frontiers | Frontiers in Robotics and AI

TYPE Original Research
PUBLISHED 15 January 2025
DOI 10.3389/frobt.2024.1435197

OPEN ACCESS

EDITED BY
Rui Pimentel de Figueiredo,
Aalborg University, Denmark

REVIEWED BY
Silvio P. Sabatini,
University of Genoa, Italy
Dimitrije Marković,
Technical University Dresden, Germany

*CORRESPONDENCE
Thomas Barbier,
thomas.barbier@uca.fr

RECEIVED 19 May 2024
ACCEPTED 14 October 2024
PUBLISHED 15 January 2025

CITATION
Barbier T, Teulière C and Triesch J (2025) A spiking neural network for active efficient coding.
Front. Robot. AI 11:1435197.
doi: 10.3389/frobt.2024.1435197

A spiking neural network for active efficient coding

Thomas Barbier[1]*, Céline Teulière[1] and Jochen Triesch[2]

[1]SIGMA Clermont, Centre National de la Recherche Scientifique, Institut Pascal, Université Clermont Auvergne, Clermont-Ferrand, France, [2]Life- and Neurosciences, Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany

Biological vision systems simultaneously learn to efficiently encode their visual inputs and to control the movements of their eyes based on the visual input they sample. This autonomous joint learning of visual representations and actions has previously been modeled in the Active Efficient Coding (AEC) framework and implemented using traditional frame-based cameras. However, modern event-based cameras are inspired by the retina and offer advantages in terms of acquisition rate, dynamic range, and power consumption. Here, we propose a first AEC system that is fully implemented as a Spiking Neural Network (SNN) driven by inputs from an event-based camera. This input is efficiently encoded by a two-layer SNN, which in turn feeds into a spiking reinforcement learner that learns motor commands to maximize an intrinsic reward signal. This reward signal is computed directly from the activity levels of the first two layers. We test our approach on two different behaviors: visual tracking of a translating target and stabilizing the orientation of a rotating target. To the best of our knowledge, our work represents the first ever fully spiking AEC model.

KEYWORDS

active efficient coding, spiking neural network, event-based cameras, unsupervised learning, reinforcement learning

1 Introduction

For humans and other mammals, learning to see is an active process involving the development of efficient neural representations and accurate eye movements. The Active Efficient Coding (AEC) framework proposes that both aspects are learned jointly to improve the overall coding efficiency of the visual system. This process is thought to occur through a self-calibrating feedback loop where sensory representations continually adapt to the statistics of visual signals sampled by eye movements, while eye movements adapt to enhance the efficient encoding of visual input.

Various AEC models have been proposed to describe the self-calibration of different aspects of vision, such as active stereo vision (Zhao et al., 2012; Lonini et al., 2013; Zhang et al., 2014; Klimmasch et al., 2021), active motion vision (Teulière et al., 2015), accommodation (Eckmann et al., 2020), and torsional eye movements (Zhu et al., 2022), as well as combinations thereof (Lelais et al., 2019). Notably, AEC has also been extended to the auditory domain (Wijesinghe et al., 2021).

However, all these studies have utilized network architectures with continuous activation model neurons. To date, no AEC model has been developed based on Spiking Neural Networks (SNNs). Such a model would be of great interest due to the increased biological plausibility of SNNs and their potential for energy efficiency, especially when sparsity can be exploited in hardware (Christensen et al., 2022; Dampfhoffer et al., 2023). Furthermore, SNNs are naturally suited for processing visual inputs from event-based

cameras. These cameras, inspired by the mammalian retina, output a continuous stream of discrete events rather than periodically sampled image frames, offering advantages like low data rates, low latency, and high dynamic range (Gallego et al., 2022).

In this study, we introduce the first fully spiking AEC model. An event-based camera provides input to the model, which comprises a two-layer efficient coding network that learns to represent visual inputs with a minimal number of spikes. A spiking reinforcement learner is trained to control eye movements, aiding in the efficient encoding of visual input. This learner utilizes an intrinsic reward signal generated by the efficient coding network. Consequently, the system learns both neural representations and visual behavior in a fully self-calibrating manner, without requiring external supervision or an extrinsically provided reward signal.

We demonstrate our approach with two visual behaviors: tracking a translating target in two dimensions through pursuit movements of a single camera, and stabilizing the orientation of a rotating target, a problem relevant, for instance, to stabilizing the horizon line during flight. The presented tasks are admittedly quite simple, but they serve as a minimal demonstration of intrinsically motivated reinforcement learning in spiking networks. Better results could be achieved with a carefully designed extrinsic reward signal, but our contribution is intrinsically motivated learning in a fully spiking framework.

2 Related work

2.1 Learning efficient representations with SNNs

SNNs have been used to take full advantage of the asynchronous nature of event streams. Several studies have modified backpropagation rules in conjunction with supervised learning for SNNs (Ponulak and Kasinski, 2010; Shrestha and Orchard, 2018; Cheng et al., 2020). However, the reliance on extensive amounts of labeled training data makes these approaches impractical for a self-calibrating vision system. Consequently, our focus is on fully unsupervised techniques.

Many researchers have explored the combination of SNNs with the Spike-Timing Dependent Plasticity (STDP) learning rule to learn effective representations. STDP naturally adapts neural representations to match input statistics and has proven effective for digit recognition (Diehl and Cook, 2015; Tavanaei et al., 2018; Iakymchuk et al., 2015; Hopkins et al., 2018). These works demonstrate that a layer of Leaky Integrate and Fire (LIF) neurons, with various homeostatic mechanisms, can produce spike patterns used to classify MNIST digits using Winner Take All (WTA) strategies or additional classifiers. While these visual representations are effective, the applicability and scalability of their networks have not been demonstrated. Moreover, they do not test their networks on real event-based data, instead relying on a conversion of MNIST images to a firing rate representation.

Kheradpisheh et al. (2018) propose a larger convolutional SNN architecture with up to seven layers, inspired by traditional deep learning. They effectively distinguish objects in images using a rate-coding scheme to convert images into spike trains. The complexity of learned representations increases with depth, from Gabor filters in the early layers to parts of objects in the final layers. While these deep spiking architectures are highly performant, they appear not well-suited for closed-loop vision-based control due to their learning speed, energy consumption, and potential processing delays. Our interest lies in more lightweight architectures that can seamlessly integrate with a reinforcement learning agent.

Paredes-Valles et al. (2019) present a multi-layered architecture for estimating optical flow from scenes, learning effective representations based on synaptic delays, and testing on real event-based data. Akolkar et al. (2015) train a SNN using actual event-based data from an event-based camera, learning neuronal receptive fields for classification tasks. Paulun et al. (2018) propose a complex neuronal architecture trained with STDP, representing a model of the primary visual cortex, and test it on the MNIST-DVS dataset.

Chandrapala and Shi (2016) propose a two-layer SNN that learns invariant feature representations similar to simple and complex cells in the brain, using gassom instead of STDP. Their model differentiates digits from the MNIST-DVS dataset with 90% precision, outperforming other event-based processing methods. Debat et al. (2021) designed a three-layer SNN to estimate trajectories, outperforming human capabilities. Guyonneau et al. (2004) show that SNNs can effectively learn efficient visual representations from the spatio-temporal structure of spike patterns, robust to fast visual stimuli, and emphasize the importance of sparsity in neural codes.

Inspired by these principles, we designed our model with a novel combination of homeostatic mechanisms, STDP rules, and architectural features based on prior work (Barbier et al., 2020; 2021). Our approach aims to address the limitations of previous models by developing a more efficient and adaptable vision system.

2.2 Spiking reinforcement learning

Learning effective policies using traditional reinforcement learning algorithms is challenging with SNNs due to their asynchronous and spiking nature. Hybrid strategies are often used, where networks are trained using conventional reinforcement learning frameworks and then converted for use with SNNs, as demonstrated by Patel et al. (2019), or where SNNs are combined with traditional deep reinforcement learning for training as in Tang et al. (2020). However, we believe that one can achieve comparable performance using fully spiking-based learning strategies, and demonstrate this approach in here.

The brain's learning process heavily relies on both extrinsic and intrinsic rewards, similar to how dopamine modulates behavior. The Reward Modulated STDP (R-STDP) learning rule mimics this process by integrating a reward factor with the classic STDP rule. Frémaux et al. (2010); Florian (2007) provide detailed descriptions of R-STDP, which can incorporate rewards directly or through eligibility traces to associate rewards with past actions Gerstner et al. (2018).

While R-STDP has been applied to tasks like MNIST classification Mozafari et al. (2019); Ghaemi et al. (2021), it lacks a traditional reinforcement learning framework involving an agent in a controlled environment. Yuan et al. (2019) address this by combining stochastic and deterministic plasticity learning rules

to create neuronal agents, designing the Stochastic-Deterministic Coordinated (SDC) spiking reinforcement learning model, though limited to the design of a logic gate and a simple 19-state random walk application.

Most spiking reinforcement learning models utilize a Temporal Difference (TD) actor-critic framework based on R-STDP. Early works such as Potjans et al. (2009); Jitsev et al. (2012); Nichols et al. (2013) used that framework on simple grid-world and robotic applications.

Weidel et al. (2021) proposed an actor-critic model with a recurrent representation layer and a reward-modulated output layer, validating their model on the mountain car environment. Anwar et al. (2022) created a network for playing a modified pong game, solving 2D visual environments using a spiking reinforcement learning framework. Their network includes a visual cortex for frame transformation, an association cortex for learning visual representations, and a motor cortex for outputting motor commands. The main limitation is the use of a rate-based frame coding approach while we are interested in fully spiking frameworks.

Frémaux et al. (2013); Frémeaux and Gerstner (2016) extended the TD learning framework to a continuous spiking model, using it with a fully spiking actor-critic agent to solve simple maze environments However, their model used fixed-place cells and did not learn efficient representations. Hence, here we extend the framework from Frémaux et al. (2013); Frémeaux and Gerstner (2016), with learnable efficient representation, and apply it to visual environments observed through event-based cameras. Additionally, we aim to learn visual representations that can be directly utilized by the actor and critic populations within the reinforcement learning framework.

2.3 Intrinsic reward

One of the biggest limitations of reinforcement learning frameworks is their dependence on externally provided or "extrinsic" rewards. While this can be biologically plausible in certain cases, such as animal experiments where humans deliver food as a reward for successful actions, it is less plausible as a model of how the brain self-calibrates basic sensorimotor loops.

Jaegle et al. (2019) showed that primates heavily rely on visual signals of novelty and curiosity to generate interest and develop specific motor behaviors. Although this is more geared toward higher-level behaviors, such as food search, it suggests that intrinsic motivation plays a significant role in visuomotor learning. It is believed that the human visual system is capable of generating intrinsic rewards based on the efficiency of visual stimuli encoding during eye gaze. Chorley and Seth (2011) attempted to predict dopamine signals using a competitive excitation/inhibition model but did not link this to the learning of concrete visual behaviors.

Gibaldi et al. (2010, 2015) were able to learn vergence control of the eyes using an internal representation of visual input. They utilized disparity-tuned neurons similar to complex cells in the brain to estimate object depth and designed reinforcement learning strategies that used these cells to verge on the target object. Effective gaze strategies were associated with efficient visual encoding and were reinforced over time. This approach was particularly successful for eye vergence, where the left and right visual fields produce very similar stimuli. However, these methods do not operate in the spiking domain and are not applied to event-based visual input. Furthermore, they rely on predefined filters found in the primate visual system and do not learn the efficient coding part.

In this paper, we present a novel approach for generating intrinsic rewards directly from efficient SNN coding layers using plastic synaptic lateral and top-down inhibitory connections. As far as we know, we are the first to propose a fully spiking reinforcement learning framework capable of solving control tasks while intrinsically generating rewards from its internal representation layers.

Figure 1A illustrates the three stages and main features of our AEC spiking architecture.

3 Methods

3.1 Efficient coding model

The efficient coding model encodes the visual event stream into a more abstract and compressed representation. We use the efficient coding model described in Barbier et al. (2021). For completeness, we summarize its main elements here. More details regarding homeostatic mechanisms, including refractory periods, threshold and spike rate adaptation, weight normalization mechanisms, as well as weight sharing, can be found in Supplementary Appendix. The model comprises two layers inspired by simple and complex cells in the primary visual cortex of mammals.

3.1.1 Sensory input

Our model is designed to work with event-based data as its input. Event streams are a sparse and asynchronous representation of visual data. Real event-based recordings were captured using a DAVIS346 event-based camera with a resolution of 346×260 pixels. Synthetic event streams were created using traditional videos at high framerate and then converted to event data using the V2E tool Hu et al. (2021). Both ON and OFF events from the event-based camera are fed into the network and handled via separate groups of synapses.

3.1.2 Neuron model

We base all our cells on a LIF neuron model, a simple yet efficient model widely used in SNNs. To complement this model, we include homeostatic mechanisms that regulate spike rates, create inter-cell competition, and increase learning diversity in the network. To limit winner-take-all behaviors where one cell becomes overly active, we added (i) a refractory period and (ii) a spike rate adaptation mechanism. A neuron's membrane potential then follows:

$$\tilde{V}(t+\Delta t) = V(t)\, e^{\frac{-\Delta t}{\tau_m}} - V_{\mathrm{SRA}}(t)\, e^{\frac{-\Delta t}{\tau_{\mathrm{SRA}}}} - \eta_{\mathrm{RP}} e^{\frac{t_s - t - \Delta t}{\tau_{\mathrm{RP}}}}, \tag{1}$$

$$V(t+\Delta t) = \begin{cases} \tilde{V}(t+\Delta t) & : \tilde{V}(t+\Delta t) < V_\theta \\ 0 & : \tilde{V}(t+\Delta t) \geq V_\theta, \end{cases} \tag{2}$$

where $V(t)$ and τ_m are the membrane potential and the membrane time constant, $V_{\mathrm{SRA}}(t)$ and τ_{SRA} are the spike rate adaptation membrane potential and the membrane time constant, η_{RP} and τ_{RP} are the refractory trace and refractory time constant, respectively.

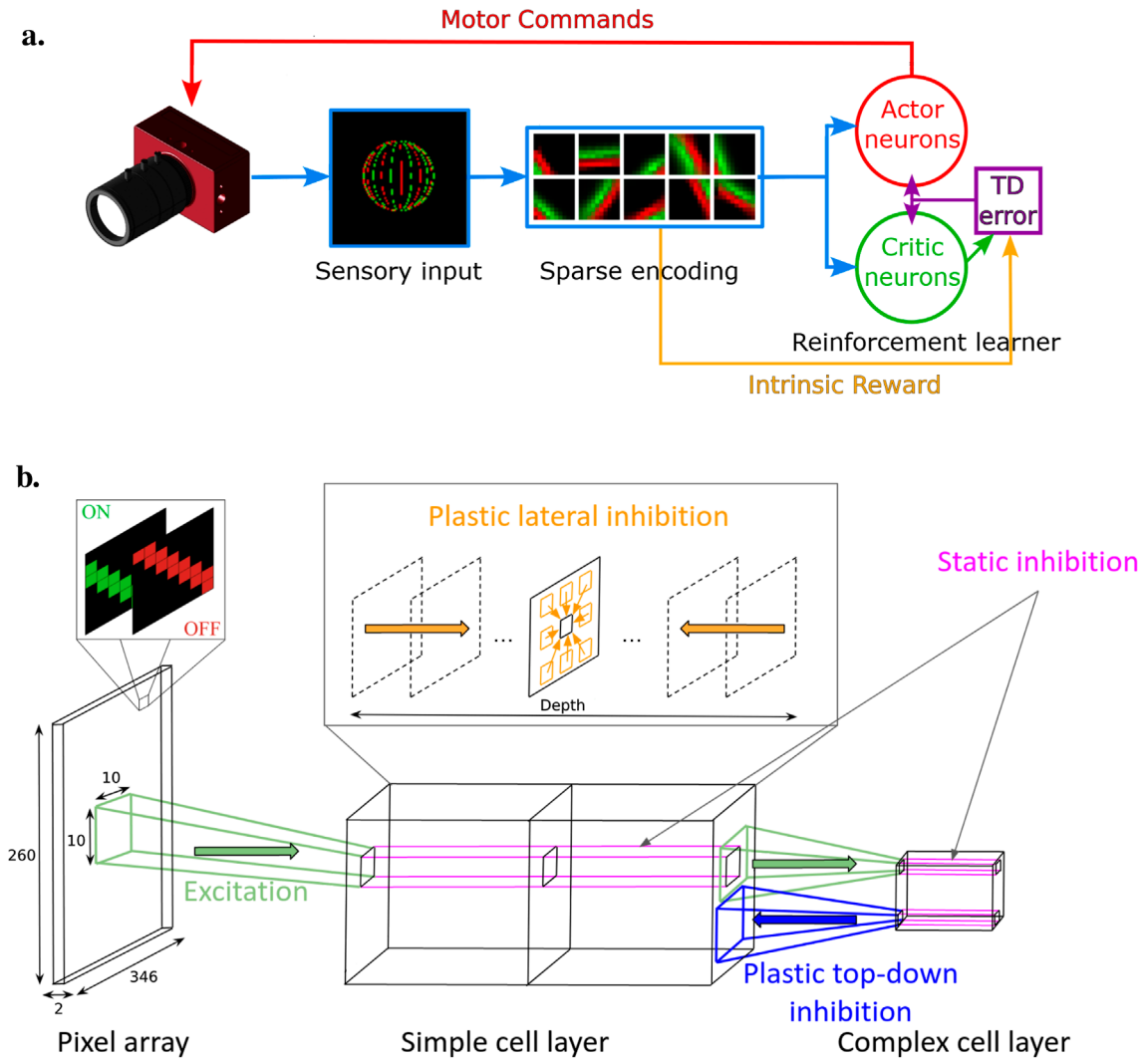

FIGURE 1
(A) AEC spiking model comprising three main blocks: sensory input, efficient coding model, and reinforcement learner. The sensory input from an event-based camera is processed by the efficient coding model to create sparse representations. The reinforcement learner uses these representations to generate motor commands, guided by intrinsic rewards from dynamic inhibition, to optimize visual encoding. **(B)** Sensory input and sparse encoding SNN architecture. The pixel array transmits visual stimuli via ON and OFF channels, which are processed through simple and complex cell layers. Excitation and inhibition mechanisms, including plastic lateral and top-down inhibition, help create sparse, efficient neural representations. Static inhibition stabilizes activity within the complex cell layer. Figure 1B reprinted with permission from N'Dri et al. (2023), Copyright © 2023, IEEE.

3.1.3 Simple cell layer

The first layer of our network aims to approximate the behavior of biological simple cells. These cells learn a more abstract representation of the visual inputs, highlighting features such as orientation, motion, or depth, while also sparsifying the representation.

Our model includes a threshold adaptation mechanism that regulates the membrane potential threshold V_θ according to cell activity $S(t)$, promoting a more uniform spike rate among all cells.

Synaptic weights are learned using an STDP learning rule. We update the weights w_i after each neuron's spike. The simple cells are connected to the input pixels in their receptive field using two different sets of synaptic weights, one for the ON and one for the OFF events. We track the exact timings of all the presynaptic spikes t_i but only the last two postsynaptic spikes t_s and t_{s-1}. The weight update is given by:

$$\begin{aligned} \Delta w_i^{\mathrm{LTP}} &= \eta_{\mathrm{LTP}} e^{\frac{t_i - t_s}{\tau_{\mathrm{LTP}}}} \\ \Delta w_i^{\mathrm{LTD}} &= -\eta_{\mathrm{LTD}} e^{\frac{t_{s-1} - t_i}{\tau_{\mathrm{LTD}}}}, \end{aligned} \tag{3}$$

where $t_s \geq t_i \geq t_{s-1}$. η_{LTP} and η_{LTD} control the height of the potentiation and depression windows, whereas τ_{LTP} and τ_{LTD} control their widths. We use a simple weight normalization mechanism to avoid unbounded growth.

To facilitate rapid and efficient representation learning, our model also includes a weight-sharing mechanism between simple cells. Neurons that look at different locations of the visual field jointly learn the same set of synaptic weights. These neurons belong to the

same "neuronal map." The use of multiple neuronal maps still allows for learning diverse simple cell tunings.

3.1.4 Complex cell layer

Complex cells pool information from multiple simple cells and have larger receptive fields. They learn a higher-level representation that comes from the combination of simple cell receptive fields. We use a different STDP window in the form of a step function with the window centered around the spike time. Both the Long Term Potentiation (LTP) and Long Term Depression (LTD) parts of the window are positive, as follows:

$$\Delta w_i^{\mathrm{LTP,c}} = \begin{cases} \eta_{\mathrm{LTP}} : |t_i - t_s| \leq \tau_{\mathrm{LTP}} \\ 0 : |t_i - t_s| > \tau_{\mathrm{LTP}}. \end{cases}$$
$$\Delta w_i^{\mathrm{LTD,c}} = \begin{cases} \eta_{\mathrm{LTD}} : |t_{s-1} - t_i| \leq \tau_{\mathrm{LTD}} \\ 0 : |t_{s-1} - t_i| > \tau_{\mathrm{LTD}}. \end{cases} \tag{4}$$

We do not use weight-sharing here since it would have limited purpose for a pooling layer such as this one. Figure 1B shows the SNN architecture in more detail.

3.1.5 Inhibition mechanisms

Our framework is based on the use of three different inhibition mechanisms, as can be seen Figure 1B that resumes our SNN architecture.

We use a simple static inhibition mechanism to encourage diversity among neurons' receptive fields. Neurons with the same receptive field location are connected via static inhibitory connections. When a simple (or complex) cell spikes, it inhibits all simple (or complex) cells with the same receptive field location, reducing their membrane potential by a fixed amount:

$$\tilde{V}(t + \Delta t) = \max\left\{V_{\min}, V(t)\, e^{\frac{-\Delta t}{\tau_m}} - \eta_{\mathrm{I}}\right\} \tag{5}$$

where η_{I} sets the strength of this inhibition. This mechanism decorrelates neural responses and facilitates the learning of a diverse set of receptive fields.

Importantly we also use two adaptive inhibition schemes that establish a relationship between the quality of the encoding of the visual input and the amount of spiking activity in the network. This is achieved by learning top-down connections from complex cells to simple cells and lateral connections between simple cells. The rationale is that visual stimuli seen more frequently will trigger more inhibition, thereby reducing network activity for these common stimuli. This activity reduction will be the basis of our intrinsic reward.

In the top-down inhibition scheme, complex cells inhibit all simple cells that excite them via learned connections that instantaneously reduce the membrane potential of the simple cells by an amount corresponding to the learned connection weights. These weights are learned similarly to the excitatory ones, using the STDP rule from Equation 3 applied to inhibitory connections. Simple cells receive inhibitory spikes from the connected complex cells and store them. Once a simple cell spikes, it updates the inhibitory connection weights using a similar STDP window as used for excitatory connections, with the same values for τ_{LTP} and τ_{LTD}, but different learning rates, η_{ILTP} and η_{ILTD}. A specific normalization factor is applied to inhibitory connections.

The primary effect of this inhibition mechanism is that cells firing together on a specific visual pattern will inhibit each other, reducing their cumulative spike rate. This effect is illustrated in Figure 2A. For instance, with a moving edge, cells locally close together will experience the pattern simultaneously, driving up their membrane potential and triggering associated complex cells. These complex cells transmit immediate inhibition signals to the simple cells, reinforcing the inhibitory connections with repeated stimuli. After multiple presentations of the same pattern, the inhibitory connections become strong enough to prevent spiking in the set of simple cells, thereby reducing the average activity rate of the network.

The lateral inhibition scheme is similar to the top-down scheme, but acts from simple cells to other simple cells. Designed so that one simple cell inhibits neighboring simple cells, this differs from static inhibition which only works on cells of different neuronal maps sharing the same receptive field location. The same general properties and learning rules as the top-down scheme from Equation 3 apply, with reduced learning rate and normalization factor due to the higher frequency of simple cell spikes.

This lateral inhibition mechanism can be interpreted as a form of predictive learning. When presented with a moving stimulus, like a moving edge, adjacent simple cells activate close in time in a specific pattern. The first set of simple cells activates and sends inhibitory signals to adjacent simple cells. As the edge moves, it activates neighboring simple cells, some of which just received inhibitory signals. These connections are reinforced, as illustrated in Figure 2B. With repeated visual patterns, the inhibitory connections become strong, making simple cells predictors of the pattern. For example, a moving edge triggering the first set of simple cells will strongly inhibit those associated with the next location, reducing total network activity for this pattern. If a different pattern, such as an edge moving in another direction, is presented, network activity remains unchanged since the inhibition does not affect simple cells triggered by this new pattern.

3.2 Reinforcement learner

3.2.1 Temporal difference error

Our reinforcement learning agent is based on a TD learning actor-critic framework by Frémaux et al. (2013). The discrete time formulation of the TD learning framework is based on an estimate $V(\boldsymbol{x}_t)$ of the true value function $V^{\pi}(\boldsymbol{x}_t)$:

$$\delta_t = \gamma V(\boldsymbol{x}_t) - V(\boldsymbol{x}_{t-1}) + R(\boldsymbol{x}_t, \boldsymbol{a}_t) \tag{6}$$

with δ_t the temporal difference error at discrete time step t, γ the discount factor, $V(\boldsymbol{x}_t)$ the value function associated to the specific state $\boldsymbol{x}_t$ and R the reward associated with the state $\boldsymbol{x}_t$ and action $\boldsymbol{a}_t$.

However, in a fully spiking environment, there are no real discrete steps of time t, or if there are (due to the limitation in temporal resolution of event-based cameras), they are too small to be used effectively. Therefore, a continuous TD error is defined as:

$$\delta_t = \dot{V}(\boldsymbol{x}_t) - \frac{1}{\tau_r} V(\boldsymbol{x}_t) + r(\boldsymbol{x}_t, \boldsymbol{a}_t) \tag{7}$$

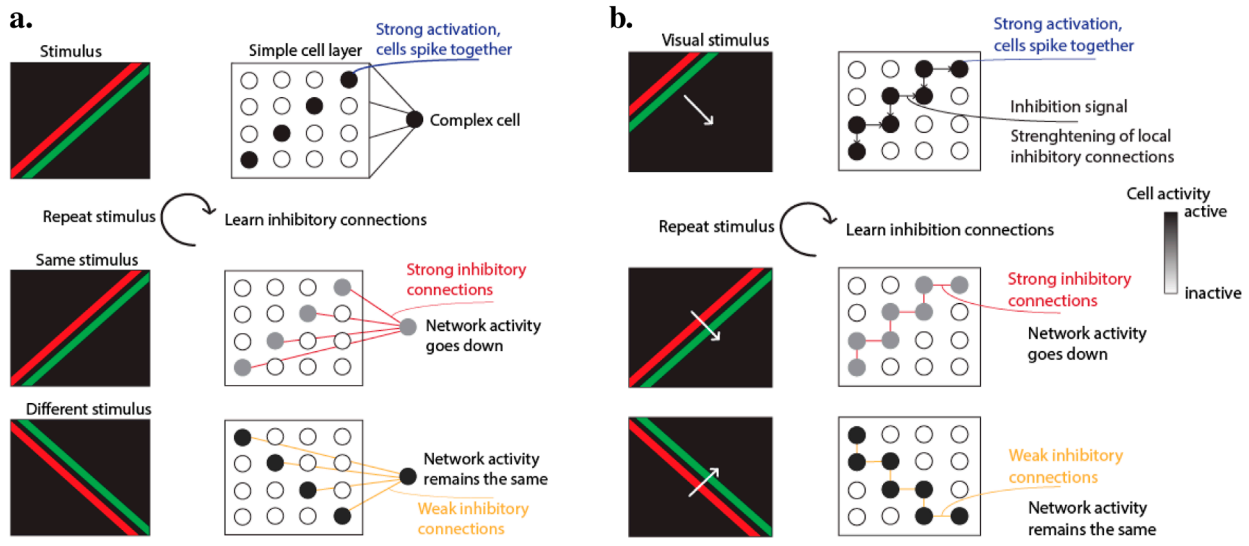

FIGURE 2
(A) Top-down inhibition. Complex cells inhibit simple cells that excite them, strengthening inhibitory connections through repeated visual stimuli, which reduces network activity. **(B)** Lateral inhibition. Simple cells inhibit neighboring simple cells, with inhibitory connections reinforced by repeated visual patterns, thereby decreasing overall network activity for frequent stimuli.

where $\dot{V}(\boldsymbol{x}_t)$ is the time derivative of the value function, and τ_r is the reward discount time constant that plays a similar role to the conventional reward discount factor.

The TD error gives us an indication of the quality of the value estimate. If the value function perfectly evaluates the real value of a state-action pair, the TD error should be equal to zero. Otherwise, it indicates if the value of a state-action pair is overestimated (negative δ_t) or underestimated (positive δ_t).

3.2.2 Critic neurons

The value function estimation is essential to learn a good policy. In SNNs, we cannot evaluate functions as easily as in traditional neural networks. Inputs are arriving continuously, which forces us to look at the evolution of activity in a population of neurons instead.

The first layers of the SNN will serve as a state representation, i.e., their activation patterns represent the state in which the agent is. By connecting those cells to a population of critic neurons, we can learn to associate specific neural activity with a state value function.

Considering one spiking neuron as a value estimator, we can define its value estimate as follows:

$$V(s_t) = v\rho(t) + V_0 \tag{8}$$

with ρ_t the firing rate of the neuron, V_0 the baseline for when the neuron has no activity, and v a scaling factor.

This equation, defines an affine relationship between neuron firing rate and value estimation. Since the firing rate of a neuron is always positive, we can obtain a negative value estimation by carefully selecting V_0 to be negative. This neuron is called a critic neuron as it performs a similar task as the critic in the discrete formulation.

To make the system more robust, we use a whole population of such critic neurons. Therefore, the Equation 8 can be written as:

$$V(s_t) = \frac{v}{N_{critic}} \sum_{i=1}^{N_{critic}} \rho_i(t) + V_0 \tag{9}$$

We usually select a population size of $N_{critic} = 100$.

The firing rate of neurons evolves continuously over time. An exponentially decaying kernel is used to evaluate the firing rate, defined by:

$$\kappa(t) = \frac{e^{\frac{-t}{\tau_k}} - e^{\frac{-t}{v_k}}}{\tau_k - v_k} \tag{10}$$

with $\tau_k = 100$ ms and $v_k = 5$ ms.

Frémaux et al. (2013) used the derivative of the critic kernel to get an approximation of the value derivative $\dot{V}(s_t)$. However, we found that the input natural variability of event rates led to instabilities in the derivative. We thus simply use a second-order numerical differentiation on the value, which can be written as:

$$\dot{V}(s_t) = \frac{\eta_{\text{actor}}}{N} \sum_{i=N-t}^{t} \frac{V(s_{i+1}) - V(s_{i-1})}{2} \tag{11}$$

with η_{actor} a scaling factor and N the number of value points we take into account (since the simulation has a minimal time step of 1 ms).

One limitation of the above formulation is that the value function estimation is proportional to the critic neuron spike rate. Neuronal spike rates are dependent on two main factors: the synaptic weight value of inputs, and the amount of input itself. When submitted to many events, the network's simple and complex cells will inevitably spike more than when fewer events are present. This, in turn, will drive the critic neurons more, which will impact the value estimation. This is an important difference compared to the work of Frémaux et al. (2013). In their model, they ensure that

the amount of activity in the population of state representation spiking neurons stays constant over time. But with event-based visual input, the amount of activity can vary substantially.

To make the value estimation depend only on the values of the weights rather than the event rate, we normalize the value function by the event rate itself. Cell rates in the network are highly correlated with the input rate, which allows us to normalize the event rate without introducing too much variation in the value function estimation.

3.2.3 Actor neurons

In an SNN there are no regular time steps at which actions can be chosen from the output of a readout layer. But similarly to the critic neurons, we can assign spiking neurons to certain actions. In biology, neurons can trigger muscles they are connected to, eliciting fine-tuned movements. Frémaux et al. (2013) used only one actor neuron per action, but to increase stability in the action selection, we use 50 actor neurons for a specific motor action in our framework.

As with the critic neurons, actor neurons are connected to the representation layers so that they can access the agent states' information. Then, we select actions at regular intervals by looking at the activity of those actor neurons. This process is somewhat of a simplification that is not very biologically plausible. We could envision selecting actions only when a certain amount of activity has been registered in the actor neurons, but that would introduce substantial variability in the control loop. The action selected is the one from the actor population that has the strongest activity. This is a simple WTA mechanism that can be found regularly in biological systems. For simplicity, we do not use a sliding window for computing the firing rate of actor neurons, but instead simply count the number of spikes that occurred since the last chosen action.

3.2.4 Three-factor learning rule

Learning associations between the representation layers and the neurons of the critic and actor is done using the R-STDP learning rule. It is composed of two parts, the traditional STDP rule 3 in addition to a third term, a reward, which is often sparse and will modulate the first rule.

In the traditional STDP rule, we modified the weights every time a neuron spiked. However, this is not possible with a R-STDP rule, as the reward can be sparse and will not always arrive at the same time as the spike of the neuron. For that reason, we only update the weights when the reward signal is transmitted to the neuron.

But we still need to keep track of all the usual changes that the STDP rule would create. To do that, we use synaptic eligibility traces. Each synapse connected to the critic or actor neuron possesses a small eligibility trace that is updated every time the neuron spikes. We apply the usual STDP potentiation and depression to it, in addition to a decay term written:

$$\Delta w_{ei}(t+\Delta t) = w_{ei}(t)\,\frac{e^{-\Delta t}}{\tau_e} \tag{12}$$

with w_{ei} the eligibility trace and τ_e the eligibility decay time constant. As the agent moves in state space, the critic and actor neurons' eligibility traces will keep track of the neuronal patterns coming from the representations layers. The time constant τ_e determines from how far back in the past information is kept. Then, once the reward is received, we apply the changes to the real weights. For that, we simply multiply the weights with the eligibility traces:

$$\Delta w_i(t) = \eta w_{ei}(t)\,\delta_t \tag{13}$$

with η the learning rate and δ_t the TD error that acts as the neuromodulator. With this rule, it is possible for the neurons to learn even with sparse and delayed rewards.

Actions are selected at regular intervals predefined beforehand, which in turn corresponds to the transmission of the reward to the actor and critic neurons. We update both the weights of the critic and actor neurons every time an action is selected, using the current TD error. Importantly, we only update the actor neurons from which the previous action was selected since the TD error estimation arises from that specific action.

If the action contributed to a positive TD error, then we strengthen the actor neurons' association with the representation layer. In the opposite case, a negative TD error will decrease the weights accordingly.

3.2.5 Exploration and exploitation strategy

Learning both the value and policy at the same time can be difficult. A good policy can only be learned after an effective value approximation has been computed, which requires properly exploring the state space. In this work, we encourage the agent to first explore its environment by selecting random actions regularly, before slowly changing to an exploitation strategy. To allow the network to explore most states at the beginning, the network selects a random action with the probability λ_{EXP}, called exploration factor. As learning progresses, we decrease this probability, which will drive the agent towards an exploitation phase. In that second phase, the agent has more time to refine its policy.

3.2.5.1 Decay intervals

We handle the change from exploration to exploitation using a simple decay mechanism. Every Δ_{decay} time interval, we decrease some of the learning parameters, including the exploration factor, action selection interval and critic and actor learning rate η following:

$$\Delta_\eta = \left(1 - \frac{\Delta_\eta \eta_{\text{decay}}}{100}\right) \tag{14}$$

with η_{decay} the decay rate.

At the beginning of the learning, the value function has not yet been learned properly. For that reason, it is preferable to reduce the actor learning rate slower than the critic learning rate. We, therefore, use a smaller η_{decay} for the actor neurons.

Table 1 details all cell parameters used in our framework. Table 2 presents the reinforcement learning parameters.

3.2.6 Intrinsic reward from activity

The reward itself is directly generated from the activity of the simple cells. To be precise, we compute a rolling average of the simple cell population activity every 1 ms as follows:

$$S_t = \alpha \overline{S_t} + (1-\alpha)\,S_{t-1} \tag{15}$$

with S_t the activity rolling average, $\overline{S_t}$ the averaged sum of all simple cell spikes in that 1 ms window, and α a smoothing factor that we set to 0.75.

TABLE 1 Parameters configuration for the network's cells in the reinforcement learning tasks.

Param	Unit	Simple Cells	Compl-ex cells	Critic Cells	Actor Cells
V_{thresh}	mV	30	3	2	2
V_{reset}	mV	−20	−20	−20	−20
η_{LTP}	mV	0.00077	0.2	0.077	0.077
η_{LTD}	mV	0.00021	0.2	0.021	0.021
η_{I}	mV	15	15		
η_{TA}	mV	1			
η_{RP}	mV	1	1		
η_{SRA}	mV	0.6			
η_{ILTP}	mV	0.0077			
η_{ILTD}	mV	0.0021			
τ_{m}	ms	18	20		
τ_{LTP}	ms	7	20	7	7
τ_{LTD}	ms	14	20	14	14
τ_{RP}	ms	20	30		
τ_{SRA}	ms	100			
S^*	sp.s^{-1}	0.75			
λ		4	10	4	4
λ_{lateral}		100			
λ_{topdown}		300			
η	mV			0.2	0.1
η_{decay}				5	1.66
ν_{k}	ms			5	
τ_{k}	ms			100	
τ_{e}	ms			250	250

Finally the intrinsic reward R is computed as:

$$R = \gamma \frac{(\beta - S_t)}{E_t} \tag{16}$$

with E_t the average event rate, which is computed similarly to the simple cell event rate. γ and β are simply scaling factors. We chose $\gamma = 5$ and $\beta = 90$ based on experimental results. This reward encourages actions that lead to efficient visual encoding (low activity).

3.3 Control on various visual tasks

In this paper, we focus our attention on agents learning to solve a task in a simulated environment. We use CoppeliaSim to create diverse scenarios upon which the agent evolves and acts. It is a robotic simulator coupled with an efficient physics engine. In our case, we used Bullet 2.78 as the physics engine.

By default, CoppeliaSim is not capable of generating event-based camera output. To solve that issue, we capture frames at a very high rate using a small simulation time step of 1 ms. This

TABLE 2 Reinforcement learning framework parameters.

	Action rate	Action rate (min)	λ_{EXP}	Δ_{decay}	V_0	τ_r	η_{actor}	ν	N
Unit	ms	ms	%	ms	mV	ms	mV		
Value	250	10	75	2000	−20	1	80	1,000	100

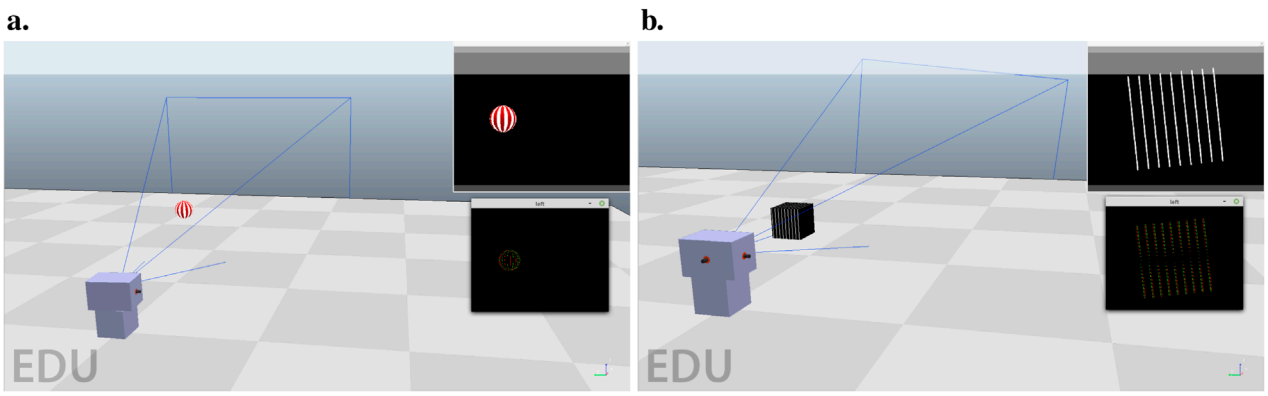

FIGURE 3
(A) Tracking environment, composed of one motorized agent (gray boxes) and one ball (with white and red stripes). The goal is to bring the ball to the center of the visual field, as seen in the top right of the image. Simulation images are sampled at high frame rates and then transformed into event streams, as seen under the visual field representation. The agent can either turn right or left in the horizontal plane. **(B)** Stabilization environment, composed of one motorized agent (gray boxes) and a grating stimulus of white bars on a black background. The goal is to align the bars to a horizontal orientation by rotating the camera around its optical axis.

gives us a frame rate of 1,000 images per second, which we then convert to event streams using an event-based camera emulator called PIX2NVS Bi and Andreopoulos (2018). Even though this is not as precise as real event-based data, the emulator and frame rates are enough to produce accurate models of event streams. Since images are created every 1 ms, we also jitter the events in time to more accurately simulate the output of a real event camera and avoid a cluster of events around frame timestamp generation.

To increase the number of events generated by the moving objects, we added a constant jittering to the camera. This is similar to eye microsaccades and ocular drift, which has been shown to increase visual acuity Intoy and Rucci (2020). It moves left, right, up, and down in saccadic movements, following an Orsnstein-Uhlenbeck stochastic process that can be written as:

$$dx_t = \theta(\mu - x_t)\,dt + \sigma dW_t \tag{17}$$

where $\theta > 0$ and $\sigma > 0$ are parameters that control, respectively, the attraction and drift components. μ is the central point to which the system goes back. W_t denotes a Wiener process, a stochastic value generator. Similar jittering techniques have been implemented in Ahissar and Arieli (2012); Testa et al. (2023). The idea behind this jitter is that the camera will continuously drift from the central point μ due to the Wiener process, while being attracted back to it over time.

We created two different environments to test our framework, each having specific properties and challenges.

3.3.1 Tracking task

The tracking task consists of one motorized camera with one axis of rotation. Figure 3A presents a screenshot of the environment. A textured ball is moving in a circle around the camera, whose radius is 1.3 m from the optical center of the camera. It is either moving clockwise or counter-clockwise. The goal of the task is simple, to be able to track the ball by keeping it in the center of the visual field. The simple cell layer is composed of a thin strip of 30 neurons in width and 6 neurons in height that covers the entire ball. We specify the parameters of the architecture in Table 3.

3.3.2 Stabilization task

The stabilization task also consists of one motorized camera. The environment is shown in Figure 3B. This time, the rotation is performed around the camera's optical axis. This task tries to simulate in a simplified way the process of a flying insect that would have to keep its flight horizontal using visual cues. We generate the visual cues from a set of straight bars. A leveled flight corresponds to the bars being horizontal.

This task is harder than the previous one since the network must be able to process the bar orientation to effectively distinguish between states. We will demonstrate in Section 4.1 that the network cells can efficiently differentiate between oriented stimuli, which makes them effective state representation for such a task. The actions are rotating clockwise or counter-clockwise to bring back the bars to the desired horizontal state.

We changed the neurons' spatial arrangement for that task to be able to fit the entire visual stimulus in the visual field of the network.

TABLE 3 Connectivity parameters. The values indicate dimensions of the following form: $(x \times y \times z)$ cells.

	Simple cells		Complex cells		Critic cells	Actor cells
Retina size	# cells	Receptive field	# cells	Receptive field	# cells	# cells
Network architecture for the tracking task						
$(300 \times 60 \times 2)$	$(30 \times 6 \times 64)$	$(10 \times 10 \times 2)$	$(10 \times 2 \times 16)$	$(3 \times 3 \times 64)$	(100)	(100)
Network architecture for the stabilization task						
$(160 \times 160 \times 2)$	$(16 \times 16 \times 144)$	$(10 \times 10 \times 2)$	$(4 \times 4 \times 16)$	$(4 \times 4 \times 144)$	(100)	(100)

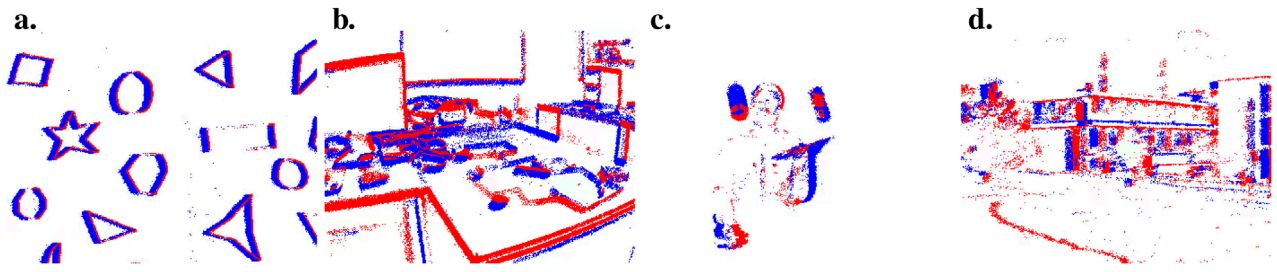

FIGURE 4
Screen capture of four event recordings, **(A)**shapes on a sheet of paper, **(B)**view of an office, **(C)**someone juggling, and **(D)**view from a robotic platform in an urban environment.

The stimulus is located in a square in the center of the visual field. We summarize the parameters of the network architecture in Table 3.

4 Results

In this section, we evaluate separately the different stages that compose our AEC model, namely the efficient coding model, the intrinsic reward calculation, and the reinforcement learner.

4.1 Efficient coding model

The efficient coding model is the foundation on which the whole AEC model rests. As discussed in Section 3.1.1, events are mostly triggered by the moving edges of objects. An edge can be characterized by different properties, such as its orientation, speed, or disparity in the case of stereovision. A good coding representation must be able to capture all this information while removing unnecessary and redundant parts of the information itself. We showed in Barbier et al. (2020, 2021); N'Dri et al. (2023) that our network is able to learn simple and complex cell-like receptive fields based on synthetic and natural event-based recordings, such as the ones presented Figure 4.

Here, we study the simple and complex cell responses to various types of event inputs. We selected four event recordings in different environments: shapes drawn on a sheet of paper moving in front of the sensor, a recording of an office, someone juggling with balls, and a mobile robotic platform in a small recreated urban road environment. Each recording is a few seconds to a few minutes long.

We created a network of similar architecture than the one presented in Table 3 for the stabilization task. We did not add actor and critic cells since we are not interested in motor control for this specific experiment. We first trained the network on the shapes recording, then we fed the four event recordings to the network and recorded the cells' activity. The shape data contain edges of multiple orientations, speeds, and widths, so that the network can efficiently learn a wide representation of oriented filters.

Event streams are already sparse by nature, but still contain up to millions of events per second. The simple cell layer, due to the integrating nature of simple cells, reduces that number by a huge factor. We recorded the number of spikes for the simple and complex cell layers, as well as the difference between the normalized event activity and the normalized simple cell activity. We present those results in Table 4.

We measure the activity reduction induced by the efficient coding layers as the number of input events divided by the total number of spikes of the layer. Here, on average, the trained efficient coding layer is able to reduce the input activity by a factor of 116 for the simple cell layer and 657 for the complex cell layer.

To verify that the encoding by the simple cell layer reflects the amount of input events we also computed the correlation coefficient between the number of events and the simple cell activity. These correlation coefficients are given in Table 4. We can observe that the correlation between the input and simple cell layer is very high, especially for the shapes recording that was used for learning the representations.

We showed in previous work N'Dri et al. (2023) that our network is capable of retaining essential information from the visual input stimuli. There we used a classical neural network approach

TABLE 4 Sparsity analysis on four different event recordings. We show the result for a network with learned representation. We present the activity reduction, the number of events divided by the number of spikes in the cell layer. We also present the correlation coefficient between the events and simple cell activity.

Scene	Urban	Office	Shapes	Juggling	Mean
# of events	11, 8M	3, 6M	6, 5M	27, 9M	
Simple cell activity reduction	173	112	67	112	116
Complex cell activity reduction	983	610	623	410	657
Simple cell correlation coefficient	0.84	0.90	0.98	0.92	0.91

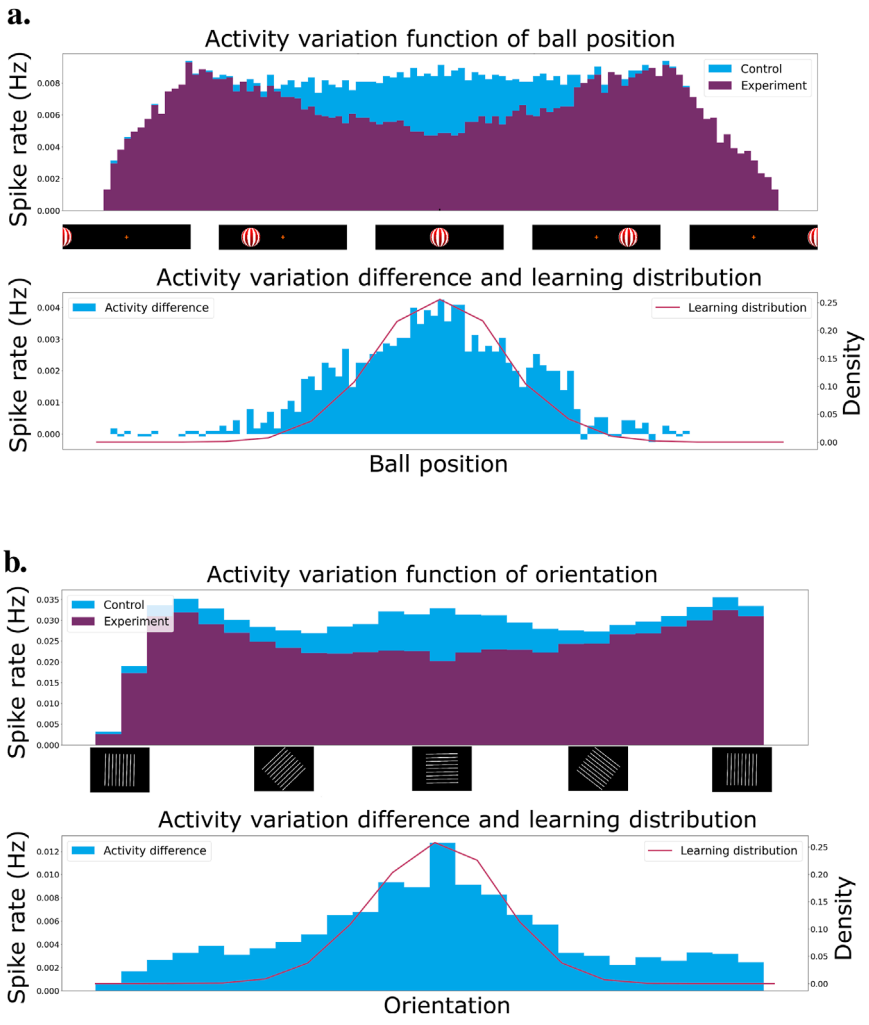

FIGURE 5
(A) Activity of the control and experiment network when presented to a ball moving from the left to the right of the visual field. In purple, the experiment network after learning the inhibition on a normal distribution of ball positions centered on the middle of the visual field. In blue, the control network is the experiment network without inhibitiory weights. The bottom graph presents the normal distribution used for learning in red superimposed on the activity difference between the control and the experiment network. **(B)** Activity of the network when presented to oriented gratings from 0 to 360°. In purple, the experiment network after learning the inhibition on a Gaussian distribution of oriented gratings centered around 0°. In blue, the control network is the experiment network with shuffled inhibition weights. The bottom graph presents the Gaussian distribution used for learning in red superimposed on the activity difference between the control and the experiment.

to reconstruct specific parameters from the activity of the simple cells layer. Information such as position, width, length, orientation and movement direction from a set of generated moving bars was correctly predicted, ranging from 80 to almost 100% accuracy depending on the type of information reconstructed. We refer the reader to the mentioned article for more detail on that experiment.

4.2 Intrinsic reward calculation

To generate an intrinsic reward signal, we rely on the ability to reduce network activity via a form of learned inhibition. This inhibition reduces network activity for frequently observed stimuli. To demonstrate that, we learn with specific well-controlled input statistics and the results show that the network activity is inversely related to the frequency of observed input patterns. Here, we are using the tracking simulation environment to generate stimuli. We show the network a repetition of a short recording that involves the ball moving back and forth around a specific part of the visual field. Since the network should show reduced activity for central visual stimuli, we need to expose the network to a majority of visual patterns happening in the center of the visual field. For that purpose, the position of the ball in the visual field is selected according to a normal distribution whose mean is located at the visual center. That way, we make sure the network is mostly exposed to visual inputs that will excite the center cells, which in turn will drive up the inhibitory connections between those cells. With enough repetitions, the network starts to learn to strongly inhibit inputs at the center of the visual field, while not so much on the borders of the visual field. In the first experiment, we fix the excitatory weights and only learn the top-down and lateral inhibitory connections.

Figure 5A presents the result of a single recording of the ball moving from the left to the right of the entire visual field at the end of learning. We recorded the total cell activity and present it as a histogram of activity over time. In blue, we have the control network, which is the network with shuffled inhibitory weights. In purple, the experiment network with inhibitory weights learned on the distribution mentioned above. The distribution can be seen in red in the bottom graph. The latter also presents the activity difference between the control and experiment network.

The activity of the experiment network drops significantly when the ball approaches the center of the visual field. More importantly, the drop is gradual. This means we can use the activity of the network as an effective intrinsic reward signal. Low activity implies a high reward, whereas high activity implies a low reward. The activity of the networks drops when the ball exits the visual field on the right or left since there are much fewer cells to excite in those regions. This does not pose a problem since we normalize the intrinsic reward by the number of events.

For the second experiment, we consider the stabilization task environment. Again, the excitatory connections are fixed and we only learn the lateral and top-down inhibitory connections. We present oriented gratings to the network whose orientations are drawn from a normal distribution centered around the horizontal orientation. The first graph in Figure 5B shows the network activity as a function of grating orientation. The distribution of orientations is shown in red in the bottom graph, along with the activity difference between the experiment and control conditions.

We observe that the activity of the network decreases significantly close to the center of the distribution, i.e., for horizontal stimuli. This is represented as a bigger activity difference between the control and experiment network. Furthermore, the decrease in activity is gradual as we change the orientation. This means we can extract a smooth reward directly from the network activity and use it for reinforcement learning.

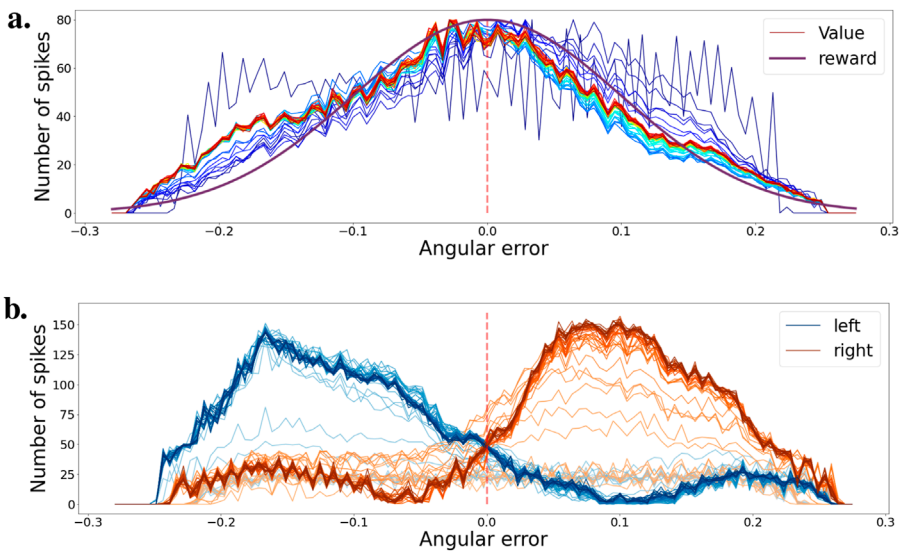

FIGURE 6
(A) Evolution of the value function during training for various ball positions. Color progression from blue (early training) to red (late training) indicates training stages. The purple curve represents the learning distribution used for learning the intrinsic reward. **(B)** Evolution of the agent's policy during training, shown as actor cell spike rates. Color intensity varies from light (early training) to heavy tones (late training). Actions for left and right movements are depicted in blue and orange, respectively.

4.3 Reinforcement learner

4.3.1 Learning the tracking task

We consider a simple scenario with only two possible actions, moving the camera to the right or to the left. We used 100 critic neurons, as well as 100 actor neurons, 50 for each action. These cells receive inputs from the simple and complex cell layers.

For this task, we first learned the weights of the efficient coding layer and fixed them to focus on the reinforcement learning framework. We learned a diverse basis similar to the one presented in Section 4.1. We start in full exploration mode, where every action is selected randomly. During that phase, only the weights of the critic neurons are updated. Every 2 s of simulation time, we decrease both the exploration rate, critic and actor learning rate as well as the time between two action selections. As we continue to decrease the exploration rate, the actions selected are less and less random, and the network fine-tunes its policy.

We recorded the weights of the network every 2 s in the simulation. Then, we observe the evolution of the network's performance in a simple validation task. The tracking environment consists of moving the ball back and forth once from the left to the right part of the visual field. For each location, we accumulate spikes from both critic and actor neurons to calculate the estimated value and policy. We submit the network to this validation for every recorded network checkpoint during training. We observe the change in value and policy as the network learns the task in Figure 6.

Figure 6A shows the evolution of the value function over the course of the whole training. Evolution over time is represented by a shift in color from blue to red. In the beginning, the value function is flat and noisy, but very quickly, over a few simulation seconds, it learns to associate a high value to states with a small angular error. As the learning rate decreases, the value stabilizes. The value function is somewhat representative of the learning distribution used in Figure 5A to generate the intrinsic reward, represented here in purple.

Figure 6B shows the policy evolution during training. The blue and orange curves correspond respectively to the left and right motor action. If the ball is in the left part of the visual field, the

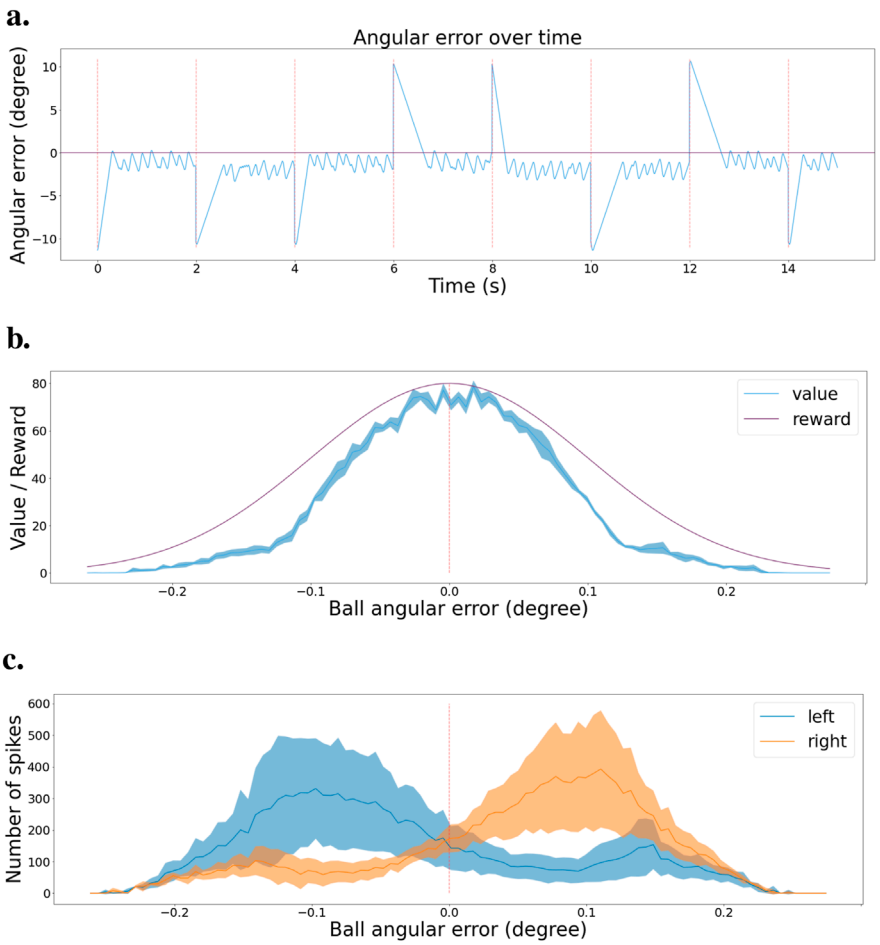

FIGURE 7
Validation scenario for the tracking task **(A)** Angular error is plotted as a function of time during the validation scenario. Red dashed lines indicate the times when the ball is reset to a random location. Zero angular error corresponds to the ball being in the center of the visual field. **(B)** Value function at the end of learning, with mean and error bands of 1 standard deviation from five experiments with different starting seeds. The purple curve represents the learning distribution used for learning the intrinsic reward, shown for visual comparison. **(C)** Agent's policy at the end of learning, depicted as actor cell spike rates with mean and error bands of 1 standard deviation from five experiments with different starting seeds. Left and right actions are shown in blue and orange, respectively.

agent must rotate the camera to the left to bring the ball back to the center. The opposite is true if the ball is in the right part of the visual field. The blue and orange curves show the actor's spike rates for the different ball positions in the visual field. As training progresses, the network learns to select the correct action depending on the position of the ball. The evolution of the policy over time during training is represented by the color intensity in the curves, from light to deep tones.

In this environment, the network was able to learn an effective policy in a very short time. After only a minute of training, the network easily differentiates the different states of the ball and can select the correct action accordingly.

To estimate network performance, we tested the final policy of the network in the simulated environment. The network selects actions every 10 ms, and every 2 s, we reset the ball to either the left or right border of the visual field. We keep track of the angular error between the center of the visual field and the center of the ball. Figure 7A shows the resulting error during the test. We observe that the network successfully brings back the ball to the center of the visual field every time the ball is reset, represented by a red dashed line. Then, the network keeps the ball in the center. We can note that the tracking is not perfectly stable, as there is some jittering when trying to keep the ball in the center, but in general, the policy is effective.

We submitted the network to a validation scenario similar to that when using two actions. The ball is going back and forth from left to right, then right to left. We kept track of the spike train of both the critic and actor neurons and created a histogram of activity based on the angular error from the ball to the center of the visual field. Figure 7B presents the critic histogram in blue, while the purple curve represents the learning distribution used to generate the intrinsic reward. We can see that the activity of the critic neurons tends to match the learning distribution. Figure 7C shows the activity of the two subgroups of actor neurons, 1 for each action. When the ball is in the left part of the visual field, the actor neurons associated with the turning left action spike the most. On the contrary, the actor neurons linked to the turning right action spike more when the ball is on the right part of the visual field. The learned policy is sensible and allows the network to efficiently track the ball in any situation.

4.3.2 Learning the stabilization task

Just like for the tracking task, we also learned the efficient coding layer independently first. We also use a similar exploration and exploitation learning strategy. We saved the weights at regular intervals during training and present the results in Figure 8. Figure 8A demonstrates that we learn an efficient value function during training. Very quickly, the value starts to reflect the learning distribution that was used for generating the intrinsic reward as seen in Figure 5B. Similarly, Figure 8B shows that the actor neurons learn to separate the states as training progresses.

We also recorded the angular error between the actual and optimal bar orientation (horizontal) during an exploitation test scenario on the fully trained network. Actions are selected every 10 ms. Figure 9A presents those results. The bar orientation is reset every second at a random orientation. We can observe that the network can bring the bars to the right orientation no matter the initial angle. Once in the right orientation, the network can keep the bars horizontal most of the time, even though it presents a small oscillating behavior.

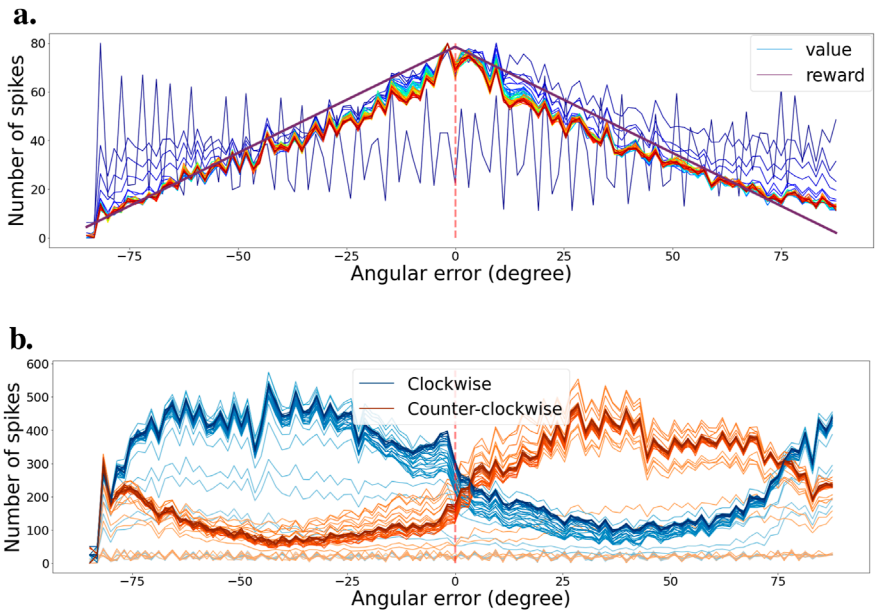

FIGURE 8
(A) Evolution of the value function during training for various bar orientations. Color progression from blue (early training) to red (late training) indicates training stages. The purple curve represents the learning distribution used for learning the intrinsic reward. **(B)** Evolution of the agent's policy during training, shown as actor cell spike rates. Color intensity varies from light (early training) to heavy tones (late training). Actions for clockwise and counter-clockwise movements are depicted in blue and orange, respectively.

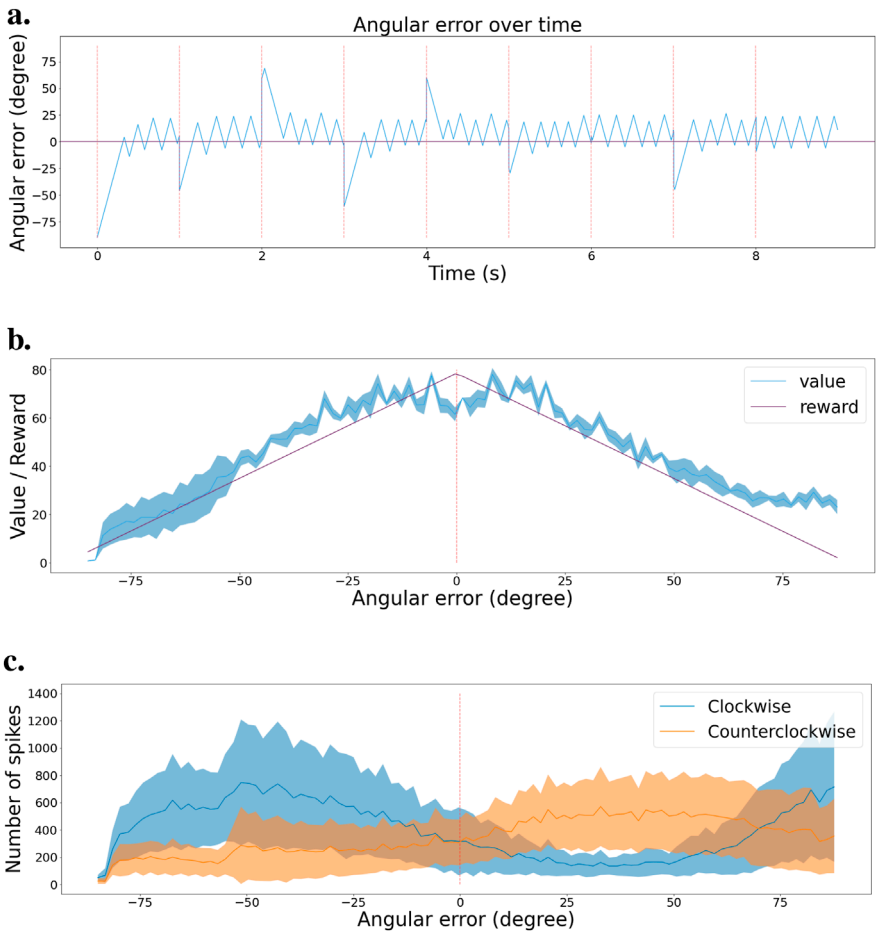

FIGURE 9
Validation scenario for the stabilization task. **(A)** Angular error is plotted as a function of time during the validation scenario. Red dashed lines indicate the times when the camera's rotation is reset to a random angle. Zero angular error corresponds to maintaining the grating horizontal. **(B)** Value function at the end of learning, with mean and error bands of 1 standard deviation from five experiments with different starting seeds. The purple curve represents the learning distribution used for learning the intrinsic reward, shown for visual comparison. **(C)** Agent's policy at the end of learning, depicted as actor cell spike rates with mean and error bands of 1 standard deviation from five experiments with different starting seeds. Clockwise and counter-clockwise actions are shown in blue and orange, respectively.

Then, similarly to the tracking task, we designed another simple validation scenario to observe the value and policy of the network when changing the bar orientations. We start with bars at −90° orientation (vertical in our case), then rotate them to +90° and back again to −90°. That way, the network observes all possible orientations in both rotation directions. During that test, we deactivated the actions and recorded the spike trains of the critic and actor neurons.

Figure 9B shows the activity histogram of the critic neurons. We averaged the clockwise and counterclockwise rotations together for more accuracy. The value function effectively associates a higher value to states with a smaller angular error. Figure 9C presents a similar representation but for the actor neurons. We can see the switch in which actor neurons spike the most around the horizontal orientation (0°). We note that the decision boundary is slightly shifted to the right, which can also be observed in Figure 9A. This shows that our spiking reinforcement learning framework works well overall, but might lack resilience for learning a precise policy on more complicated visual tasks.

5 Discussion

We have presented the first spiking neural network implementation of the AEC approach. From visual sensing with an event-based camera all the way to motor output, everything works in the spiking domain. Our network autonomously learns simple visual behaviors. The efficient coding component learns efficient visual representations in an unsupervised manner, while the reinforcement learning component uses an intrinsically generated reward based on the spiking activity of the efficient coding component. Thus, the method requires neither supervision nor an externally provided ("extrinsic") reward signal.

We demonstrated the feasibility of the approach with two admittedly very simple tasks: tracking a moving object through pursuit movements or stabilizing the orientation of a rotating (horizon) line. As such, our work should be viewed as a proof of concept rather than a highly performant solution to these particular tasks. It should be noted, however, that AEC is quite general and has been applied to various active

perception skills in the past including active stereo vision Zhao et al. (2012); Lonini et al. (2013); Klimmasch et al. (2021), active motion vision Zhang et al. (2014); Teulière et al. (2015), accommodation control Eckmann et al. (2020), torsional eye movements Zhu et al. (2022) and combinations thereof Lelais et al. (2019) as well as echolocation in bats Wijesinghe et al. (2021). This suggests that our spiking approach to AEC could be extended to any of these behaviors and possibly more.

A limitation of the approach presented here is that in order to establish the intrinsic reward signal, we enforced a particular distribution of sensory inputs (ball positions or grating orientations) during learning of the lateral and top-down inhibitory connections. Therefore, the current approach does not learn fully autonomously. This is in contrast to previous (non-spiking) AEC models which did not require such interventions.

Many parts of our model are inspired by findings on the mammalian brain, such as the presence of simple and complex cells in visual cortex or the utilization of temporal difference learning. However, the model is not fully biologically plausible. Some obvious limitations in this respect are the use of instantaneous synaptic transmission in excitatory or inhibitory connections or the violation of Dale's law, i.e. single units in our model can both excite and inhibit other units, which does not seem to be the case in biological neural networks. Also, the reinforcement learner's action choices are based on an external clock and sampled at regular intervals. We believe that some of those limitations with respect to biological plausibility could be overcome relatively easily, but this was not our focus.

Since our model is formulated entirely as a spiking neural network, it would be interesting to implement it on neuromorphic hardware Javanshir et al. (2022); Rathi et al. (2023). This could speed up computation and allow learning visual behaviors in real-time, opening the door to real-world applications of the approach.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/comsee-research/Neuvisys.

Author contributions

TB: Conceptualization, Data curation, Methodology, Software, Validation, Visualization, Writing–original draft, Writing–review and editing. CT: Conceptualization, Formal Analysis, Methodology, Supervision, Validation, Writing–review and editing. JT: Conceptualization, Formal Analysis, Methodology, Supervision, Validation, Writing–review and editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was sponsored by a public grant overseen by the French National Agency as part of the "Investissements d'Avenir" through the IMobS3 Laboratory of Excellence (ANR-10-LABX-0016) and the IDEX-ISITE initiative CAP 20–25 (ANR-16-IDEX-0001). Financial support was also received from Clermont Auvergne Metropole through a French Tech-Clermont Auvergne professorship. JT acknowledges support from the Johanna Quandt foundation.

Conflict of interest

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.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frobt.2024.1435197/full#supplementary-material

References

Ahissar, E., and Arieli, A. (2012). Seeing via miniature eye movements: a dynamic hypothesis for vision. *Front. Comput. Neurosci.* 6, 89. doi:10.3389/fncom.2012.00089

Akolkar, H., Panzeri, S., and Bartolozzi, C. (2015). Spike time based unsupervised learning of receptive fields for event-driven vision. *Proc. - IEEE Int. Conf. Robotics Automation* 2, 4258–4264. doi:10.1109/icra.2015.7139786

Anwar, H., Caby, S., Dura-Bernal, S., D'Onofrio, D., Hasegan, D., Deible, M., et al. (2022). Training a spiking neuronal network model of visual-motor cortex to play a virtual racket-ball game using reinforcement learning. *PLoS ONE* 17, e0265808. doi:10.1371/journal.pone.0265808

Barbier, T., Teulière, C., and Triesch, J. (2020). Unsupervised learning of spatio-temporal receptive fields from an event-based vision sensor. *ICANN Artif. Neural Netw. Mach. Learn.*, 622–633. doi:10.1007/978-3-030-61616-8_50

Barbier, T., Teulière, C., and Triesch, J. (2021). Spike timing-based unsupervised learning of orientation, disparity, and motion representations in a spiking neural network. *CVPR 2021 Work.* 25, 1377–1386.

Bi, Y., and Andreopoulos, Y. (2018). "Pix2nvs: parameterized conversion of pixel-domain video frames to neuromorphic vision streams," in Proceedings - International Conference on Image Processing, 2017–September (ICIP), 1990–1994.

Chandrapala, T. N., and Shi, B. E. (2016). "Invariant feature extraction from event based stimuli," in Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, 2016–July, 1–6. doi:10.1109/biorob.2016.7523449

Cheng, X., Hao, Y., Xu, J., and Xu, B. (2020). LISNN: improving spiking neural networks with lateral interactions for robust object recognition. *IJCAI Int. Jt. Conf. Artif. Intell.* 1, 1519–1525. doi:10.24963/ijcai.2020/211

Chorley, P., and Seth, A. K. (2011). Dopamine-signaled reward predictions generated by competitive excitation and inhibition in a spiking neural network model. *Front. Comput. Neurosci.* 5, 21. doi:10.3389/fncom.2011.00021

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., et al. (2022). 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Comput. Eng.* 2, 022501. doi:10.1088/2634-4386/ac4a83

Dampfhoffer, M., Mesquida, T., Valentian, A., and Anghel, L. (2023). Are snns really more energy-efficient than anns? an in-depth hardware-aware study. *IEEE Trans. Emerg. Top. Comput. Intell.* 7, 731–741. doi:10.1109/TETCI.2022.3214509

Debat, G., Chauhan, T., Cottereau, B. R., Masquelier, T., Paindavoine, M., and Baures, R. (2021). Event-based trajectory prediction using spiking neural networks. *Front. Comput. Neurosci.* 15, 47. doi:10.3389/fncom.2021.658764

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi:10.3389/fncom.2015.00099

Eckmann, S., Klimmasch, L., Shi, B. E., and Triesch, J. (2020). Active efficient coding explains the development of binocular vision and its failure in amblyopia. *Proc. Natl. Acad. Sci. U. S. A.*, 117.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Comput.* 19, 1468–1502. doi:10.1162/neco.2007.19.6.1468

Frémaux, N., Sprekeler, H., and Gerstner, W. (2010). Functional requirements for reward-modulated spike-timing-dependent plasticity. *J. Neurosci.* 30, 13326–13337. doi:10.1523/jneurosci.6249-09.2010

Frémaux, N., Sprekeler, H., and Gerstner, W. (2013). Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS Comput. Biol.* 9, 1003024. doi:10.1371/journal.pcbi.1003024

Frémeaux, N., and Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circuits* 9, 85. doi:10.3389/fncir.2015.00085

Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., et al. (2022). Event-based vision: a survey. *IEEE Trans. Pattern Analysis Mach. Intell.* 44, 154–180. doi:10.1109/tpami.2020.3008413

Gerstner, W., Lehmann, M., Liakoni, V., Corneil, D., and Brea, J. (2018). Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Front. neural circuits* 12, 53. doi:10.3389/fncir.2018.00053

Ghaemi, H., Mirzaei, E., Nouri, M., and Kheradpisheh, S. R. (2021). Biolcnet: reward-modulated locally connected spiking neural networks. *ArXiv*.

Gibaldi, A., Canessa, A., and Sabatini, S. P. (2015). "Vergence control learning through real v1 disparity tuning curves," in International IEEE/EMBS Conference on Neural Engineering, NER, 2015-July, 332–335. doi:10.1109/ner.2015.7146627

Gibaldi, A., Chessa, M., Canessa, A., Sabatini, S. P., and Solari, F. (2010). A cortical model for binocular vergence control without explicit calculation of disparity. *Neurocomputing* 73, 1065–1073. doi:10.1016/j.neucom.2009.11.016

Guyonneau, R., VanRullen, R., and Thorpe, S. J. (2004). Temporal codes and sparse representations: a key to understanding rapid processing in the visual system. *J. Physiology-Paris* 98, 487–497. doi:10.1016/j.jphysparis.2005.09.004

Hopkins, M., Pineda-Garcia, G., Bogdan, P. A., and Furber, S. B. (2018). Spiking neural networks for computer vision. *Interface Focus* 8, 20180007. doi:10.1098/rsfs.2018.0007

Hu, Y., Liu, S.-C., and Delbruck, T. (2021). v2e: From video frames to realistic dvs events

Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J. F., Bataller-Mompeán, M., and Francés-Víllora, J. V. (2015). Simplified spiking neural network architecture and stdp learning algorithm applied to image classification. *Eurasip J. Image Video Process.* 2015, 4–11. doi:10.1186/s13640-015-0059-4

Intoy, J., and Rucci, M. (2020). Finely tuned eye movements enhance visual acuity. *Nat. Commun.* 11, 795–811. doi:10.1038/s41467-020-14616-2

Jaegle, A., Mehrpour, V., and Rust, N. (2019). Visual novelty, curiosity, and intrinsic reward in machine learning and the brain. *Curr. Opin. Neurobiol.* 58, 167–174. doi:10.1016/j.conb.2019.08.004

Javanshir, A., Nguyen, T. T., Mahmud, M. A. P., and Kouzani, A. Z. (2022). Advancements in algorithms and neuromorphic hardware for spiking neural networks. *Neural Comput.* 34, 1289–1328. doi:10.1162/neco_a_01499

Jitsev, J., Morrison, A., and Tittgemeyer, M. (2012). Learning from positive and negative rewards in a spiking neural network model of basal ganglia. *Proc. Int. Jt. Conf. Neural Netw.* 80, 1–8. doi:10.1109/ijcnn.2012.6252834

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., and Masquelier, T. (2018). Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Netw.* 99, 56–67. doi:10.1016/j.neunet.2017.12.005

Klimmasch, L., Schneider, J., Lelais, A., Fronius, M., Shi, B. E., and Triesch, J. (2021). The development of active binocular vision under normal and alternate rearing conditions. *eLife* 10, e56212. doi:10.7554/elife.56212

Lelais, A., Mahn, J., Narayan, V., Zhang, C., Shi, B. E., and Triesch, J. (2019). Autonomous development of active binocular and motion vision through active efficient coding. *Front. Neurorobotics* 13, 49. doi:10.3389/fnbot.2019.00049

Lonini, L., Forestier, S., Teulière, C., Zhao, Y., Shi, B. E., and Triesch, J. (2013). Robust active binocular vision through intrinsically motivated learning. *Front. Neurorobotics* 7, 20. doi:10.3389/fnbot.2013.00020

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., Thorpe, S. J., and Masquelier, T. (2019). Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recognit.* 94, 87–95. doi:10.1016/j.patcog.2019.05.015

N'Dri, A. W., Barbier, T., Teulière, C., and Triesch, J. (2023). "Predictive coding light: learning compact visual codes by combining excitatory and inhibitory spike timing-dependent plasticity ∗," in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Vancouver, BC, Canada, 3997–4006. doi:10.1109/CVPRW59228.2023.00417

Nichols, E., McDaid, L. J., and Siddique, N. (2013). Biologically inspired snn for robot control. *IEEE Trans. Cybern.* 43, 115–128. doi:10.1109/tsmcb.2012.2200674

Paredes-Valles, F., Scheper, K. Y. W., and Croon, G. C. H. E. D. (2019). Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: from events to global motion perception. *IEEE Trans. Pattern Analysis Mach. Intell.*, 1.

Patel, D., Hazan, H., Saunders, D. J., Siegelmann, H. T., and Kozma, R. (2019). Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. *Neural Netw.* 120, 108–115. doi:10.1016/j.neunet.2019.08.009

Paulun, L., Wendt, A., and Kasabov, N. (2018). A retinotopic spiking neural network system for accurate recognition of moving objects using NeuCube and dynamic vision sensors. *Front. Comput. Neurosci.* 12, 42. doi:10.3389/fncom.2018.00042

Ponulak, F., and Kasinski, A. (2010). Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting. *Neural Comput.* 22, 467–510. doi:10.1162/neco.2009.11-08-901

Potjans, W., Morrison, A., and Diesmann, M. (2009). A spiking neural network model of an actor-critic learning agent. *Neural Comput.* 21, 301–339. doi:10.1162/neco.2008.08-07-593

Rathi, N., Chakraborty, I., Kosta, A., Sengupta, A., Ankit, A., Panda, P., et al. (2023). Exploring neuromorphic computing based on spiking neural networks: algorithms to hardware. *ACM Comput. Surv.* 55, 1–49. doi:10.1145/3571155

Shrestha, S. B., and Orchard, G. (2018). Slayer: spike layer error reassignment in time. *Adv. Neural Inf. Process. Syst. 2018-Decem*, 1412–1421.

Tang, G., Kumar, N., Yoo, R., and Michmizos, K. P. (2020). Deep reinforcement learning with population-coded spiking neural network for continuous control

Tavanaei, A., Masquelier, T., and Maida, A. (2018). Representation learning using event-based stdp. *Neural Netw.* 105, 294–303. doi:10.1016/j.neunet.2018.05.018

Testa, S., Sabatini, S. P., and Canessa, A. (2023). Active fixation as an efficient coding strategy for neuromorphic vision. *Sci. Rep.* 13, 7445. doi:10.1038/s41598-023-34508-x

Teulière, C., Forestier, S., Lonini, L., Zhang, C., Zhao, Y., Shi, B., et al. (2015). Self-calibrating smooth pursuit through active efficient coding. *Robotics Aut. Syst.* 71, 3–12. doi:10.1016/j.robot.2014.11.006

Weidel, P., Duarte, R., and Morrison, A. (2021). Unsupervised learning and clustered connectivity enhance reinforcement learning in spiking neural networks. *Front. Comput. Neurosci.* 15, 543872. doi:10.3389/fncom.2021.543872

Wijesinghe, L. P., Wohlgemuth, M. J., So, R. H., Triesch, J., Moss, C. F., and Shi, B. E. (2021). Active head rolls enhance sonar-based auditory localization performance. *PLoS Comput. Biol.* 17, e1008973. doi:10.1371/journal.pcbi.1008973

Yuan, M., Wu, X., Yan, R., and Tang, H. (2019). Reinforcement learning in spiking neural networks with stochastic and deterministic synapses. *Neural Comput.* 31, 2368–2389. doi:10.1162/neco_a_01238

Zhang, C., Zhao, Y., Triesch, J., and Shi, B. E. (2014). Intrinsically motivated learning of visual motion perception and smooth pursuit. *Proc. - IEEE Int. Conf. Robotics Automation* 37, 1902–1908. doi:10.1109/icra.2014.6907110

Zhao, Y., Rothkopf, C. A., Triesch, J., and Shi, B. E. (2012). "A unified model of the joint development of disparity selectivity and vergence control," in 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL), 1–6. doi:10.1109/devlrn.2012.6400876

Zhu, Q., Zhang, C., Triesch, J., and Shi, B. E. (2022). Learning torsional eye movements through active efficient coding. *Neuromorphic Comput. Eng.* 2, 034007. doi:10.1088/2634-4386/ac84fd