shown in **Figure 13**.

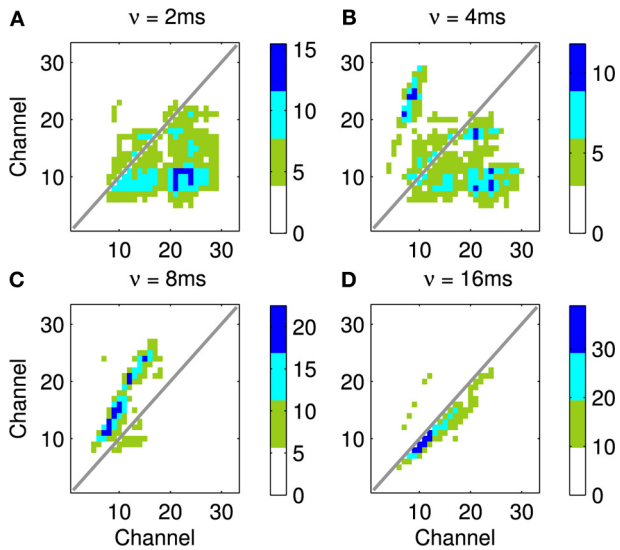



**FIGURE 13 | This figure shows the predicted connectivity of the network, as Figure 11D, but for the Weddle seal call; the spectrogram for this call is shown in Figure 12.** There are four predictions based on a range of values for the propagation rate between channels. **(A–D)** show the predicted pattern of connectivity with increasing values of ν corresponding to the network adapting to correlations at progressively longer time scales. Note that in **(A)** the distance from the diagonal to the lower left local maximum represents the "chirp" FM rate. In **(C)** a clear apparent up-sweep is caused by the low frequency maxima of each chirp and the majority of the next successive chirp being at higher frequency. The **(B)** shows features of **(A,C)**. The **(D)** captures less of the dynamic nature of the stimulus.

## 4. DISCUSSION AND CONCLUSION

The results presented here show that the previously published results and approach (Sheik et al., 2011) are not limited to simple stereotypical stimuli and, even in this highly challenging arena, that there is scope for implementing systems that are robust to realistic signal variability. The stimuli used in these studies exhibit a range of different spectro-temporal properties, are presented continuously rather than in isolation, exhibit wide variability due to added noise, and have a variable presentation rate. All of these complications, and distortions, represent a substantial challenge and are necessary prerequisites to the development of systems that can deployed in real situations.

In section 2.1.5 we discuss how the network is capable of simultaneously representing the position, rate of change, and spectral distance (and to a more limited extent temporal distance) between features in the stimuli. Adaptive sensitivity to all of these has been demonstrated in hardware and software. The robustness in the system is derived from the fact that, although noise and variable presentation rate alter or degrade these patterns of features, it requires either or both types of variability to be present to a very large degree for the degradation to cause the correlations to be masked completely.

We have introduced an information theoretic characterization of the performance of the network, the SSI, based on the variability of the stimuli and the consequent range of responses to a single stimulus class. This represents a method of quantifying the performance of a hardware system that has not been previously reported in an engineering context, but has direct parallels in physiological measurements. The substitution of an information theoretic measure for a classifier is deliberate, because it focusses on the information present in the response rather than the design or performance of the classifier. Our results, summarized in **Figures 8** and **9** indicate that the adaptation of the network to the formative stimulus produces a differential response that is informative with respect to all classes.

Sensory stimuli, in particular auditory stimuli, contain both short and long range temporal correlations. The techniques currently employed in the hardware implementation primarily address correlations only over time scales of the order of synaptic or membrane time constants, up to those represented by the propagation of excitation to adjacent regions. However we have shown that the principles embodied in the network could be extended to longer time scales making it feasible to build systems capable of adapting to complex stimuli, such as animal communication calls. In hardware, longer time scales could be addressed using many levels of recurrence between widely separated layers, as is observed in the mammalian auditory system. Alternatively, from a pragmatic perspective, it could be tackled with working memory and neuromorphic implementations of state machine based approaches (Neftci et al., 2013).

Alongside our previously reported results (Sheik et al., 2011) we pointed out that in order to be useful, the properties of the neuromorphic system we described would have to be validated against noise and other variations in the stimulus, and to be shown to work with more realistic stimuli. We also promised to go beyond the demonstration of emergent sensitivity to a stimulus parameter, and to quantify the increase in acuity in information-theoretic terms; thus providing a basis for the quantitative comparison of networks, connectivity patterns, and learning strategies in the future. In this work we have made significant progress in all of these aims. The approach has been shown to be capable of handling considerable stimulus variation, changes in presentation rate, and the increased complexity of stimulus. Had it fallen at any of these hurdles then the feasibility of the approach would have been called in to question. It is clear, then, that each of these new results is evidence that the approach could lead to a neuromorphic subsystem engineered for dynamic pattern recognition in real world applications.

## ACKNOWLEDGMENTS

### FUNDING

This work was supported by the European Community's Seventh Framework Programme (grants no.231168-SCANDLE and no.257219-neuroP), and by the Cluster of Excellence 277 (CITEC, Bielefeld University). We would like to thank our colleague Robert Mill for many helpful suggestions and contributions made during the work underlying this manuscript.

## REFERENCES

Bibbona, E., Panfilo, G., and Tavella, P. (2008). The Ornstein–Uhlenbeck process as a model of a low pass filtered white noise. *Metrologia* 45, S117. doi: 10.1088/0026-1394/45/6/S17

Brader, J. M., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput.* 19, 2881–2912. doi: 10.1162/neco.2007.19.11.2881

Butts, D. A., and Goldman, M. S. (2006). Tuning curves, neuronal variability, and sensory coding. *PLoS Biol.* 4:e92. doi: 10.1371/journal.pbio.0040092

Chan, V., Liu, S.-C., and van Schaik, A. (2007). AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Trans. Cir. Syst. I* 54, 48–59. doi: 10.1109/TCSI.2006.887979

Coath, M., Mill, R., Denham, S., and Wennekers, T. (2010). "The emergence of feature sensitivity in a recurrent model of auditory cortex with spike timing dependent plasticity," in *Proceedings of BICS 2010* (Madrid).

Ellis, D. P. W. (2005). *Sinewave Speech Analysis/Synthesis in Matlab.* Web resource available online at: http://www.ee.columbia.edu/ln/labrosa/matlab/sws/

Fasnacht, D., and Indiveri, G. (2011). "A PCI based high-fanout AER mapper with 2 GiB RAM look-up table, 0.8 µs latency and 66 mhz output event-rate," in *Conference on Information Sciences and Systems, CISS 2011* (Baltimore, MD: Johns Hopkins University), 1–6.

Fawcett, T. (2006). An introduction to roc analysis. *Patt. Recogn. Lett.* 27, 861–874. doi: 10.1016/j.patrec.2005.10.010

Friston, K. (2005). A theory of cortical responses. *Philos. Trans. R. Soc. Lond. B Biol. Sci.* 360, 815–836. doi: 10.1098/rstb.2005.1622

Glasberg, B. R., and Moore, B. C. (1990). Derivation of auditory filter shapes from notched noise data. *Hear. Res.* 47, 103–138. doi: 10.1016/0378-5955(90)90170-T

Insanally, M. N., Köver, H., Kim, H., and Bao, S. (2009). Feature-dependent sensitive periods in the development of complex sound representation. *J. Neurosci.* 29, 5456–5462. doi: 10.1523/JNEUROSCI.5311-08.2009

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proc. Natl. Acad. Sci. U.S.A.* 110, E3468–E3476. doi: 10.1073/pnas.1212083110

Razak, K. A., and Fuzessery, Z. M. (2008). Facilitatory mechanisms underlying selectivity for the direction and rate of frequency modulated sweeps in the auditory cortex. *J. Neurosci.* 28, 9806–9816. doi: 10.1523/JNEUROSCI.1293-08.2008

Razak, K. A., and Fuzessery, Z. M. (2009). GABA shapes selectivity for the rate and direction of frequency-modulated sweeps in the auditory cortex. *J. Neurophysiol.* 102, 1366–1378. doi: 10.1152/jn.00334.2009

Razak, K. A., and Fuzessery, Z. M. (2010). Development of parallel auditory thalamocortical pathways for two different behaviors. *Front. Neuroanat.* 4:134. doi: 10.3389/fnana.2010.00134

Sheik, S., Chicca, E., and Indiveri, G. (2012). "Exploiting device mismatch in neuromorphic vlsi systems to implement axonal delays," in *The 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane), 1–6.

Sheik, S., Coath, M., Indiveri, G., Denham, S., Wennekers, T., and Chicca, E. (2011). Emergent auditory feature tuning in a real-time neuromorphic vlsi system. *Front. Neurosci.* 6:17. doi: 10.3389/fnins.2012.00017

Ye, C., Poo, M., Dan, Y., and Zhang, X. (2010). Synaptic mechanisms of direction selectivity in primary auditory cortex. *J. Neurosci.* 30, 1861–1868. doi: 10.1523/JNEUROSCI.3088-09.2010

Zhang, L. I., Bao, S., and Merzenich, M. M. (2001). Persistent and specific influences of early acoustic environments on primary auditory cortex. *Nat. Neurosci.* 4, 1123–1130. doi: 10.1038/nn745

Zhang, L. I., Tan, A. Y. Y., Schreiner, C. E., and Merzenich, M. M. (2003). Topography and synaptic shaping of direction selectivity in primary auditory cortex. *Nature* 424, 201–205. doi: 10.1038/nature01796

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 24 September 2013; accepted: 30 December 2013; published online: 17 January 2014.*

*Citation: Coath M, Sheik S, Chicca E, Indiveri G, Denham SL and Wennekers T (2014) A robust sound perception model suitable for neuromorphic implementation. Front. Neurosci. 7:278. doi: 10.3389/fnins.2013.00278*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*

frontiers in
NEUROSCIENCE

ORIGINAL RESEARCH ARTICLE
published: 04 February 2014
doi: 10.3389/fnins.2014.00010

# An efficient automated parameter tuning framework for spiking neural networks

***Kristofor D. Carlson[1], Jayram Moorkanikara Nageswaran[2], Nikil Dutt[3] and Jeffrey L. Krichmar[1,3]****

[1] *Department of Cognitive Sciences, University of California Irvine, Irvine, CA, USA*
[2] *Brain Corporation, San Diego, CA, USA*
[3] *Department of Computer Science, University of California Irvine, Irvine, CA, USA*

***Edited by:***
*Tobi Delbruck, ETH Zurich and University of Zurich, Switzerland*

***Reviewed by:***
*Michael Schmuker, Freie Universität Berlin, Germany*
*Siddharth Joshi, University of California, San Diego, USA*

***Correspondence:**
*Jeffrey L. Krichmar, Department of Cognitive Sciences, University of California Irvine, 2328 Social and Behavioral Sciences Gateway, Irvine, CA 92697-5100, USA*
*e-mail: jkrichma@uci.edu*

As the desire for biologically realistic spiking neural networks (SNNs) increases, tuning the enormous number of open parameters in these models becomes a difficult challenge. SNNs have been used to successfully model complex neural circuits that explore various neural phenomena such as neural plasticity, vision systems, auditory systems, neural oscillations, and many other important topics of neural function. Additionally, SNNs are particularly well-adapted to run on neuromorphic hardware that will support biological brain-scale architectures. Although the inclusion of realistic plasticity equations, neural dynamics, and recurrent topologies has increased the descriptive power of SNNs, it has also made the task of tuning these biologically realistic SNNs difficult. To meet this challenge, we present an automated parameter tuning framework capable of tuning SNNs quickly and efficiently using evolutionary algorithms (EA) and inexpensive, readily accessible graphics processing units (GPUs). A sample SNN with 4104 neurons was tuned to give V1 simple cell-like tuning curve responses and produce self-organizing receptive fields (SORFs) when presented with a random sequence of counterphase sinusoidal grating stimuli. A performance analysis comparing the GPU-accelerated implementation to a single-threaded central processing unit (CPU) implementation was carried out and showed a speedup of 65× of the GPU implementation over the CPU implementation, or 0.35 h per generation for GPU vs. 23.5 h per generation for CPU. Additionally, the parameter value solutions found in the tuned SNN were studied and found to be stable and repeatable. The automated parameter tuning framework presented here will be of use to both the computational neuroscience and neuromorphic engineering communities, making the process of constructing and tuning large-scale SNNs much quicker and easier.

**Keywords: spiking neural networks, parameter tuning, evolutionary algorithms, GPU programming, self-organizing receptive fields, STDP**

## INTRODUCTION

Although much progress has been made in simulating large-scale spiking neural networks (SNNs), there are still many challenges to overcome before these neurobiologically inspired algorithms can be used in practical applications that can be deployed on neuromorphic hardware (Boahen, 2005; Markram, 2006; Nageswaran et al., 2010; Indiveri et al., 2011). Moreover, it has been difficult to construct SNNs large enough to describe the complex functionality and dynamics found in real nervous systems (Izhikevich and Edelman, 2008; Krichmar et al., 2011; Eliasmith et al., 2012). Foremost among these challenges are the tuning and stabilization of large-scale dynamical systems, which are characterized by many state values and open parameters (Djurfeldt et al., 2008). The task of tuning SNNs is becoming more difficult as neuroscientists move away from simpler models toward more realistic, but complex models to describe the properties of network elements (van Geit et al., 2008). For example, many modelers have moved away from simple "integrate and fire" neuron models to models which capture a wider range of neuronal dynamics, but have more open parameters (Izhikevich, 2003; Brette and Gerstner, 2005). A similar shift in complexity is occurring when simulating synaptic plasticity (Abbott and Nelson, 2000), as new types of plasticity models such as homeostatic synaptic scaling (Watt and Desai, 2010; Carlson et al., 2013), short-term plasticity (Mongillo et al., 2008), and spike-timing dependent plasticity (STDP) (Song et al., 2000; van Rossum et al., 2000) are being incorporated into SNNs. In addition, network topologies are shifting from conventional feed-forward connectivity to recurrent connectivity, which have more complex dynamics and require precise tuning of synaptic feedback for stable activity (Seung et al., 2000).

For these reasons, the process of hand-tuning SNNs is often extremely time-consuming and inefficient which has led to interest among researchers in automating this process. To address these challenges, we present an automated tuning framework that utilizes the parallel nature of graphics processing units (GPUs) and the optimization capabilities of evolutionary algorithms (EAs) to tune open parameters of SNNs in a fast and efficient manner.

The present article describes a means to automate parameter tuning of spiking neural networks which are compatible with present and future neuromorphic hardware. However, it is

important to first examine the role SNN models play in the development of neuromorphic hardware. Recent neuromorphic science funding initiatives, such as the SyNAPSE project in the USA and the FACETS/BrainScaleS projects in Europe, have resulted in the construction of neuromorphic chips. Not surprisingly, the research groups involved in producing these neuromorphic hardware devices have also spent a great deal of time building software simulation and interface frameworks (Amir et al., 2013; Thibeault and Srinivasa, 2013). This is because the novel hardware requires new software environments and methodologies to run applications (Brüderle et al., 2011). There are two main software development tasks required to run neuromorphic applications on a hardware device. First, the neuromorphic application must be designed and tuned to perform a particular cognitive or computational function. This is the focus of our present study. Second, the model description of the neuromorphic application must be mapped onto the neuromorphic hardware device computing elements. There have been a number of recent studies that have applied various optimization techniques to solve this mapping problem with some success (Ehrlich et al., 2010; Sheik et al., 2011; Gao et al., 2012; Neftci et al., 2013). Although both tasks are integral to developing neuromorphic hardware applications, the latter is outside the scope of present study.

There has been a great deal of work in the computational neuroscience community on automating the process of parameter tuning neuronal models. A variety of different methodologies have been used to automate parameter tuning in neural models, many of which are summarized in the review by van Geit et al. (2008). Svensson et al. (2012) fit a nine-parameter model of a filter-based visual neuron to experimental data using both gradient following (GF) methods and EAs. Some groups have used optimization techniques to tune ion channels kinetics for compartmental neurons (Hendrickson et al., 2011; Ben-Shalom et al., 2012) while other groups have used quantum optimization techniques and EAs to tune more abstract networks of neurons (Schliebs et al., 2009, 2010). Additionally, brute force methods of searching the parameter space were used to examine a three-neuron model of a lobster stomatogastric circuit by creating large databases of compartmental neurons with varying membrane conductance values and testing the resulting functional behavior of this circuit (Prinz et al., 2003, 2004). Some automated parameter-search tools have been developed as interfaces between neural simulators and the optimization methods used to tune them such as Neurofitter (van Geit et al., 2008). Other tools allow for the automatic compilation of very large sets of simulation runs across a wide range of parameter values and experimental conditions (Calin-Jageman and Katz, 2006).

Unlike these parameter tuning methodologies, which have been applied to small neural circuits, single neurons or networks of hundreds of neurons, our focus is the automated tuning of much *larger* neural systems (on the scale of $10^3$–$10^6$ neurons). Neural networks at these scales become more useful for the description of cognitive models and closer to the scale of SNNs currently being designed to run on neuromorphic hardware (Esser et al., 2013; Thibeault and Srinivasa, 2013). Recent work by Rossant et al. (2011) and Eliasmith et al. (2012) has focused on tuning large-scale SNNs; we compare these approaches with our tuning framework in the discussion section.

A parallel line of research in automated parameter tuning has taken place where larger, more abstract artificial neural networks (ANNs) are constructed using EAs (Fogel et al., 1990). The building of ANNs using EAs can be broken into two basic methodologies: direct encoding and indirect encoding. Much work has been done using the direct encoding approach, where the genetic description of the individual, or the genotype, is directly mapped to the actual traits of the individual, or the phenotype (Hancock, 1992; Gomez and Miikkulainen, 1997; Stanley and Miikkulainen, 2002). An EA is said to use direct encoding when there is a one-to-one correspondence between parameter values, like synaptic weight values and genotype values. Drawbacks of this approach include poor genotype scaling for large network encodings and very large parameter spaces due to the lack of geometrical constraints of the networks. Alternatively, indirect encoding allows the genotype to specify a rule or method for growing the ANN instead of specifying the parameter values directly (Husbands et al., 1998; Beer, 2000; Floreano and Urzelai, 2001; Stanley and Miikkulainen, 2003). NeuroEvolution of Augmented Topologies (NEAT) and HyperNEAT use indirect encoding to evolve network topologies, beginning with a small network and adding complexity to that network as evolution progresses (Stanley and Miikkulainen, 2002; Stanley et al., 2009; Clune et al., 2011; Risi and Stanley, 2012). HyperNEAT has been used to encode networks with as many as 8 million connections and networks evolved using NEAT have been used in food-gathering tasks (Stanley et al., 2009), in a checkers-playing ANN that exhibits topographic mappings (Gauci and Stanley, 2010), and in evolving robot gaits in hardware (Yosinski et al., 2011). The present study utilizes the indirect encoding approach, in which the learning parameters are evolved, as opposed to the direct encoding approach where the synaptic weights are evolved. This allows for a large reduction in the parameter space. Although EAs are an effective tool for constructing ANNs, they often require long execution times to produce well-tuned networks. A number of parallel computing techniques can be used to reduce the execution time of EAs, however, this paper focuses mainly on parallelization via GPU computing.

With the development of mature GPU parallel computing platforms like CUDA (Nickolls et al., 2008) and OpenCL (Stone et al., 2010), GPU accelerated algorithms have been applied to a variety of tasks in scientific computing. GPU acceleration has been used to increase the throughput of EAs (Maitre et al., 2009), simulate neural field models of the primary visual cortex V1 (Baladron et al., 2012), and search parameter spaces in bio-inspired object-recognition models (Pinto et al., 2009). In addition to these applications, a number of research groups in the computational neuroscience community (Brette and Goodman, 2012) have developed and implemented parallel implementations of SNNs on GPUs (Bernhard and Keriven, 2006; Fidjeland et al., 2009; Nageswaran et al., 2009b; Bhuiyan et al., 2010; Han and Taha, 2010; Hoffmann et al., 2010; Yudanov et al., 2010; Ahmadi and Soleimani, 2011; Nowotny, 2011; Thibeault et al., 2011; de Ladurantaye et al., 2012; Mirsu et al., 2012; Pallipuram et al., 2012). GPU-driven SNN simulators have already been used

in SNN models of the basal forebrain (Avery et al., 2012), the basal ganglia (Igarashi et al., 2011), the cerebellum (Yamazaki and Igarashi, 2013), and the olfactory system (Nowotny, 2010).

Our present study drastically decreases the time it takes to tune SNN models by combining a freely available EA library with our previous work (Nageswaran et al., 2009b; Richert et al., 2011), which consisted of a parallelized GPU implementation of an SNN simulator. Although other research groups have used EAs and GPUs to tune SNNs (Rossant et al., 2011), our approach is more general as it tunes a variety of SNN parameters and utilizes fitness functions that can be broadly applied to the behavior of the entire SNN. As a proof of concept, we introduce a parameter tuning framework to evolve SNNs capable of producing self-organized receptive fields similar to those found in V1 simple cells in response to patterned inputs. An indirect encoding approach was utilized as the parameters tuned in the SNN governed Hebbian learning, homeostasis, maximum input stimulus firing rates, and synaptic weight ranges. A performance analysis compared the parallelized GPU implementation of the tuning framework with the equivalent central processing unit (CPU) implementation and found a speedup of 65× (i.e., 0.35 h per generation vs. 23.5 h per generation) when SNNs were run concurrently on the GPU. Using affordable, widely-accessible GPU-powered video cards, the software package presented here is capable of tuning complex SNNs in a fast and efficient manner. The automated parameter tuning framework is publicly available and could be very useful for the implementation of large-scale SNNs on neuromorphic hardware or for the development of large-scale SNN simulations that describe complex brain functions.

## METHODS

### GPU ACCELERATED SNNs IN CARLsim

An important feature of the automated parameter tuning framework is the ability to run multiple SNNs in parallel on the GPU, allowing significant acceleration of the EA evaluation phase. We first briefly review the approaches CARLsim uses to run SNNs in parallel before describing the general layout of the automated parameter tuning framework and describe how a researcher would use the tool to tune SNNs. **Figure 1** shows the basic CUDA GPU architecture, which consists of a multiple streaming multiprocessors (SMs) and a global memory, accessible to all SMs. Each SM is comprised of multiple floating-point scalar processors (SPs), at least one special function unit (SFU), and a cache/shared memory. CUDA code is distributed and executed in groups of 32 threads called warps. Each SM has at least one warp scheduler that ensures maximum thread concurrency by switching from slower to faster executing warps. Our simulations utilized an NVIDIA Tesla M2090 GPU with 6 GB of global memory, 512 cores (each operating at 1.30 GHz) grouped into 16 SMs (32 SPs per SM), and a single precision compute power of 1331.2 GFLOPS.

The CARLsim parallel GPU implementation was written to optimize four main performance metrics: parallelism, memory bandwidth, memory usage, and thread divergence which are discussed in greater detail in (Nageswaran et al., 2009a). The term parallelism refers to both the degree to which the application data is mapped to parallel threads and the structure of the mapping itself. CARLsim utilizes both neuronal parallelism (N-parallelism), where individual neurons are mapped to processing elements and simulated in parallel, and synaptic parallelism (S-parallelism), where synaptic data are mapped to processing elements and simulated in parallel. Anytime a neuronal state variable is updated, N-parallelism is used, and anytime a weight update is necessary, S-parallelism is used. Sparse representation techniques such as the storage of SNN data structures using the reduced Address Event Representation (AER) format and the use of a circular queue to represent firing event data decrease both memory and memory bandwidth usage. GPUs execute many threads concurrently (1536 threads per SM in the Tesla M2090) and manage these threads by providing a thread scheduler for each SM which organizes groups of threads into warps.

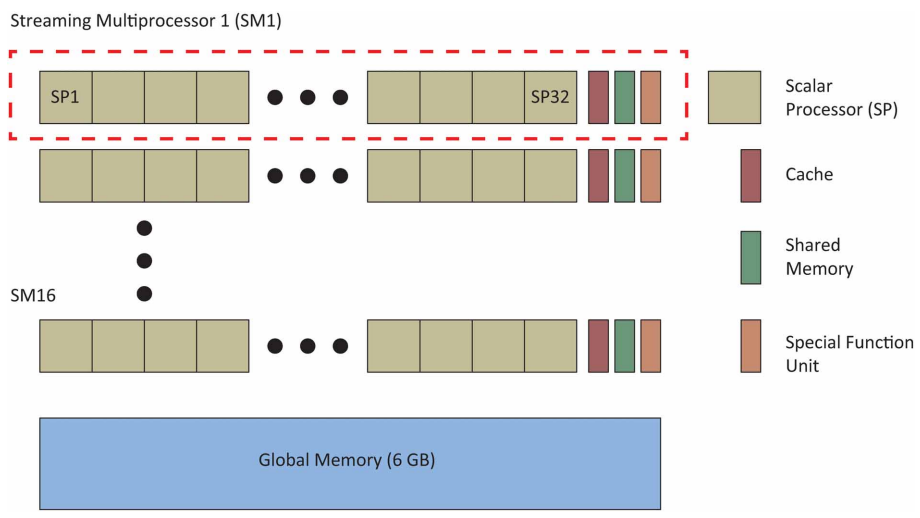



**FIGURE 1 | A simplified diagram of NVIDIA CUDA GPU architecture (adapted from Nageswaran et al., 2009a,b).** Our simulations used an NVIDIA Tesla 2090 GPU that had 16 streaming multiprocessors (SM) made up of 32 scalar processors (SPs) and 6 GB of global memory.

Thread/warp divergence occurs when threads in a single warp execute different operations, forcing the faster executing threads to wait until the slower threads have completed. In CARLsim, thread/warp divergence is minimized during diverging loop executions by buffering the data until all threads can execute the diverging loop simultaneously.

### AUTOMATED PARAMETER TUNING FRAMEWORK DESCRIPTION

To test the feasibility of an automated parameter tuning framework, our group used EAs to tune open parameters in SNNs running concurrently on a GPU. As a proof of concept, the SNNs were evolved to produce orientation-dependent stimulus responses similar to those found in simple cells of the primary visual cortex (V1) through the formation self-organizing receptive fields (SORFs). The general evolutionary approach was as follows: (1) A population of neural networks was created, each with a unique set of neural parameter values that defined overall behavior. (2) Each SNN was then ranked based on a fitness value assigned by the objective function. (3) The highest ranked individuals were selected, recombined, and mutated to form the offspring for the next generation. (4) This process continued until a desired fitness was reached or until other termination conditions were met (**Figure 2A**).

The automated parameter tuning framework consisted of three software packages and is shown in **Figure 2B**. The framework includes: (1) the CARLsim SNN simulator (Richert et al., 2011), (2) the Evolving Objects (EO) computational framework, a publically available evolutionary computation toolkit (Keijzer et al., 2002), and (3) a Parameter Tuning Interface (PTI), developed by our group, to provide an interface between CARLsim and EO. Evolving Objects is available at http://eodev.sourceforge.net/ and both CARLsim and the PTI are available at http://www.socsci.uci.edu/~jkrichma/CARLsim/. The EO computational framework runs the evolutionary algorithm on the user-designated parameters of SNNs in CARLsim. The PTI allows the objective function to be calculated independent of the EO computation framework. Parameter values are passed from the EO computation framework through the PTI to the SNN in CARLsim where the objective function is calculated. After the objective function is executed, the results are passed from the SNN in CARLsim through the PTI back to the EO computation framework for processing by the EA. With this approach, the fitness function calculation, which involves running each SNN in the population, can be run in parallel on the GPU while the remainder of EA calculations can be performed using the CPU (**Figure 2B**).

### USING THE PARAMETER TUNING INTERFACE

In addition to providing a means for CARLsim and EO to exchange data, the PTI hides the low level description and configuration of EO from the user by providing a simple application programming interface (API). Before using the PTI, the user must have a properly configured EO parameter file, which is a plain text file that provides the user with control over an EO

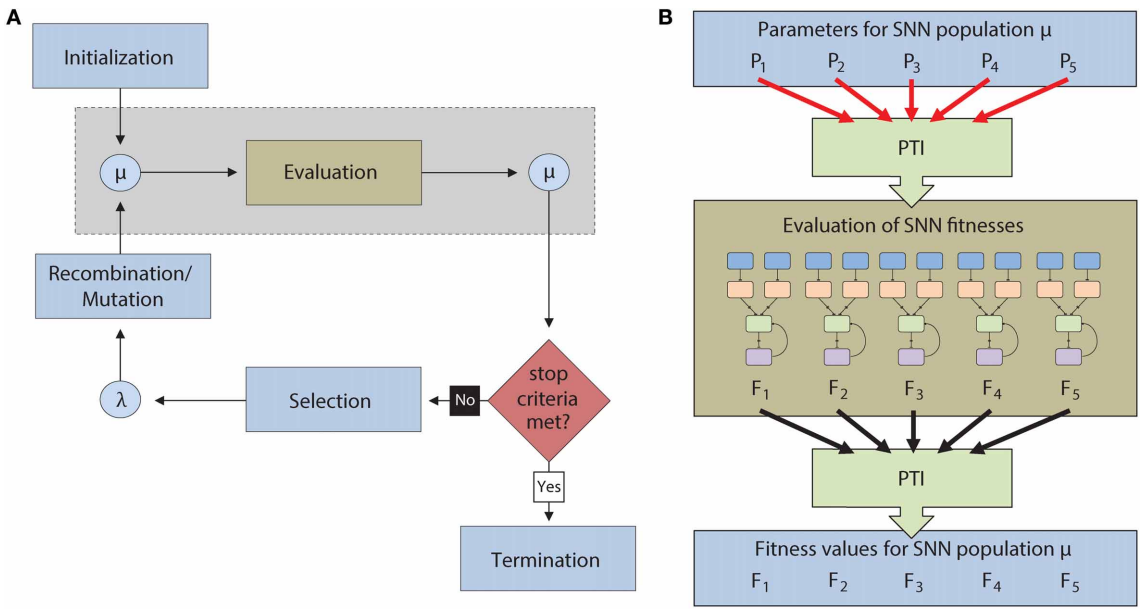



**FIGURE 2 | (A)** Flow chart for the execution of an Evolutionary Algorithm (EA). A population of individuals (μ) is first initialized and then evaluated. After evaluation, the most successful individuals are selected to reproduce via recombination and mutation to create an offspring generation (λ). The offspring then become parents for a new generation of the EA. This continues until a termination condition is reached. The light blue boxes denote operations that are carried out serially on the CPU while the light brown box denotes operations carried out in parallel on the GPU. The operations inside the dotted gray box are described in greater detail in **(B)**. **(B)** Description of the automated parameter tuning framework consists of the CARLsim SNN simulator (light brown), the EO computational framework (light blue), and the Parameter Tuning Interface (PTI) (light green). The PTI passes tuning parameters ($P_N$) to CARLsim for evaluation in parallel on the GPU. After evaluation, fitness values ($F_N$) are passed from CARLsim back to EO via the PTI.

configuration. An example of an EO parameter file is shown in Supplementary 1 of the supplementary materials. At execution, EO reads the parameter file to configure all aspects of the EA, including selection, recombination, mutation, population size, termination conditions, and many other EA properties. Beginners to EO can use the example EO parameter files included with the EO source code for the automated parameter tuning framework presented here. A sample program overview of the PTI and a summary of the PTI-API are included in Supplementary materials sections 2 and 3. Additional EO examples and documentation can be found online at http://eodev.sourceforge.net/eo/tutorial/html/eoTutorial.html. After creating a valid EO parameter file, the user is ready to use the PTI and CARLsim to tune SNNs.

## EVOLVING SNNs WITH V1 SIMPLE CELL RESPONSES AND SORF FORMATION

As a proof of concept, the ability of the automated parameter tuning network to construct an SNN capable of producing SORFs and orientation-dependent stimulus responses was examined. This set of simulations was presented with grayscale counterphase gratings of varying orientations. The EO computation framework evolved SNN parameters that characterized spike-timing dependent plasticity (STDP), homeostasis, the maximum firing rates of the neurons encoding the stimuli, and the range of weight values for non-plastic connections. The network topology of the SNN, shown in **Figure 3**, modeled the visual pathway from the lateral geniculate nucleus (LGN) to the primary visual cortex (V1).

Each individual in the population participated in a training phase, where synaptic weights were modified according to STDP and homeostatic learning rules, and a testing phase where a multi-component objective function was used to evaluate an individual's ability to reproduce V1 simple-cell behavior. The training phase consisted of the presentation of 40 grayscale sinusoidal grating patterns of varying orientation (from $\pi/40$ to $\pi$) in random sequence to the SNN for approximately 100 min. Each pattern was presented to the network for 2 s while 1 Hz Poisson noise was applied to the network for 500 ms between pattern presentations. During the testing phase eight grating orientations (from $\pi/8$ to $\pi$) were presented to the network and the firing rate responses of the four output neurons in the Exc group were recorded. STDP and homeostasis were enabled during the training phase but were disabled for the testing phase. The evolutionary algorithm began with the random initialization of the parent population, consisting of 10 SNNs, and produced 10 offspring per generation. Ten SNN configurations ran in parallel. To evolve V1 simple cell responses, a real-valued optimization algorithm called Evolution Strategies (De Jong, 2002) was used with deterministic tournament selection, weak-elitism replacement, 40% Gaussian mutation and 50% crossover. Weak-elitism ensures the overall fitness monotonically increases each generation by replacing the worst fitness individual of the offspring population with the best fitness individual of the parent population. Fourteen parameters were evolved: four parameters associated with E → E STDP, four parameters associated with E → I STDP, the homeostatic target firing rates of the Exc and Inh groups, the strength of the fixed uniformly random On(Off)Buffer → Exc group connections, the strength of the plastic Exc → Inh group connections, the strength of the fixed uniformly random Inh → Exc group connections, and the maximum firing rate response to the input stimuli. The range of allowable values for each parameter is shown

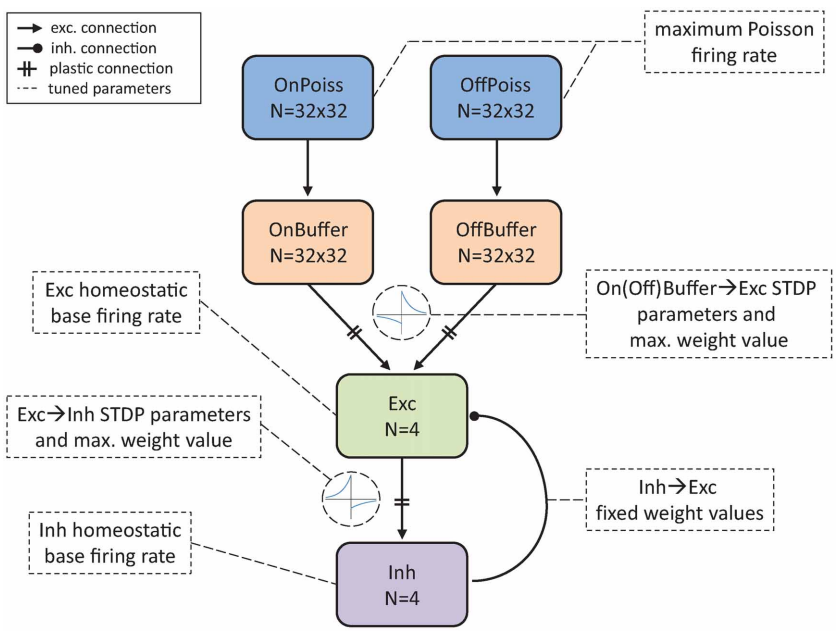



**FIGURE 3 | Network architecture of the SNN tuned by the parameter tuning framework to produce V1 simple cell response and SORFs.** N represents the number of neurons used in different groups. E → E and E → I STDP curves are included to describe plastic On(Off)Buffer → Exc and Exc → Inh connections. Tuned parameters are indicated with dashed arrows and boxes.

in **Table 1**. The parameter ranges for the STDP time windows were constrained by experimental data (Caporale and Dan, 2008) while the remaining parameter ranges were chosen to produce SNNs with biologically realistic firing rates.

The multi-component objective function was constructed by requiring output neurons to have desirable traits in neuronal response activity, namely, decorrelation, sparseness, and a description of the stimulus that employs the entire response range. The total fitness function to be maximized, $\text{fitness}_{\text{total}}$, is described by Equation (1) and required each fitness component in the denominator to be minimized. Fitness values were normalized by the highest fitness value and ranged from 0 to 1. The fitness function consisted of three fitness components, $\text{fitness}_{\text{decorr}}$, $\text{fitness}_{\text{Gauss}}$, $\text{fitness}_{\text{maxRate}}$, and a scaling factor $K$ which had a value of 4.4 in all simulations discussed here.

$$\text{fitness}_{\text{total}} = \frac{1}{\text{fitness}_{\text{decorr}} + \text{fitness}_{\text{Gauss}} + K_{\text{scaling factor}} \cdot \text{fitness}_{\text{maxRate}}} \tag{1}$$

Here $\text{fitness}_{\text{decorr}}$, described in Equation (2), was minimized if each output neuron responded uniquely and preferentially to a grating orientation, causing the average firing rates of each neuron to be decorrelated. The fitness component, $\text{fitness}_{\text{Gauss}}$, was minimized when each Exc group neuron had an idealized Gaussian tuning curve response and is defined in Equation (4). The fitness component, $\text{fitness}_{\text{maxRate}}$, was minimized when the maximum firing rate of the output neurons achieved a target firing rate, which helped neuronal activity remain stable and sparse, and is defined in Equation (6). A scaling term, $K_{\text{scaling factor}} = 4.4$, was used to correctly balance the maximum firing rate requirement against the decorrelation and Gaussian tuning curve curve requirements. Taken together, both $\text{fitness}_{\text{maxRate}}$ and $\text{fitness}_{\text{Gauss}}$ result in the assignment of high fitness values to neurons that have a stimulus response that utilizes the entire neuronal response range from approximately 0 to 60 Hz, which is an important aspect of neuronal activity.

**Table 1 | Range of allowable values for parameters optimized by the automated parameter tuning framework.**

| Parameters | Range |
|---|---|
| Max. Poiss. Rate | 10–40 Hz |
| Buff → Exc Wts | 4.0e-3–1.6e-2 |
| Exc → Inh Wts | 0.1–1.0 |
| Inh → Exc Wts | 0.1–0.5 |
| $R_{\text{target}}$ Exc | 10–30 Hz |
| $R_{\text{target}}$ Inh | 40–100 Hz |
| $A_+$ Exc | 9.6e-6–4.8e-5 |
| $A_-$ Exc | 9.6e-6–4.8e-5 |
| $\tau_+$ Exc | 10–60 ms |
| $\tau_-$ Exc | 5–100 ms |
| $A_+$ Inh | 9.6e-6–4.8e-5 |
| $A_-$ Inh | 9.6e-6–4.8e-5 |
| $\tau_+$ Inh | 10–60 ms |
| $\tau_-$ Inh | 5–100 ms |

*Weight ranges and STDP $A_+$ and $A_-$ parameters are dimensionless and their relative magnitudes are important for creating a functional SNN.*

The $\text{fitness}_{\text{decorr}}$ component of the fitness function enforced decorrelation in the Exc group neuronal firing rates so that each neuron responded maximally to a different stimulus presentation angle. Equation (2) ensured the angles of maximum response $\theta^i_{\text{max}}$ for each neuron, $i$, were as far from one another as possible by minimizing the difference between the two closest maximum angles ($D^i_{\text{min}}$) and the maximum possible value of $D^i_{\text{min}}$, called $D_{\text{target}}$. $D^i_{\text{min}}$ is described in Equation (3) and $D_{\text{target}}$ had a value of $\pi/4$.

$$\text{fitness}_{\text{decorr}} = \sum_{i=1}^{N=4} \left| D^i_{\text{min}} - D_{\text{target}} \right| \tag{2}$$

$$D^i_{\text{min}} = \min\left( \left| \theta^i_{\text{max}} - \theta^j_{\text{max}} \right| \right) \ \forall\ j \neq i \tag{3}$$

The next fitness component $\text{fitness}_{\text{Gauss}}$ ensured that each Exc group neuron had a Gaussian tuning curve response similar to that found in V1 simple cells. The difference between the normalized firing rate $R^i_j$ and a normalized Gaussian $G^i_j$ was calculated for every presentation angle for each Exc group neuron and was summed over all angles and neurons. This is shown in Equation (4) while a description of the Gaussian is shown in Equation (5), where $r^i_{\text{max}}$ is the maximum firing rate for the $i$th neuron, $\theta^i_{\text{max}}$ is the angle of maximum response of the $i$th neuron, $\theta_j$ is the $j$th stimulus angle, and $\sigma$ was chosen to be $15\pi/180$ to match experimental observations (Henry et al., 1974).

$$\text{fitness}_{\text{Gauss}} = \sum_{i=i}^{N=4} \sum_{j=1}^{M=40} \left| R^i_j - G^i_j \right| \tag{4}$$

$$G^i_j = r^i_{\text{max}} \exp\left[ -\frac{1}{2} \left( \frac{\theta_j - \theta^i_{\text{max}}}{\sigma} \right)^2 \right] \tag{5}$$

The $\text{fitness}_{\text{maxRate}}$ component, in combination with the Inh → Exc group connections, helped to enforce the requirement that the Exc group neurons had sparse firing rates by limiting the firing rate of each neuron to a maximum target firing rate $R^{\text{max}}_{\text{target}}$ of 60 Hz. The difference between the maximum firing rate $R^i_{\text{max}}$ of each Exc group neuron and the maximum target firing rate was calculated and summed over all Exc group neurons as shown in Equation (6).

$$\text{fitness}_{\text{maxRate}} = \sum_{i=1}^{N=4} \left| R^i_{\text{max}} - R^{\text{max}}_{\text{target}} \right| \tag{6}$$

Each fitness component had a fitness constraint imposed on it which caused the individual to be assigned a poor overall fitness if it fell outside a particular range of values. Recall that the fitness components are in the denominator of the total fitness equation making lower fitness component values more fit than higher fitness component values. The constraints are expressed as upper limits. Those individuals with fitness components larger than the

upper limit were assigned poor overall fitness values by adding 240 to the denominator of Equation (1). The fitness component $\text{fitness}_{\text{decorr}}$ had an upper limit constraint of 15, the fitness component $\text{fitness}_{\text{Gauss}}$ had an upper limit of 1300, and the fitness component $\text{fitness}_{\text{maxRate}}$ had an upper limit of 160.

### NETWORK MODEL

The input to the network consisted of a $32 \times 32$ grid of grayscale pixels, ranging from -1 to 1, which were connected to a pair of $32 \times 32$ Poisson spiking neuron groups with one-to-one topology to model the On/Off receptive fields found in the LGN. One Poisson spiking neuron group, the OnPoiss group, had linear spiking responses corresponding to Equation (7) while the OffPoiss group had responses corresponding to Equation (8). Here, $r_{i,\text{On}}$ ($r_{i,\text{Off}}$) represent the firing rate of neuron $i$, of the On(Off)Poiss group in response to the value of the input $p$, pixel $i$. The rates had maximum values of 1 and were scaled with the *Max. Poiss. Rate* parameter. Each On(Off)Poiss group had fixed excitatory synapses with one-to-one connectivity to a pair of $32 \times 32$ spiking neuron groups consisting of regular spiking (RS) Izhikevich neurons (Izhikevich et al., 2004), called the On(Off)Buffer groups. The On(Off)Buffer group neurons have a refractory period and were included to produce more realistic spike dynamics in response to the stimulus input. The On(Off) Buffer groups were included because Poisson spiking neurons with a built-in delay period were not part of the standard NVIDIA CUDA Random Number Generation (cuRAND) library and were therefore, more difficult to generate. On(Off)Buffer neurons had plastic excitatory synapses with all-to-all connectivity to an output group of RS neurons called the Exc group. Finally, to induce competition and encourage sparse firing, the Exc group made plastic excitatory all-to-all connections to a fast-spiking (FS) inhibitory neuron group (Izhikevich et al., 2004), which made fixed inhibitory all-to-all connections back to the Exc group.

$$r_{i,\text{On}}\left(p_i\right) = \begin{cases} p_i, & p_i > 0 \\ 0, & p_i \leq 0 \end{cases} \tag{7}$$

$$r_{i,\text{Off}}\left(p\right) = \begin{cases} 0, & p_i > 0 \\ \left|p_i\right|, & p_i \leq 0 \end{cases} \tag{8}$$

The mathematical description of the Poisson spiking neurons used in the simulation is shown in Equation (9).

$$t_{i+1} = t_i - \ln\left(x_i\right)/r \tag{9}$$

The spike times were generated iteratively by generating inter-spike intervals (ISIs) from an exponential distribution (Dayan and Abbott, 2001). Here $t_i$ is the spike time of the current spike, $t_{i+1}$ is the spike time of the next spike, $r$ is the average firing rate, and $x_i$ is the current random number (uniformly distributed between 0 and 1) used to generate the next spike time.

The spiking neurons used in the simulation were Izhikevich-type neurons and were chosen because they are computationally efficient and able to produce neural dynamics with a high degree of accuracy (Izhikevich, 2003). All excitatory neurons were modeled as RS neurons while all inhibitory neurons were modeled as FS neurons. The dynamics of Izhikevich neurons are shown in Equations (10, 11) and consist of a 2D system of ordinary differential equations.

$$\frac{d\upsilon}{dt} = 0.04\upsilon^2 + 5\upsilon + 140 - u + I \tag{10}$$

$$\frac{du}{dt} = a(b\upsilon - u) \tag{11}$$

Here, $\upsilon$ is the membrane potential of the neuron and $u$ is the recovery variable. The neuron dynamics for spiking are as follows: If $\upsilon \geq 30$ mv, then $\begin{cases} \upsilon \leftarrow c \\ u \leftarrow u + d \end{cases}$. The variables $a$, $b$, $c$, and $d$ are specific to the type of Izhikevich neuron being modeled. For RS neurons, $a = 0.02$, $b = 0.2$, $c = -65.0$, and $d = 8.0$. For FS neurons, $a = 0.1$, $b = 0.2$, $c = -65.0$, and $d = 2.0$. The synaptic input for the spiking neurons consisted of excitatory NMDA and AMPA currents and inhibitory $\text{GABA}_\text{A}$ and $\text{GABA}_\text{B}$ currents (Izhikevich and Edelman, 2008) and has the form shown in Equation (12). Each conductance has the general form of $g(\upsilon - \upsilon_0)$ where $g$ is the conductance, $\upsilon$ is the membrane potential, and $\upsilon_0$ is the reversal potential.

$$\begin{aligned} I = {} & g_{\text{NMDA}} \frac{\left[\frac{\upsilon+80}{60}\right]^2}{1+\left[\frac{\upsilon+80}{60}\right]^2}(\upsilon - 0) + g_{\text{AMPA}}\left(\upsilon - 0\right) \\ & + g_{\text{GABA}_\text{A}}\left(\upsilon + 70\right) + g_{\text{GABA}_\text{B}}\left(\upsilon + 90\right) \end{aligned} \tag{12}$$

The conductances obey the first order dynamics shown in Equation (13).

$$\frac{dg_i}{dt} = -\frac{g_i}{\tau_i} \tag{13}$$

Here $i$ denotes a particular conductance (NMDA, AMPA, $\text{GABA}_\text{A}$, or $\text{GABA}_\text{B}$) and $\tau$ denotes the decay constant for the conductance. The decay constants are $\tau_{\text{NMDA}} = 100$ ms, $\tau_{\text{AMPA}} = 5$ ms, $\tau_{\text{GABA}_\text{A}} = 6$ ms, and $\tau_{\text{GABA}_\text{B}} = 150$ ms.

All plastic connections used a standard nearest-neighbor STDP implementation (Izhikevich and Desai, 2003) but distinct STDP rules were used for STDP occurring between excitatory-to-excitatory ($\text{E} \rightarrow \text{E}$) neurons and STDP occurring between excitatory-to-inhibitory ($\text{E} \rightarrow \text{I}$) neurons. Excitatory-to-excitatory plastic connections had traditional STDP curves as detailed in (Bi and Poo, 1998) while excitatory-to-inhibitory plastic connections used STDP curves where potentiation occurred for pre-after-post pairings and depression occurred for pre-before-post pairings as found in experiments (Bell et al., 1997). A model for homeostatic synaptic scaling (Carlson et al., 2013) was also included to prevent runaway synaptic dynamics that often arise in STDP learning rules.

The STDP update rule used in our simulations is shown in Equation (14).

$$\frac{dw_{i,j}}{dt} = \delta + \beta\left(\text{LTP}_{i,j} + \text{LTD}_{i,j}\right) \tag{14}$$

The synaptic weight from presynaptic neuron $i$ to postsynaptic neuron $j$ is indicated by the variable $w_{i,j}$. Additionally, $\delta$ is a bias often set to zero or a positive number to push the network toward positive weight increases for low synaptic input, while $\beta$ is the learning rate. The weight changes were updated every time step (1 ms) but the weights themselves are modified once every 1 s.

To model homeostatic synaptic plasticity the STDP update rule was modified as shown in Equation (15) where $\alpha = 0.1$ and $\beta = 1.0$.

$$\frac{dw_{i,j}}{dt} = \left[\alpha \cdot w_{i,j}\left(1 - \frac{\overline{R}}{R_{\text{target}}}\right) + \beta\left(\text{LTP}_{i,j} + \text{LTD}_{i,j}\right)\right] \cdot \text{K} \quad (15)$$

Here, $\alpha$ is the homeostatic scaling factor while $\overline{R}$ and $R_{\text{target}}$ are the average and target firing rates, respectively, for the postsynaptic neuron, $j$. A stability term denoted by K, damps oscillations in the weight updates and speeds up learning. K is defined as:

$$\text{K} = \frac{\overline{R}}{T \cdot \left(1 + \left|1 - \overline{R}/R_{\text{target}}\right| \cdot \gamma\right)} \quad (16)$$

In Equation (16), $\gamma$ is a tuning factor and $T$ is the time scale over which the firing rate of the postsynaptic neuron is averaged. Here $\gamma = 50$ and $T = 10$ s.

### SIMULATION DETAILS

The SORF formation and performance analysis simulations were developed and implemented on a publically available neural simulator (Nageswaran et al., 2009b; Richert et al., 2011) and the forward Euler method (Giordano and Nakanishi, 2006) was used to integrate the difference equations with a step size of 1 ms for plasticity equations and 0.5 ms for neuronal activity equations. The CPU version of CARLsim was run on a system with an Intel Core i7 2.67 GHz quad-core processor with 6 GB of memory. The GPU version of CARLsim was run on a NVIDIA Tesla GPU M2090 card, with 6 GB of total memory and 512 cores. The GPU was capable of 665 GFLOPS of double precision, 1.33 TFLOPs of single precision, and had a memory bandwidth of 117 GB/s. The GPU was in a 12-core CPU cluster with 24 GB of memory and 4 GPU cards. Simulations executed on the CPU were single-threaded, while simulations executed on the GPU were parallelized, but only on a single GPU.

## RESULTS

An SNN capable of SORF formation and V1 simple cell like responses to counterphase grating stimuli presentation was constructed using the automated parameter tuning framework described above. Using a configuration with 10 SNNs running simultaneously on the GPU, each having 4104 neurons, the automated parameter tuning framework took 127.2 h to complete 287 generations of the EA and used a stopping criterion that halted the EA after the completion of 100 generations without a change in the fitness of the best individual or after the completion of 500 generations. The average and best fitness values for every generation are shown in red and blue, respectively, in **Figure 4**. The automated parameter tuning framework constructed 128 SNNs out of 2880 total SNNs (4.4%) that displayed SORF formation and V1 simple cell like responses and produced the first of these

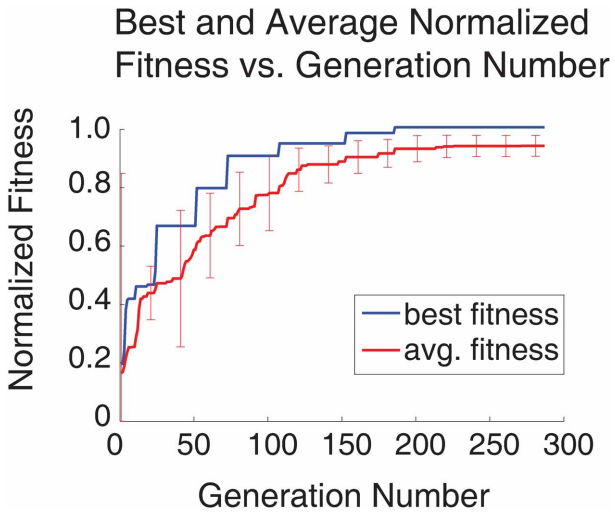



**FIGURE 4 | Plot of best and average fitness vs. generation number for entire simulation run (287 generations, 4104 neuron SNNs, 10 parallel configurations).** All values were normalized to the best fitness value. The error bars denote the standard deviation for the average fitness at intervals of once per 20 generations. Initially the standard deviation of the average fitness is large as the EA explores the parameter space, but over time, the standard deviation decreases as the EA finds better solutions.

**Table 2 | Sorted fitness values (higher is better) for the initial and final SNN populations.**

| Initial population fitness values | Final population fitness values |
|---|---|
| 0.1949 | 1.0000 |
| 0.1780 | 0.9807 |
| 0.1594 | 0.9481 |
| 0.1551 | 0.9454 |
| 0.1444 | 0.9384 |
| 0.1399 | 0.9294 |
| 0.1212 | 0.9146 |
| 0.1006 | 0.9107 |
| 0.0977 | 0.9105 |
| 0.0913 | 0.9040 |

*Entries with a shaded background denote SNNs with V1 simple cell responses and SORF formation for every Exc group neuron (4).*

SNNs at generation 52. **Table 2** shows the fitness values of the 10 initial SNNs and the fitness values after 287 generations. Shaded table entries denote SNNs that produced SORFs and V1 simple cell-like tuning curves for *all* four Exc group neurons while unshaded table entries indicate SNNs that failed to produce these neural phenomena. All SNNs from the initial population had very low fitness, produced no orientation selectivity, and had no SORF formation. All SNNs from the final population except for the last individual (fitness = 0.9040) had high fitness values, produced V1 simple cell-like tuning curve responses, and produced SORFs. The last individual in **Table 2**, had a high fitness, but only produced V1 simple-cell like tuning curve responses and SORFs in three of the four Exc group neurons.

## EVOLVING SNNs WITH V1 SIMPLE CELL RESPONSES AND SORF FORMATION

A single set of parameter values from the highest fitness individual (row 1, column 2 in **Table 2**) was used to generate **Figures 5–7A**, **10**, these parameter values can be found in Supplementary 4 of the supplementary materials. **Figure 5** shows the firing rates of four output neurons from the Exc group in response to all 40 input stimulus grating orientations. Each plot represents the firing rate of an individual Exc group neuron, denoted by a blue line, along with an idealized Gaussian tuning curve similar to those found in simple cell responses in visual cortical area V1 of the visual cortex (Henry et al., 1974), denoted by a dashed red line. The firing rate responses from the Exc group neurons qualitatively match the idealized V1 simple cell Gaussian tuning curves. The maximum firing rate responses of Exc group neurons were constrained by the sparsity requirement of the fitness function and peaked at an average value of 67 Hz. The firing rate responses were also decorrelated, another requirement of the fitness function, which lead to different preferred orientations for each of the Exc group neurons.

To examine the ability of the automated parameter tuning framework to construct SNNs capable of SORF formation, the synaptic weights between the On(Off)Buffer groups and the Exc group were visualized in **Figure 6** for the highest fitness SNN. Each plot is a composite of the connections between the On(Off)Buffer group and a single Exc group neuron, where light regions represent strong synaptic connections and dark regions represent weak synaptic connections. **Figure 6A** shows

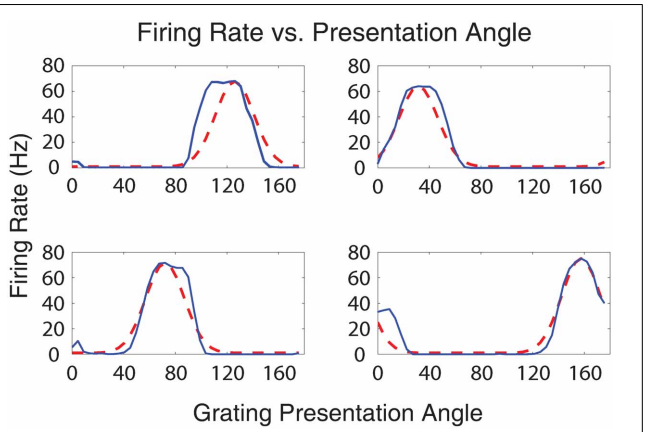



**FIGURE 5 | Plot of the firing rate response of Exc group neurons vs. grating presentation orientation angle.** The blue lines indicate the firing rate of a neuron in the simulation while the dotted red lines indicate idealized Gaussian tuning curves. Together, the four excitatory neurons cover the stimulus space of all the possible presentation angles.

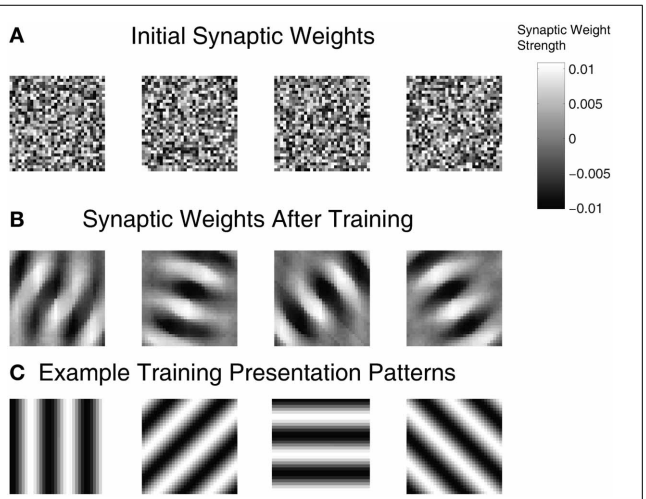



**FIGURE 6 | Synaptic weights for the On(Off)Buffer → Exc connections of a high fitness SNN individual. (A)** Initial weight values before training. **(B)** After training for approximately 100 simulated minutes with STDP and homeostasis, the synaptic weight patterns resemble Gabor filters. **(C)** Four example orientation grating patterns are shown.

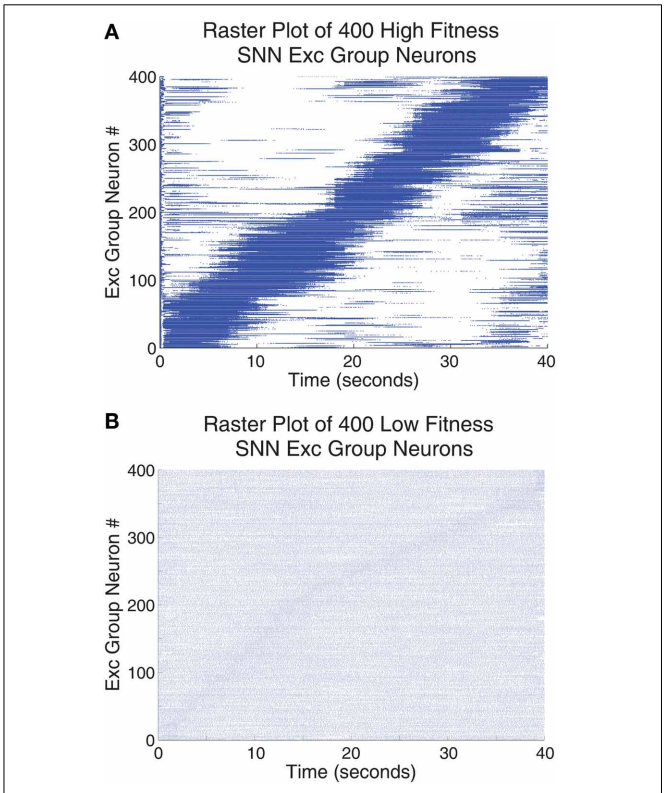



**FIGURE 7 | The responses of the Exc group neurons (identified by their neuron id on the y-axis) were tested for all 40 grating orientations.** One orientation was presented per second and the test ran for 40 s (x-axis). **(A)** Neuronal spike responses of 400 neurons trained with the highest fitness SNN parameters found using the parameter tuning framework. **(B)** Neuronal spike responses of 400 neurons trained using a single set of low fitness parameters. The neurons were arranged such that those neurons responding to similar orientations were grouped together for both **(A,B)**. This accounts for the strong diagonal pattern found in **(A)** and the very faint diagonal pattern found in **(B)**. Neuronal spike responses in **(A)** are sparse in that relatively few neurons code for one orientation while neuronal spike responses in **(B)** are not sparse. Additional, many of the neuronal spike responses in part **(A)** employ a wide range of firing rates to describe a subset of the orientation stimulus space while spike responses in **(B)** have similar firing responses across all angles in all cases.

the initial randomized synaptic weights while **Figure 6B** shows the final synaptic weights after 100 min of simulation time during the training period. The synaptic connections between the On(Off)Buffer neurons and the Exc neurons in **Figure 6B** formed receptive fields that resembled Gabor filters, which have been used extensively to model V1 simple cell responses (Jones and Palmer, 1987). **Figure 6C** shows four example counterphase sinusoidal grating orientations used as visual input into the SNN.

**Figure 7** shows a raster plot of 400 Exc group neurons from 100 SNNs that were trained using the highest fitness parameter values taken from row 2, column 1 of **Table 2** and shown in **Figure 7A** compared with a set of very low fitness parameters, fitness = 0.0978, shown in **Figure 7B**. Neurons that have similar preferred orientation angles have been placed close to one another. The high fitness neurons in **Figure 7A** have responses that are sparse (only a small subset of the neurons respond to any particular stimulus angle) and orthogonal (different neurons respond to different stimulus orientations) while neurons in **Figure 7B** do not have these properties. Although each high fitness neuron responds to a small subset of stimulus orientations, taken together the high fitness neurons have responses that cover all the possible stimulus orientations while low fitness neurons do not have responses that carry meaningful information in this respect.

**Figure 8** compares the evolved parameters of "high fitness" SNNs with "low fitness" SNNs. We judged an SNN to be high fitness if its three fitness component values met the following cutoffs: $\text{fitness}_{\text{decorr}}$ had a cutoff value of 15, $\text{fitness}_{\text{Gauss}}$ had a cutoff value of 950, and $\text{fitness}_{\text{maxRate}}$ had a cutoff value of 50. We found these cutoffs produced SNNs with SORFs in the receptive fields of at least 3 out of 4 of the Exc group neurons. There were 128 high fitness SNNs and 2752 low fitness SNNs out of the 2880 total SNNs constructed and tested by the parameter tuning framework.

**Figure 8** shows a comparison between homeostatic target firing rate parameters for Exc and Inh groups for high fitness SNNs (shown in **Figure 8A**) found using the parameter tuning framework along with the remaining low fitness parameter values (shown in **Figure 8B**). Each point represents a target Exc and Inh firing rate pair for a given SNN. The homeostatic target firing rate parameter for Exc groups in high fitness SNNs is clustered around a relatively small region (10–14 Hz) when compared to the total allowed target firing rate ranges of the Exc and Inh groups which are 10–30 and 40–100 Hz, respectively. The low fitness SNNs have Exc and Inh groups with target firing rates that have a much wider range of values. It is interesting that successful SNNs cluster around a low Exc group homeostatic firing rate (10–14 Hz). This may be due to the interplay between STDP time windows or the maximum input Poisson firing rate. In high fitness SNNs, Inh groups with higher homeostatic target firing rates are rare, but the distribution of firing rates is broader.

We next examined the relationship between STDP plasticity parameters among high fitness SNNs individuals exclusively. **Figure 9A** shows the LTD/LTP decay constant ratios, which dictate the size of the LTP and LTD time windows, for Buffer to Exc group connections and Exc to Inh group connections. **Figure 9B** shows a comparison between LTD/LTP amplitude ratios for Buffer to Exc group connections and Exc to Inh group connections. The overall parameter ranges can be found in **Table 1**. The Buffer to Exc decay constant ratio in **Figure 9A** is within close range of experimental observations by (Bi and Poo, 1998), that show the LTD decay constant as being roughly twice as large at the LTP decay constant. The Exc to Inh LTD/LTP decay constant ratio in **Figure 9A** has a broader distribution of values that ranged from approximately 1 to 4. These values also fall within the range of experimental measurements of the LTD/LTP decay constant ratio of approximately one (Bell et al., 1997). High fitness SNNs in **Figure 9B** show a narrow distribution of LTD/LTP amplitude ratios that favor an LTD/LTP ratio less than one for Buffer to Exc group connections while Exc to Inh group connections show significantly broader LTD/LTP amplitude ratios with values ranging from approximately 1 to 4.

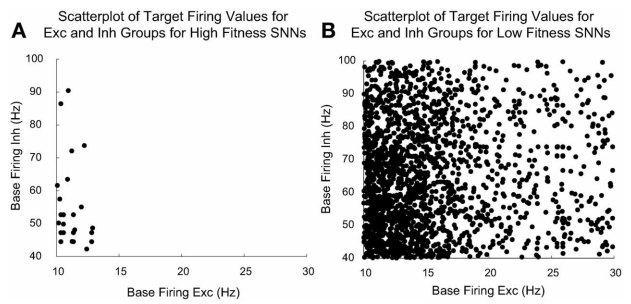



**FIGURE 8 | Plot of the target homeostatic firing rate parameters for Exc group and Inh group for high fitness SNNs shown in (A) and low fitness SNNs shown in (B).** The Exc group homeostatic target firing rate is significantly more constrained (between the ranges of 10–14 Hz) for the high fitness SNNs as opposed to the corresponding parameters for the low fitness SNNs. There were 128 high fitness SNNs and 2752 low fitness SNNs out of a total of 2880 individuals. EAs allow parent individuals to pass high value parameter values directly to their offspring, because of this, there are many offspring with identical high fitness values. This explains why there are not 128 distinct points distinguishable in **(A)**.

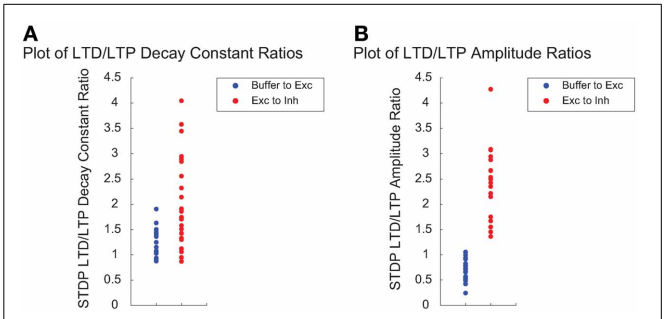



**FIGURE 9 | The time windows in which STDP occurs are often modeled as decaying exponentials and each of the LTP and LTD windows can be characterized by single decay constant.** The degree to which the weight is increased during LTP or decreased during LTD is often called the LTP/LTD amplitude or magnitude. **(A)** Ratio of the STDP LTD/LTP decay constant for the Buffer to Exc group connections (blue) and the Exc to Inh group connections (red) for high fitness SNNs. **(B)** The ratio of the STDP LTD/LTP amplitude for the Buffer to Exc group connections (blue) and the Exc to Inh group connections (blue) for high fitness SNNs.

## STABILITY ANALYSIS

To ensure that the solutions found by the automated tuning framework were stable, the parameter set from the highest fitness SNN was used to train and test an SNN for an additional 100 trials, allowing the SORFs to be learned through STDP and tested as described in the previous section. That is, a single set of parameters was tested to ensure that the ability of a naïve SNN to form SORFs was repeatable and independent of stimulus presentation order. Thus, the order of stimulus presentations was randomized between trials and each trial consisted of training and testing phases. Parameter values were deemed stable if the SNNs consistently produced V1 simple cell-like responses and SORFs for the majority of the trials. A robustness analysis on the effect of small perturbations on the functional behavior of the SNNs was not performed. To further analyze the stability of the parameter values, firing rate responses from the Exc group over all 100 trials were used to decode eight test angles presented to the trained SNNs with a widely used population decoding algorithm (Dayan and Abbott, 2001). At each presentation of the eight orientation test angles, the neuronal firing rate and its preferred stimulus orientation (i.e., the orientation for which the neuron fired maximally) were used to create a population vector for all the Exc neurons from the 100 trials (4 Exc neurons per trial × 100 trials = 400 neurons in total). The neuronal population vectors were averaged and the resultant vector was compared to the stimulus orientation angle.

The results of the 100 training and testing trials for the identical set of parameters were as follows. 76% of the trials had SNNs with tuning curves that qualitatively matched V1 simple cell responses and produced Gabor filter-like SORFs. The remaining 24% of the trials had SNNs with three Exc group neurons that produced good behavior and a single Exc group neuron with a bimodal tuning curve and a SORF that resembled two overlapping Gabor filters at different angles. A population code from the firing rate responses of the 400 Exc group neurons was used to decode the orientation of the presented test stimuli. **Figure 10** shows the population decoding results for eight presented test angles. The smaller black arrows are neuronal responses from the 400 neurons which sum to the population vector, shown with a blue arrow. The lengths of the individual neural response vectors (black arrows) were normalized by dividing the mean firing rate by 2. The length of the population vector (blue arrow) was normalized by dividing the sum of the individual responses by the magnitude of the vector. The population vector was very close to the presented test stimulus orientation for every case with a mean error of 3.4° and a standard deviation of 2.3°.

## PERFORMANCE ANALYSIS

To test the computational performance of the automated parameter tuning framework, three different sized SNNs were run using either a serial CPU implementation or a parallel GPU implementation of CARLsim. Each SNN had identical topology except for the size of the On(Off)Poiss and On(Off)Buffer groups which were either 16 × 16, 24 × 24, or 32 × 32 giving rise to networks with 1032, 2312, and 4104 neurons, respectively. The number of configurations executed in parallel on the GPU was varied from 5 to 30 for all network sizes and execution times were recorded.

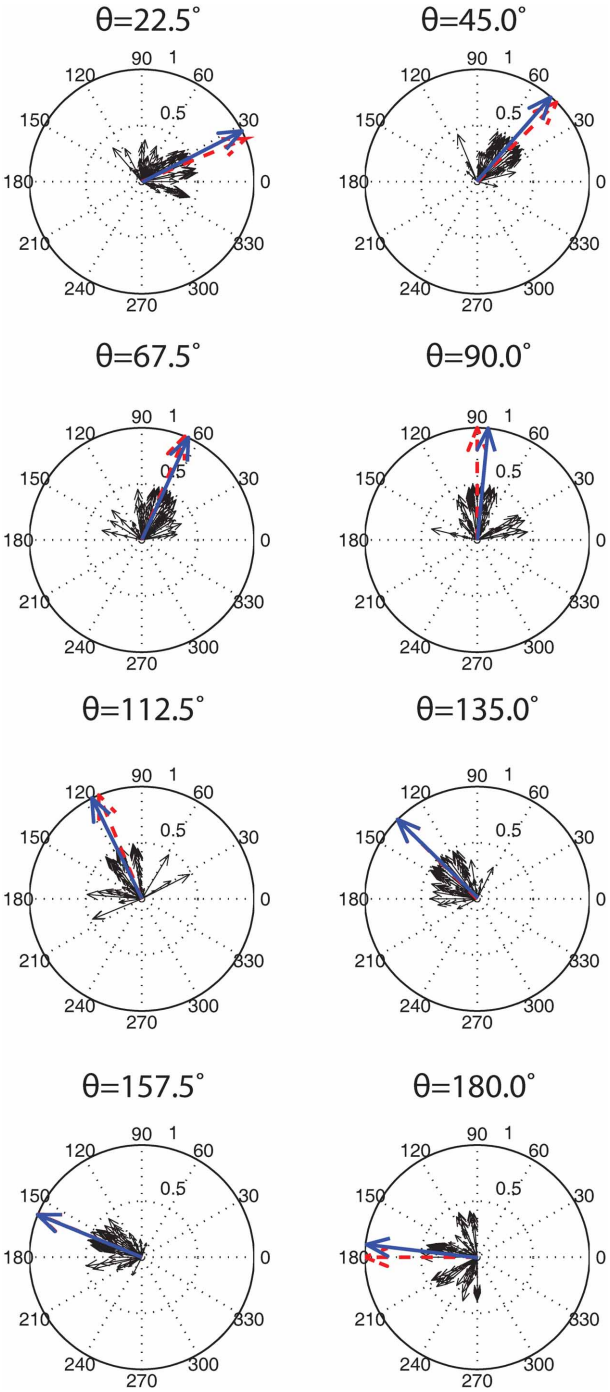



**FIGURE 10 | Population decoding of eight test presentation angles.** The test presentation angle θ, is shown above each population decoding figure. 100 simulation runs, each with identical parameter values but different training presentation orders, were conducted and the firing rates of the Exc group neurons were recorded. The individual responses of each of the 400 neurons (4 Exc neurons × 100 runs) are shown with solid black arrows. These individuals were summed to give a population vector (shown with a blue arrow) that was compared to the correct presentation angle (shown with a red arrow). Both the population vectors and correct presentation angle vectors were normalized while the component vectors were scaled down by a factor of 2 for display purposes (see text for details).

The parallelized GPU SNN implementation showed impressive speedups over the CPU SNN implementation (**Figure 11**). The largest speedup (65×) was found when 30 SNN configurations, each with 4104 neurons, were run in parallel, which took 21.1 min to complete a single generation, whereas 10 SNN configurations with 4104 neurons required 26.4 min to complete a single generation. In contrast, the CPU took 23.5 h for a single generation. It would be interesting to compare the GPU performance with a multi-threaded CPU simulation and there may be gains in such an approach. However, in our experience SNNs on such systems do not optimize or scale as well as GPUs. Because the calculation of SNN neuronal and synaptic states can be cast as single instruction multiple data (SIMD), parallel computation of SNNs is more suited to GPUs having thousands of simple cores, rather than multithreaded CPUs having many less, but more powerful cores.

As the number of concurrent SNN configurations grows, the speedup increases slowly and nearly plateaus for 30 parallel SNN configurations. These speedup plateaus are mostly likely due to the limitations of the GPU core number and clock-frequency, and not the GPU global memory size as 99% of the GPU was utilized but less than 20% of the GPU memory was utilized for the largest simulation configurations. It should be noted that although the single SNN configuration was moderately sized, all 30 configurations together comprised a large-scale network (i.e., 123,120 total neurons) that was running simultaneously. This parameter tuning approach can be scaled to tune larger SNNs by running fewer configurations in parallel or by spreading the computation and memory usage across multiple GPUs with an MPI/CUDA implementation.

## DISCUSSION

With the growing interest in large-scale neuromorphic applications using spiking neural networks, the challenge of tuning the vast number of open parameters is becoming increasingly important. We introduced an automated parameter tuning framework that can quickly and efficiently tune SNNs by utilizing inexpensive, off-the-shelf GPU computing technology as a substitute for more expensive alternatives such as supercomputing clusters. The automated parameter tuning framework consists solely of freely available open source software. As a proof of concept, the framework was used to tune 14 neural parameters in an SNN ranging from 1032 to 4104-neurons. The tuned SNNs evolved STDP and homeostasis parameters that learned to produce V1 simple cell-like tuning curve responses and SORFs. We observed speedups of 65× using the GPU for parallelization over a CPU. Additionally, the solutions found by the automated parameter tuning framework were shown to be stable.

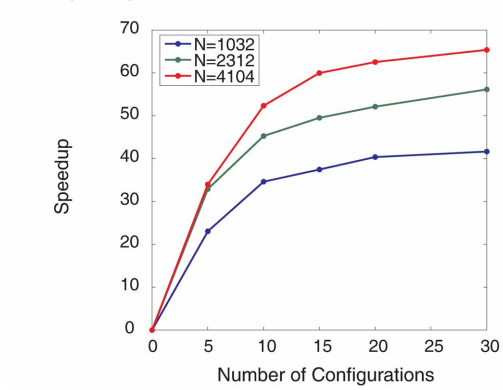



**FIGURE 11 | Plot of GPU speedup over CPU vs. number of SNNs run in parallel for different sized SNNs and different numbers of SNNs run in parallel.** Three different SNN sizes were used, the blue line denotes SNNs with 1032 neurons, the green line denotes SNNs with 2312 neurons, and the red line denotes SNNs with 4104 neurons.

There are a few research groups that have designed software frameworks capable of tuning large-scale SNNs. Eliasmith et al. (2012) constructed a 2.5 million neuron simulation that demonstrated eight diverse behavioral tasks by taking a control theoretic approach called the Neural Engineering Framework (NEF) to tune very large-scale models. The NEF is implemented in a neural simulator called Nengo and can specify the connection weights between two neuronal populations given the input population, the output population, and the desired computation to be performed on those representations. Our parameter tuning framework takes a different approach, allowing the user to tune not only individual synaptic weights but also parameters related to plasticity rules, connection topology, and other biologically relevant parameters. Our framework does not require the user to specify the desired computations between two neuronal populations but rather leaves it to the user to specify the exact fitness function. The popular SNN simulator, Brian (Goodman and Brette, 2009), also has support for parameter tuning in the form of a parallelized CPU/GPU tuning toolkit. Their toolkit has been used to match individual neuron models to electrophysiological data and also to reduce complex biophysical models to simple phenomenological ones (Rossant et al., 2011). Our tuning framework is focused more on tuning the overall SNN behavior as opposed to tuning a spiking model neuron that captures electrophysiological data.

SNNs constructed and tuned with our framework could be converted to run on any neuromorphic device that incorporates the AER format for spike events and supports basic connection topologies. This is the case for many neuromorphic hardware devices (Furber et al., 2012; Cruz-Albrecht et al., 2013; Esser et al., 2013; Pfeil et al., 2013). Although the framework presented here was run on the CARLsim simulator, which utilizes the Izhikevich neuron and STDP, the automated tuning framework presented here could readily be extended to support any spiking model, such as the leaky integrate-and-fire neuron or the adaptive exponential integrate-and-fire neuron (Brette and Gerstner, 2005).

SNNs with thousands of neurons, multiple plasticity rules, homeostatic mechanisms, and feedback connections, similar to the SNN presented here, are notoriously difficult to construct and tune. The automated parameter tuning framework presented here can currently be applied to much larger SNNs (on the scale of $10^6$ neurons) with more complex network topologies but GPU memory constraints limit the tuning of larger SNNs. Currently, CARLsim SNN simulations are limited to approximately 500 K neurons and 100 M synapses on a single Tesla M2090 GPU, but a

version that allows SNN simulations to run across multiple GPUs is in development and will increase the size of SNNs that can be tuned using this framework. The combination of a multi-GPU version of CARLsim and the implementation of more advanced evolutionary computation principles, such as multi-objective fitness functions and co-evolving populations, should allow the framework to be scalable and capable of tuning large-scale SNNs on the scale of millions of neurons. The highly efficient automated parameter tuning framework presented here can reduce the time researchers spend constructing and tuning large-scale SNNs and could prove to be a valuable contribution to both the neuromorphic engineering and computational neuroscience research communities.

## ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency (DARPA) subcontract 801888-BS and by NSF Award IIS/RI-1302125. We thank Micah Richert for his work developing the custom spiking neural network simulator and homeostatic plasticity model. We also thank Michael Avery and Michael Beyeler for valuable feedback and discussion on this project. Finally, we thank the reviewers for their feedback which greatly improved the accuracy and clarity of the manuscript.

## SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fnins.2014.00010/abstract

## REFERENCES

Abbott, L. F., and Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nat. Neurosci.* 3, 1178–1183. doi: 10.1038/81453

Ahmadi, A., and Soleimani, H. (2011). "A GPU based simulation of multilayer spiking neural networks," in *Proceedings of the 2011 Iranian Conference on Electrical Engineering (ICEE)* (Tehran), 1–5.

Amir, A., Datta, P., Risk, W. P., Cassidy, A. S., Kusnitz, J. A., Esser, S. K., et al. (2013). "Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX). doi: 10.1109/IJCNN.2013.6707078

Avery, M., Krichmar, J. L., and Dutt, N. (2012). "Spiking neuron model of basal forebrain enhancement of visual attention," in *Proccedings of the 2012 International Joint Conference on Neural Networks (IJCNN)* (Brisbane, QLD), 1–8. doi: 10.1109/IJCNN.2012.6252578

Baladron, J., Fasoli, D., and Faugeras, O. (2012). Three applications of GPU computing in neuroscience. *Comput. Sci. Eng.* 14, 40–47. doi: 10.1109/MCSE.2011.119

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends Cogn. Sci.* 4, 91–99. doi: 10.1016/S1364-6613(99)01440-0

Bell, C. C., Han, V. Z., Sugawara, Y., and Grant, K. (1997). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature* 387, 278–281. doi: 10.1038/387278a0

Ben-Shalom, R., Aviv, A., Razon, B., and Korngreen, A. (2012). Optimizing ion channel models using a parallel genetic algorithm on graphical processors. *J. Neurosci. Methods* 206, 183–194. doi: 10.1016/j.jneumeth.2012.02.024

Bernhard, F., and Keriven, R. (2006). "Spiking neurons on GPUs," in *Computational Science—ICCS 2006 Lecture Notes in Computer Science*, eds V. Alexandrov, G. Albada, P. A. Sloot, and J. Dongarra (Berlin; Heidelberg: Springer), 236–243.

Bhuiyan, M. A., Pallipuram, V. K., and Smith, M. C. (2010). "Acceleration of spiking neural networks in emerging multi-core and GPU architectures," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on* (Atlanta, GA), 1–8. doi: 10.1109/IPDPSW.2010.5470899

Bi, G. Q., and Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *J. Neurosci.* 18, 10464–10472.

Boahen, K. (2005). Neuromorphic microchips. *Sci. Am.* 292, 56–63. doi: 10.1038/scientificamerican0505-56

Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.* 94, 3637–3642. doi: 10.1152/jn.00686.2005

Brette, R., and Goodman, D. F. M. (2012). Simulating spiking neural networks on GPU. *Network* 23, 167–182. doi: 10.3109/0954898X.2012.730170

Brüderle, D., Petrovici, M. A., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., et al. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biol. Cybern.* 104, 263–296. doi: 10.1007/s00422-011-0435-9

Calin-Jageman, R. J., and Katz, P. S. (2006). A distributed computing tool for generating neural simulation databases. *Neural Comput.* 18, 2923–2927. doi: 10.1162/neco.2006.18.12.2923

Caporale, N., and Dan, Y. (2008). Spike timing-dependent plasticity: a Hebbian learning rule. *Annu. Rev. Neurosci.* 31, 25–46. doi: 10.1146/annurev.neuro.31.060407.125639

Carlson, K. D., Richert, M., Dutt, N., and Krichmar, J. L. (2013). "Biologically plausible models of homeostasis and STDP: stability and learning in spiking neural networks," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX). doi: 10.1109/IJCNN.2013.6706961

Clune, J., Stanley, K. O., Pennock, R. T., and Ofria, C. (2011). On the performance of indirect encoding across the continuum of regularity. *IEEE Trans. Evol. Comput.* 15, 346–367. doi: 10.1109/TEVC.2010.2104157

Cruz-Albrecht, J. M., Derosier, T., and Srinivasa, N. (2013). A scalable neural chip with synaptic electronics using CMOS integrated memristors. *Nanotechnology* 24, 384011. doi: 10.1088/0957-4484/24/38/384011

Dayan, P., and Abbott, L. F. (2001). *Theoretical Neuroscience*. Cambridge: MIT press.

De Jong, K. A. (2002). *Evolutionary Computation: A Unified Approach*. Cambridge: The MIT Press.

de Ladurantaye, V., Lavoie, J., Bergeron, J., Parenteau, M., Lu, H., Pichevar, R., et al. (2012). A parallel supercomputer implementation of a biological inspired neural network and its use for pattern recognition. *J. Phys. Conf. Ser.* 341, 012024. doi: 10.1088/1742-6596/341/1/012024

Djurfeldt, M., Ekeberg, O., and Lansner, A. (2008). Large-scale modeling—a tool for conquering the complexity of the brain. *Front. Neuroinformatics* 2:1. doi: 10.3389/neuro.11.001.2008

Ehrlich, M., Wendt, K., Zühl, L., Schüffny, R., Brüderle, D., Müller, E., et al. (2010). "A software framework for mapping neural networks to a wafer-scale neuromorphic hardware system," in *Proceedings of ANNIIP* (Funchal, Madeira), 43–52.

Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., et al. (2012). A large-scale model of the functioning brain. *Science* 338, 1202–1205. doi: 10.1126/science.1225266

Esser, S. K., Andreopoulus, A., Appuswamy, R., Datta, P., Barch, D., Amir, A., et al. (2013). "Cognitive computing systems: algorithms and applications for networks of neurosynaptic cores," in *Proceedings of the 2013 International Joint Conference on Neural Networks (IJCNN)* (Dallas, TX). doi: 10.1109/IJCNN.2013.6706746

Fidjeland, A. K., Roesch, E. B., Shanahan, M. P., and Luk, W. (2009). "NeMo: a platform for neural modelling of spiking neurons using GPUs," in *Application-specific Systems, Architectures and Processors, 2009 ASAP 2009. 20th IEEE International Conference on*, (Boston, MA), 137–144.

Floreano, D., and Urzelai, J. (2001). Neural morphogenesis, synaptic plasticity, and evolution. *Theory Biosci.* 120, 225–240. doi: 10.1007/s12064-001-0020-1

Fogel, D. B., Fogel, L. J., and Porto, V. W. (1990). Evolving neural networks. *Biol. Cybern.* 63, 487–493. doi: 10.1007/BF00199581

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., et al. (2012). Overview of the SpiNNaker system architecture. *IEEE Trans. Comput.* 62, 2454. doi: 10.1109/TC.2012.142

Gao, P., Benjamin, B. V., and Boahen, K. (2012). Dynamical system guided mapping of quantitative neuronal models onto neuromorphic hardware. *IEEE Trans. Circuits Syst. Regul. Pap.* 59, 2383–2394. doi: 10.1109/TCSI.2012.2188956

Gauci, J., and Stanley, K. O. (2010). Autonomous evolution of topographic regularities in artificial neural networks. *Neural Comput.* 22, 1860–1898. doi: 10.1162/neco.2010.06-09-1042

Giordano, N. J., and Nakanishi, H. (2006). *Computational Physics*. 2nd Edn. Upper Saddle River, NJ: Pearson Prentice Hall.

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adapt. Behav.* 5, 317–342. doi: 10.1177/105971239700500305

Goodman, D. F. M., and Brette, R. (2009). The brian simulator. *Front. Neurosci.* 3:192–197. doi: 10.3389/neuro.01.026.2009

Han, B., and Taha, T. M. (2010). "Neuromorphic models on a GPGPU cluster," in *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)* (Barcelona), 1–8. doi: 10.1109/IJCNN.2010.5596803

Hancock, P. J. B. (1992). "Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification," in *International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992, COGANN-92*, (Baltimore, MD), 108–122. doi: 10.1109/COGANN.1992.273944

Hendrickson, E. B., Edgerton, J. R., and Jaeger, D. (2011). The use of automated parameter searches to improve ion channel kinetics for neural modeling. *J. Comput. Neurosci.* 31, 329–346. doi: 10.1007/s10827-010-0312-x

Henry, G., Dreher, B., and Bishop, P. (1974). Orientation specificity of cells in cat striate cortex. *J. Neurophysiol.* 37, 1394–1409.

Hoffmann, J., El-Laithy, K., Güttler, F., and Bogdan, M. (2010). "Simulating biological-inspired spiking neural networks with OpenCL," in *Artificial Neural Networks—ICANN 2010 Lecture Notes in Computer Science*, eds K. Diamantaras, W. Duch, and L. Iliadis (Berlin; Heidelberg: Springer), 184–187.

Husbands, P., Smith, T., Jakobi, N., and O'Shea, M. (1998). Better living through chemistry: evolving GasNets for robot control. *Connect. Sci.* 10, 185–210. doi: 10.1080/095400998116404

Igarashi, J., Shouno, O., Fukai, T., and Tsujino, H. (2011). Real-time simulation of a spiking neural network model of the basal ganglia circuitry using general purpose computing on graphics processing units. *Neural Netw.* 24, 950–960. doi: 10.1016/j.neunet.2011.06.008

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., van Schaik, A., Etienne-Cummings, R., Delbruck, T., et al. (2011). Neuromorphic silicon neuron circuits. *Front. Neurosci.* 5:73. doi: 10.3389/fnins.2011.00073

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Trans. Neural Netw.* 14, 1569–1572. doi: 10.1109/TNN.2003.820440

Izhikevich, E. M., and Desai, N. S. (2003). Relating STDP to BCM. *Neural Comput.* 15, 1511–1523. doi: 10.1162/089976603321891783

Izhikevich, E. M., and Edelman, G. M. (2008). Large-scale model of mammalian thalamocortical systems. *Proc. Natl. Acad. Sci. U.S.A.* 105, 3593–3598. doi: 10.1073/pnas.0712231105

Izhikevich, E. M., Gally, J. A., and Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cereb. Cortex* 14, 933–944. doi: 10.1093/cercor/bhh053

Jones, J., and Palmer, L. (1987). An evaluation of the two-dimensional gabor filter model of simple receptive-fields in cat striate cortex. *J. Neurophysiol.* 58, 1233–1258.

Keijzer, M., Merelo, J. J., Romero, G., and Schoenauer, M. (2002). "Evolving objects: a general purpose evolutionary computation library," in *Artficial Evolution*, eds P. Collet, C. Fonlupt, J. K. Hao, E. Lutton, and M. Schoenauer (Berlin: Springer-Verlag), 231–242.

Krichmar, J. L., Dutt, N., Nageswaran, J. M., and Richert, M. (2011). "Neuromorphic modeling abstractions and simulation of large-scale cortical networks," in *Proceedings of the 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (San Jose, CA), 334–338. doi: 10.1109/ICCAD.2011.6105350

Maitre, O., Baumes, L. A., Lachiche, N., Corma, A., and Collet, P. (2009). "Coarse grain parallelization of evolutionary algorithms on GPGPU cards with EASEA," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (Montreal, QC), 1403–1410.

Markram, H. (2006). The blue brain project. *Nat. Rev. Neurosci.* 7, 153–160. doi: 10.1038/nrn1848

Mirsu, R., Micut, S., Caleanu, C., and Mirsu, D. B. (2012). Optimized simulation framework for spiking neural networks using GPU's. *Adv. Electr. Comp. Eng.* 12, 61–68. doi: 10.4316/aece.2012.02011

Mongillo, G., Barak, O., and Tsodyks, M. (2008). Synaptic theory of working memory. *Science* 319, 1543–1546. doi: 10.1126/science.1150769

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., and Veidenbaum, A. (2009a). "Efficient simulation of large-scale spiking neural networks using CUDA graphics processors," in *Proceedings of the 2009 International Joint Conference on Neural Networks* (IJCNN). (Piscataway, NJ: IEEE Press), 3201–3208.

Nageswaran, J. M., Dutt, N., Krichmar, J. L., Nicolau, A., and Veidenbaum, A. V. (2009b). A configurable simulation environment for the efficient simulation of large-scale spiking neural networks on graphics processors. *Neural Netw.* 22, 791–800. doi: 10.1016/j.neunet.2009.06.028

Nageswaran, J. M., Richert, M., Dutt, N., and Krichmar, J. L. (2010). "Towards reverse engineering the brain: modeling abstractions and simulation frameworks," in *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, (Madrid), 1–6. doi: 10.1109/VLSISOC.2010.5642630

Neftci, E., Binas, J., Rutishauser, U., Chicca, E., Indiveri, G., and Douglas, R. J. (2013). Synthesizing cognition in neuromorphic electronic systems. *Proc. Natl. Acad. Sci. U.S.A.* 110:E3468–E3476. doi: 10.1073/pnas.1212083110. Available online at: http://www.pnas.org/content/early/2013/07/17/1212083110

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with CUDA. *Queue* 6, 40–53. doi: 10.1145/1365490.1365500

Nowotny, T. (2010). "Parallel implementation of a spiking neuronal network model of unsupervised olfactory learning on NVidia CUDA," in *The Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)* (Barcelona), 1–8. doi: 10.1109/IJCNN.2010.5596358

Nowotny, T. (2011). Flexible neuronal network simulation framework using code generation for NVidia(R) CUDATM. *BMC Neurosci.* 12:P239. doi: 10.1186/1471-2202-12-S1-P239

Pallipuram, V. K., Smith, M. C., Raut, N., and Ren, X. (2012). "Exploring multi-level parallelism for large-scale spiking neural networks," in *Proceedings of the International Conference on Parallel and Distributed Techniques and Applications (PDPTA 2012) held in conjunction with WORLDCOMP* 2012, (Las Vegas, NV), 773–779.

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Schmuker, M., et al. (2013). Six networks on a universal neuromorphic computing substrate. *Front. Neurosci.* 7:11. doi: 10.3389/fnins.2013.00011

Pinto, N., Doukhan, D., DiCarlo, J. J., and Cox, D. D. (2009). A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput. Biol.* 5:e1000579. doi: 10.1371/journal.pcbi.1000579

Prinz, A. A., Billimoria, C. P., and Marder, E. (2003). Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *J. Neurophysiol.* 90, 3998–4015. doi: 10.1152/jn.00641.2003

Prinz, A. A., Bucher, D., and Marder, E. (2004). Similar network activity from disparate circuit parameters. *Nat. Neurosci.* 7, 1345–1352. doi: 10.1038/nn1352

Richert, M., Nageswaran, J. M., Dutt, N., and Krichmar, J. L. (2011). An efficient simulation environment for modeling large-scale cortical processing. *Front. Neuroinform.* 5:19. doi: 10.3389/fninf.2011.00019

Risi, S., and Stanley, K. O. (2012). An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artif. Life* 18, 331–363. doi: 10.1162/ARTL_a_00071

Rossant, C., Goodman, D. F., Fontaine, B., Platkiewicz, J., Magnusson, A. K., and Brette, R. (2011). Fitting neuron models to spike trains. *Front. Neurosci.* 5:9. doi: 10.3389/fnins.2011.00009

Schliebs, S., Defoin-Platel, M., Worner, S., and Kasabov, N. (2009). Integrated feature and parameter optimization for an evolving spiking neural network: exploring heterogeneous probabilistic models. *Neural Netw.* 22, 623–632. doi: 10.1016/j.neunet.2009.06.038

Schliebs, S., Kasabov, N., and Defoin-Platel, M. (2010). On the probabilistic optimization of spiking neural networks. *Int. J. Neural Syst.* 20, 481–500. doi: 10.1142/S0129065710002565

Seung, H. S., Lee, D. D., Reis, B. Y., and Tank, D. W. (2000). Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron* 26, 259–271. doi: 10.1016/S0896-6273(00)81155-1

Sheik, S., Stefanini, F., Neftci, E., Chicca, E., and Indiveri, G. (2011). "Systematic configuration and automatic tuning of neuromorphic systems," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, (Rio de Janeiro), 873–876. doi: 10.1109/ISCAS.2011.5937705

Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* 3, 919–926. doi: 10.1038/78829

Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. (2009). A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life* 15, 185–212. doi: 10.1162/artl.2009.15.2.15202

Stanley, K. O., and Miikkulainen, R. (2002). "Efficient evolution of neural network topologies," in *The Proceedings of the Genetic and Evolutionary Computation Conference*, eds W. B. Langdon, E. Cantu-Paz, K. E. Mathias, R. Roy, D. Davis, R. Poli, et al. (Piscataway, NJ; San Francisco, CA: Morgan Kaufmann), 1757–1762.

Stanley, K. O., and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artif. Life* 9, 93–130. doi: 10.1162/106454603322221487

Stone, J. E., Gohara, D., and Shi, G. (2010). OpenCL: a parallel programming standard for heterogeneous computing systems. *Comput. Sci. Eng.* 12, 66–73. doi: 10.1109/MCSE.2010.69

Svensson, C. M., Coombes, S., and Peirce, J. W. (2012). Using evolutionary algorithms for fitting high-dimensional models to neuronal data. *Neuroinformatics* 10, 199–218. doi: 10.1007/s12021-012-9140-7

Thibeault, C. M., Hoang, R. V., and Harris, F. C. (2011). "A novel multi-GPU neural simulator," in *Proceedings of the 2011 International Conference on Bioinformatics and Computational Biology (BICoB)* (New Orleans, LA), 146–151.

Thibeault, C. M., and Srinivasa, N. (2013). Using a hybrid neuron in physiologically inspired models of the basal ganglia. *Front. Comput. Neurosci.* 7:88. doi: 10.3389/fncom.2013.00088

van Geit, W., de Schutter, E., and Achard, P. (2008). Automated neuron model optimization techniques: a review. *Biol. Cybern.* 99, 241–251. doi: 10.1007/s00422-008-0257-6

van Rossum, M. C. W., Bi, G. Q., and Turrigiano, G. G. (2000). Stable hebbian learning from spike timing-dependent plasticity. *J. Neurosci.* 20, 8812–8821.

Watt, A. J., and Desai, N. S. (2010). Homeostatic plasticity and STDP: keeping a neuron's cool in a fluctuating world. *Front. Synaptic Neurosci.* 2:5. doi: 10.3389/fnsyn.2010.00005

Yamazaki, T., and Igarashi, J. (2013). Realtime cerebellum: a large-scale spiking network model of the cerebellum that runs in realtime using a graphics processing unit. *Neural Netw.* 47, 103–111. doi: 10.1016/j.neunet.2013.01.019

Yosinski, J., Clune, J., Hidalgo, D., Nguyen, S., Zagal, J. C., and Lipson, H. (2011). "Evolving robot gaits in hardware: the HyperNEAT generative encoding vs. parameter optimization," in *Proceedings of the 20th European Conference on Artificial Life* (Paris).

Yudanov, D., Shaaban, M., Melton, R., and Reznik, L. (2010). "GPU-based simulation of spiking neural networks with real-time performance and high accuracy," in *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)* (Barcelona), 1–8. doi: 10.1109/IJCNN.2010.5596334

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

*Received: 26 August 2013; accepted: 17 January 2014; published online: 04 February 2014.*

*Citation: Carlson KD, Nageswaran JM, Dutt N and Krichmar JL (2014) An efficient automated parameter tuning framework for spiking neural networks. Front. Neurosci.* ***8****:10. doi: 10.3389/fnins.2014.00010*

*This article was submitted to Neuromorphic Engineering, a section of the journal Frontiers in Neuroscience.*