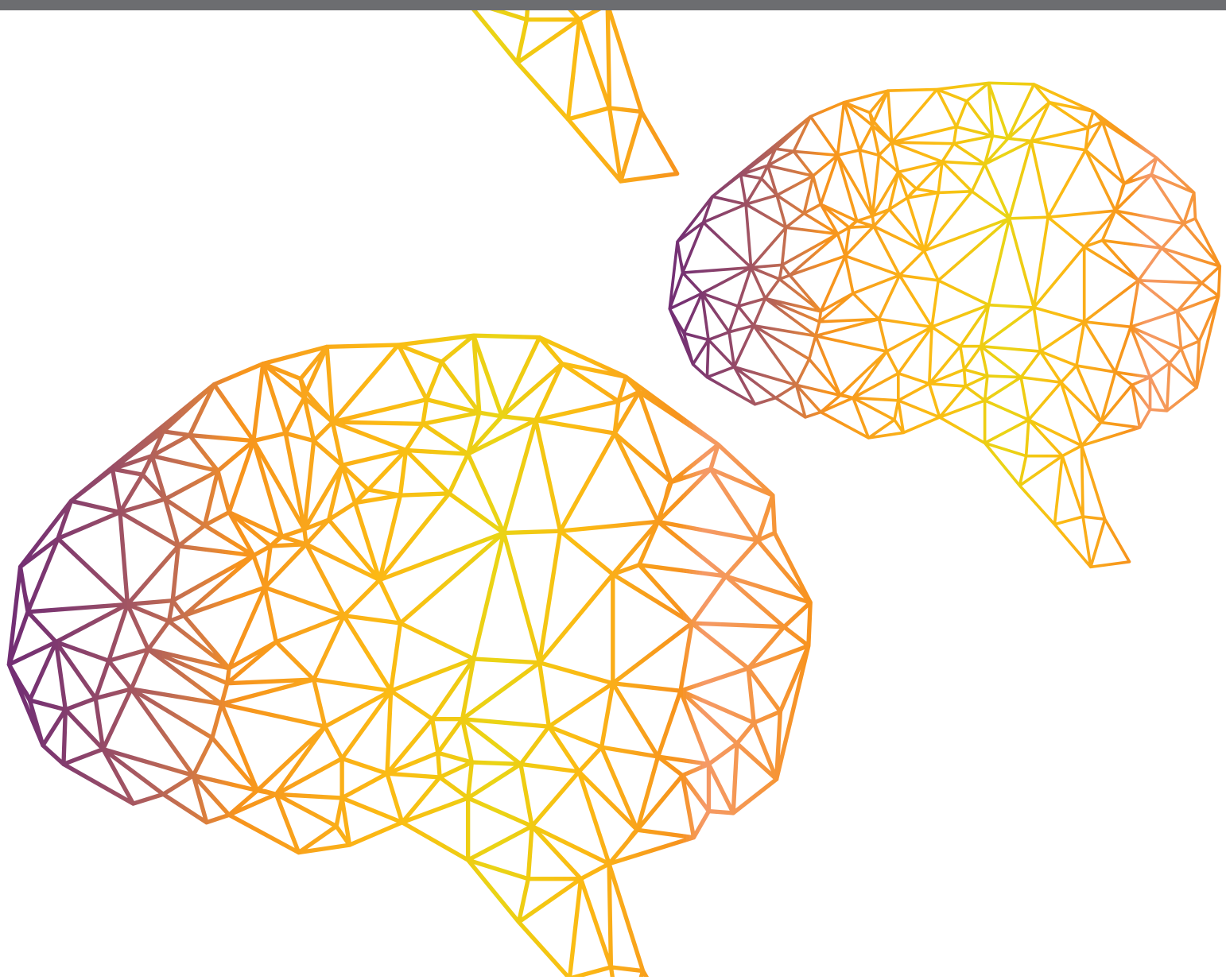
NEW ADVANCES AT THE INTERSECTION OF BRAIN-INSPIRED LEARNING AND DEEP LEARNING IN AUTONOMOUS VEHICLES AND ROBOTICS

EDITED BY: Guang Chen, Pascual Campoy, Changhong Fu and Caixia Cai

PUBLISHED IN: Frontiers in Neurorobotics



frontiers Research Topics

frontiers

ISSN 1664-8714
ISBN 978-2-88963-971-7
DOI 10.3389/978-2-88963-971-7

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

NEW ADVANCES AT THE INTERSECTION OF BRAIN-INSPIRED LEARNING AND DEEP LEARNING IN AUTONOMOUS VEHICLES AND ROBOTICS

Topic Editors:
Guang Chen, Tongji University, China
Pascual Campoy, Polytechnic University of Madrid, Spain
Changhong Fu, Tongji University, China
Caixia Cai, Agency for Science, Technology and Research (A*STAR), Singapore

Citation: Chen, G., Campoy, P., Fu, C., Cai, C., eds. (2020). New Advances at the Intersection of Brain-Inspired Learning and Deep Learning in Autonomous Vehicles and Robotics. Lausanne: Frontiers Media SA. doi: 10.3389/978-2-88963-971-7

Table of Contents

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 04 April 2019
doi: 10.3389/fnbot.2019.00011

Robust Learning Control for Shipborne Manipulator With Fuzzy Neural Network

Zhiqiang Xu, Wanli Li and Yanran Wang*

School of Mechanical Engineering, Tongji University, Shanghai, China

The shipborne manipulator plays an important role in autonomous collaboration between marine vehicles. In real applications, a conventional proportional-derivative (PD) controller is not suitable for the shipborne manipulator to conduct safe and accurate operations under ocean conditions, due to its bad tracing performance. This paper presents a real-time and adaptive control approach for the shipborne manipulator to achieve position control. This novel control approach consists of a conventional PD controller and fuzzy neural network (FNN), which work in parallel to realize PD+FNN control. Qualitative and quantitative tests of simulations and real experiments show that the proposed PD+FNN controller achieves better performance in comparison with the conventional PD controller, in the presence of uncertainty and disturbance. The presented PD+FNN eliminates the requirements for precise tuning of the conventional PD controller under different ocean conditions, as well as an accurate dynamics model of the shipborne manipulator. In addition, it effectively implements a sliding mode control (SMC) theory-based learning algorithm, for fast and robust control, which does not require matrix inversions or partial derivatives. Furthermore, simulation and experimental results show that the angle compensation deviation of the shipborne manipulator can be improved in the range of $\pm 1^{\circ}$.

Keywords: shipborne manipulator, real-time adaptive control, conventional PD controller, fuzzy neural network, sliding mode control, experiment verification

OPEN ACCESS

Edited by:
Caixia Cai,
*Agency for Science, Technology and Research (A*STAR), Singapore*

Reviewed by:
Yingbai Hu,
Technische Universität München, Germany
Dawei Gong,
Juilliard School, United States

****Correspondence:***
Wanli Li
1710213@tongji.edu.cn

Received: *14 January 2019*
Accepted: *13 March 2019*
Published: *04 April 2019*

Citation:
Xu Z, Li W and Wang Y (2019) Robust Learning Control for Shipborne Manipulator With Fuzzy Neural Network. Front. Neurorobot. 13:11. doi: 10.3389/fnbot.2019.00011

1. INTRODUCTION

The shipborne manipulator has become the most important tool in achieving autonomous cargo reloading between marine vehicles. With the use of the shipborne manipulator, onboard physical labor can be greatly reduced. However, unpredictable ship motion has a great impact on the maneuverability of the manipulator in real applications due to the complexity of the marine environment. If the sea state reaches Level-4, i.e., the height of a sea wave is larger than 1.52 m and wind speed exceeds 10.8 m/s, arm movement of the manipulator is extremely limited because of the influence of sling inertia and non-linear ship pose variation. In this case, the operational capacity of the manipulator is reduced by more than 50% or the manipulator is even temporarily suspended.

For the shipborne manipulator, changing boom inclination is realized through the expansion and contraction of its amplitude cylinder. The energy-saving and vibration-damping function of its accumulator is of great significance for improving manipulator control. In literature, the cylinder-accumulator in a manipulator has been studied in different types of engineering applications (Xiao et al., 2014; Shen et al., 2015; Zhao et al., 2017; Xia et al., 2018).

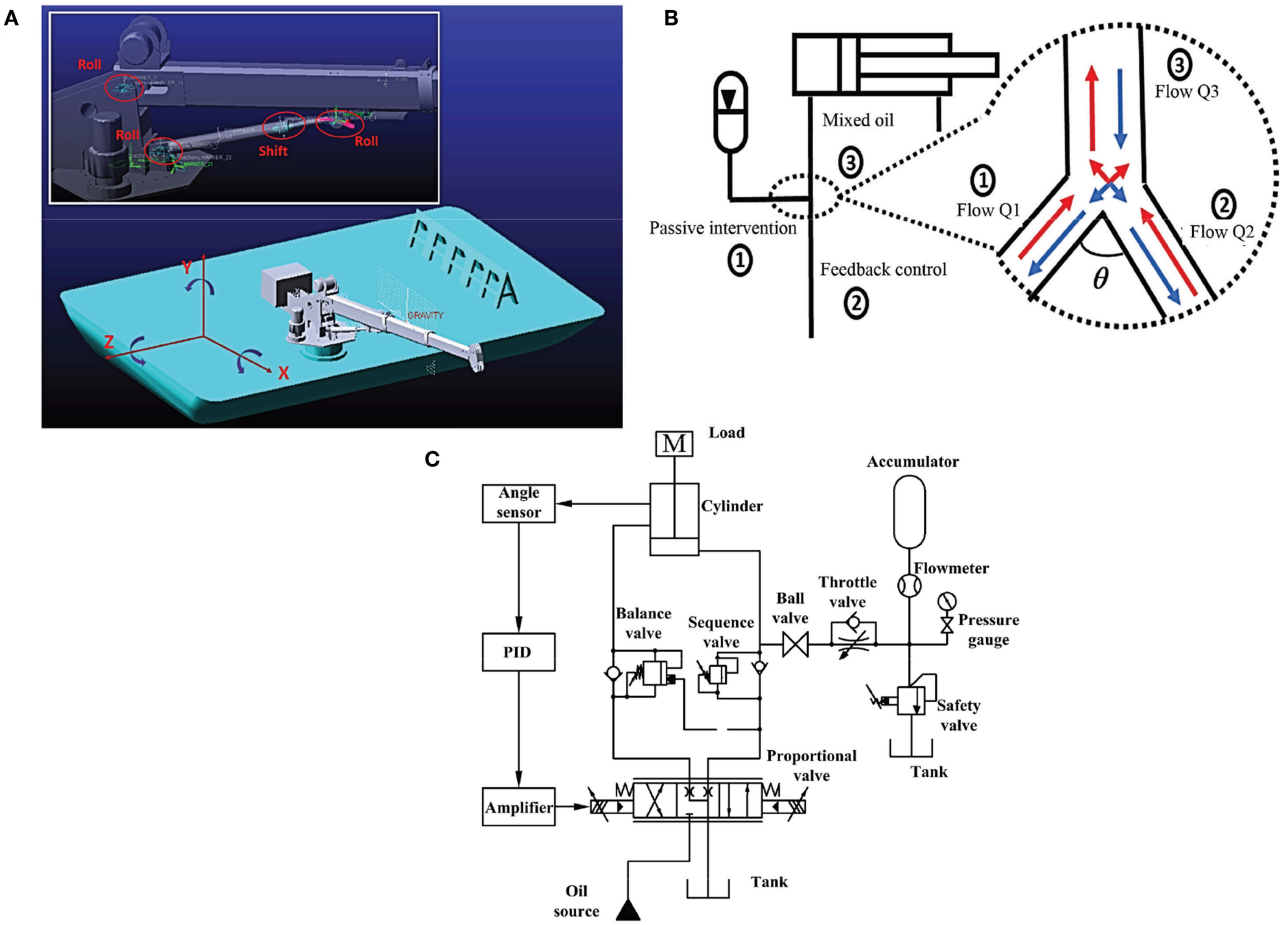


FIGURE 1 | Cylinder control strategy with the intervention of an accumulator, **(A)** is the joints of manipulator, **(B)** is the flow direction indication, **(C)** is the hydraulic system diagram with control signals.

Specifically, the accumulator is a key component in the design of hydraulic hybrid structures, which ensures acceptable shock absorption performance and system energy consumption. Its different designs are used in other construction machines such as a rock drill (Yang et al., 2017), Scraper (Junke and Zhen, 2017), and Fast Forging Press (Zhang et al., 2016). However, there are few investigations related to the influence of the accumulator on the valve control system. According to the valve-controlled cylinder-accumulator model, the accumulator is used as an energy-saving and oil damping source in parallel with the rodless cavity of the amplitude cylinder. **Figure 1** shows the details of connecting the accumulator with the cylinder.

In literature, the conventional PD controller is often used to control different types of manipulators (Cervantes and Alvarez-Ramirez, 2001; Alvarez-Ramirez et al., 2003; Su et al., 2010). However, it is not suitable for controlling the hydraulic system discussed in this work, due to its high-order non-linearity, time-varying and hysteresis characteristics. It cannot control the hydraulic system in time and is vulnerable to environmental interference. Additionally, a significant steady-state error still exists, even if plenty of time is used to tune the appropriate values for the conventional PD controller. In order to control the manipulator, calculating torque is the simplest control approach, but this approach relies on the accurate mechanical model of the system. To overcome this issue, the model-free approach has gained respectable attention since it does not require the precise model of the system and is more robust in response to uncertainty and disturbance.

The fuzzy logic controller (FLC) is widely applied to handle uncertainty and disturbance in many systems (Hasanien and Matar, 2015; Dabbaghjamanesh et al., 2016; Vaidyanathan and Azar, 2016), especially in different kinds of robots (Fu et al., 2013; Tai et al., 2016; Sarabakha et al., 2018). However, the FLC also requires a lot of time in order to tune the proper parameters to achieve a satisfied control performance. Recently, the FLC has been combined with an artificial neural network (ANN), i.e., fuzzy neural network (FNN), to overcome the aforementioned weakness of the FLC. In literature, the FNN has been successfully applied in identification and non-linear system controlling (Lin et al., 2015; Tang et al., 2017; He and Dong, 2018). At the same time, the sliding mode control (SMC) theory-based algorithm has been presented as a faster learning approach for tuning

the FNN parameters, due to its faster convergence speed and higher robustness to uncertainty and disturbance (Lin et al., 2014). Moreover, the FNN, trained with the SMC theory-based algorithm, has been successfully used in controlling a spherical rolling robot (Kayacan et al., 2013), a robotic arm (Wai and Muthusamy, 2013), and a gyroscope (Yan et al., 2017).

In this work, an FNN is proposed to work in parallel with a conventional PD controller to achieve a PD+FNN controller. It not only overcomes the original defects of conventional PD control but also significantly enhances self-learning abilities and adaptabilities. In order to validate the proposed control strategy, simulation and experimental tests have been implemented. The qualitative and quantitative results show that the presented strategy is feasible and practical. In addition, it outperforms the conventional PD controller. The main contributions of this work are listed below:

- Designing a novel control strategy for real-time control of shipborne manipulator. The presented control strategy consists of a conventional PD controller and FNN, which combines a fuzzy logic controller and artificial neural network.
- Developing online adaptation laws to eliminate the requirement for precise tuning of the controller in the shipborne manipulator.
- Qualitative and quantitative tests in the simulation and real experiments have been conducted to evaluate the control performance of the presented PD+FNN control strategy.

The organization of this paper is as follows: In section 2, the dynamic model of the shipborne manipulator is introduced. In section 3, the PD+FNN control strategy is described. In section 4, different simulation tests are conducted in order to verify the proposed control strategy. In section 5, the real experimental tests on the swaying platform are performed to validate the proposed controller. Finally, conclusions are drawn in section 6.

2. DYNAMIC MODEL

Table 1 shows the simulation parameters. **Figure 2** shows the flow direction of the oil. The accumulator is linked with the rodless cavity of the cylinder. When the cylinder extends, the accumulator and the valve supply oil to the cylinder simultaneously, making the cylinder stretch out faster. The coupling dynamics model of the parallel accumulator of the valve-controlled cylinder system is established based on the flow continuity equation and the dynamic equation.

2.1. Cylinder Dynamic Equation

The cylinder dynamic equation can be obtained by ignoring the cylinder cavity pressure, which is defined as:

$$pA = m_t\ddot{x} + B_P\dot{x} + k_z x\,, \tag{1}$$

where p denotes the working pressure, N/m^2; A denotes the action area of rodless cavity, m^2; m_t denotes the mass of the piston, kg; x denotes the displacement of pistol, m; B_P denotes the viscosity damping coefficient, $N \cdot s/m$; k_z denotes the spring stiffness, N/m.

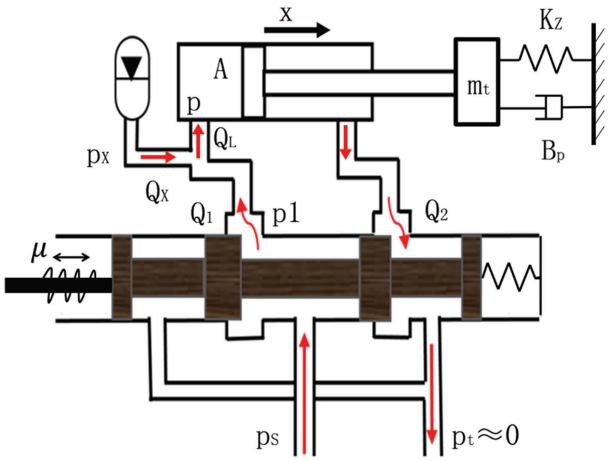


FIGURE 2 | Connection schematic diagram of valve and accumulator.

2.2. Cylinder Flow Equation

The cylinder flow continuity equation is:

$$Q_L = A\dot{x} + C_i p\,, \tag{2}$$

where C_i denotes the internal leakage coefficient, $m^5/N \cdot s$; Q_L denotes the total flow into rodless cavity of cylinder, m^3/s, which is defined as:

$$Q_L = Q_1 + Q_X\,, \tag{3}$$

where Q_1 denotes the oil flow from valve, m^3/s; Q_X denotes the oil flow from the accumulator, m^3/s. Accumulator energy release process can be regarded as interference according to Gaussian distribution, i.e.,:

$$Q_X = \mathcal{N}\left(\mu_v, \sigma_v^2\right)\,, \tag{4}$$

where μ_v and σ_v are the mean value and the standard deviation of the accumulator output flow, respectively.

2.3. Valve Flow Equation

The spool flow is a function of the working pressure and the displacement of the spool. The spool can be viewed as a zero-open four-way spool valve. The valve flow equation is defined as follows:

$$Q_1 = C_d w x_v \sqrt{\frac{2}{\rho}\left(p_S - p_1\right)} = k_q x_v\,, \tag{5}$$

where C_d denotes the flow coefficient of valve; w denotes the valve area gradient, m^2/m; x_v denotes the displacement of valve core, m; $x_v = k_v \cdot \mu$, k_v denotes the spool scale factor; μ denotes current signal; ρ denotes the oil density, kg/m^3; ps denotes the system pressure, N/m^2; p_1 denotes the pressure of the rodless cavity, N/m^2; k_q denotes the flow gain coefficient, m^2/s, which is defined as:

$$k_q = C_d w \sqrt{\frac{2}{\rho}\left(p_S - p_1\right)}\,, \tag{6}$$

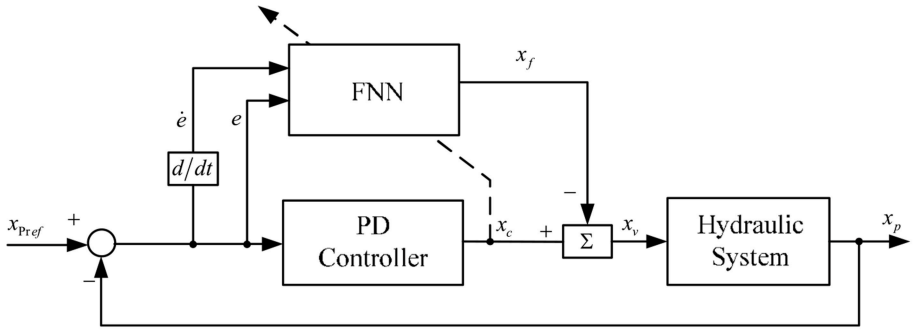


FIGURE 3 | Presented PD+FNN control strategy for shipborne manipulator.

Assume that the state of the system is $x_1 = x$, $x_2 = \dot{x}$, the dynamic model of the valve-controlled cylinder system can be defined as follows:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \theta_1 x_1 + \theta_2 x_2 + g\mu + d\,, \\ x = x_1 \end{cases} \tag{7}$$

where

$$\begin{cases} \theta_1 = -\frac{k_z}{m_t} \\ \theta_2 = -\left(\frac{A^2 + C_i B_P}{m_t C_i}\right) \\ g = \frac{k_q k_v A}{m_t C_i} \\ d = \frac{\mathcal{N}(\mu_v, \sigma_v^2) A}{m_t C_i} \end{cases}. \tag{8}$$

3. PD+FNN CONTROL STRATEGY

3.1. Overview of Control Strategy

Figure 3 shows the presented PD+FNN control strategy, in which the conventional PD controller works in parallel with the fuzzy-neuro controller, as the FNN block shows in **Figure 3**. The PD controller is utilized to not only trace the target value by system error, i.e., $e = x_{pref} - x_p$, but also to provide learning errors to train the FNN online. The FNN is supposed to improve the control accuracy and offset the effects of system interference.

3.2. Fuzzy Neural Network Construction

The proposed FNN consists of two input signals, i.e., $x_1 = e$ and $x_2 = \dot{e}$, and one output signal x_f. Takagi-Sugeno-Kang (TSK) fuzzy model (Lin et al., 2015; Precup et al., 2015) is used in which the antecedent part is the fuzzy set and the consequent part consists of only crisp numbers. The rth rule of a zero-order TSK model with two input variables x_1 and x_2 can be defined as follows:

$$\text{IF } x_1 \text{ is } M_{1i} \text{ and } x_2 \text{ is } M_{2j}\,, \text{ THEN } f_{ij} = d_{ij}\,, \tag{9}$$

where f_{ij} is the time-varying parameter of the consequent part. d_{ij} is the coefficient of the output function for the rth rule, and M_{1i} and M_{2j} are fuzzy sets. Therefore, the inputs can be represented as μ_{1i} and μ_{2j}, respectively. The firing strength of the rth rule is computed as the T-norm (multiplication) of the member functions (MFs) in the antecedent part (Imanberdiyev and Kayacan, 2018):

$$W_{ij} = \mu_{1i}(x_1)\mu_{2j}(x_2)\,, \tag{10}$$

The output signal of the system can be derived using the normalized values of the firing strength $\tilde{W}_{ij}$ with the following form (Biglarbegian et al., 2010):

$$u_f = \sum_{i=1}^{I}\sum_{j=1}^{J} f_{ij}\tilde{W}_{ij}\,, \tag{11}$$

where J and I represent the number of MFs for x_2 and x_1, respectively. $\tilde{W}_{ij}$ is expressed as follows:

$$\tilde{W}_{ij} = \frac{W_{ij}}{\sum_{i=1}^{I}\sum_{j=1}^{J} W_{ij}}\,. \tag{12}$$

Overall control input u to the system is defined as follows:

$$x_v = x_c - x_f\,, \tag{13}$$

where x_c and x_f are the control signals produced by the PD controller and the FNN controller, respectively.

3.3. Triangular Fuzzy MFs

In the FNN, the fuzzy MFs play the important role of overcoming environmental interference. These MFs have already shown promising results for control (Khanesar et al., 2015) and identification (Khanesar et al., 2011) purposes. In this work, typical triangular fuzzy MFs are chosen in order to achieve a faster and robust control performance. The MF is defined as follows:

$$\mu(x) = \begin{cases} \left(\mathbf{1} - \left|\frac{x-c}{d}\right|\right) & \text{if } c - d < x < c + d \\ \mathbf{0} & \text{otherwise} \end{cases}, \tag{14}$$

where x is the input, d and c are the width and the center of the MF. The stability proof can be found in Kayacan and Khanesar (2015) according to sliding mode control theory.

3.4. Sliding Mode Control Theory-Based Training Approach

In this paper, SMC based parameter update rules are proposed to guarantee the stability of the system and provide favorable robustness. By utilizing the principles of the SMC theory, the zero dynamics of the learning error coordinate $x_c(t)$ can be described as a time-varying sliding surface S_c in the following form:

$$S_c(x_f, x_v) = x_c(t) = x_f(t) + x_v(t) = 0\,. \tag{15}$$

If this condition is satisfied, the FNN structure is trained to become the non-linear regulator which assists the parallel controller (in our case PD controller), and the desired performance of the system can be obtained. Therefore, the sliding surface for the non-linear system under control is given by

$$S_p(e, \dot{e}) = \dot{e} + \chi e\,, \tag{16}$$

with $\chi > 0$ being a positive parameter which defines the desired trajectory of the error signal.

The time-varying parameter of the consequent part $\dot{f}_{ij}$ is updated based on the following adaptation law (Kayacan and Khanesar, 2015):

$$\dot{f}_{ij} = -\frac{\tilde{W}_{ij}}{\prod^T \prod} \alpha \text{sign}(x_c), \tag{17}$$

where

$$\prod = \left(\sum_{i=1}^{I} \sum_{j=1}^{J} \tilde{W}_{ij} \right)\,, \tag{18}$$

the learning rate $\alpha > 0$ is updated based on the following equation:

$$\dot{\alpha} = |x_c|\,, \tag{19}$$

The adaptation law for the premise part is given as follows (Kayacan and Khanesar, 2015):

$$\dot{c}_{1i} = -\gamma_1 \left|d_{1i}\right| (1 - T_{1i})\, \text{sgn}\,(x_1 - c_{1i}) \times H\left(x_1, c_1 - d_1, c_1 + d_1\right), \tag{20}$$

$$\dot{d}_{1i} = -\gamma_1 \frac{(1 - T_{1i})\left|d_{1i}\right|}{|x_1 - c_{1i}|} \text{sgn}\left(d_{1i}\right) \times H\left(x_1, c_1 - d_1, c_1 + d_1\right), \tag{21}$$

$$\dot{c}_{2i} = -\gamma_1 \left|d_{2j}\right| \left(1 - T_{2j}\right) \text{sgn}\,(x_2 - c_{2i}) \times H\left(x_2, c_2 - d_2, c_2 + d_2\right), \tag{22}$$

$$\dot{d}_{2i} = -\gamma_1 \frac{\left(1 - T_{2j}\right)\left|d_{2j}\right|}{|x_1 - c_{2j}|} \text{sgn}\left(d_{2j}\right) \times H\left(x_2, c_2 - d_2, c_2 + d_2\right), \tag{23}$$

where

$$H(x, c, d) = \begin{cases} x \text{ if } c - d < x < c + d \\ 0 \text{ otherwise} \end{cases}\,, \tag{24}$$

$$\begin{cases} T_{1,i} = \left|\frac{x_1 - c_i}{d_i}\right| \\ T_{2,i} = \left|\frac{x_2 - c_i}{d_i}\right| \end{cases}\,. \tag{25}$$

For the γ_1, it needs to be selected as positive (Kayacan and Khanesar, 2015).

4. SIMULATION AND RESULTS ANALYSIS

4.1. Simulation Parameter

The control gains for the PD controller are chosen as follows: $k_p = 10, k_d = 5$.

4.2. Simulation Results

Table 1 shows the simulation parameters. **Figures 4–7** shows the simulation results without noise. The adaptive learning capabilities of the PD+FNN structure can provide superior performance in different conditions. It is able to solve limitations such as the lack of modeling and existing uncertainties in the environment and is therefore more suitable for real-time applications.

As seen from **Figure 4**, the PD+FNN controller has a faster and more stable response performance. **Figure 5** shows that the controller has a better adaptive learning property to lessen the

TABLE 1 | Simulation parameters.

Parameter name	Description	Value	Unites
A	action area of rodless cavity	6.36×10^{-3}	m^2
m_t	mass of the piston	7.0	kg
B_P	viscosity damping coefficient	0.2	$N{\cdot}s/m$
k_z	spring stiffness	8.0×10^2	N/m
C_i	leakage coefficient	5.1×10^{-7}	$m^5/N{\cdot}s$
k_q	flow gain coefficient	0.868	m^2/s
k_v	proportional coefficient of core	10^{-3}	

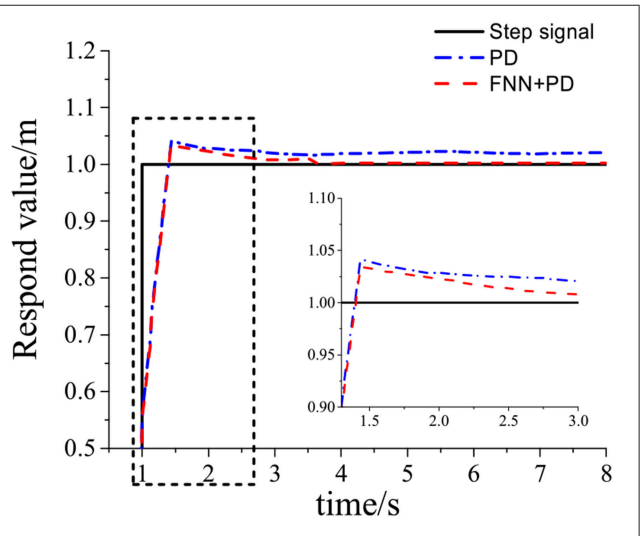


FIGURE 4 | Tracking response of step signal.

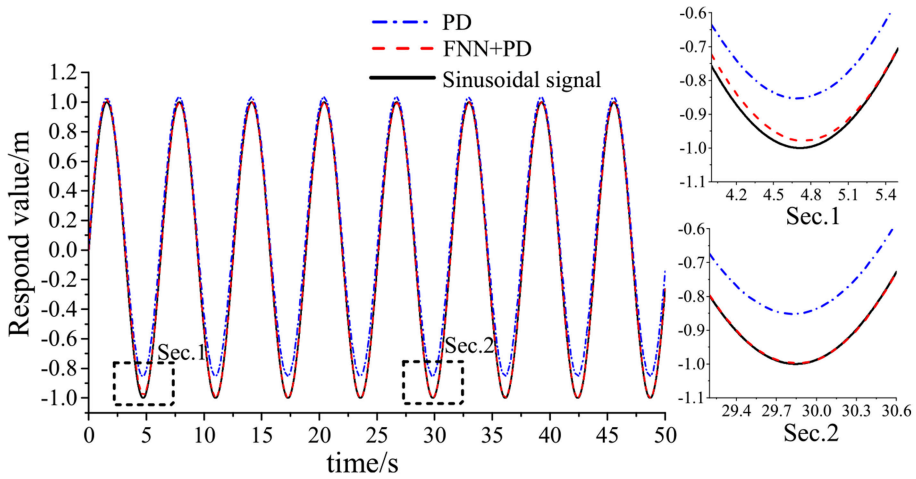


FIGURE 5 | Tracking response of sinusoidal signal.

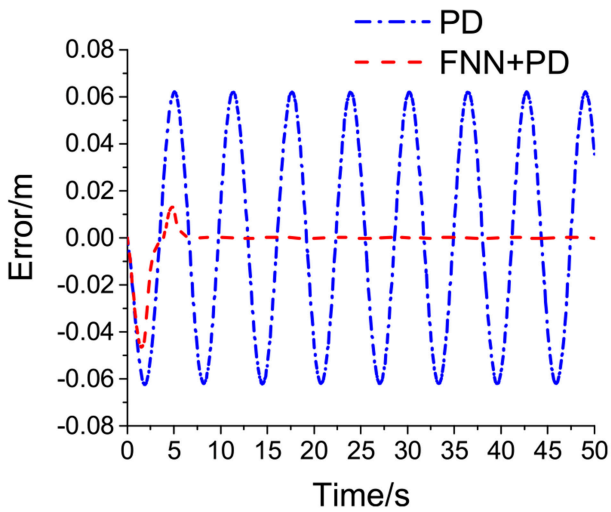


FIGURE 6 | Euclidean error.

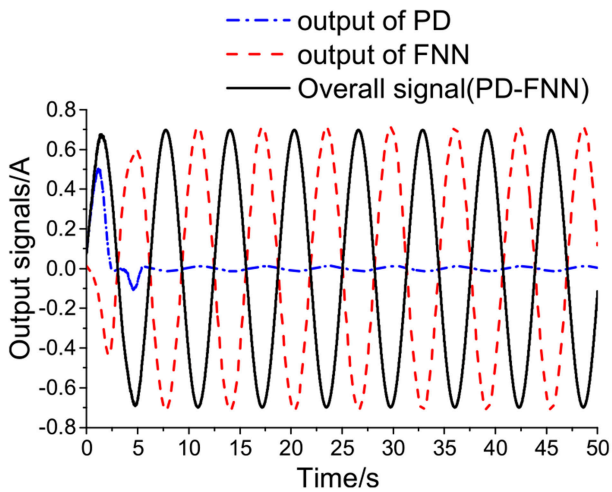


FIGURE 7 | Control signals of PD+FNN.

error gradually. As shown in **Figure 6**, although the PD controller ensures the error signal is bounded in the neighborhood of zero, significant steady-state errors that occur from internal or external interferences cannot be eliminated. Compared to the PD controller, the PD+FNN controller eliminates the steady-state error.

Figure 7 shows curves of the overall signal (which is defined as $x_v = x_c - x_f$), the output of FNN (x_f), and the output of the conventional PD controller (x_c). As seen in **Figure 7**, the overall control signal is close to the conventional PD controller at the beginning of the simulation, then the FNN learns the dynamics of the system and takes responsibility for the system. Simultaneously, the output of the PD controller tends to go to zero.

As discussed in the section 2.2, the accumulator energy release process can be regarded as interference according to Gaussian distribution. In order to create different noise levels, four different mean values of the accumulator output flow are chosen, i.e., $\mu_v = \{100, 150, 200, 300\}$ [L/min]. For the standard deviation, it is selected as $\sigma_v = 70$ L/min.

Figure 8 shows the manipulator position errors under PD controller with the noise level from 0.12 to 0.24 m.

Figure 8 shows that the PD controller cannot handle steady-state errors that occur from internal or external interferences. However, the PD+FNN controller can eliminate steady-state errors through an adaptive learning algorithm, and can be used in a real time control to cope with noisy measurements and uncertainties in the system more effectively.

Figure 9 shows the output signals of PD+FNN in different levels of noise. **Figure 9**, shows that the overall control signal is determined by the FNN controller, and the output signal from PD tends to be close to zero. Therefore, the FNN learns the

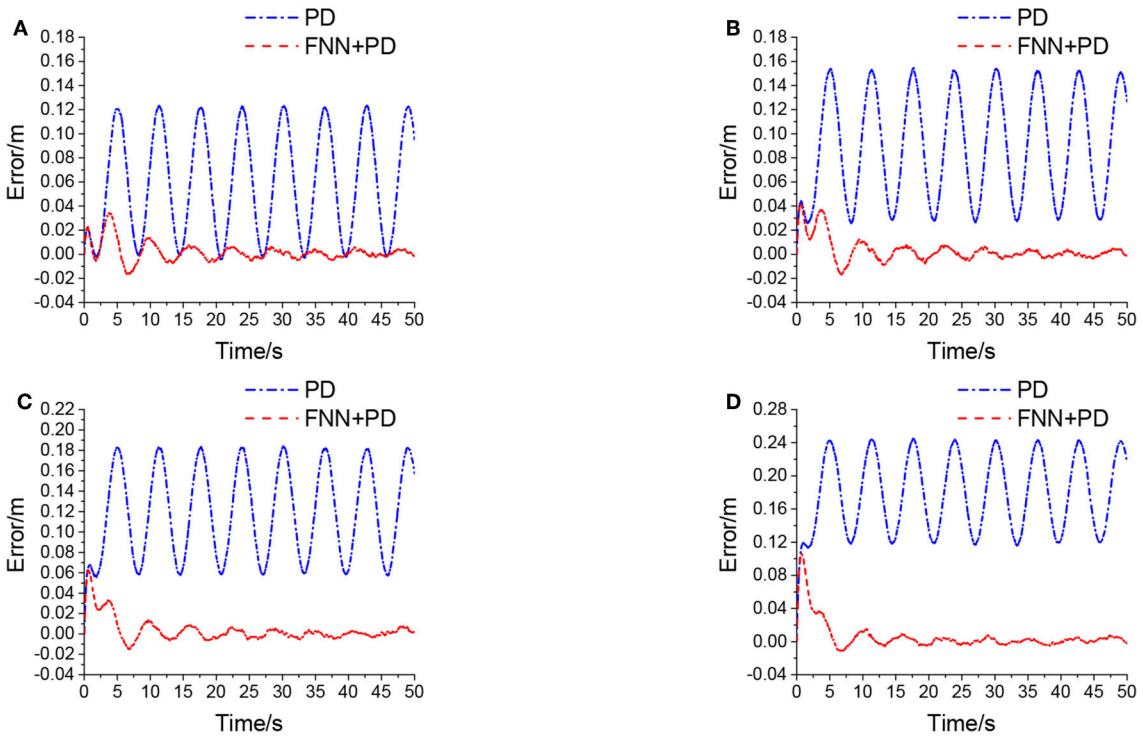


FIGURE 8 | Euclidean error in different levels of noise, **(A)** is $\mu_V = 100$ *L/min*, $\sigma_V = 70$ *L/min*, **(B)** is $\mu_V = 150$ *L/min*, $\sigma_V = 70$ *L/min*, **(C)** is $\mu_V = 200$ *L/min*, $\sigma_V = 70$ *L/min*, **(D)** is $\mu_V = 300$ *L/min*, $\sigma_V = 70$ *L/min*.

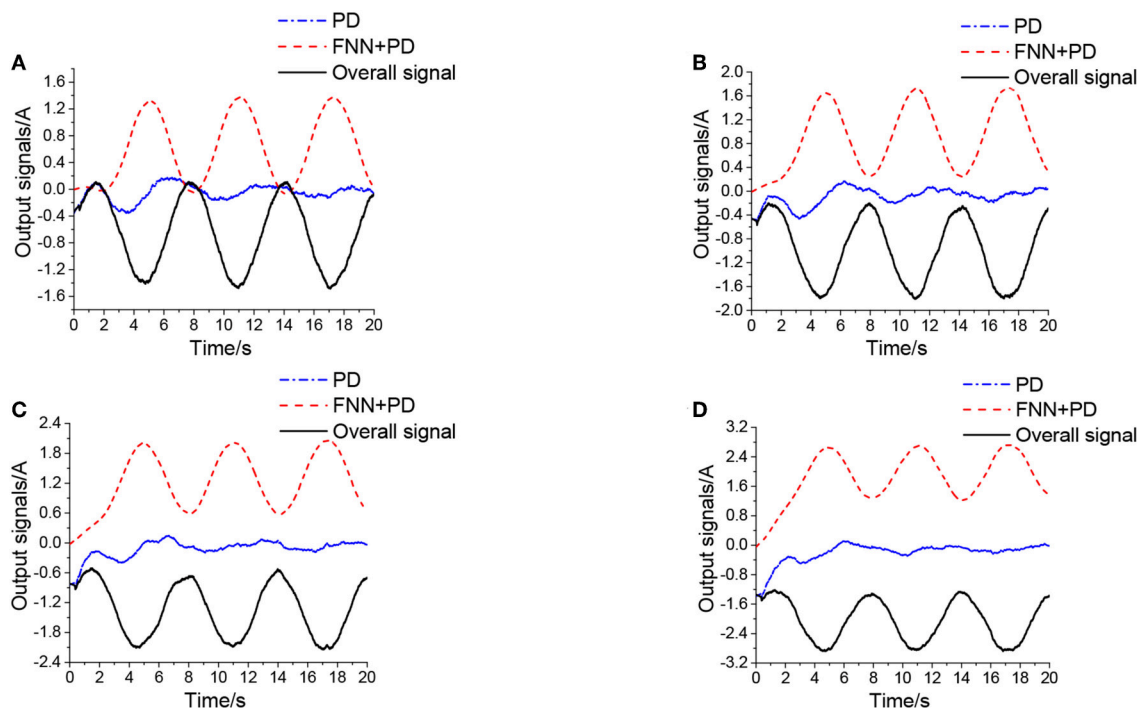


FIGURE 9 | Output signals of PD+ FNN in different levels of noise, **(A)** is $\mu_V = 100$ *L/min*, $\sigma_V = 70$ *L/min*, **(B)** is $\mu_V = 150$ *L/min*, $\sigma_V = 70$ *L/min*, **(C)** is $\mu_V = 200$ *L/min*, $\sigma_V = 70$ *L/min*, **(D)** is $\mu_V = 300$ *L/min*, $\sigma_V = 70$ *L/min*.

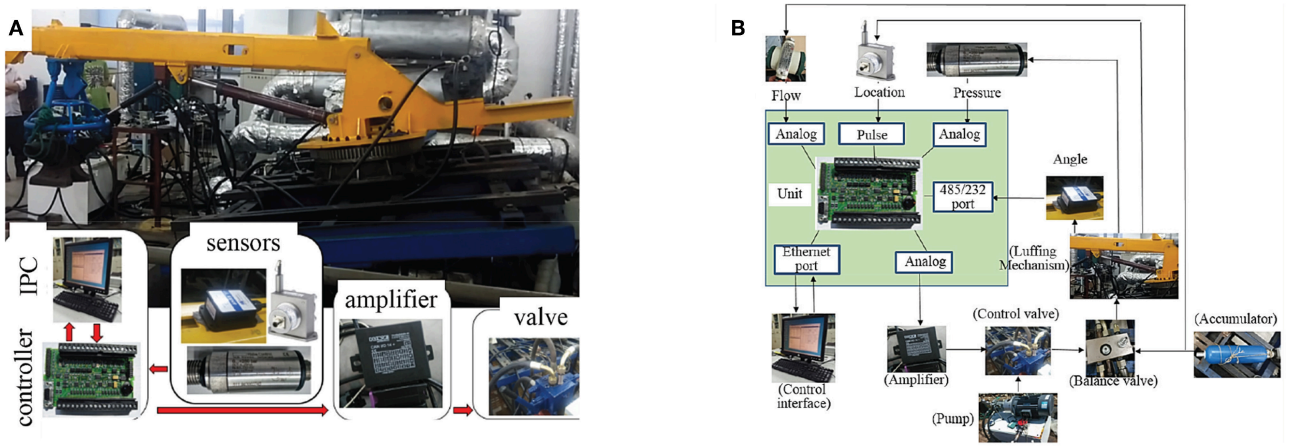


FIGURE 10 | Experimental equipment, **(A)** is the main equipment, **(B)** is the relationships between sensors and the control unit.

dynamics of the system and ultimately takes responsibility for the system.

5. EXPERIMENTAL VERIFICATION

5.1. Introduction of Experimental Equipment

The crane experimental model was placed on a swaying table to simulate the sea state. The relationship between each part of manipulators is shown in **Figure 10**.

The sensor and controller parameters are shown in **Table 2**.

5.2. Analysis of Step Signal Response

The test results are shown in **Figure 11** with the step extension signal.

The release process of the accumulator is shown in **Figure 11**. When the cylinder is extended, the accumulator releases the oil non-linearly. The cylinder protrudes quickly, and the extension time is about 0.25 s. The color block diagram on the right side of **Figure 11** shows that the energy release rate of the accumulator is significantly correlated to the rodless chamber pressure and the cylinder extension speed.

5.3. Experimental Strategy and Data Analysis

The swaying table was controlled with a sinusoidal signal in the frequency of 0.11Hz and an amplitude of $\pm 7^{\circ}$. The comparison experiment of automatic wave compensation was performed using a PD algorithm and PD+FNN to control the angular at 0°.

Figure 12 shows the wave compensation results of the conventional PD controller. During the wave compensation process, the cylinder pressure is stable, and the fluctuation range is approximately 90–105 Bar. However, the expansion and contraction movement of the cylinder is irregular due to the vicious cycle caused by the accumulator intervening. The performance of the cylinder control varies greatly, and the delay is severe. The wave compensation angle error is $\pm 3^{\circ}$. **Figure 13** shows that the accumulator flow output is irregular.

Figure 14 shows the experimental results of the PD+FNN controller. The optimization strategy effectively solves the speed shock caused by the accumulator. The cylinder control signal is consistent with the motion of the swinging table, which largely reduces the vibration of the cylinder. The wave compensation effect has been improved, and the angle compensation deviation is stable at $\pm 1^{\circ}$.

TABLE 2 | Parameters of components.

Name	Content	Parameters
Crane	Weight /*kg*	4.8×10^3
	Maximum system pressure /*Nm*	1.8×10^7
Luffing mechanism	Maximum operating range /*m*	1.5
	Maximum working velocity /$m{\cdot}s^{-1}$	1
	The pitching angle /°	$-20 \sim 58$
	Maximum lifting weight /*kg*	1.25×10^3
Accumulator	Model type	Bladder
	Volume /m^3	6.3×10^{-3}
	Pre-charge pressure /*Nm*	6×10^6
Cylinder	Bore-rod /*mm*	90–45
Pressure sensor	Range /*Nm*	$0 - 6 \times 10^8$
	Response time /*ms*	< 2
	Accuracy	0.3%
	Linearity	$\leq 0.5\%$
Flow sensor	Range /$m^3{\cdot}h^{-1}$	0.2–1.2
	Accuracy	±1% Range

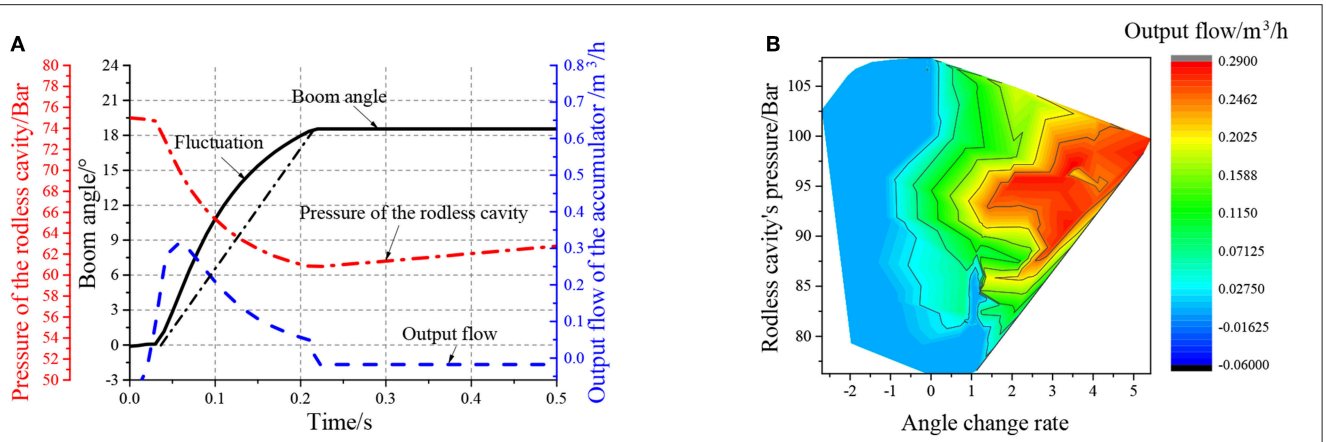


FIGURE 11 | Outstretch simulation of step signal, **(A)** is the fluctuation diagram, **(B)** is the output flow of the cylinder under different angle change rates.

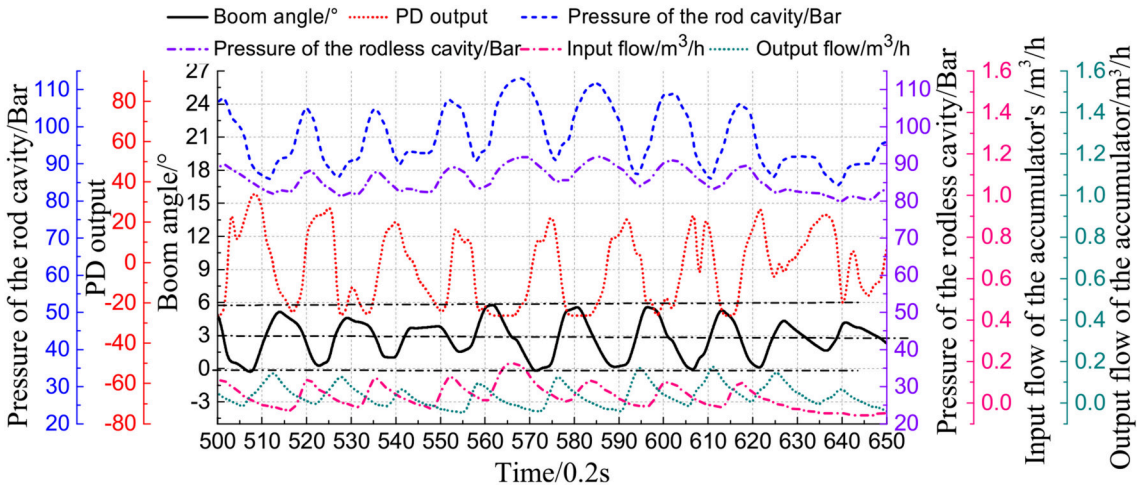


FIGURE 12 | Compensation test with PD strategy.

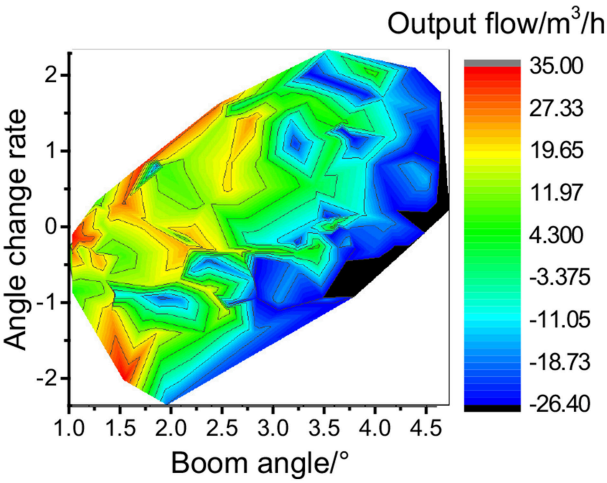


FIGURE 13 | Accumulator output flow with PD strategy.

Figure 15 shows the highest value of the accumulator flow output concentrated in the negative angle of the cylinder (cylinder extension).

6. CONCLUSION

In this work, a novel control strategy, i.e., PD+FNN approach, is designed to control a shipborne manipulator. Specifically, it is able to handle the high-order non-linearity, time-varying and hysteresis characteristics of the valve-controlled cylinder under the intervention of the accumulator. In addition, it can solve the overshoot generated by the wave compensation process when the accumulator releases energy and the cylinder reacts quickly in the extended stage. Moreover, the presented control strategy is capable of solving the problem of pressure fluctuation. The control precision is improved compared to using a conventional PD controller. Qualitative and quantitative tests on the simulation and real experiments have shown that the proposed controller is capable of significantly reducing

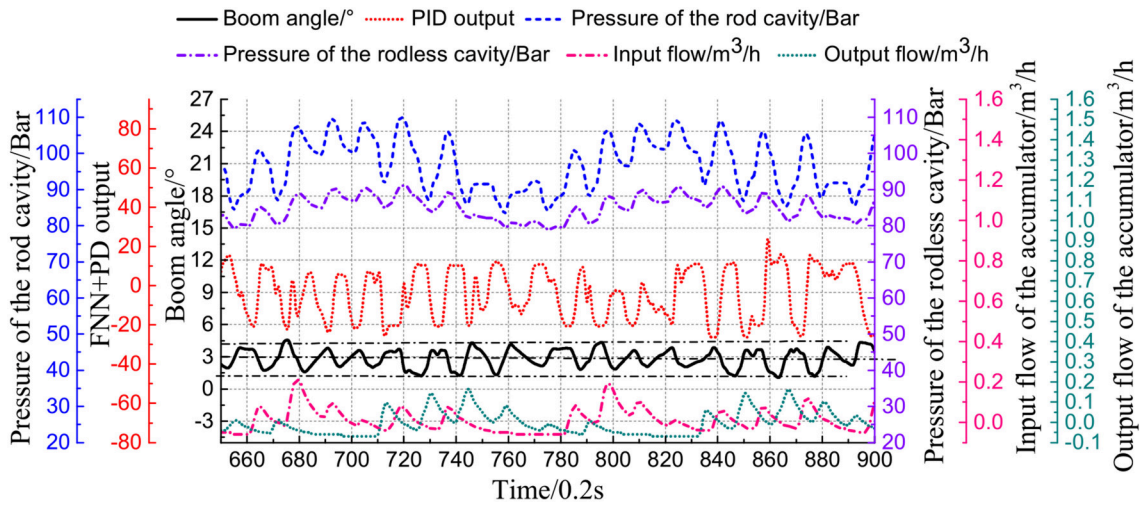


FIGURE 14 | Compensation test with PD+FNN strategy.

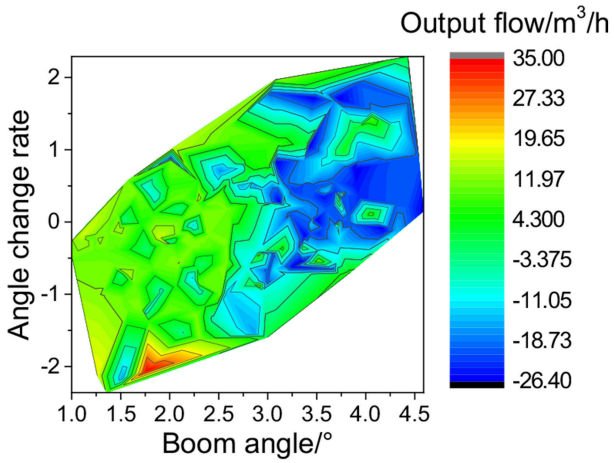


FIGURE 15 | Accumulator output flow with PD+FNN strategy.

steady state-errors and in overcoming the disturbances caused by the accumulator and uncertainties. The deviation angle of compensation is $\pm 1^\circ$ instead of $\pm 3^\circ$ compared to the conventional PD controller. We believe that the results of this work will motivate a wider use of the proposed PD+FNN approach, for autonomous collaboration of marine vehicles with a shipborne manipulator.

DATA AVAILABILITY

All datasets generated for this study are included in the manuscript and/or the supplementary files.

AUTHOR CONTRIBUTIONS

ZX and WL conceived and designed the experiments. YW performed the experiments. ZX and YW analyzed the data and wrote the paper.

REFERENCES

Alvarez-Ramirez, J., Kelly, R., and Cervantes, I. (2003). Semiglobal stability of saturated linear pid control for robot manipulators. *Automatica* 39, 989–995. doi: 10.1016/S0005-1098(03)00035-9

Biglarbegian, M., Melek, W. W., and Mendel, J. M. (2010). On the stability of interval type-2 tsk fuzzy logic control systems. *IEEE Trans. Syst. Man Cybern. B* 40, 798–818. doi: 10.1109/TSMCB.2009.2029986

Cervantes, I., and Alvarez-Ramirez, J. (2001). On the pid tracking control of robot manipulators. *Syst. Control Lett.* 42, 37–46. doi: 10.1016/S0167-6911(00)00077-3

Dabbaghjamanesh, M., Moeini, A., Ashkaboosi, M., Khazaei, P., and Mirzapalangi, K. (2016). High performance control of grid connected cascaded h-bridge active rectifier based on type ii-fuzzy logic controller with low frequency modulation technique. *Int. J. Electr. Comput. Eng.* 6, 484–494. doi: 10.11591/ijece.v6i2.9442

Fu, C., Olivares-Mendez, M. A., Campoy, P., and Suarez-Fernandez, R. (2013). "Uas see-and-avoid strategy using a fuzzy logic controller optimized by cross-entropy in scaling factors and membership functions," in *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, Atlanta, GA: 532–541.

Hasanien, H. M., and Matar, M. (2015). A fuzzy logic controller for autonomous operation of a voltage source converter-based distributed generation system. *IEEE Trans. Smart Grid* 6, 158–165. doi: 10.1109/TSG.2014.2338398

He, W., and Dong, Y. (2018). Adaptive fuzzy neural network control for a constrained robot using impedance learning. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 1174–1186.

Imanberdiyev, N., and Kayacan, E. (2018). A fast learning control strategy for unmanned aerial manipulators. *J. Intell. Robot. Syst.* 90, 1–20. doi: 10.1007/s10846-018-0884-7

Junke, H., and Zhen, C. (2017). "Research on lifting cylinder's pressure stability control method of active scraper," in *Mechanical, System and Control*

Engineering (ICMSC), 2017 International Conference on, Saint Petersburg, 250–254.

Kayacan, E., Kayacan, E., Ramon, H., and Saeys, W. (2013). Adaptive neuro-fuzzy control of a spherical rolling robot using sliding-mode-control-theory-based online learning algorithm. *IEEE Trans. Cybern.* 43, 170–179. doi: 10.1109/TSMCB.2012.220290

Kayacan, E., and Khanesar, M. A. (2015). *Fuzzy Neural Networks for Real Time Control Applications: Concepts, Modeling and Algorithms for Fast Learning* (Butterworth-Heinemann), Londan.

Khanesar, M. A., Kayacan, E., Reyhanoglu, M., and Kaynak, O. (2015). Feedback error learning control of magnetic satellites using type-2 fuzzy neural networks with elliptic membership functions. *IEEE Trans. Cybern.* 45, 858–868. doi: 10.1109/TCYB.2015.2388758

Khanesar, M. A., Kayacan, E., Teshnehlab, M., and Kaynak, O. (2011). "Levenberg marquardt algorithm for the training of type-2 fuzzy neuro systems with a novel type-2 fuzzy membership function," in *Advances in Type-2 Fuzzy Logic Systems (T2FUZZ), 2011 IEEE Symposium on*, Paris, 88–93.

Lin, F.-J., Hung, Y.-C., and Ruan, K.-C. (2014). An intelligent second-order sliding-mode control for an electric power steering system using a wavelet fuzzy neural network. *IEEE Trans. Fuzzy Syst.* 22, 1598–1611. doi: 10.1109/TFUZZ.2014.2300168

Lin, F.-J., Lu, K.-C., Ke, T.-H., Yang, B.-H., and Chang, Y.-R. (2015). Reactive power control of three-phase grid-connected pv system during grid faults using takagi–sugeno–kang probabilistic fuzzy neural network control. *IEEE Trans. Indus. Electron.* 62, 5516–5528. doi: 10.1109/TIE.2015.2407851

Precup, R.-E., Sabau, M.-C., and Petriu, E. M. (2015). Nature-inspired optimal tuning of input membership functions of takagi-sugeno-kang fuzzy models for anti-lock braking systems. *Appl. Soft Comput.* 27, 575–589. doi: 10.1016/j.asoc.2014.07.004

Sarabakha, A., Fu, C., Kayacan, E., and Kumbasar, T. (2018). Type-2 fuzzy logic controllers made even simpler: from design to deployment for UAVs. *IEEE Trans. Indus. Electron.* 65, 5069–5077. doi: 10.1109/TIE.2017.2767546

Shen, W., Jiang, J., Su, X., and Karimi, H. R. (2015). Control strategy analysis of the hydraulic hybrid excavator. *J. Franklin Inst.* 352, 541–561. doi: 10.1016/j.jfranklin.2014.04.007

Su, Y., Muller, P. C., and Zheng, C. (2010). Global asymptotic saturated pid control for robot manipulators. *IEEE Trans. Control Syst. Technol.* 18, 1280–1288. doi: 10.1109/TCST.2009.2035924

Tai, K., El-Sayed, A.-R., Biglarbegian, M., Gonzalez, C., Castillo, O., and Mahmud, S. (2016). Review of recent type-2 fuzzy controller applications. *Algorithms* 9, 39. doi: 10.3390/a9020039

Tang, J., Liu, F., Zou, Y., Zhang, W., and Wang, Y. (2017). An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Trans. Intell. Transport. Syst.* 18, 2340–2350. doi: 10.1109/TITS.2016.2643005

Vaidyanathan, S., and Azar, A. T. (2016). Takagi-sugeno fuzzy logic controller for liu-chen four-scroll chaotic system. *Int. J. Intell. Eng. Informatics* 4, 135–150. doi: 10.1504/IJIEI.2016.076699

Wai, R.-J., and Muthusamy, R. (2013). Fuzzy-neural-network inherited sliding-mode control for robot manipulator including actuator dynamics. *IEEE Trans. Neural Netw. Learn. Syst.* 24, 274–287. doi: 10.1109/TNNLS.2012.2228230

Xia, L., Quan, L., Ge, L., and Hao, Y. (2018). Energy efficiency analysis of integrated drive and energy recuperation system for hydraulic excavator boom. *Energy Convers. Manage.* 156, 680–687. doi: 10.1016/j.enconman.2017.11.074

Xiao, Y., Guan, C., and Lai, X. (2014). Research on the design and control strategy for a flow-coupling-based hydraulic hybrid excavator. *Proc. Inst. Mech. Eng. D J. Automobile Eng.* 228, 1675–1687. doi: 10.1177/0954407013502326

Yan, W., Hou, S., Fang, Y., and Fei, J. (2017). Robust adaptive nonsingular terminal sliding mode control of mems gyroscope using fuzzy-neural-network compensator. *Int. J. Mach. Learn. Cybern.* 8, 1287–1299. doi: 10.1007/s13042-016-0501-7

Yang, S.-Y., Ou, Y.-B., Guo, Y., and Wu, X.-M. (2017). Analysis and optimization of the working parameters of the impact mechanism of hydraulic rock drill based on a numerical simulation. *Int. J. Precis. Eng. Manufact.* 18, 971–977. doi: 10.1007/s12541-017-0114-4

Zhang, Q., Fang, J., Wei, J., Xiong, Y., and Wang, G. (2016). Adaptive robust motion control of a fast forging hydraulic press considering the nonlinear uncertain accumulator model. *Proc. Inst. Mech. Eng. I J. Syst. Control Eng.* 230, 483–497. doi: 10.1177/0959651816628994

Zhao, P.-Y., Chen, Y.-L., and Zhou, H. (2017). Simulation analysis of potential energy recovery system of hydraulic hybrid excavator. *Int. J. Precis. Eng. Manufactur.* 18, 1575–1589. doi: 10.1007/s12541-017-0187-0

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.

frontiers in Neurorobotics

METHODS
published: 16 April 2019
doi: 10.3389/fnbot.2019.00015

Mobile Robot Path Planning Based on Ant Colony Algorithm With A* Heuristic Method

Xiaolin Dai[1,2], *Shuai Long*[1], *Zhiwen Zhang*[1] *and Dawei Gong*[1,2]*

[1] School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, China, [2] Center of Robot, University of Electronic Science and Technology of China, Chengdu, China

This paper proposes an improved ant colony algorithm to achieve efficient searching capabilities of path planning in complicated maps for mobile robot. The improved ant colony algorithm uses the characteristics of A* algorithm and MAX-MIN Ant system. Firstly, the grid environment model is constructed. The evaluation function of A* algorithm and the bending suppression operator are introduced to improve the heuristic information of the Ant colony algorithm, which can accelerate the convergence speed and increase the smoothness of the global path. Secondly, the retraction mechanism is introduced to solve the deadlock problem. Then the MAX-MIN ant system is transformed into local diffusion pheromone and only the best solution from iteration trials can be added to pheromone update. And, strengths of the pheromone trails are effectively limited for avoiding premature convergence of search. This gives an effective improvement and high performance to ACO in complex tunnel, trough and baffle maps and gives a better result as compare to traditional versions of ACO. The simulation results show that the improved ant colony algorithm is more effective and faster.

Keywords: path planning, ant colony algorithm, A* algorithm, bending suppression, retraction mechanism

OPEN ACCESS

Edited by:
Caixia Cai,
*Agency for Science, Technology and Research (A*STAR), Singapore*

Reviewed by:
Wenchao Gao,
*Institute for Infocomm Research (A*STAR), Singapore*
Ran Duan,
Hong Kong Polytechnic University, Hong Kong
Bonan Huang,
Northeastern University, China

***Correspondence:**
Dawei Gong
pzhzhx@126.com

Received: *20 December 2018*
Accepted: *25 March 2019*
Published: *16 April 2019*

Citation:
Dai X, Long S, Zhang Z and Gong D (2019) Mobile Robot Path Planning Based on Ant Colony Algorithm With A Heuristic Method. Front. Neurorobot. 13:15. doi: 10.3389/fnbot.2019.00015*

INTRODUCTION

Path planning is a key issue in the field of mobile robot research. Its main purpose is to find an optimal or suboptimal, safe and collision-free path from the starting point to the target point in the environment with obstacle (Cheng et al., 2010; Deepak et al., 2012; Zhou et al., 2013). According to the degree of intelligence in the process of path planning, mobile robot path planning can be divided into traditional path planning and intelligent path planning. The traditional path planning algorithm includes simulated annealing algorithm (Miao and Tian, 2013), potential function theory (Cetin and Yilmaz, 2014; Nair et al., 2015), fuzzy logic algorithm (Li et al., 2013; Jiang and Li, 2014; Bakdi et al., 2016) and so on. However, these traditional methods can't be further improved in path search efficiency and path optimization. Intelligent path planning algorithm includes Ant Colony Optimization (ACO) (Jovanovic et al., 2016; Wang et al., 2016), genetic algorithm (Arantes et al., 2017; Lin et al., 2017), neural network (He et al., 2016a, 2017a,b) and particle swarm algorithm (Das et al., 2016; Song et al., 2016) and so on. The ant colony algorithm has the advantages of strong robustness, good global optimization ability and inherent parallelism. Moreover, it easily combines with multiple heuristic algorithms to improve the performance of algorithms. So it is widely used in path planning.

However, due to the randomness of probabilistic transfer and the inappropriateness of pheromone intensity update, the traditional ACO will easily fall into the local optimum and tend to poor convergence. To this end, many scholars delivered a variety of improved methods to solve problems regarding pheromone update and path search strategy (Stützle and Hoos, 2000; Zeng et al., 2016; Zhao et al., 2016; Zhang et al., 2017). In Stützle and Hoos (2000), an Ant Colony System (ACS) algorithm was proposed to speed up the convergence rate of ACO by updating pheromones on the path of the optimal ant of each generation. In Zhao et al. (2016), by adaptively changing the volatilization rate and adjusting the pheromone updating formula, the search ability of the ant colony and the convergence rate of the algorithm were improved. In Zhao et al. (2016), some intelligent algorithm was proposed to generate an initial path, which can be transformed into the initial pheromone distribution to avoid blind search of ant colony. In Zhang et al. (2017), the path information (such as the crowded path and the steep path weight) was added into the initial pheromone matrix, which could affects the efficiency of the algorithm. In Zhao et al. (2016), the heuristic function was adjusted to improve the convergence rate of the algorithm according to the target point. In Zeng et al. (2016), it unlimited step length of finding optimal path so that the improved ACO could find a shorter path and its convergence was better. In addition, many scholars have combined the ant colony algorithm with other (intelligent) algorithms (He et al., 2016b; Liu et al., 2016; Yen and Cheng, 2016; He and Zhang, 2017) to improve the convergence rate and the smooth of path. In Liu et al. (2016), the geometric method was used to optimize path. Also in Liu et al. (2016), the force factor in the artificial potential field method is transformed into local diffusion pheromone to improve the ability of the ant colony algorithm to find the obstacle. In Yen and Cheng (2016), the fuzzy ant colony optimization method was proposed to minimize the iterative learning error.

In this paper, an effective version of ant colony algorithm is achieved. It utilizes the evaluation function of A* algorithm to improve the heuristic information of Ant colony algorithm, which accelerates the convergence speed during the search. And MAX–MIN Ant System is used to make the global search ability better by updating the path pheromone of the optimal network. At the same time, the bending suppression operator is introduced to improve heuristic information, which aims to optimize the smoothness of the path. The problem of deadlock is solved by using the retraction mechanism. All these procedures not only give an effective improvement and better performance to ACO, but also give the best results as compare to traditional versions of the algorithm (Zhao et al., 2016) and ACO in complex tunnel, trough and baffle maps. The simulation results show that the proposed algorithm is effective and fast.

MATERIALS AND METHODS

Environment Model

The work environment is built by using the grid model, which divides the robot working space into *N*N squares*. As shown in **Figure 1**, the gray grids are represented as obstacles (the grid with barriers) and the white grids are represented as free grid squares (the robot can move). In order to identify obstacles, the white grid cell is represented by 0 and the gray grid unit is represented by 1. The grid method is simple and effective to create and maintain grid model. Moreover, the grid method have strong adaptability for obstacle. This method is convenient for computer storage and processing.

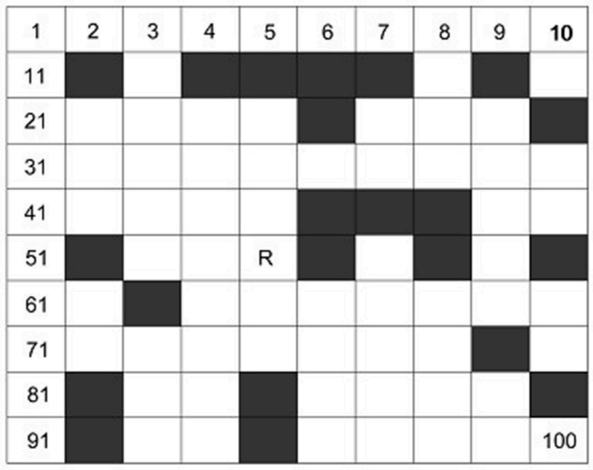


FIGURE 1 | Environment model.

The grid model was placed into two-dimensional coordinate system. And then serial number method is adopted to mark each grid. In *N*N* grid map, the starting node is named after *Start* and the target node is named after *Goal*. The position coordinates (x, y) corresponding to any grid whose grid number is *R* as follow:

$$\begin{cases} x = \begin{cases} mod\,(R,N) - 0.5 & if\ mod\,(R,N)! = 0 \\ N + mod\,(R,N) - 0.5 & otherwise \end{cases} \\ y = N + 0.5 - ceil\left(\frac{R}{N}\right) \end{cases} \tag{1}$$

Where *mod* is the surplus operation, *ceil* rounds the elements to the nearest integers toward infinity.

Ant Colony Algorithm

Heuristic Strategy With Direction Information

In the traditional ACO, the probability of the next node is selected by roulette wheel method as follows:

$$\begin{aligned} P_{ij}^{k}(t) &= \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}\cdot(\eta_{ij}(t))^{\beta}}{\sum_{s\in allow_k}(\tau_{is}(t))^{\alpha}\cdot(\eta_{is}(t))^{\beta}} & s \in allow_k \\ 0 & s \notin allow_k \end{cases} \\ \eta_{ij}(t) &= \frac{1}{d_{ij}} \\ d_{ij} &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \end{aligned} \tag{2}$$

Where τ_{ij} is the pheromone trail of the path grid *i* to grid *j*, and η_{ij} is the heuristic information of the path grid *i* to grid *j*. α *is* the stimulating factor of pheromone concentration which determine the relative influence of the pheromone trail. β is the stimulating

factor of visibility which determine the relative influence of the heuristic information. d_{ij} is the distance between node i and node j. (x_i, y_i) and (x_j, y_j) is the coordinates of grid i and grid j. $allow_k$ is the collection of grids which ants can choose when ants in the grid i (in other words, they are the grids around the grid i except the obstacle grid and taboo grid).

Coverage and Updating Strategy

According to the traditional ACO, the next node is decided by the roulette wheel method and it is repeated until the target point is obtained. After each iteration is completed, pheromone trails are updated in line with the length of path planning. For each trial during pheromone update, all imperfect pheromones evaporates and only the best pheromones are updated to trials history, because it enables ants to neglect all substandard pheromone trails and improve its coverage efficiency to find a shorter path. Formula (3) is used to update the pheromone quantity on each vertex at the end of each cycle:

$$\begin{cases} \tau_{ij} &= (1-\rho)\,\tau_{ij} + \Delta\tau_{ij} \\ \Delta\tau_{ij} &= \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \end{cases} \quad , \; 0 < \rho < 1 \tag{3}$$

where m is the number of ants. ρ is pheromone evaporation rate. $\Delta\tau_{ij}^{k}$ represents the value of pheromone that the ant k leaves in the path of grid i to grid j. This article uses the ant-cycle-system model, and $\Delta\tau_{ij}^{k}$ is defined as follows:

$$\Delta\tau_{ij}^{k}(k) = \begin{cases} Q_1/L_k(t) & \textit{if arc}\,(i,j)\ \textit{is used by k in iteration t} \\ 0 & \textit{otherwise} \end{cases} \tag{4}$$

Where Q_1 is a constant. $L_K(t)$ is the length of the path that the ant k is looking for.

Improved Ant Colony Algorithm

The traditional ACO has the following shortcomings: Due to the lack of initial pheromone and the unapparent difference of the heuristic value between adjacent grids, it usually requires a longer search time, which leads to the slow convergence rate. When grid model is complex, the robot maybe fall into a deadlock state in which the robot cannot move to the surrounding grids. In the grid map, the path planning with traditional ACO may have more bending times and big cumulative bending angle. To solve the above problems, this paper makes the following improvements.

Heuristic Information Based on A* Algorithm

A* algorithm (Duchon et al., 2014) is the most effective direct search method for solving the shortest path in static road network. It is developed on the basis of Dijkstra algorithm, which can avoid blind search to improve search efficiency. In this paper, A* algorithm is used as the heuristic information of path searching to improve the convergence speed of the algorithm and obtain the better path. The bending suppression operator is added to the heuristic information to reduce bending times and cumulative bending angle.

The heuristic cost of A* algorithm is expressed by the estimated function, and the estimated function equation $f(n)$ is as follows:

$$\begin{aligned} f(n) &= g(n) + h(n) \\ h(n) &= \left((n_x - g_x)^2 + (n_y - g_y)^2\right)^{1/2} \\ g(n) &= \left((n_x - s_x)^2 + (n_y - s_y)^2\right)^{1/2} \end{aligned} \tag{5}$$

where $g(n)$ is the minimum cost from the source node to the current node. $h(n)$ is the minimum cost from the current node to the destination node. n_x and n_y are the coordinates of the current node n. g_x and g_y are the coordinates of the target node g, s_x, and s_y are the coordinates of the initial node s.

The estimated function of A* algorithm is used as heuristic information to search for global optimal path in ant colony algorithm, and the bending suppression operator is added to the heuristic value of ant colony algorithm to reduce the number of bending times and the large cumulative turning angle. The improved heuristic information formula is as follows:

$$\begin{aligned} \eta_{ij}(t) &= \frac{Q_2}{h(n) + g(n) + cost(bend)} \\ cost\,(bend) &= \varphi \cdot turn + \psi \cdot thita \end{aligned} \tag{6}$$

where Q_2 is a constant more than 1. $cost(bend)$ is a bending suppression operator. $turn$ is the number of turns from node $n-1$(previous node) to node $n+1$ (next node). $thita$ is the angle between the line segment of node $n-1$ to node n (current node) and the line segment of node n to node $n+1$. φ is the coefficient converting turning times into grid length. ψ is the coefficient converting angle into grid length.

Solve the Deadlock Problem

When the robot environment is more complex (especially the ants go into the environment of concave obstacles), due to the presence of the taboo table, the ants may fall into a deadlock state without the next grid to move. As shown in **Figure 2**, when the ant travels from the grid T to the grid S, the next optional grid is C. At this time, the ant is trapped in a deadlock state and it cannot move to its surrounding grid.

For the deadlock problem, Wang and Yu (2011) adopted the early death strategy, which deleted the ants trapped in a deadlock state from the ant colony and did not update the global pheromone. However, when more of the ants are trapped in the deadlock state, the number of ants that can reach the goal is significantly reduced, which results in a decrease in the diversity of solutions and is not conducive to the search of optimal path for ants. In this paper, the improvement measure is that the ants adopt retraction mechanism when they fall into the deadlock state. As shown in **Figure 2**, the ant, which has walked into the grid, is trapped in the deadlock state, and the improved strategy allows the ant to roll back one step and updates the taboo table information. If the ant is still trapped into a deadlock state, the ant will continue to rollback untill grid T. At this moment, the ant escapes the deadlock area. Since the deadlock

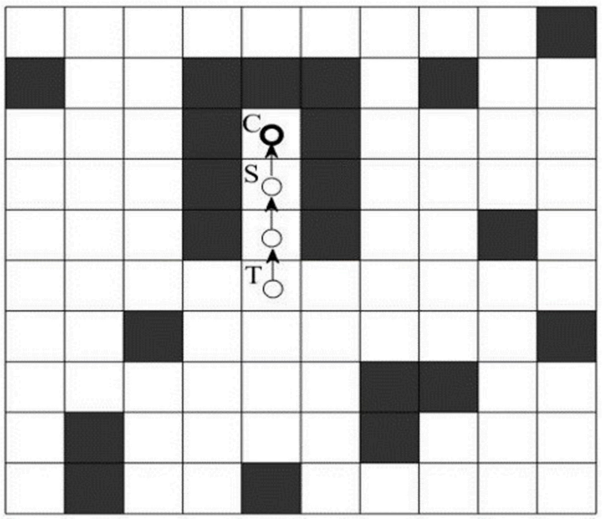


FIGURE 2 | Deadlock state diagram.

edge may be the part of global optimal path, no pheromone punishment is carried out on the deadlock edge. The retraction mechanism cannot prevent ants from entering a deadlock state, but it lets the deadlocked ants return back to the previous grid until there is a feasible grid around the ants, so the ACO with the retraction mechanism has higher efficiency and fewer iterations. The ACO with the retraction mechanism and without the retraction mechanism is compared in section The Retraction Mechanism Results Analysis below.

Max–Min Ant System

As the traditional ant colony algorithm may cause premature convergence and precocious phenomenon, it needs to improve algorithm to solve these problem. The MAX-MIN Ant System (MMAS) (Stützle and Hoos, 2000) can solve these problems well.

(1) Pheromone trail updating. After each iteration trial, the pheromone is submitted into update history in traditional ant colony algorithm. While in the MMAS, only the path pheromone of the optimal network is updated after the iteration is completed. Accordingly, the modified pheromone trail update rule is stated by:

$$\begin{aligned} \tau_{ij}(t+1) &= (1-\rho)\,\tau_{ij}(t) + \Delta\tau_{ij}^{best} \\ \Delta\tau_{ij}^{k}(t) &= \frac{Q_1}{L^{best}} + \frac{Q_3}{C_{turn}^{best}} \\ C_{turn}^{best} &= \omega_1 Cals\left(l\right) + \omega_2 Turns\left(l\right) \\ \omega_1 &= \frac{V_{robot}}{W_{robot}} \\ \omega_2 &= V_{robot} \times t_a \end{aligned} \tag{7}$$

where Q_3 is a constant more than 1.L^{best} denotes to the shortest path currently found by the algorithm. *Cals*(*l*) represents the sum of all the angles of turning on the best optimized path. *Turns*(*l*) is the sum of the turns on the best optimized path. w_1 and w_2 represent different weight coefficient and are set by analyzing the robot's structure and kinematics (Wu et al., 2013; Li et al., 2017). The w_1 and w_2 can convert turning angle and turning times into grid length, respectively. V_{robot} represents the constant speed of a mobile robot. W_{robot} represents the angular speed of a mobile robot as it turns. t_a represents the time of acceleration and deceleration as the mobile robot turns once.

(2) Pheromone trail limits. In order to avoid the situation that the traditional ant colony algorithm may falls into local optimal solution and loses the further search space ability by pheromone accumulation, the pheromone trail of the MMAS is limited in the upper limits and lower limits $\left[T\mathrm{au}_{min,}T\mathrm{au}_{max,}\right]$. The formula is:

$$\mathrm{Tau} = \begin{cases} \mathrm{Tau}_{min}, & \mathrm{Tau} \le \mathrm{Tau}_{min} \\ \mathrm{Tau}, & \mathrm{Tau}_{min} < \mathrm{Tau} \le \mathrm{Tau}_{max} \\ \mathrm{Tau}_{max}, & \mathrm{Tau} > \mathrm{Tau}_{min} \end{cases} \tag{8}$$

Aco Procedure

To sum up, specific steps of mobile robot path planning based on the improved ant colony algorithm are as follows:

Step 1: The working environment is modeled by the grid method, and the starting point *start* and the target point *goal* of the mobile robot are given.

Step 2: Initialize the ant system. Set the number m of ants, parameter α which determines the relative influence of the pheromone trail, parameter β which determines the heuristic value, the global pheromone volatilization coefficient ρ, pheromone intensity Q_1 and other related parameters.

Step 3: Update taboo table. Place the ant k ($k = 1, 2, \cdots, m$) on the current node and add the current node to the corresponding taboo table.

Step 4: Process deadlock. According to the taboo table, it will judge whether ants are trapped in a deadlock state. If the ants are in a deadlock state, the retraction mechanism will be adopted and the deadlock node will be added to the taboo table. Conversely, it will judge whether the ants reach the target point. If the ants reach the target point, it will turn to Step 6, otherwise it will turn to Step 5.

Step 5: Select the next grid. It will calculate the heuristic function according to formula (6), and calculate the probability function according to formula (2). Finally, it will use the roulette method to select the next feasible grid. If the ants reach the target grid, it will turn to Step 6, otherwise it will turn to Step 3.

Step 6: If the ants reach the target node, it will repeat Step 3 until each ant completes the search target during its iteration process and then turn to Step 7.

Step 7: Update pheromone. After each iteration, if the number of iterations satisfies inequality $N \le N_{\max}$, it will update the path pheromone and determine whether it meets the convergence conditions. If it meets the convergence conditions, it will withdraw. If it does not meet, it will turn to Step 3. If the number of iterations satisfies inequality $N > N_{\max}$, it will be not counted further. The final result is output as long as the end condition is satisfied.

The implementation process of improved ant colony algorithm is as in **Table 1**.

RESULTS

In order to verify the effectiveness of the improved ant colony algorithm, this paper uses MATLAB software to simulate. It is more convincing to use comparative method to carry out experiments under the same experimental conditions. In the simulation, the main parameter values of the ACO should be determined firstly. The main parameters include number of ants, stimulating factor of pheromone concentration, stimulating factor of visibility and pheromone evaporation coefficient. Through parameter analysis method (Wu et al., 2010), the relationship between each parameter and simulation results (path length, number of iterations) can be obtained. According to the relationship between each parameter and simulation results (Shi et al., 2014), we can get the value of the main parameters in the ACO. In the simulation, value of each parameter in the ACO is as in **Table 2**:

Comparative Analysis of Path Planning Algorithms

The experiment was divided into four parts according to four types of maps(the common map, the tunnel map, the trough map and the baffle map) and three algorithms (the traditional ant colony algorithm, the algorithm (Zhao et al., 2016) and the improved ant colony algorithm proposed in this paper) are simulated on each map in turn**.** The convergence speed, shortest path length and bending suppression effect of those algorithms are compared.

(1) *Example 1.* In this example, the environment of the robot was built into the 20*20 grid model and the three algorithms are tested on the common map. The coordinates of grid *Start* and grid *Goal* is (0.5, 19.5) and (19.5, 0.5) (shown in **Figure 3**), respectively.
(2) *Example 2.* In this example, the environment of the robot was built into the 30*30 grid model and the three algorithms are tested on the tunnel map. The coordinates of grid *Start* and grid *Goal* is (0.5, 8.5) and (15.5, 18.5) (shown in **Figure 4**), respectively.
(3) *Example 3.* In this example, the environment of the robot was built into the 40*40 grid model and the three algorithms are tested on the trough map. The coordinates of grid *Start* and grid *Goal* is (5.5, 34.5) and (28.5, 5.5) (shown in **Figure 5**), respectively.
(4) *Example 4.* In this example, the environment of the robot was built into the 20*20 grid model and the three algorithms are tested on the baffle map. The coordinates of grid *Start* and grid *Goal* is (0.5, 14.5) and (14.5, 14.5) (shown in **Figure 6**), respectively.

As shown in **Figure 3**, the optimized path length of the improved ant colony algorithm is 29.2133 and the number of bending times is 6. The improved ant colony algorithm is basically as same as the path planning effect of the ant colony algorithm (Zhao et al., 2016) on the path length, but it is 25% lower on bending times than the ant colony algorithm (Zhao et al., 2016). Compared with the traditional ant colony algorithm, it is 73% reduction in the number of bending times.

As shown in **Figure 4**, the optimized length of the improved ant colony algorithm is 37.3849, and the number of bending times is 7. In the shortest path length, the improved ant colony algorithm is basically as same as the algorithm (Zhao et al., 2016). In the number of bending times, it is 50% decrease than the traditional ant colony algorithm and is 22% decrease than the algorithm (Zhao et al., 2016).

As shown in **Figure 5**, the optimized path length of the improved ant colony algorithm is 51.1128. In **Figure 6**, the optimized path length of the improved ant colony algorithm is 50.7280. But in **Figures 5**, **6**, both the traditional ant colony algorithm and the algorithm (Zhao et al., 2016) can't search the global optimized path. Even as the scale of the problem expands and the environment map becomes more and more complex, the improved algorithm can still perform very well.

The results of the three algorithms that run 100 times in same map environments are shown in **Table 3**. Compared with the traditional ant colony algorithm and the algorithm (Zhao et al., 2016), the improved algorithm has a good performance on the efficiency. At the same time, it has a good adaptability in a complicated area. The improved algorithm proposed in this paper can be used not only in the path planning of mobile robots, but also in the path planning of robot manipulators (Yang et al., 2017, 2018).

TABLE 1 | Description of ACO algorithm for solving path planning.

Algorithm A*MMAS
Begin
create grid environment
initialize the ant colony system
Repeat
  for each ant k **do**
    if grid $i \in allow_k$ **then**
      if grid $i \in taboo_{deadlock}$ **then**
        fallback
      end if
  according to formula (2) and (6) select next grid j
    Update taboo
  end if
Update pheromone on each iteration by improved MMAS method according to formula (7) and (8)
Until algorithm convergence
Return best grid serial number
END

TABLE 2 | Values of the main parameters in the ACO.

Number of ants m	Stimulating factor of pheromone concentration α	Stimulating factor of visibility β	Pheromone evaporation coefficient ρ	Pheromone intensity Q
50	1	5	0.5	10

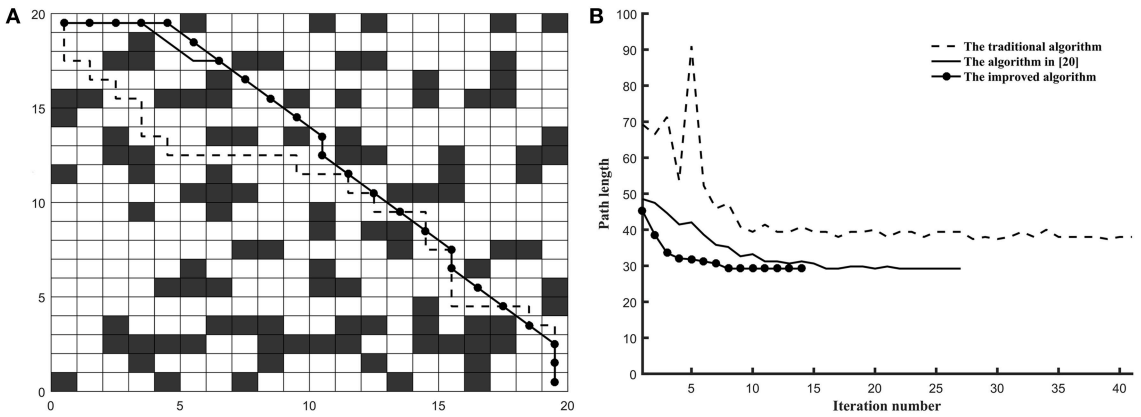


FIGURE 3 | The test results of three algorithms run on common map. **(A)** Simulation results in 20*20 grid. **(B)** Convergence curve.

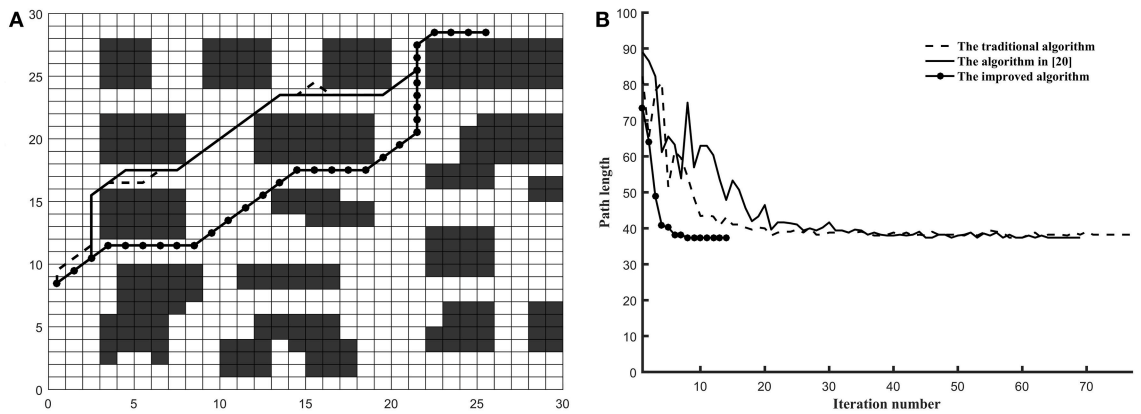


FIGURE 4 | The test results of three algorithms run on tunnel map. **(A)** Simulation results in 30*30 grid. **(B)** Convergence curve.

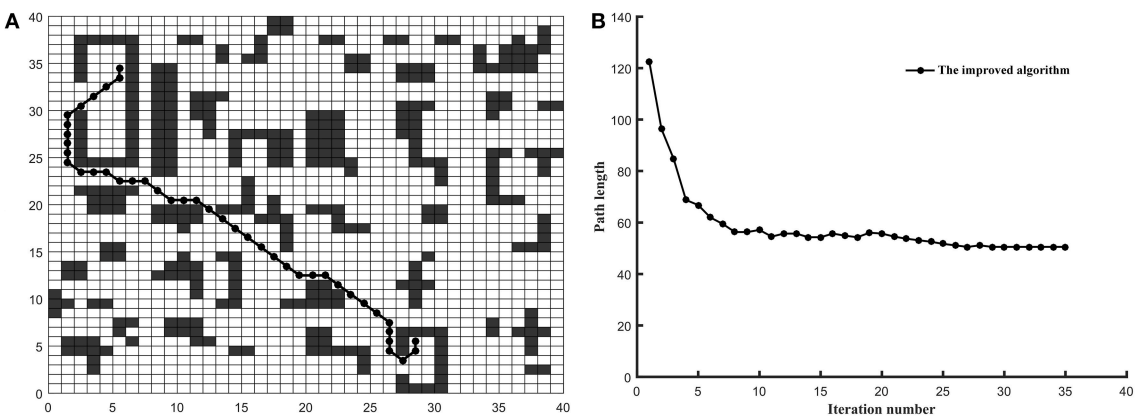


FIGURE 5 | The test results of three algorithms run on trough map. **(A)** Simulation results in 40*40 grid. **(B)** Convergence curve. (Other two algorithms is failed in trough map).

The Retraction Mechanism Results Analysis

In order to show the function of retraction mechanism, the ACO with the retraction mechanism and ACO without the retraction mechanism are tested on the trough map and the baffle map, respectively.

As shown in **Figures 7A**, **8A**, ACO with the retraction mechanism has higher efficiency and fewer iteration than ACO

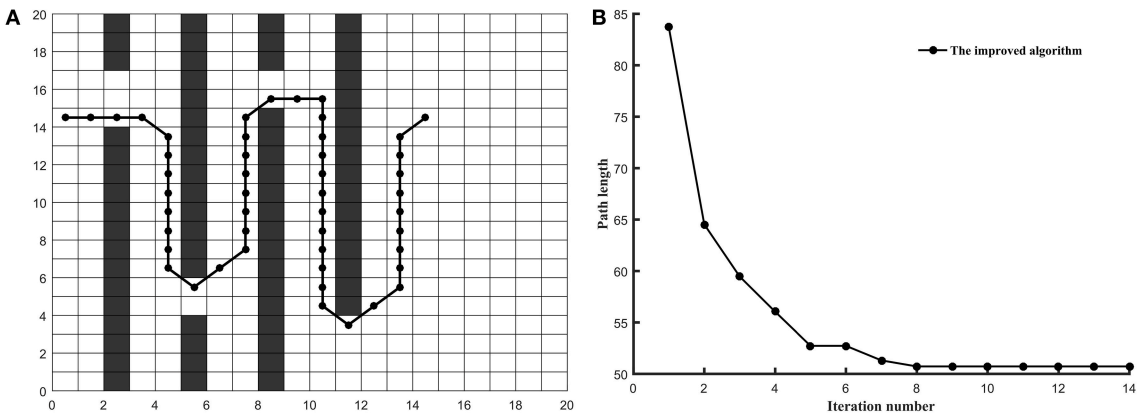


FIGURE 6 | The test results of three algorithms run on baffle map. **(A)** Simulation results in 20*20 grid. **(B)** Convergence curve. (Other two algorithms is failed in trough map).

TABLE 3 | Test results for three algorithms under different maps.

Map	Algorithm	Optimal solution of the algorithm	The average of the shortest distance	Average iteration times	Average time-consuming(sec)	Number of bends
Common map	①	37.4143	38.6335	40	9.22	22
	②	29.2133	29.4506	33	7.26	10
	③	29.2133	29.3807	12	4.89	10
Tunnel map	①	38.2133	38.6325	47	26.92	17
	②	37.3849	38.4813	35	20.62	12
	③	37.3849	38.1262	16	17.97	10
Trough map	①	–	–	–	–	–
	②	–	–	–	–	–
	③	51.1128	51.8471	40	88.20	13
Baffle map	①	–	–	–	–	–
	②	–	–	–	–	–
	③	50.7280	51.0605	15	8.40	13

①: The traditional ant colony algorithm.
②: The algorithm [20]. ③: the improved ant colony algorithm.

without the retraction mechanism. When ants fall into deadlock state, the retraction mechanism is used to replace the early death strategy, which avoids a large number of ant deaths in one iteration. Therefore, each ant can obtain a path by using the retraction mechanism, which increases the diversity of results and is beneficial to find the optimal path. As shown in **Figures 7B**, **8B**, the number of ant retracted in the initial stage of the algorithm is higher than in the middle and later stage of the algorithm and the retraction mechanism can effectively suppress the decline of the number of ants.

DISCUSSION

This paper makes a valuable contribution to the improvement of ant colony algorithm in complicated maps for the mobile robot, especially the improvement on convergence speed, shortest path length and bending suppression effect. The estimated function of improved A* algorithm is used as the heuristic function to improve search efficiency and smoothness of path. By employing the retraction mechanism and the improved MAX–MIN Ant System method, the problem of ant deadlock is solved and the global search ability of the algorithm is improved.

Three algorithms are researched on path planning in the common map, tunnel map, trough map and baffle map, respectively. Compared with the traditional ant colony algorithm and the algorithm (Zhao et al., 2016), the improved ant colony algorithm is better in the convergence rate and the bending suppression effect. Compared with the traditional ant colony algorithm, the improved ant colony algorithm has more than 65% reduction in number of iterations and 41% decrease in bending suppression. In addition, the improved ant colony algorithm is 54% lower than the algorithm (Zhao et al., 2016) in number of iterations. To sum up, this paper proves the effectiveness, rapidity and adaptability of the improved ant colony algorithm in the complex map environment.

AUTHOR CONTRIBUTIONS

XD and DG proposed the innovation and designed the experiment in this study. ZZ and SL performed the simulation experiment and analyzed the experiment results. XD checked the

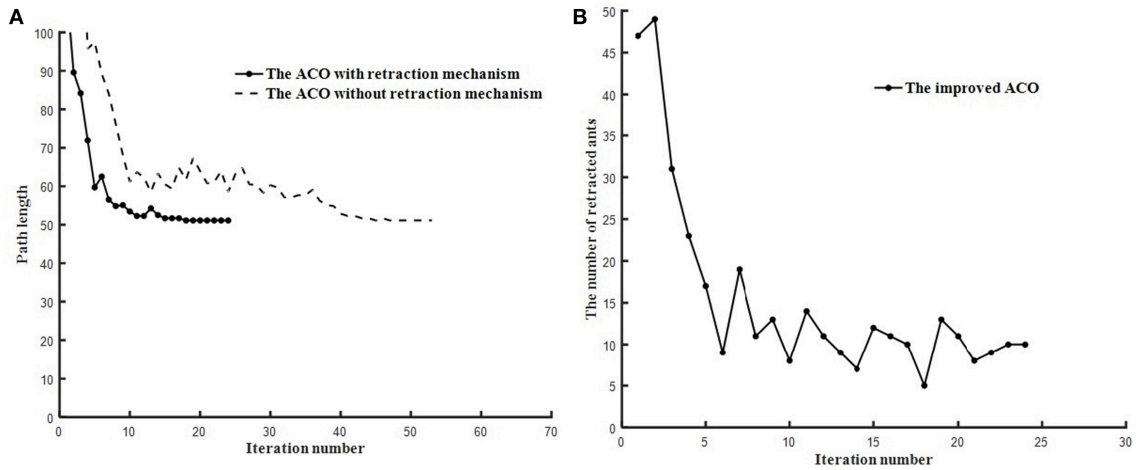


FIGURE 7 | The test results of two algorithms run on trough map. **(A)** Path planning comparison. **(B)** Ant retraction number curve.

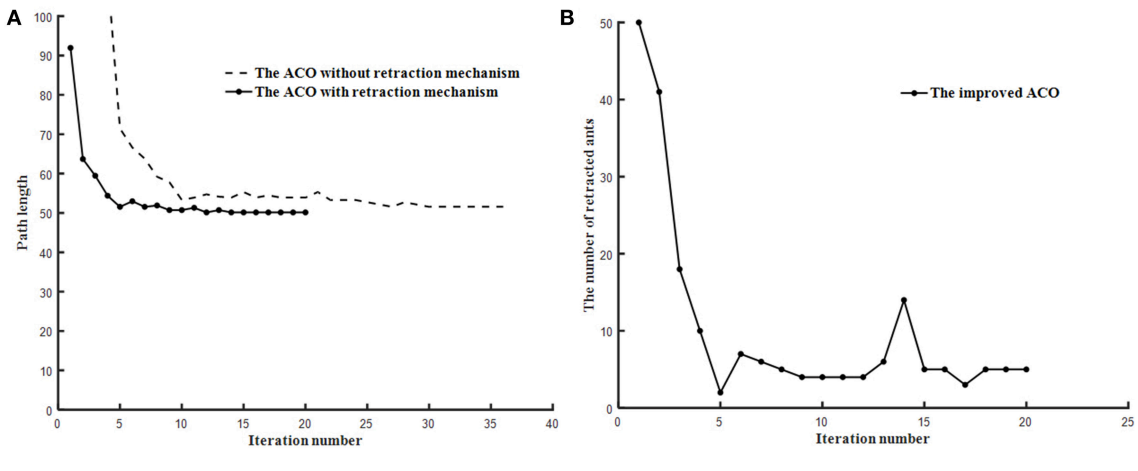


FIGURE 8 | The test results of two algorithms run on baffle map. **(A)** Path planning comparison. **(B)** Ant retraction number curve.

results. SL wrote the manuscript and DG provided writing advice of manuscript.

FUNDING

This work was supported by the National Natural Science Foundation of China (61603076) (51305066), Fundamental Research Funds for the Central Universities (ZYGX2016J116), and Science and Technology Support Program of Sichuan Province (2016GZ0198).

ACKNOWLEDGMENTS

We acknowledge the support of the School of Mechatronics Engineering and Center of Robot in University of Electronic Science and Technology of China.

REFERENCES

Arantes, J. D., Arantes, M. D., Toledo, C. F., Júnior, O. T., and Williams, B. C. (2017). Heuristic and genetic algorithm approaches for UAV path planning under critical situation. *Int. J. Artificial Intelligence Tools* 26, 1–30. doi: 10.1142/S0218213017600089

Bakdi, A., Hentout, A., Boutami, H., Maoudj, A., Hachour, O., and Bouzouia, B. (2016). Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control. *Robot. Autonomous Syst.* 89, 95–109. doi: 10.1016/j.robot.2016.12.008

Cetin, O., and Yilmaz, G. (2014). Sigmoid limiting functions and potential field based autonomous air refueling path planning for UAVs. *J. Intelligent Robot. Syst.* 73, 797–810. doi: 10.1007/s10846-013-9902-y

Cheng, C. T., Fallahi, K., Leung, H., and Tse, C. K. (2010). An AUVs path planner using genetic algorithms with a deterministic crossover operator. *IEEE Int. Conference Robot. Automat.* 2010, 2995–3000. doi: 10.1109/ROBOT.2010.5509335

Das, P. K., Behera, H. S., and Panigrahi, B. K. (2016). A hybridization of an improved particle swarm optimization and gravitational search

algorithm for multi-robot path planning. *Swarm Evol. Comput.* 28, 14-28. doi: 10.1016/j.swevo.2015.10.011

Deepak, B. B. V. L., Parhi, D. R., and Kundu, S. (2012). Innate immune based path planner of an autonomous mobile robot. *Procedia Eng.* 38, 2663–2671. doi: 10.1016/j.proeng.2012.06.313

Duchon, F., Babinec, A., Kajan, M., Beno, P., Florek, M., Fico, T., et al. (2014). Path planning with modified A star algorithm for a mobile robot. *Procedia Eng.* 96, 59-69. doi: 10.1016/j.proeng.2014.12.098

He, W., Chen, Y., and Yin, Z. (2016a). Adaptive neural network control of an uncertain robot with full-state constraints. *IEEE Trans. Cybernet.* 46, 620–629. doi: 10.1109/TCYB.2015.2411285

He, W., Dong, Y., and Sun, C. (2016b). Adaptive neural impedance control of a robotic manipulator with input saturation. *IEEE Trans. Syst. Man Cybernet. Syst.* 46, 334–344. doi: 10.1109/TSMC.2015.2429555

He, W., Yan, Z., Sun, C., and Chen, Y. (2017a). Adaptive neural network control of a flapping wing micro aerial vehicle with disturbance observer. *IEEE Trans. Cybernet.* 47, 3452–3465. doi: 10.1109/TCYB.2017.2720801

He, W., Yin, Z., and Sun, C. (2017b). Adaptive neural network control of a marine vessel with constraints using the asymmetric barrier lyapunov function. *IEEE Trans. Cybernet.* 47, 1641–1651. doi: 10.1109/TCYB.2016.2554621

He, W., and Zhang, S. (2017). Control design for nonlinear flexible wings of a robotic aircraft. *IEEE Trans. Control Syst. Technol.* 25, 351–357. doi: 10.1109/TCST.2016.2536708

Jiang, K., and Li, C. G. (2014). Path planning based on fuzzy logic algorithm for robots in hierarchical control. *Appl. Mech. Mater.* 644–650,701–704. doi: 10.4028/www.scientific.net/amm.644-650.701-704

Jovanovic, R., Tuba, M., and Vo,ß, S. (2016). An ant colony optimization algorithm for partitioning graphs with supply and demand. *Appl. Soft Comput.* 41, 317–330. doi: 10.1016/j.asoc.2016.01.013

Li, Q., Zhang, C., Han, C. W., Zhang, T., and Zhang, W. (2013). Path planning based fuzzy logic algorithm for mobile robots in dynamic environments. *J. Central South Univ. Sci. Technol.* 2013, 105–107. doi: 10.1109/CCDC.2013.6561434

Li, W. H., Yang, C. G., Jiang, Y. M., and Su, C. Y. (2017). Motion planning for omnidirectional wheeled mobile robot by potential field method. *J. Adv. Transport.* 3, 1–11. doi: 10.1155/2017/4961383

Lin, D., Shen, B., Liu, Y., Alsaadi, F. E., Alsaedi, A., and Cheng, H. (2017). Genetic algorithm-based compliant robot path planning: an improved Bi-RRT-based initialization method. *Assembly Automat.* 37, 261–270. doi: 10.1108/AA-12-2016-173

Liu, J. , Yang, J., Liu, H., Tian, X., and Gao, M. (2016). An improved ant colony algorithm for robot path planning. *Soft Comput.* 1, 1–11. doi: 10.1007/s00500-016-2161-7

Miao, H., and Tian, Y. C. (2013). Dynamic robot path planning using an enhanced simulated annealing approach. *Appl. Math. Comput.* 222, 420–437. doi: 10.1016/j.amc.2013.07.022

Nair, R. R., Behera, L., Kumar, V., and Jamshidi, M. M. (2015). Multisatellite formation control for remote sensing applications using artificial potential field and adaptive fuzzy sliding mode control. *IEEE Syst. J.* 9, 508–518. doi: 10.1109/jsyst.2014.2335442

Shi, E., Chen, M., Li, J., and Huang, Y. (2014). Research on method of global path-planning for mobile robot based on ant-colony algorithm. *Trans. Chin. Soc. Agri. Machinery* 45, 53–57. doi: 10.6041/j.issn.1000-1298.2014.06.009

Song, B., Wang, Z., and Zou, L. (2016). On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm. *Cogn. Comput.* 9, 5–17. doi: 10.1007/s12559-016-9442-4

Stützle, T., and Hoos, H. H. (2000). MAX-MIN ant system. *Fut. Gen. Comp. Syst.* 16, 889–914. doi: 10.1016/S0167-739X(00)00043-1

Wang, D. S., and Yu, H. F. (2011). Path planning of mobile robot in dynamic environments. *Int. Conference Intelligent Control Info. Process.* 2, 691–696. doi: 10.1109/ICICIP.2011.6008338

Wang, P., Lin, H. T., and Wang, T. S. (2016). An improved ant colony system algorithm for solving the IP traceback problem. *Elsevier Science Inc.* 326, 172–187. doi: 10.1016/j.ins.2015.07.006

Wu, J., Li, T., and Xu, B. (2013). Force optimization of planar 2-DOF parallel manipulators with actuation redundancy considering deformation. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* 227, 1371-1377. doi: 10.1177/0954406212458055

Wu, J., Wang, J., and You, Z. (2010). An overview of dynamic parameter identification of robots. *Robot. Computer Integr. Manufacturing* 26, 414–419. doi: 10.1016/j.rcim.2010.03.013

Yang, C., Huang, K., Cheng, H., Li, Y., and Su, C. (2017). Haptic identification by ELM-controlled uncertain manipulator. *IEEE Trans. Syst. Man Cybernet. Syst.* 47, 2398–2409. doi: 10.1109/TSMC.2017.2676022

Yang, C., Jiang, Y., He, W., Na, J., Li, Z., and Xu, B. (2018). Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. *IEEE Trans. Ind. Electronics* 65, 8112–8123. doi: 10.1109/TIE.2018.2803773

Yen, C. T., and Cheng, M. F. (2016). A study of fuzzy control with ant colony algorithm used in mobile robot for shortest path planning and obstacle avoidance. *Microsyst. Technol.* 2016, 1–11. doi: 10.1007/s00542-016-3192-9

Zeng, M., Xi, L., and Xiao, A. (2016). The free step length ant colony algorithm in mobile robot path planning. *Adv. Robot.* 30, 1509–1514. doi: 10.1080/01691864.2016.1240627

Zhang, W., Gong, X., Han, G., and Zhao, Y. (2017). An improved ant colony algorithm for path planning in one scenic area with many spots. *IEEE Access.* 5, 13260–13269. doi: 10.1109/ACCESS.2017.2723892

Zhao, J., Cheng, D., and Hao, C. (2016). An improved ant colony algorithm for solving the path planning problem of the omnidirectional mobile vehicle. *Math. Prob. Eng.* 2016, 1–10. doi: 10.1155/2016/7672839

Zhou, Z., Nie, Y., and Min, G. (2013). Enhanced ant colony optimization algorithm for global path planning of mobile robots. *Int. Conference Comput. Info. Sci.* 2013, 698–701. doi: 10.1109/ICCIS.2013.189

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 10 May 2019
doi: 10.3389/fnbot.2019.00012

Neural Network Based Uncertainty Prediction for Autonomous Vehicle Application

Feihu Zhang[1], Clara Marina Martinez[2], Daniel Clarke[3], Dongpu Cao[4] and Alois Knoll[5]*

[1] School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China, [2] Porsche Engineering Services GmbH, Bietigheim-Bissingen, Germany, [3] Cogsense Technologies Limited, London, United Kingdom, [4] Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON, Canada, [5] Department of Informatics, Technical University of Munich, Munich, Germany

This paper proposes a framework for uncertainty prediction in complex fusion networks, where signals become available sporadically. Assuming there is no information of the sensor characteristics available, a surrogated model of the sensor uncertainty is yielded directly from data through artificial neural networks. The strategy developed is applied to autonomous vehicle localization through odometry sensors (speed and orientation), so as to determine the location uncertainty in the trajectory. The results obtained allow for fusion of autonomous vehicle location measurements, and effective correction of the accumulated odometry error in most scenarios. The neural networks applicability and generalization capacity are proven, evidencing the suitability of the presented methodology for uncertainty estimation in non-linear and intractable processes.

Keywords: neural network, autonomous driving, uncertainty prediction, localization, odometry

OPEN ACCESS

Edited by:
Changhong Fu,
Tongji University, China

Reviewed by:
Liang Lu,
Polytechnic University of Madrid, Spain
Chao Zhang,
Shanghai Jiao Tong University, China
Hui Xie,
Western Sydney University, Australia

****Correspondence:***
Feihu Zhang
feihu.zhang@nwpu.edu.cn

Received: *08 October 2018*
Accepted: *18 March 2019*
Published: *10 May 2019*

Citation:
Zhang F, Martinez CM, Clarke D, Cao D and Knoll A (2019) Neural Network Based Uncertainty Prediction for Autonomous Vehicle Application. Front. Neurorobot. 13:12. doi: 10.3389/fnbot.2019.00012

1. INTRODUCTION

Mobility has become a serious challenge in a society with a gradually aging population and a perpetually increasing traffic. This situation is motivating the automotive industry and governments to invest heavily in highly automatized vehicles toward full autonomy. Nevertheless, the progress in autonomous vehicle integration is being hindered from an engineering perspective, due to limits in vehicle sensing and infrastructure modernization requirements (Ma et al., 2018; Taeihagh and Lim, 2019). Moreover, considerable advances in electronics and control theory, safety and robust autonomous driving can only be achieved in conditions that vehicles are fully aware of the driving scenario (Han et al., 2012; Li et al., 2014).

High quality measurements can be obtained using expensive sensors (Elfring et al., 2016), as installed in the well-known Google Car. This vehicle includes an advanced laser range finder, between other sensors, able to process the environment real time (Poczter and Jankovic, 2014). Nonetheless, these advanced devices are generally associated with elevated cost, and therefore are not feasible for serial production vehicles. An alternative solution is to compensate measurements quality with higher redundancy by installing larger number of low cost devices based on different technologies. As a consequence, the features perception becomes a complex problems where heterogeneous signals need to be registered, transformed into a common level and conveniently combined to guarantee safety (Jiang et al., 2011). This process is known as data fusion and usually involves noisy measurements and highly non-linear transformations.

Data fusion can be executed in either centralized or decentralized architectures. Whilst the first involves a common processor, and decentralized architectures consist of networks where each

sensor has its own processing unit (Grime and Durrant-Whyte, 1994; Durrant-Whyte et al., 2001; Garcia-Ligero et al., 2012). On the one hand, centralized architectures need to be re-designed when changes in the sensing units take place, which implies costly and time-consuming development. On the other hand, decentralized solutions are particularly convenient in networks where sensors can be dynamically added and removed from the networks as a result of being sporadically available. Nevertheless, although a plurality of measurements might become accessible in decentralized architectures, fusion requires knowledge of the uncertainty associated to them. Furthermore, measurements are not used directly, but information extracted from them hereby referred to as features, which generally involves non-linear transformations. Consequently, the conversion of sensor noise into feature noise is a complex task that usually involves arduous mathematical derivations and can be intractable in many applications.

Several attempts to uncertainty prediction for inertial measurement units (IMU) have been presented in the literature. These include methodologies to fuse odometric measurements with global positioning system measurements, geographical information systems and laser scanners, between others. Vision systems are used in Park et al. (2012) and the disparity between the image space and the Cartesian space is used to derive the uncertainty mathematical model. Vision based controlled is used by Fu et al. (2018) in a system based on an onboard camera and an IMU. The authors use a non-singleton fuzzy logic controller able to handle high uncertainties. The Kalman filter has been also widely used to deal with noisy measurement and models. The parameters estimation was performed using methods such as random walk, Gaussian-Markov and autoregressive processes (El-Diasty and Pagiatakis, 2008). Extended Kalman filters have been proposed by other authors such as Bry et al. (2012) and Fabrizi et al. (2000), where the noise assumption is taken using Gaussian white noise.

This paper presents a solution to facilitate data fusion in decentralized architectures. The proposed paper enhances our previous system for feature extraction (Martinez et al., 2017), where either the source of information, or the sensor noise is unknown. These networks require an appropriate estimation of the signals uncertainty so as to properly fuse them into an "improved measurement," rather than worsen the fusion output. The uncertainty allows evaluating the quality of the signals and provides a combined result of higher accuracy where the information retained is maximized.

Despite its importance, in the literature revised sensor noise is either assumed to be known, or fitted with simple Gaussian distributions. Hereby, a methodology directly for uncertainty prediction from raw data is proposed based on Artificial Neural Networks (ANN), assuming no information is prior available about the sensor characteristics. The applicability of this data-driven strategy extends to highly non-linear and even intractable feature transformation, avoiding tedious mathematical derivations. Proof of this is supported by its implementation for autonomous vehicle location through odometry data, obtaining satisfactory results in varied scenarios.

2. PROBLEM DEFINITION

Autonomous driving highly depends on the sensor measurement, uncertainty and fusion. Nonetheless, sensor models are not generality provided by sensor manufacturers. This lack of data significantly increases challenges in decentralized architectures, where new sensors can be "plugged & played" within the *ad-hoc* network.

2.1. Motivation

The operational limits of the sensor technology condition safety in autonomous driving, as a consequence of their strong dependence on the measurements quality (Zheng and McDonald, 2003; Michalke et al., 2011). The GPS precision limits are a well-known example of the noise effect in systems performance, experienced daily by the general public through of the shelf navigation devices (Schrader et al., 2012). This involves the fusion of pseudo-range GPS signals of vehicles, used to minimize the error produced by uncontrolled sources like satellite clock bias, atmospheric delay, and acquisition noise. Nevertheless, despite the complexity of the error origin, previous studies model noise using Gaussian distributions (Liu et al., 2013).

The main barriers for sensor fusion in application, such as vehicle localization, are found in the uncertainty of sensor technology integrated in each vehicle. This inevitably affects the uncertainty characteristics, and the different nature of the signals to fuse, involving highly non-linear feature transformations (Xu et al., 2014). Furthermore, incorrect uncertainty estimation could reduce the fusion accuracy and produce a security hazard by deteriorating the system performance. Regardless of its importance, most research in this area has limited the error prediction to single vehicle model-based approaches usually using Gaussian distributions, developed either theoretically or empirically. Therefore, more accurate uncertainty estimation would be of great benefit for these applications, and would solve issues that hinder the implementation of the autonomous technology nowadays.

2.2. Problem Statement

Odometry measurement for vehicle location is subjected to sensor noise applied to velocity d and orientation θ. This uncertainty is extended to the features x and y coordinates, which are calculated from the noisy signals through geometrical transformations (Choi and Huhtala, 2016). Consequently, the vehicle location error is accordingly described by non-linear mathematical equations and accumulates along the path with every sampling time. The absence of information about the features uncertainty, x and y covariance, prevents from fusing odometry data with additional measurements that might become available along the trajectory. By means for this, the estimation of the location uncertainty is of great interest to efficiently correct the accumulated error (Zhang et al., 2013). Hereby a feature noise estimator is obtained from data with independence of the sensor characteristics, and complexity of the feature transformation. This solution allows for sensor noise prediction, avoids the use of complex mathematical formulations and facilitates sensor fusion under any use case.

2.3. Data-Driven Modeling

Feature transformations are often difficult to derive in mathematical terms, and calculations that are usually time consuming and occasionally intractable. Nevertheless, vehicle trajectory data collection under real-life conditions is generally possible using limited resources. These evidences support the use of data-driven algorithms, methods that can efficiently manage big data and yield insightful conclusions from unknown complex processes (McAfee et al., 2012; Hou and Wang, 2013). Various algorithms can be used to derive the so-called surrogate models without requiring actual understanding of the relationship between inputs and outputs. These models are compact, normally inexpensive to evaluate compared to their homologous strictly mathematically derived. Furthermore, they are mathematically tractable and can estimate the process with high-fidelity at least locally to the training set (Gorissen et al., 2010; Koziel et al., 2011). Some examples of surrogate modeling techniques include: polynomial regressions, kernel methods, kriging, support vector machine, Radial Basis Functions (RBF), and Neural Networks (NN) (Jin et al., 2001; Razavi et al., 2012). Each method has different characteristics in terms of operation, complexity, design flexibility and fidelity capability. For instance, support vectors perform particularly well with high dimensional spaces when only scarce training data is available (Forrester and Keane, 2009). Highly non-linear and complex process are better captured using RBF, kriging, and NN, which require determining a specific number of parameters by trial and error. From a high level analysis, the structural limits of RBF are relaxed with kriging, which assumes the model response has stochastic behavior and fits it with a statistical basis. In the kriging method the basis function variance is considered a parameter, providing larger flexibility and resultant increase in training time.

Artificial Neural Networks allow modeling the relationship between inputs and outputs from data. This characteristic, applied to sensor noise, is expected to be able to find the underlying relation between measurement and noise associated to them. Furthermore, NN accept multiple inputs, which can be used to determine additional features affecting the noise and their correlation. With these precedents, NN offer an exceptional framework for implementing and testing the suitability of models generated from data applied to noise magnitude prognosis of sensor measurements. Hereby, NN are selected within the above strategies to exploit their potential for sensor noise estimation, and explore their high level of flexibility owing to their substantial number of defining parameters: network structure, neuron function, number of hidden layers, and number of neurons per layer.

2.4. Design for Surrogate Model Development

In terms of number of hidden layers, the criterion applied focuses on the trade-off between accuracy and generalization capabilities. Sensor fusion algorithms will benefit from a guidance to assess the extent of the error covariance of new sensor measurements. This information will allow the system to identify the degree of information present in the new measurement and perform the data fusion accordingly, ensuring the output maximizes the information content. It is therefore acceptable to obtain a guidance value of this error covariance and not highly precise results, reason why the network structure selected for sensor noise estimation is formed by a single hidden layer. This simplified structure might prevent from learning accurate noise behavior as observed in deep learning, but would also facilitate training and avoid noise fitting when applied to noisy sensor signals. Generalization capability is hereby prioritize against results accuracy with the selection of the single hidden layer structure.

The network layers, named input, hidden and output, can be connected through either feedforward (FF) configuration or using a feedback (FB) connection. Layers in FFNNs only receive information from forward layers, whilst in FBNNs any neurons can connect with each other. Consequently, signals in FBNN are repeatedly transformed and lean toward steady state or vibration state. By introducing feedback delays, this structure is also able to capture the relationship between past inputs and current output, influence that is completely ignored in the FF configuration. In the following, both FF and FB configurations are examined as candidates of NN structure so as to determine the most suitable configuration with the support of the training and test results.

Once the network structure is defined, the size of the layers need to be determined. Input and output layers are constrained by the input and output signals selected set, but the hidden layer is a prior a free parameter, closely related with the process complexity. By reason of lack of known mathematical formulation of this particular case, this number has to be determined by trial and error. The optimal size criteria should consider a trade-off between complexity, accuracy, and generalization capability of the neural network candidates. Excessively complex networks not only raise training time, but also increase the risk of over fitting, which would return high accurate results over the training set and poor generalization capacity on new data (Hagan et al., 2014). The growing method is used in this application in order to prevent for over fitting by establishing an initial network with relative small size, and increasing it gradually with special attention over both the training and testing accuracy.

3. TRAINING DATA GENERATION AND ANALYSIS

To encourage acceptable performance under all possible scenarios, the amount and variability of the training data should ideally account for any conceivable use case. The data selected for training proceeds from six different trajectories that combine disparate direction, length, orientation, and speed as depicted in **Figure 1**.

3.1. Training Data Generation

The trajectories contain highly precise vehicle locations in Cartesian coordinates, and yaw signals, regarded as Ground

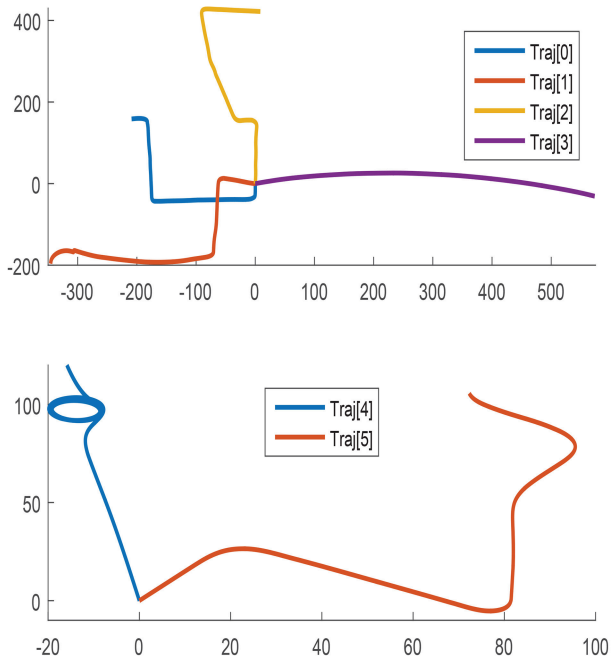


FIGURE 1 | Path followed in all trajectories for training.

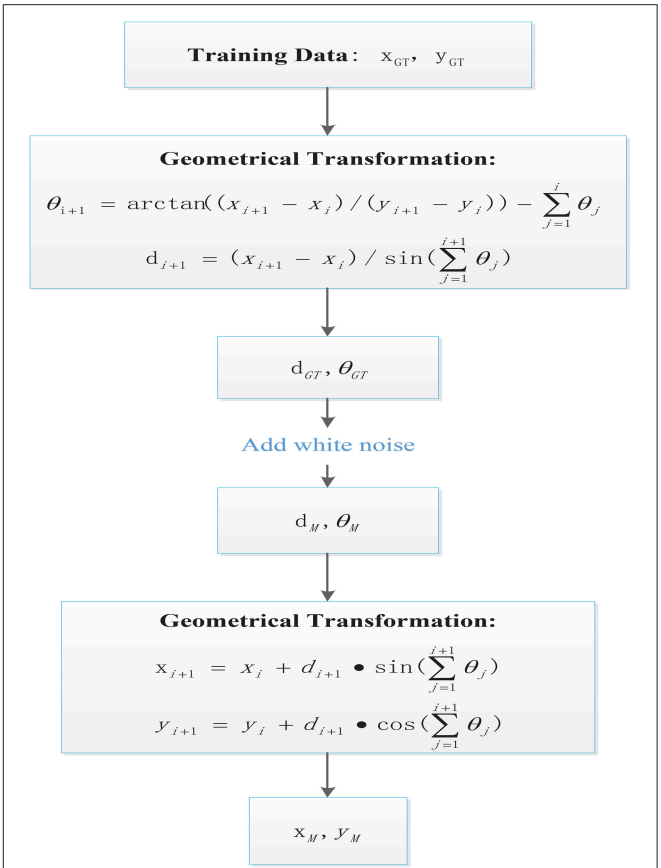


FIGURE 2 | Flow diagram illustrating the ground truth data processing to emulate feature extraction from noisy signals.

Truth (GT). The GT signals need to be processed to generate realistic data complying with a real case scenario. Training data is generated assuming the vehicle velocity and orientation, identified herewith with the symbols d and θ respectively, are collected with sensors characterized by white noise. For the purpose of the following investigations, the standard deviation values 0.1 and 0.001 are respectively selected for speed and orientation measurements. These values are based on experience and are considered representative of general noise measure in odometry sensors.

- Calculate d and θ GT, d_{GT} and θ_{GT}, from x_{GT} and y_{GT}.
- Add white noise artificially to d_{GT} and θ_{GT}, through a Monte Carlo (MC) simulation with 1000 iterations. These results in d and θ measured (M), d_M and θ_M, and emulates sensor noisy data acquisition.
- Use the inverse equations to calculate x_M and y_M from d_M and θ_M, which in essence is the signal to feature transformation.
- Use the 1000 MC noisy versions of the trajectories to calculate the location error standard deviation (std) in x and y and the location covariance (cov_{xy}).

Figure 2 illustrates the detailed process in a flow diagram describing the steps and signals obtained from ground truth to measured feature data. As included in the respective steps for geometrical transformations, the signals d and θ could be obtained along the sampling steps in a cumulative manner as detailed in following equations (Zhang et al., 2013):

$$x_n = \sum_{i=1}^{n} d_i \cdot \sin\left(\sum_{j=1}^{i} \theta_j\right) \tag{1}$$

$$y_n = \sum_{i=1}^{n} d_i \cdot \cos\left(\sum_{j=1}^{i} \theta_j\right) \tag{2}$$

where n refers to the current time step. By combining Equations (1) and (2), the variables of interest can be obtained for every sampling time:

$$\theta_{i+1} = \arctan\left((x_{i+1} - x_i)/(y_{i+1} - y_i)\right) - \sum_{j=1}^{i} \theta_j \tag{3}$$

$$d_{i+1} = (x_{i+1} - x_i) \Big/ \sin\left(\sum_{j=1}^{i+1} \theta_j\right) \tag{4}$$

Noise can be artificially added to the GT results of Equations (3) and (4) by randomly generating numbers with the previously designated standard deviation. The results are regarded as sensor measurements and can be used to obtain the measured features

following Equations (1) and (2), implemented as:

$$x_{i+1} = x_i + d_{i+1} \cdot \sin\left(\sum_{j=1}^{i+1} \theta_j\right) \tag{5}$$

$$y_{i+1} = y_i + d_{i+1} \cdot \cos\left(\sum_{j=1}^{i+1} \theta_j\right) \tag{6}$$

The ground truth original data and measured features can be compared to determine the uncertainty over the vehicle location at every point of the trajectories. This is defined by the standard deviation of the feature estimation error:

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(ex_i - \mu_x)} \tag{7}$$

where N corresponds to the number of MC iterations of the same trajectory. The estimation error and the mean estimation error can be calculated as follows:

$$ex = x_{GT} - x_M \tag{8}$$

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} ex_i \tag{9}$$

Similarly, the equations can be applied to y coordinate to obtain ey, μ_y, σ_y. Finally, the covariance of the errors in x and y is obtained by:

$$\text{cov}_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)\left(y_i - \mu_y\right) \tag{10}$$

The uncertainty is defined as σ_x, σ_y, and cov_{xy}. Mean error in x and y bias the location measurement, but are not considered as estimation targets in this particular study. The previous transformation provides information of the uncertainty of the vehicle location, when measured through noisy velocity and orientation sensors subjected to a specific level of white noise. This measurement allows evaluating the quality of the current location through odometry, and therefore to which extent this measure should contribute into a sensor fusion framework when compared to other sources.

σ_x obtained for each of the trajectories is illustrated in **Figures 3**, **4**, by assuming the vehicle is perfectly located at the initial point. **Figure 3** represents σ_x growth along the entire trajectories until the end point of the longest one, whilst **Figure 4** illustrates a zoom in the σ_x to better visualize the uncertainty accumulated in shorter paths. σ_x always presents and increasing trend due to the cumulative characteristics of the error in the vehicle location. Nonetheless, this tendency of accumulation differs between trajectories, which suggests that the shape of the trajectory and the characteristics of the displacement affect the uncertainty growth. The curves are therefore dissimilar between trajectories and presumably influenced by variables such as $\Delta x, \Delta y$, and Δyaw. Analogous behavior is observed when analyzing σ_y and cov_{xy}, which agrees with the previous assumption.

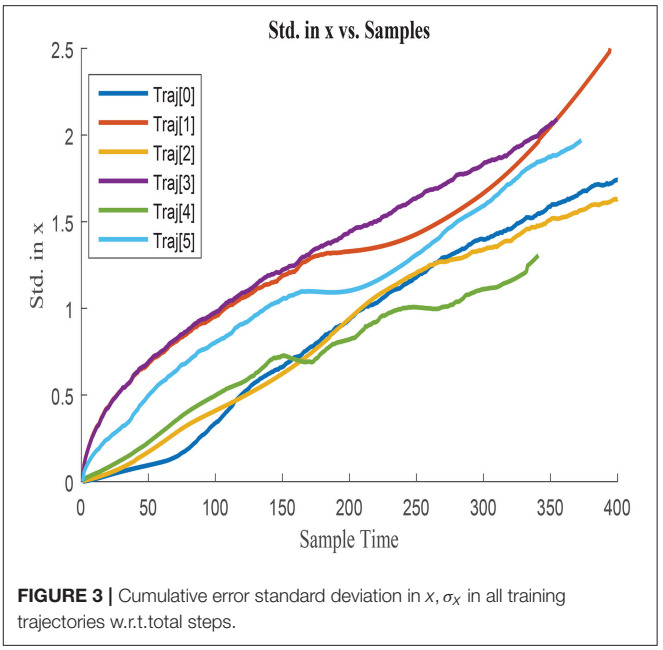


FIGURE 3 | Cumulative error standard deviation in x, σ_x in all training trajectories w.r.t.total steps.

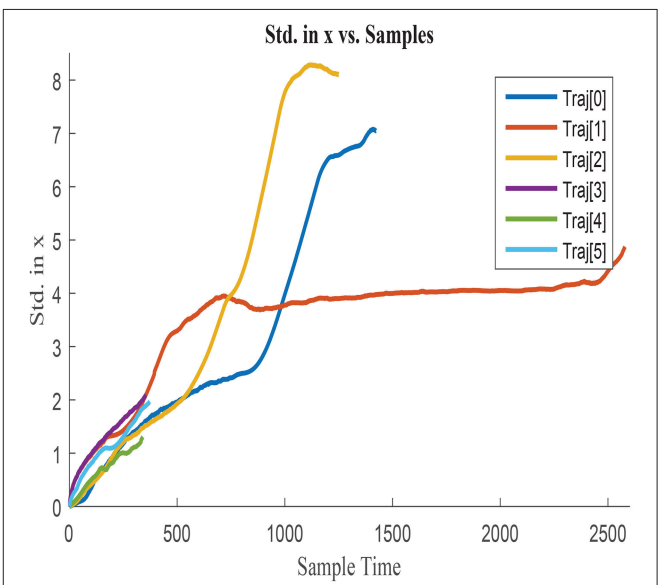


FIGURE 4 | Zoom in cumulative error standard deviation in x with shorter duration.

The growth of the combined uncertainty in all directions σ_x, σ_y, and cov_{xy} is illustrated in **Figures 5**, **6**, which is represented via ellipses that increase in area as the error accumulates. A first visual examination allows identifying how larger increments in a specific

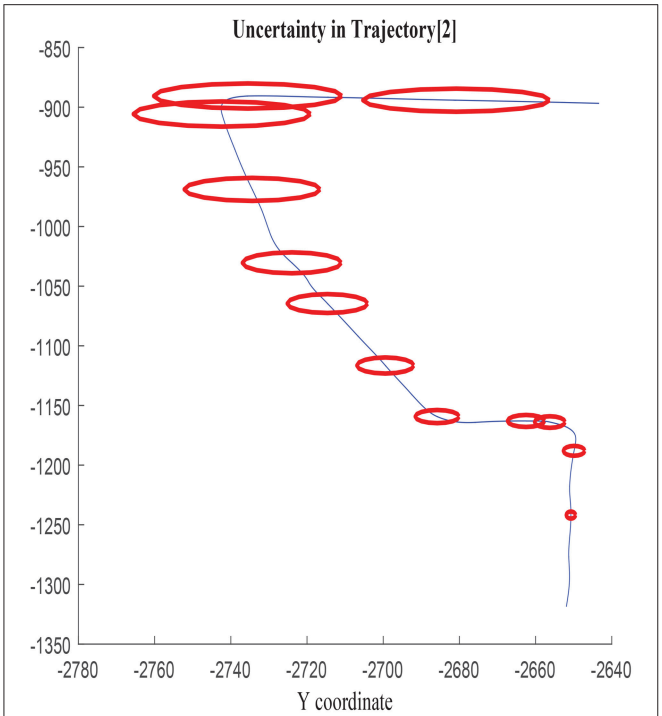


FIGURE 5 | Error ellipses in σ_x, σ_y, and cov_{xy} in trajectory 2.

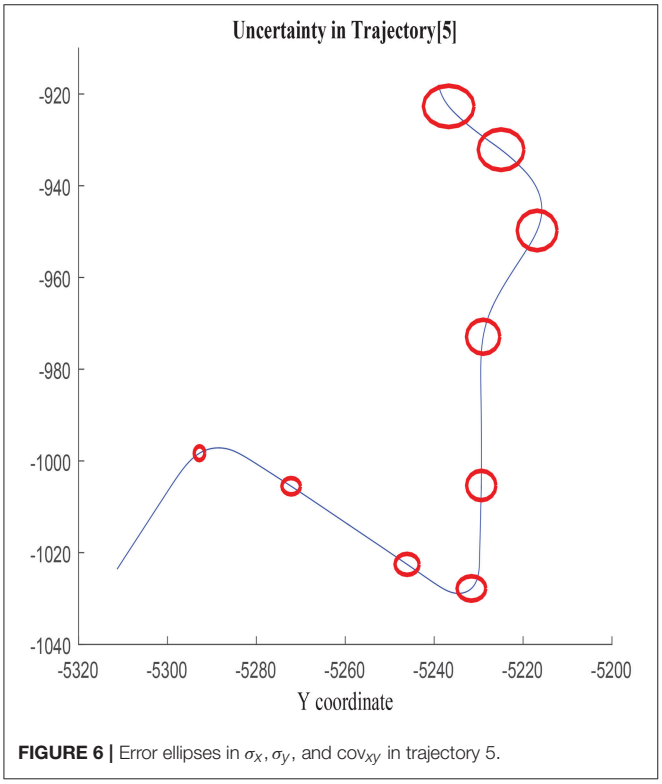


FIGURE 6 | Error ellipses in σ_x, σ_y, and cov_{xy} in trajectory 5.

direction affect to the growth of the uncertainty differently, which also allows drawing a priori hypothesis of the variables closely related. In the following, the study concentrates on σ_x, although the results and conclusions are expected to be dimension agnostic and applicable to both σ_y and cov_{xy}.

3.2. Training Data Analysis

As aforementioned, one of the remaining design parameters of the NN is the number of inputs. Ideally, the input variables should contain the maximum number influencing factors over the target to estimate, but its scope should be constraint to prevent from excessive training time and network complexity and overfitting. The optimal input selection should aim to gather maximum relevant information for the prediction and minimum non-relevant data. Non-useful information would increase the model complexity, and might introduce misleading data that deteriorate the generalization capability.

Inputs can be selected using common sense in easily interpretable applications, albeit there are alternative correlation analysis able to evaluate numerically their level of dependence with respect to the target (Sudheer and Ramasastri, 2002). In addition to a prior evaluation of inputs and outputs, NNs can be themselves used for signals selection. In simple network structures, the importance of each signal can be identified by looking into the magnitude of the weights that connect them to the successive layers. A formal analysis of the signals weights is included in Giordano et al. (2014), where a criteria for input selection is derived mathematically and tested.

As a result of the noisy characteristics of the variables used in this particular application, complex signal evaluation is not considered of interest. Instead, the procedure follows inputs selection, by consisting the combination of a correlation analysis with the training and testing results interpretation of various input candidates. First, the signals linear correlation is studied calculating the Pearson correlation coefficient with respect to the output. The high sampling rate used compared to the input candidates' variability, allows assuming no time lag exists between inputs and outputs, simplifying the evaluation.

Although immediate effect of inputs over the outputs is impracticable, the study of the signals' variability with respect to delayed ones suggests this assumption is acceptable (Maier and Dandy, 1997). The output of the correlation provides useful information to select several input candidates, which are later tested at a second stage to conclude into the most suitable option. The best candidate is assessed in terms of training and testing results in an iterative process.

3.3. Inputs vs. Output Correlation

The signals available to use as inputs are: x, y, yaw, θ, d. The use of values such as real x and y directly impairs generalization, as the network would learn from training trajectories characterized by specific absolute location points. Moreover, the incremental tendency of the uncertainty suggests additive behavior happening in every trajectory with independence from the initial and relative vehicle location. This reasoning supports the use of signals increments between sampling steps, rather than absolute values with respect to a pre-defined reference system.

Another hypothesis that can be reasonably stated is the effect of the direction of the displacement over the uncertainty growth;

that is, whether variables increments or absolute variables increments are suitable for the input set. This conjuncture would be resolved when determining the effect of the inputs increments sign on the uncertainty accumulation. It is sensible to assume that the features uncertainty is affected by the actual magnitude of the displacement with independence on the direction; the uncertainty should not be affected by the reference system. Consequently, it could be deduced that the sign omission would avoid needless information to be fed into the NN, and therefore would encourage the generalization capacity.

All previous hypotheses are considered to determine the signal candidates to evaluate in the correlation analysis. **Table 1** contains the Pearson correlation coefficients between input candidates and output in all training trajectories. The formula used is the base line Pearson equation, where r, cov, and σ represent respectively correlation coefficient, covariance and variance of the designated signals (Lee Rodgers and Nicewander, 1988).

$$r_{xy} = \mathrm{cov}(var_1, var_2)/\sigma_{\mathrm{var1}}\sigma_{\mathrm{var2}} \tag{11}$$

Table 1 shows high correlation between σ_x and the signals yaw, x, and y. d appears to have secondary importance, although it proves to be relevant in training trajectories 3 and 4, when compared to 1 and 5 for instance. These differences are associated with the mean value and mean absolute value of the speed, observed to be higher in trajectories 3 and 4. In contrast, θ correlation seems to be negligible.

The correlation coefficients change substantially when analyzing variables increments. The weight of Δx and Δy weights reduce, with respect to the original variables and Δyaw becomes practically independent to the output. Δy shows stronger relationship with $\Delta\sigma_x$ when compared to Δx. In contrast, Δyaw are only tangible in trajectories characterized by substantial direction changes, as happens in trajectories 4 and 5. Δd also loses relevance when compared to the absolute variable analysis, and θ influence is barely affected and kept negligible. When focusing on absolute values of the increments, the correlation results produce similar values compared to relative increments, supporting a priori hypothesis over the uncertainty independence with respect to the sign of the movement. A part from the magnitude of the correlation coefficients, the sign can be also interpreted. The broad variety of values and sign within trajectories prevents from selecting a single preferred combination of training signals, reason why several candidates are selected to further pinpoint the suitability.

As a final remark, it is worth highlighting that Δy presents larger correlation with σ_x than Δx, and vice versa. That is, in trajectories with more movements in x direction, σ_y grows quicker than σ_x and similarly, in trajectories with larger displacement in y direction, σ_x grows quicker than σ_y. This is observed in all trajectories with exception of trajectory 4, where both uncertainties are similar probably due to the followed direction in repeated circles. An explanation of this phenomenon might be found, in the relative amplitude of the actual displacement every sampling and the error magnitude. Whilst the error might be negligible after a large displacement, it could be of the same order of magnitude of short movements, causing larger distortion in the vehicle location. Consequently, large growth of σ_x could be associated to low Δx instead of being related to Δy as it was initially deduced from the results of the analysis.

3.4. Delayed Signals Correlation

Inputs to output correlation analysis is complemented with the signals delay study, also evaluated using the Pearson coefficient. The aim of this test is to determine the possible relationship between old inputs and current outputs; that is, the influence of past changes in the vehicle movement and location on the accumulation of the current uncertainty in the vehicle positioning. The first correlation test of delayed signals analyses the relationship between delayed inputs and current output. The steps used are 0, 1, and 2 sampling times. Next, in order to draw a holistic understanding of the signals interrelation, a second correlation test between current input signals and the same ones delayed 1 and 2 sampling steps is also analyzed.

The results of the first test show similar correlation between input and outputs with independence of the delay implemented. Nonetheless, the second test also shows strong correlation between inputs and the respective delayed inputs. Although from the first results it could be considered that the output depends on past input signals, the second analysis discredits this assumption as they could also be due to the high similarity between current and past inputs. Consequently, no conclusive assumptions can be derived from this correlation test.

The results from the second correlation test effectively show that the inputs and delayed version of the inputs are practically identical, and consequently show similar correlation with the output. As previously states, this similarity might be due to the small sampling step implemented with respect to the input signals variability in time. Further investigations should be conducted to arise conclusive answers to the previous hypothesis. Accordingly, additional study with respect to the delay effect of the feedback NN states is considered during training.

4. TRAINING SETS CANDIDATES

Hereby, a training set is defined as the union of a specific combination of input signals, obtained from a selected number of training trajectories in an enclosed array used as training data. That is, the training set is defined by the signals used between the candidates previously analyzed in the correlation analysis and the trajectories from which signals are extracted. The training sets can contain data proceeding exclusively from a single trajectory or from the combination of more than one. Furthermore, the same trajectory can be repeated in each set in more than one occasion by implementing different noisy version from the 1000 MC simulations, practice that intents to encourage the response robustness to the presence of noise in the inputs.

4.1. Training Sets

The input training sets are designed in terms of number of signals, trajectory characteristics and amount of trajectories used, and always contain noisy data so as to simulate with

TABLE 1 | Pearson correlation coefficients analysis of input candidate signals, signal increments, and absolute signal increments.

		Signal					Signal increment					Signal absolute increment				
Target	**Traj**	**x**	**y**	**yaw**	**d**	θ	**x**	**y**	**yaw**	**d**	θ	**x**	**y**	**yaw**	**d**	θ
x std/Increment x std	[0]	–0.8	0.9	–0.7	0.4	0	0.2	0.6	0.1	0	0	0.3	0.6	0.1	0	0.1
	[1]	–0.9	–1	–0.4	0.1	0	–0.1	–0.5	0	–0.1	0	0.1	0.5	0.1	0.2	0.1
	[2]	–0.8	1	–0.7	–0.4	0	–0.5	0.6	0	0.1	0	–0.4	0.6	–0.2	0	–0.1
	[3]	1	–0.5	–1	–0.8	–0.1	–0.2	0.4	0	0	–0.2	–0.2	0.1	0	0	0.2
	[4]	–0.7	0.8	0.9	–0.5	0	–0.2	0.6	–0.3	0.3	0	0.4	0.4	0	0.4	0.1
	[5]	0.9	0.8	0.7	0.2	0	0.3	–0.1	–0.3	–0.1	0	0.5	0	0.4	0.1	0.1

TABLE 2 | Input sets candidates proposed for training and testing including: signals selected, data used, and hypothesis to verify/reject in the training results.

Set	**Inputs**	**Data**	**Explanation**
1	$\Delta x_M, \Delta y_M, \Delta yaw_M$	All trajectories-3 times	Input:relative increment
2	$\Delta x_M, \Delta y_M, \Delta yaw_M$	Trajectory[0]-10 times	Data:generalization capacity
3	abs($\Delta x_M, \Delta y_M, \Delta yaw_M$)	Trajectory[0]-10 times	Input:generalization of abs. inc
4	abs($\Delta x_M, \Delta y_M, \Delta yaw_M$)	Traj.[1],[4,][5]-10 times	Data:generalization disparate data
5	abs($\Delta x_M, \Delta y_M, \Delta yaw_M$)	Traj.[1],[4,][5]-5 times	Data:generalization disparate data
6	abs($\Delta x_M, \Delta y_M, \Delta yaw_M$ and $\Delta\theta_M$)	Traj.[1],[4,][5]-5 times	Input:proof of correction analysis
7	abs($\Delta x_M, \Delta y_M$ and $\Delta\theta_M$)	Traj.[1],[4,][5]-5 times	Input:proof of correction analysis
8	abs(Δy_M and $\Delta\theta_M$)	Traj.[1],[4,][5]-5 times	Input:proof of correction analysis
9	abs($\Delta x_M, \Delta y_M$ and Δyaw_M)	All trajectories-3 times	Additional testing
10	abs(Δy_M and $\Delta\theta_M$)	All trajectories-3 times	Additional testing

maximum fidelity real case studies. **Table 2** includes the training set candidates carefully designed to determine: the most suitable combination of inputs, optimal network structure and size and data variability requirements.

The second column in **Table 2** specifies the input signals used in each set, where abs and Δ indicates absolute value and signals increment, respectively. The amount of data used in each training set is detailed in the third column, alluding to the variability of trajectories used and amount of MC noisy versions of each trajectory. For instance, set 1 considers all trajectories repeated three times each; including therefore three MC noisy versions of each. The use of larger number of trajectories or specific ones is thoroughly defined, so as to reflect changes in the generalization capability with respect to training data variability. Consequently, by repeating noisy version of the same trajectory, the data variability should be much less than noisy versions of different trajectories.

The incremental variables specified in **Table 2** are calculated, with respect to the constant sampling rate in training sets. As an attempt to encourage generalization, alternative incremental inputs are proposed by using random dynamic sampling within the boundary of 1 to 9 sampling steps. Nonetheless, the networks trained with this data were not able to estimate the target variable, reason why they are neither included in **Table 2** nor in the test results. The failure to capture the process could be excused in the data variability and complexity introduced through variable sampling. The networks trained with this data were presumably required to emulate a behavior more complicated than the one described with constant sampling. Consequently, it might be the case that the amount of training data and network size used were not suitable to capture efficiently the underlying process. **Figure 7** illustrates a flow diagram that clarifies the design process and characteristics that define a training set. The input selection is partitioned in three stages: selection of the key signals combination, format of the signals preferred (real value, increments, or absolute increments) and trajectories used to extract the data.

4.2. Training Candidates

Sets 1 and 2, similarly to 3 and 4, compare the effect of the data variability on the generalization capacity by implementing identical input variables, but using data from different trajectories. Sets 2 and 3 determine the effect of the inputs sign again with respect to generalization, and intend to provide numerical support to proof the independence between uncertainty accumulation and movement direction. Sets 5 to 8 implement identical data, but use different input candidates so as to obtain the optimal signals combination.

5. NN DESIGN AND TRAINING

The NN candidates are compared in terms of: network size, structural complexity and results output quality. The accuracy of the estimation is evaluated using various error measurements applied to both network output, $\Delta\sigma_x$, and target variable σ_x.

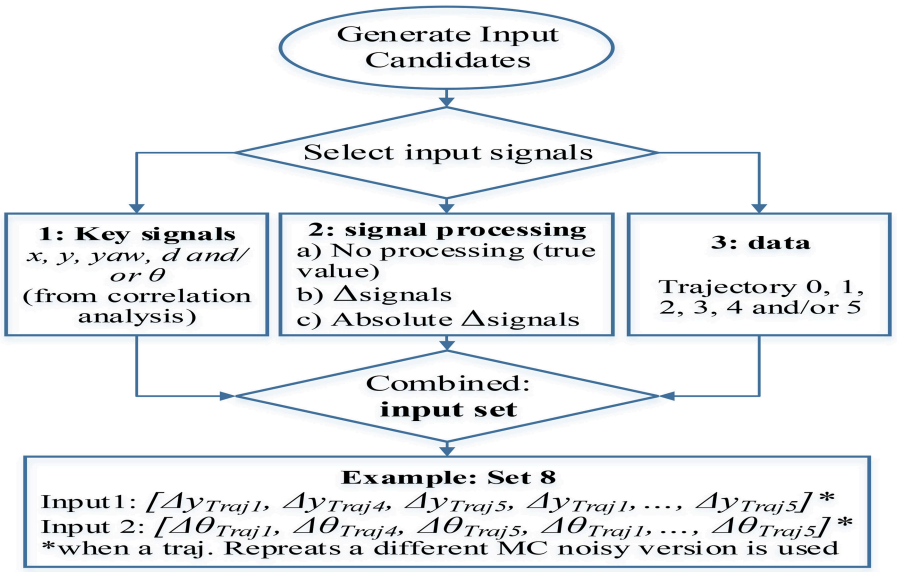


FIGURE 7 | Training sets generation from signals proceeding from the available trajectories including signals selection (from correlation analysis), signal processing (true value, increment, or absolute increment), and training trajectories used (data variability).

TABLE 3 | Training results comparison and analysis in terms of structural and data complexity, training time and performance, and accuracy indexes as convey for error evaluation.

	Structure	Performance	Epochs	Data Q	RMS	rms Cmltv	End error
Set 2	100FF	5.99E-06	136	14130	0.059	9	19
Set 2	20LR	9.23E-07	200	14130	0.0447	7.11	9.07
Set 3	20LR	1.07E-06	200	14130	0.0286	4.96	7.46
Set 3	30LR	1.25E-06	82	14130	0.032	4.7	6.6
Set 4	30LR	8.12E-04	67	32900	0.265	119	197
Set 5	30LR	5.92E-05	110	16450	0.22	84.2	67.7
Set 6	30LR	2.60E-05	150	16450	0.24	106.9	88.2
Set 7	30LR	4.50E-05	150	16450	0.27	100.4	80.7
Set 8	30LR	6.15E-05	127	16450	0.18	43.4	30.4
Set 9	40LR	4.67E-06	223	18927	0.0190	2.1	2
Set 10	40LR	4.71E-06	250	18927	0.0220	3.6	4.8
Set 9	30RL	3.56E-06	137	18927	0.0223	3.8	3.9
Set 9	50LR	4.71E-06	300	18927	0.0185	2.6	3.6

5.1. Error Measurements

The results are evaluated by using Root-Mean-Square (RMS) error and relative error measures between the network output and reference data, GT variables. The error measurement include: RMS of σ_x and relative error of the accumulated uncertainty obtained at the end of the trajectory, σ_x(end). The first assesses the actual performance of the network given the fact that the target variable is the increment of the uncertainty, $\Delta\sigma_x$, and not the cumulative one, σ_x. The second, σ_x RMS, evaluates the variable of interest and determines the possible predicted error accumulation and the actual one. Finally, the relative error of the final cumulative uncertainty analyzes how well the NN would perform in case a new sensor becomes available at the end of the vehicle path and fusion is required.

These indicators are calculated for each of the trajectories separately, so that it is possible to compare the performance in both, data used for training and data not seen before. It is also worth mentioning that the data used for training in all cases consist of 70% of the total amount that defined the training set, whilst the rest is used for testing and validation during the training process. None of the sets or NN configurations converge during the training process, when using the LR architecture as a consequence of the noisy characteristics of the data used. Nonetheless, this behavior is not necessarily detrimental due to the fact that the generalization capability is preferred to the

estimation accuracy of a specific trajectory; it is of importance to avoid noise fitting. Consequently, the networks are trained up to a certain performance value or number of iterations, epochs, and early stopping is used before the training gradient stabilizes. The output to estimate is the relative increment of the error in x std in all cases, $\Delta\sigma_x$.

5.2. Training Algorithms

Two algorithms are used for training, Levenberg-Marquardt (LM) and Scaled Conjugate Gradient (SCG). These aim to compensate from deficiencies in terms of robustness, and convergence time of the well-known Error Backpropagation (EBP) and Gauss-Newton algorithms (Moller, 1993; Yu and Wilamowski, 2011). Both differ in the selection of the step size and direction during convergence. Ideally, longer steps should be implemented at early stages and gradually smaller ones should be considered to encourage the result finesse in later stages. Moreover, the error shape might also change, affecting simultaneously to the optimal step direction. SCG implements optimized step size and direction, whilst LM alternates EBP and Gauss-Newton methods depending on the error shape. LM combines the advantages of both strategies taking advantage of the speed convergence of Gauss-Newton with quadratic error, and the robustness of EBP convergence behavior under conditions of non-advantageous for Gauss-Newton.

The training results usually benefit from LM when compared to SCG throughout the test cases. Furthermore, LM presents low μ values, variable that determines the alternation between methods, but it converges neither into Gauss-Newton nor into the steepest descent method.

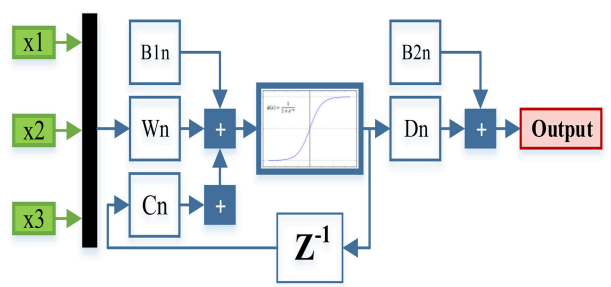


FIGURE 8 | Final structure to find the hidden layer.

5.3. Training Results: Input Signals Selection

Table 3 summarizes the results obtained after training specific network structures, second column, with the training sets enumerated in the first column. Training set 2 is used to compare FF and LR networks. In all cases, the training with identical set and structure is repeated in more than one occasion, typically up to six times. This practice is recommended due to the possible effect that the random weights initialization can cause over the end solution, which can potentially be trapped into local optima. The results included in **Table 3** are taken from the best network obtained after training several candidates. These figures are compared between equal amounts of iterations.

Sets 1 and 2 are omitted for brevity, as the conclusion coincides with the analysis of sets 3 and 4. Set 2 is used to implement FF and FB configurations as included in the first two rows of **Table 3**. The noise filtering capability of FBNN improves the estimation accuracy notably, reason why the LR structure is concluded as the most suitable and successively used in the subsequent training sets.

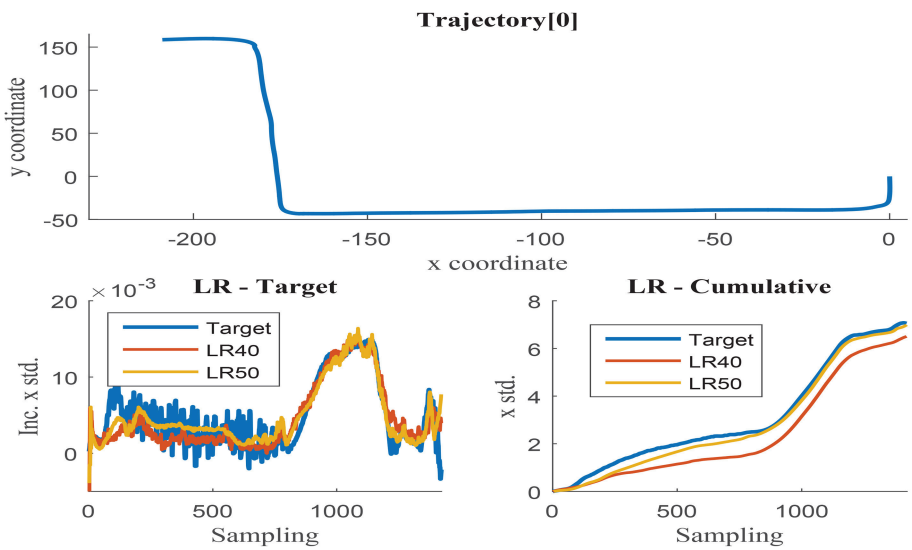


FIGURE 9 | (Top) x and y coordinates of trajectory 0; **(Bottom-Left)** $\Delta\sigma_x$ in trajectory 0 with respect to the sensor sampling; **(Bottom-Right)** $\Delta\sigma_x$ in trajectory 0 with respect to the sensor sampling.

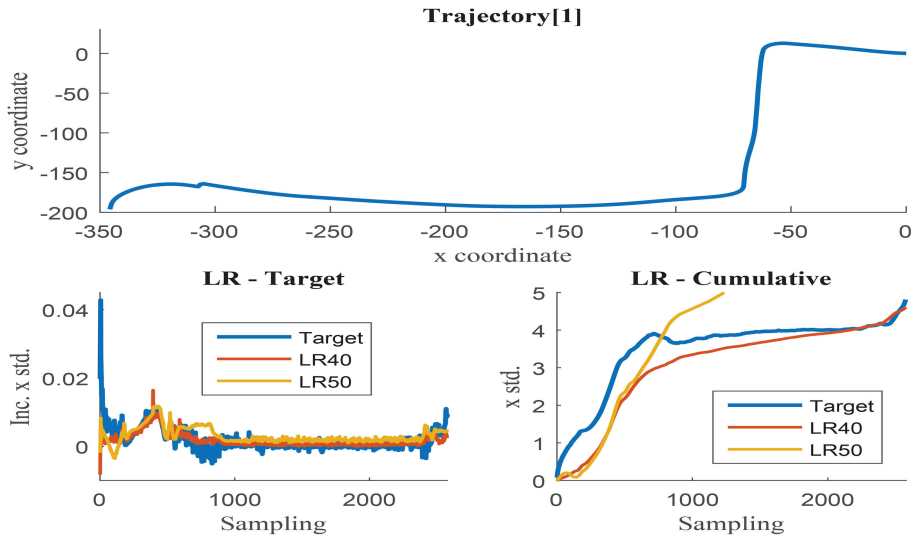


FIGURE 10 | (Top) x and y coordinates of trajectory 1; **(Bottom-Left)** $\Delta\sigma_x$ in trajectory 1 with respect to the sensor sampling; **(Bottom-Right)** $\Delta\sigma_x$ in trajectory 1 with respect to the sensor sampling.

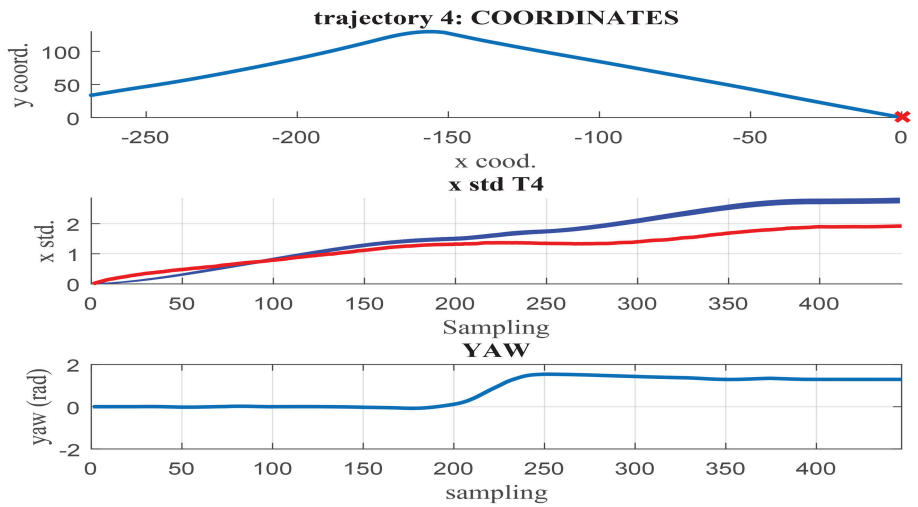


FIGURE 11 | (Middle) σ_x estimation in MC noisy trajectories (blue) respect to path coordinates **(Top)** and yaw **(Bottom)** in trajectory 4.

Six networks trained with set 2 are compared to six networks trained set 3, to determine the effect of sign in the generalization capacity. The best candidate from set 3 shows that the error is 30% lower compared to the best candidate provided by set 2. The results evidence the benefits of reducing non-relevant data in the input set, and corroborate the independence of the location error accumulation with respect to the movement direction.

Sets 3 and 4 evaluate the relationship between generalization and variability of the training sets. The candidates are trained for similar number of epochs using the same structure and size. Networks trained with set 4 are expected to have improved generalization capacity, when compared to the ones trained with set 3, oppositely to the results detailed in rows four and five. These results are not conclusive due to the deficient amount of training time allowed for set 4 network candidates, but show evident differences between networks complexity when trained with each set. Larger data variability would presumably imply also higher network complexity, and therefore longer training time and number of epochs.

Sets 5 to 8 implement the same data variability, but use different combination of input signals. The training is programmed for similar number of epochs in all cases, and therefore the results are expected to benefit simpler training sets. By comparing sets 5 and 7 it can be deduced that the yaw is preferred to θ, and therefore set 6 does not benefit from the additional information. Nevertheless, the result of set 8 do

not corroborate this hypothesis, reason why extra training is programmed with sets 9 and 10. These last sets incorporate maximum data variability, and are trained for a larger number of iterations up to a satisfactory performance. As expected from the preliminary results obtained in sets 5, 6, and 7, the combination of inputs used in sets 5 and 9, Δx, Δy, and Δyaw, is superior to the other candidates. These results were already anticipated by looking into the relationship between coordinates x and y and odometry signals d and θ, which geometrically dependent through (1) and (2). Consequently, it is reasonable to assume that the combination of any of the previous pairs would not provide extra information to the training set. In contrast, yaw proceeds from a three dimensional displacement incorporating new data that seems to the valuable for uncertainty prognosis.

5.4. Delay Effect on the Training Results

The effect of the feedback delay over the estimation accuracy is studied in the LR configuration by implementing: 1 sample delay, 2 samples delay and the combination of both. The results obtained do not vary substantially with incremental delays, which supports the hypothesis of independence between delayed inputs and current outputs. Nevertheless, the output precision benefit in all cases from the noise filtering effect of feedback structures, reason why 1 step delayed is selected.

Figure 8 presents the structure used to find the optimal hidden layer size, where n corresponds to the number of neurons in the hidden layer.

5.5. Hidden Layer Optimal Size

Further training with set 9 using 20, 30, 40, and 50 neurons was programmed to determine the optimal hidden layer size. Networks with 20 and 30 neurons presented unacceptable error measures as they were not able to capture the process complexity. Fifty neurons networks were able to accurately estimate the cumulative uncertainty in most of the trajectories, but presented inconsistent behavior in some cases. The estimation results obtained with 40 and 50 neurons networks are visually compared in **Figures 9**, **10**. These illustrations are formed by three graphs, the trajectory shape at the top level and the uncertainty estimation at the low level, including the uncertainty increment of the left and the accumulation on the right.

Figure 9 illustrates trajectory 0, in which the 50-neurons network returns improved results when compared to the 40-neurons networks. Higher number of neurons are able to filter the noisy inputs more effectively, as illustrated in the left graph, and seem to follow better the uncertainty increment, almost matching the cumulative value at the end of the trajectory. The estimation results of the 40-neurons network also match the increments in uncertainty and the shape of the accumulated error, but it is not able to effectively filter the noise. It could be deduced from the previous results that the more the noise is filtered, the better the estimation accuracy obtained. Nevertheless, this is not the case observed when analyzing trajectory 1 as illustrated in **Figure 10**. Although again 50-neurons networks filter the noise in the uncertainty increments in the left graph, the tendency of the cumulative uncertainty diverges from the target variable causing inconsistent behavior.

TABLE 4 | Comparison of training sets 9 and 11 in terms of set characteristics and testing results in trajectories not used for training.

	Set 9	**Set 11**
Input signals	abs(Δx,Δy,Δyaw)	abs(Δx,Δy,Δyaw)
Training	Train trajectories	Train trajectories 0-5 & Test trajectories 1,2,3, 4,9,14,22 and 24
TRAINING DETAILS		
NN structure	40 neurons LR	40 neurons LR
No.epochs	223	126
Training Performance	$4.67{\cdot}10^{-6}$	$8.20{\cdot}10^{-6}$
AVERAGE TEST RESULTS IN ALL TRAJECTORIES		
$\Delta\sigma_X$ agv.RMS	0.00469	0.004691
σ_X agv.RMS	0.4	0.45
avg.end error	0.54	0.52
AVERAGE TEST RESULTS IN SET 9 TRAINING TRAJECTORIES		
$\Delta\sigma_X$ agv.RMS	0.00445	0.004518
σ_X agv.RMS	0.41	0.35
avg.end error	0.52	0.43
AVERAGE TEST RESULTS IN SET 11 TRAINING TRAJECTORIES		
$\Delta\sigma_X$ agv.RMS	0.004838	0.004798
σ_X agv.RMS	0.4	0.51
avg.end error	0.55	0.58

Oppositely, 40-neurons networks are able to both filter the noise and follow the cumulative uncertainty tendency, returning reasonably accurate results at the end point of the trajectory.

Although 50-neurons network are able to return very accurate results in most of the training trajectories, they show inconsistent behavior at times, which notably diverge from the target. As previously stated, the generalization capability primes in front of the estimation accuracy in the specific application of sensor fusion. Consequently, 40-neurons networks are considered to be the best candidate to model the uncertainty increment, and are therefore considered as reference size in the following tests.

These results agree with the so-called Ockham's Razor principle, which prefers simpler networks structures able to provide acceptable level of accuracy, rather than complex and more accurate ones. The 50-neurons network is able to capture higher non-linear process than the smaller versions. This extra modeling capacity could be either trained to fit the process more accurately, or to capture other processes such as inputs noise, excusing the divergent results depicted in **Figure 10**. Nonetheless, it is not a suitable network size for the characteristics of the available training data.

6. TEST RESULTS

Given the fact that the NN has seen 70% of the data used for testing, although different version of the noisy trajectories are used, these still share similar characteristics which might

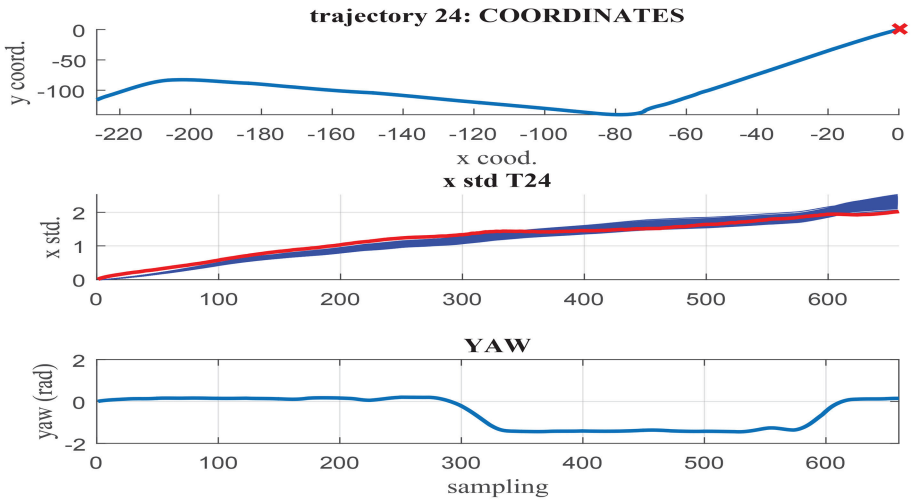


FIGURE 12 | (Middle) σ_x estimation in MC noisy trajectories (blue) respect to path coordinates **(Top)** and yaw **(Bottom)** in trajectory 24.

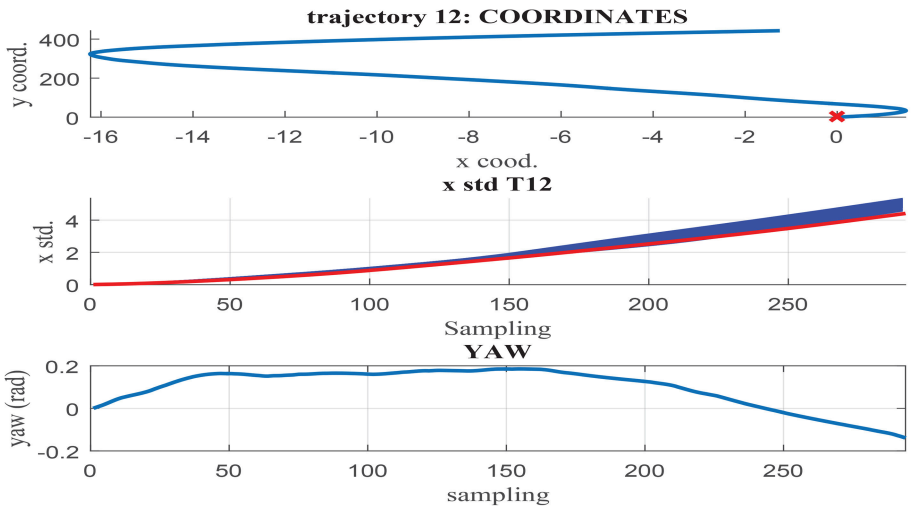


FIGURE 13 | (Middle) σ_x estimation in MC noisy trajectories (blue) respect to path coordinates **(Top)** and yaw **(Bottom)** in trajectory 12.

prevent from yielding final conclusions. Furthermore, NNs can be considered as blackbox models whose robustness cannot be tested with conventional methods. In order to further analyze the candidate performance, 28 new trajectories combining straight lines, sharp turning and winding routes along urban areas are used for testing.

6.1. Set Candidate 9

Likewise to the methodology followed to process the training trajectories, the test trajectories are converted into noisy features emulating data collection through noisy sensors. Again 1000 MC noisy versions are generated to model the uncertainty accumulation along the path. The estimation performance of the 40-neurons network is tested in all 1000 noisy version of each of the 28 trajectories so as to obtain the average error: RMS of $\Delta\sigma_x$, RMS of σ_x, and end error of the cumulative uncertainty. Although the estimation results are in average satisfactory, the network candidate is not able to fit error accumulation with appropriate accuracy in all test cases. Due to the characteristics of blackbox model of the NN, it is difficult to predict in which case scenarios the network will be able to capture the uncertainty growth.

Figures 11–13 illustrate the results obtained in test trajectories 4, 24, and 12, respectively. These contain three graphs including the trajectory coordinates, the cumulative uncertainty and the yaw signal respectively from top to bottom. The middle graph illustrates in red the targeted cumulative uncertainty and 1000 estimation outputs of the network in blue. Although the cumulated uncertainty is well-captured in trajectories 24 and

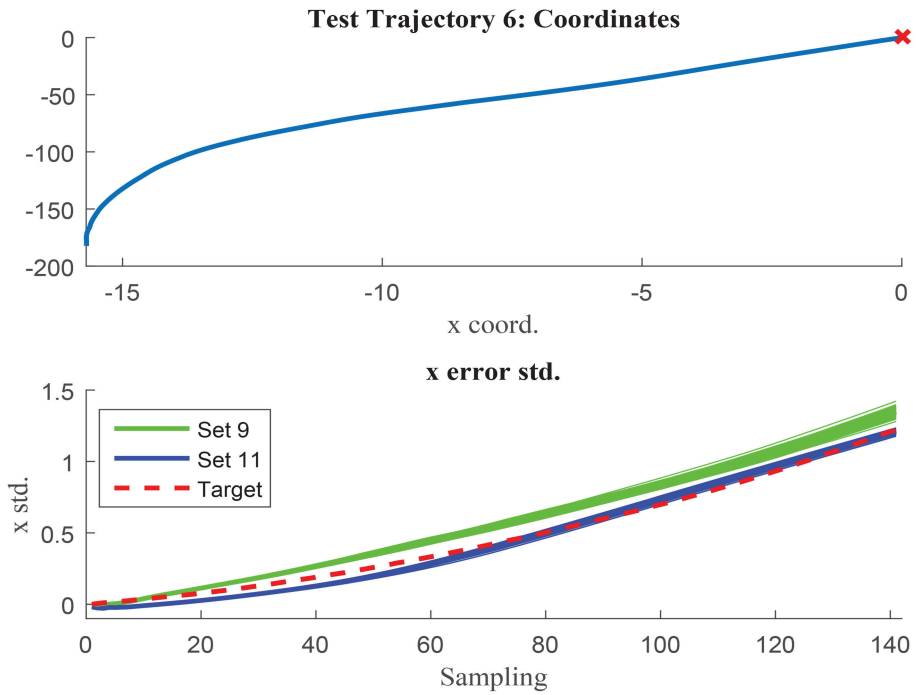


FIGURE 14 | Test trajectory not used in either set 9 or set 11 where set 11 outperforms set 9. Test trajectory 6 comparison of σ_x.

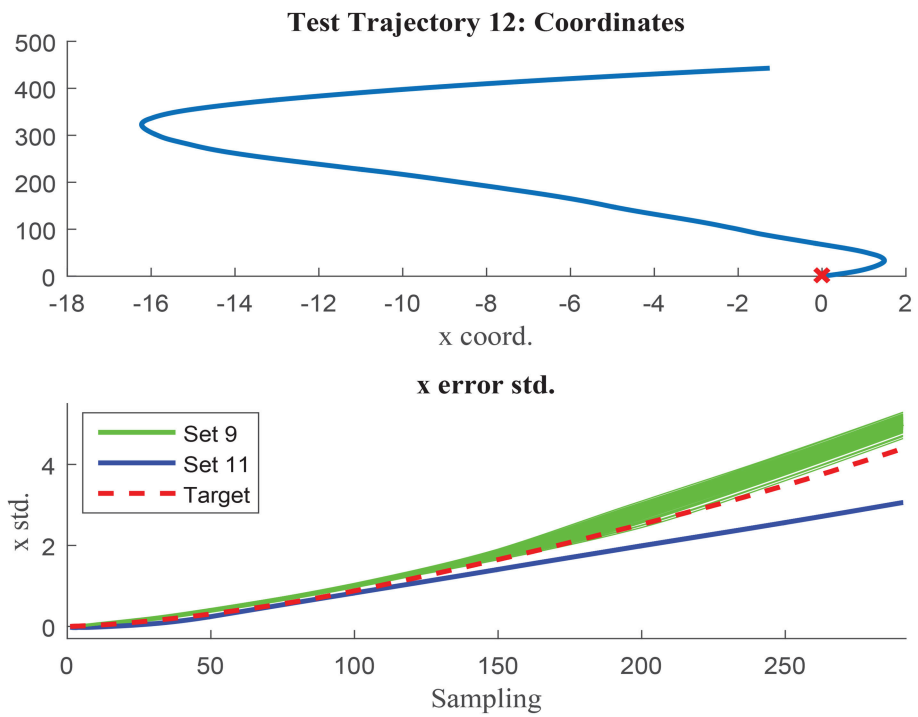


FIGURE 15 | Test trajectory not used in either set 9 or set 11 where set 9 outperforms set 11. Test trajectory 12 comparison of σ_x.

12, this is not the case of trajectory 4. These results can be explained through the values of the *yaw* signal in this test scenarios. The first 200 sampling steps are characterized by close-to-zero yaw, followed by a large increment in yaw and a close-to-constant value until the end of the route. When looking into the tendency of the accumulated uncertainty, the shape seems to match the growth only in the last 200 sampling steps. This behavior is observed in some of the testing trajectories, suggesting a consistent response. Furthermore, when assessing the training trajectories, no scenario with large yaw changes is observed. As a consequence of this deficient training data, the network is not able to capture the uncertainty when these conditions take place in the testing trajectories.

In contrast to testing trajectory 4, the trajectories 24 and 12 do not present zero *yaw* values and maintain it mostly constant along the whole path. As a consequence, the network is able to accurately fit the uncertainty accumulation, with accuracy acceptable to allow for sensor fusion at any point of the route. It can be concluded from the previous that the generalization capacity of the network could be potentially improved provided that the training set includes the test cases missing.

6.2. Set Candidate 11

The deficiencies of training set 9 data variability are corrected in candidate 11. This implements the same input signals, absolute value of Δx, Δy, and Δyaw, but includes a larger amount of trajectories and therefore a larger number of case scenarios that include possible changes in the *yaw* signal, not previously captured. Whilst set 9 only considers data from the training trajectories 0 to 5, set 11 also includes some of the trajectories previously used for testing; test trajectories 1, 2, 3, 4, 9, 14, 22, and 24. **Table 4** contains the details of the sets characteristics, data used, network structure, training and testing results. This table includes the results corresponding to the best network trained with sets 9 and 11. Again, new networks are trained with identical structure and data, and the best is selected to avoid deceiving results caused by random weights initialization. The results included in **Table 4** evaluate the networks performance in all testing trajectories, including the ones also used for training in set 11.

In average, the prediction accuracy of the increment in the uncertainty, $\Delta\sigma_x$, is identical in both cases when analyzing the average accuracy in all test trajectories. This result suggest that the 40 neurons LRNN has reached a performance limit with set 9 and does not admit the further complexity provided in set 11. Moreover, set 9 presents better accuracy when analyzing the average RMS error in σ_x and worst results when comparing the end error.

When looking into the trajectories used to train set 11, as expected the results provided by the network trained with set 9 are worse. In this case set 11 has the advantage of having implemented 70% of those trajectories during training. Nevertheless, the accuracy of the network trained with set 11 is worse when looking exclusively to the trajectories not used in any of the sets. This result could be explained considering the loss of generalization capability, when the training data complexity overcomes the non-linear capacity of the network structure.

The networks results are visually compared in **Figures 14**, **15**, where test trajectories 6 and 12 are illustrated. None of these trajectories have been used to train any of the networks and therefore, the results can be interpreted as pure testing. The uncertainty estimation in the 1000 MC using the network trained with set 9 is represented in green, whilst the respective results of the network trained with set 11 are illustrated in blue. The target curve is illustrated in red in both cases. **Figure 14** shows how set 11 outperforms set 9 prediction results, whilst **Figure 15** illustrates the opposite case. Although the results of both networks are rather similar in terms of accuracy, set 11 prediction in the 1000 MC noisy versions of each trajectory are less spread than the equivalent ones from set 9. The larger variability of data used in set 11 seems to have the effect to improve the prediction robustness to noise, and therefore reduce the variability of the prediction when noisy versions of the same trajectory are used. It can be deduced that the consistency of the results is improved when implementing sets with larger variability due to improved noise robustness.

After identifying the most suitable set for training, the optimized number of neurons was investigated. The previous results suggested that the optimal number of neurons is close to 40, probably situated between 40 and 50 neurons. Therefore, further tests were performed using 38, 42, 45, and 47 neurons. The training was maintained until stabilization of the networks performance and allowing generally higher number of epochs for larger sizes. The results verify the a priori hypothesis and situate the best candidates within 38 and 42 neurons. In particular the 42-neurons candidate improves the RMS error in 14 and 18% the 40 and 45-neurons candidates respectively. The cumulative RMS and end errors are also noticeable improved. The RMS error is improved by 0.0296 and 0.0184 and the end error by 0.161 and 0.457, when compared to the 40 and 45-neurons networks respectively. It can be therefore concluded that the optimal network size is 42-neurons hidden layer.

7. CONCLUSIONS

This paper proposes a strategy for feature uncertainty estimation directly from data without prior knowledge of the sensors characteristics. NNs learning capability of non-linear processes is tested in the particular application of vehicle location through odometry measurements. Both input set and network structure design are based on training and testing results obtained with various neural network candidates. The final results confirm NNs as suitable surrogate modeling technique robust to changes in the testing data, inputs noise and variable case scenarios, provided that the training data captures enough data variability and the network

size and structure complexity is able to resemble the process non-linear characteristics.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61703335, the Fundamental Research Funds for the Central Universities under Grants G2017KY0302, and the 111 Project under Grant B18041.

REFERENCES

Bry, A., Bachrach, A., and Roy, N. (2012). "State estimation for aggressive flight in gps-denied environments using onboard sensing," in *IEEE International Conference on Robotics and Automation* (St. Paul, MN: IEEE), 1–8.

Choi, J. W., and Huhtala, K. (2016). Constrained global path optimization for articulated steering vehicles. *IEEE Trans. Vehicul. Technol.* 65, 1868–1879. doi: 10.1109/TVT.2015.2424933

Durrant-Whyte, H., Stevens, M., and Nettleton, E. (2001). "Data fusion in decentralised sensing networks," in *International Conference on Information Fusion* (Montreal, QC: IEEE), 302–307.

El-Diasty, M., and Pagiatakis, S. (2008). Calibration and stochastic modelling of inertial navigation sensor errors. *J. Global Posit. Syst.* 7, 170–182. doi: 10.5081/jgps.7.2.170

Elfring, J., Appeldoorn, R., van den Dries, S., and Kwakkernaat, M. (2016). Effective world modeling: multisensor data fusion methodology for automated driving. *Sensors* 16:E1668. doi: 10.3390/s16101668

Fabrizi, E., Oriolo, G., Panzieri, S., and Ulivi, G. (2000). "Mobile robot localization via fusion of ultrasonic and inertial sensor data," in *In Proceedings of the 8th International Symposium on Robotics with Applications* (Maui, HI: IEEE).

Forrester, A. I. J., and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Prog. Aerospace Sci.* 45, 50–79. doi: 10.1016/j.paerosci.2008.11.001

Fu, C., Sarabakha, A., K. Kayacan, E., Wagner, C., John, R., and Garibaldi, J. M. (2018). Input uncertainty sensitivity enhanced nonsingleton fuzzy logic controllers for long-term navigation of quadrotor UAVs. *IEEE ASME Trans. Mechatron.* 23, 725–734. doi: 10.1109/TMECH.2018.2810947

Garcia-Ligero, M. J., Hermoso-Carazo, A., and Linares-Perez, J. (2012). Distributed and centralized fusion estimation from multiple sensors with markovian delays. *Appl. Math. Comput.* 219, 2932–2948. doi: 10.1016/j.amc.2012.09.017

Giordano, F., La Rocca, M., and Perna, C. (2014). Input variable selection in neural network models. *Commun. Stat. Theor. Methods* 43, 735–750. doi: 10.1080/03610926.2013.804567

Gorissen, D., Couckuyt, I., Demeester, P., Dhaene, T., and Crombecq, K. (2010). A surrogate modeling and adaptive sampling toolbox for computer based design. *J. Mach. Learn. Res.* 11, 2051–2055. doi: 10.1007/s10846-010-9395-x

Grime, S., and Durrant-Whyte, H. (1994). Data fusion in decentralized sensor networks. *Control Eng. Pract.* 2, 849–863.

Hagan, M. T., Demuth, H. B., Beale, M. H., and De Jesus, O. (2014). *Neural Network Design*. Martin Hagan.

Han, J., Kim, D., Lee, M., and Sunwoo, M. (2012). Enhanced road boundary and obstacle detection using a downward-looking lidar sensor. *IEEE Trans. Vehicul. Technol.* 61, 971–985. doi: 10.1109/TVT.2012.2182785

Hou, Z. S., and Wang, Z. (2013). From model-based control to data-driven control: survey, classification and perspective. *Inform. Sci.* 235, 3–35. doi: 10.1016/j.ins.2012.07.014

Jiang, L., Wang, Y. P., Cai,B., Jian, W., and Wei, S. (2011). "Multi-sensor based vehicle autonomous navigation for vehicle infrastructure integration: Concept and simulation analysis," in *Proc. TMEE* (Changchun: IEEE), 698–702.

Jin, R., Chen, W., and Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Struct. Multidiscipl. Optim.* 23, 1–13. doi: 10.1007/s00158-001-0160-4

Koziel, S., Ciaurri, D. E., and Leifsson, L. (2011). Surrogate-based methods. *Comput. Optim. Methods Algorithms* 356, 33–59. doi: 10.1007/978-3-642-20859-1_3

Lee Rodgers, J., and Nicewander, W. A. (1988). Thirteen ways to look at the correlation coefficient. *Am. Stat.* 42, 59–66.

Li, Q., Chen, L., Li, M., Shaw, S.-L., and Nuchter, A. (2014). A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Trans. Vehicul. Technol.* 63, 540–555. doi: 10.1109/TVT.2013.2281199

Liu, Y., Lim, H.-B. Frazzoli, E., Ji, H., and Lee, V. C. S. (2013). Improving positioning accuracy using gps pseudorange measurements for cooperative vehicular localization. *IEEE Trans. Vehicul. Technol.* 63, 2544–2556. doi: 10.1109/TVT.2013.2296071

Ma, G., Ghasemi, M., and Song, X. (2018). Integrated powertrain energy management and vehicle coordination for multiple connected hybrid electric vehicles. *IEEE Trans. Vehicul. Technol.* 67, 2893–2899. doi: 10.1109/TVT.2017.2780268

Maier, H., and Dandy, G. (1997). Determining inputs for neural network models of multivariate time series. *Comput. Aided Infrastruct. Eng.* 12, 353–368.

Martinez, C. M., Zhang, F., Clarke, D., Hinz, G., and Cao, D. (2017). "Feature uncertainty estimation in sensor fusion applied to autonomous vehicle location," in *International Conference on Information Fusion* (Xi'an: IEEE), 1–7.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D. J., and Barton, D. (2012). Big data. The management revolution. *Harvard Bus. Rev.* 90, 61–67. doi: 10.1007/s11623-013-0105-2

Michalke, T. P., Jebens, A., and Schäfers, L. (2011). "A dynamic approach for ensuring the functional reliability of next-generation driver assistance systems," in *in Proc. ITSC* (Washington, DC: IEEE), 408–415.

Moller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Netw.* 6, 525–533.

Park, J. H., , Shin, Y. D., Bae, J. H., and Baeg, M. H. (2012). Spatial uncertainty model for visual features using a kinect sensor. *Sensors* 12, 8640–8662. doi: 10.3390/s120708640

Poczter, S. L., and Jankovic, L. M. (2014). The google car: driving toward a better future? *J. Bus. Case Stud.* 10, 7–14. doi: 10.19030/jbcs.v10i1.8324

Razavi, S., Tolson, B. A., and Burn, D. H. (2012). Review of surrogate modeling in water resources. *Water Resour. Res.* 48, 7401–7423. doi: 10.1029/2011WR011527

Schrader, D. K., Min, B. C., Matson, E. T., and Dietz, J. E. (2012). "Combining multiple, inexpensive gps receivers to improve accuracy and reliability," in *IEEE Sensors Applications Symposium* (Brescia: IEEE), 1–6.

Sudheer, K. P., Gosain, A. K., and Ramasastri, K. S. (2002). A data-driven algorithm for constructing artificial neural network rainfall-runoff models. *Hydrol. Process.* 16, 1325–1330. doi: 10.1002/hyp.554

Taeihagh, A., and Lim, H. (2019). Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks. *Transp. Rev.* 39, 103–128. doi: 10.1080/01441647.2018.1494640

Xu, L., Wang, L. Y., Yin, G., and Zhang, H. (2014). Communication information structures and contents for enhanced safety of highway vehicle platoons. *IEEE Trans. Vehicul. Technol.* 63, 4206–4220. doi: 10.1109/TVT.2014.2311384

Yu, H., and Wilamowski, B. W. (2011). *Levenberg-Marquardt Training Industrial Electronics Handbook, Vol. 5 Intelligent Systems, 2nd Edn*. CRC Press.

Zhang, F., Simon, C., Chen, G., Buckl, C., and Knoll, A. (2013). "Cumulative error estimation from noisy relative measurement," in *in Proc. ITSC* (Hague: IEEE).

Zheng, P., and McDonald, M. (2003). The effect of sensor errors on the performance of collision warning systems. *Intell. Transp. Syst.* 1, 469–474. doi: 10.1109/ITSC.2003.1251998

Conflict of Interest Statement: CM was employed by company Porsche Engineering Services GmbH, DC was employed by company Cogsense Technologies Limited. All other authors declare no competing interests.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 04 June 2019
doi: 10.3389/fnbot.2019.00031

SVM-Based Classification of sEMG Signals for Upper-Limb Self-Rehabilitation Training

*Siqi Cai[1], Yan Chen[1], Shuangyuan Huang[1], Yan Wu[2], Haiqing Zheng[3], Xin Li[3] and Longhan Xie[1]**

[1] Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, China, [2] A*STAR Institute for Infocomm Research, Singapore, Singapore, [3] The Third Affiliated Hospital, Sun Yat-sen University, Guangzhou, China

Robot-assisted rehabilitation is a growing field that can provide an intensity, quality, and quantity of treatment that exceed therapist-mediated rehabilitation. Several control algorithms have been implemented in rehabilitation robots to develop a patient-cooperative strategy with the capacity to understand the intention of the user and provide suitable rehabilitation training. In this paper, we present an upper-limb motion pattern recognition method using surface electromyography (sEMG) signals with a support vector machine (SVM) to control a rehabilitation robot, ReRobot, which was built to conduct upper-limb rehabilitation training for post-stroke patients. For poststroke rehabilitation training using the ReRobot, the upper-limb motion of the patient's healthy side is first recognized by detecting and processing the sEMG signals; then, the ReRobot assists the impaired arm in conducting mirror rehabilitation therapy. To train and test the SVM model, five healthy subjects participated in the experiments and performed five standard upper-limb motions, including shoulder flexion, abduction, internal rotation, external rotation, and elbow joint flexion. Good accuracy was demonstrated in experimental results from the five healthy subjects. By recognizing the model motion of the healthy side, the rehabilitation robot can provide mirror therapy to the affected side. This method can be used as a control strategy of upper-limb rehabilitation robots for self-rehabilitation training with stroke patients.

Keywords: surface electromyography, support vector machine, rehabilitation robot, upper limb, motion pattern recognition

OPEN ACCESS

Edited by:
Changhong Fu,
Tongji University, China

Reviewed by:
Zhaohui Xia,
Rensselaer Polytechnic Institute, United States
Dongdong Zheng,
National University of Singapore, Singapore
Qinyuan Ren,
Zhejiang University, China

***Correspondence:**
Longhan Xie
melhxie@scut.edu.cn

Received: *12 February 2019*
Accepted: *09 May 2019*
Published: *04 June 2019*

Citation:
Cai S, Chen Y, Huang S, Wu Y, Zheng H, Li X and Xie L (2019) SVM-Based Classification of sEMG Signals for Upper-Limb Self-Rehabilitation Training. Front. Neurorobot. 13:31. doi: 10.3389/fnbot.2019.00031

INTRODUCTION

Stroke is the leading cause of adult disability around the world (Burton et al., 2017), with upper-limb motor impairments being the main factor influencing the quality of life in stroke survivors (Stinear et al., 2017). Repetitive motor training on movement has a notable curative effect on the restoration of arm function in stroke patients, and the patients' degree of recovery is positively influenced by treatment intensity (Steven et al., 2006; Gittler and Davis, 2018). Conventionally, stroke patients usually rehabilitate with the assistance of therapists. However, the involvement of therapists is challenging because rehabilitation training is a time-consuming and labor-intensive process. Many stroke survivors experience upper-limb impairment with few rehabilitation

opportunities due to a lack of rehabilitation therapists. Robot-assisted therapy devices, which can provide the affected arm with high intensity and repetitive treatment, have been increasingly used in rehabilitation training and can potentially enhance upper-limb functional recovery in stroke survivors (Yoo and Kim, 2015; Veerbeek et al., 2017).

Various rehabilitation robotic devices have been developed for upper-limb training in stroke patients. Among them, MIT-Manus (Krebs et al., 1999) was one of the first systems to be developed and can provide stroke survivors with plane movements. Furthermore, MIME (Lum et al., 2006), GENTLE/s (Coote et al., 2008), T-WREX (Domien et al., 2011), and NEREBOT (Stefano et al., 2014) were proposed to permit three-dimensional exercise training for patients with impaired arms.

Different control strategies have been developed and applied to rehabilitation robots for the recovery of the affected arm. Motion parameters of the patient's arm are one of the major inputs in the rehabilitation robot's control system. Many types of mechanical inputs, such as switches (e.g., Aubin et al., 2013; Artz et al., 2015), force sensors (e.g., Diftler et al., 2014), and computer vision (e.g., Taati et al., 2012), have been used as feedback in the controllers of rehabilitation robots. The surface electromyography (sEMG) signal, which is composed of the action potentials from groups of muscle fibers, is one of the major sources of information about neural control and can reflect the degree of activity of the muscles (Yang et al., 2016). During the rehabilitation training of the upper limb, sEMG signals can be captured, interpreted, and used as input for the control algorithms of rehabilitation robots (Rosen et al., 2001; Kiguchi and Hayashi, 2012; Peternel et al., 2016). Considering that rich motor control information and the user's intention can be detected from sEMG signals, the sEMG-based control scheme is one of the most appropriately suited approaches for upper-limb rehabilitation robots (Singh et al., 2014).

However, the sEMG signal is affected by many factors and is not stable, which can lead to low accuracy in recognizing patient motion intentions. Some studies have shown that machine learning techniques can be employed for classifying different tasks and improving the robustness and accuracy of the identification and classification of arm movements through the exploitation of sEMG signals (Lucas et al., 2008; Young et al., 2013; Suberbiola et al., 2015). The SVM algorithm is a well-established technique to learn how to classify new data starting from a collection of classified events and has been widely applied in machine learning problems (Vapnik, 1995; Suykens et al., 2015) and sEMG processing (Song et al., 2007) because of its simplicity and robustness. With the determination of a few additional tuning parameters, SVM solutions are characterized by a convex quadratic optimization problem (Platt, 1999). Considering that the availability and quality of sEMG signals can vary from patient to patient, it is difficult to obtain a large number of training samples. SVM is suitable for solving learning tasks where the number of attributes is large relative to the number of training examples (Suykens et al., 2002).

Aimed at developing a control strategy for upper-limb rehabilitation robots with the capacity to understand the intention of the patients and provide the corresponding rehabilitation training, this paper proposed an sEMG-based control framework based on SVM classifiers for intention identification of the upper limb. The control strategy was applied to the upper-limb rehabilitation robot, ReRobot, to perform the rehabilitative exercise training. The motion of the patient's healthy side is first recognized from the measured and processed sEMG signals, and then the ReRobot assists the affected side in conducting the corresponding rehabilitation therapy. Based on the developed sEMG-based control strategy, self-rehabilitation training in stroke patients can be conducted.

METHODS

Data Collection

Many stroke patients have trouble moving their upper limbs on the affected side, and they must receive much rehabilitation training to recover motion ability (Merletti et al., 1999). Stroke patients often show abnormal shoulder motor ability, so shoulder rehabilitation actions, including shoulder forward flexion, shoulder level adduction, and shoulder level abduction (Chen and Zhou, 2015), should be carried out.

The coordinates were defined where the coronal axis of the patient is the *X*-axis, the sagittal axis is the *Y*-axis, the vertical

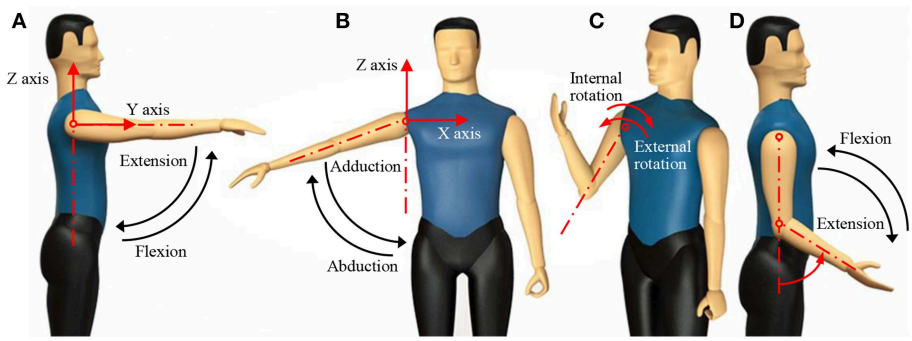


FIGURE 1 | Human upper-limb motions. **(A)** Shoulder flexion; **(B)** shoulder abduction; **(C)** shoulder internal rotation and external rotation; **(D)** elbow flexion.

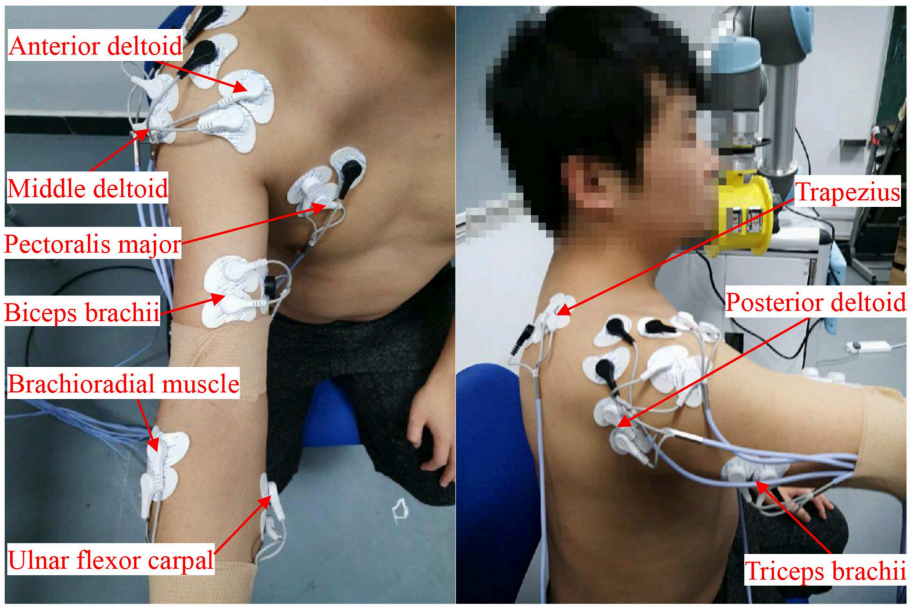


FIGURE 2 | Locations of the sEMG electrodes: middle deltoid, anterior deltoid, pectoralis major, biceps brachii, brachioradial muscle, ulnar flexor carpal, trapezius, posterior deltoid, and triceps brachii. Written informed consent was obtained from the individual in this image for the publication.

axis is the Z-axis, and the acromioclavicular joint is the origin of the coordinates. Five motions were used to test the performance of this model, including shoulder flexion, abduction, internal rotation, external rotation, and elbow joint flexion, as shown in **Figure 1**.

Five healthy subjects ($N = 5$, age 25 ± 4 years, body mass 70 ± 5 kg, height 174 ± 6 cm, all male and all right-handed) participated in the experiments. All subjects gave their informed consent before participation. The experimental procedures were conducted in accordance with the Declaration of Helsinki and approved by the Ethic Board of Medical School, South China University of Technology. Each subject performed five repetitions in accordance with five standard motions. Since stroke patients are mostly elderly people with lower motion abilities on their healthy side compared to young adults, the designed motions were imitated as movements of elderly stroke patients on their healthy side.

For each test, the test subject did not carry weight, and the movement lasted for 1–2 s. After the end of each movement, the subject took at least 1 min to rest to prevent muscle fatigue. There are many muscles involved in the movement of the shoulder and elbow joints. Muscles play two roles in the movement process: proximal stability and distal activity. In this paper, eight superficial muscles involved in the distal shoulder and elbow movements were selected as monitoring objects. To ensure safety, patients controlled the emergency stop of the equipment according to their own comfort level. In this paper, the patient's clenched fist is used as the emergency stop action, and the flexor radialis is used as the detection channel for the emergency stop action. Therefore, a total of nine muscles were selected as test objects. The sEMG signals of nine muscles in the upper limb were acquired, including the middle deltoid, anterior deltoid, pectoralis major, biceps brachii, brachioradial muscle, ulnar flexor carpal, trapezius, posterior deltoid, and triceps brachii. The first eight muscles were used to evaluate how the muscle works while the signal changes. The ulnar flexor carpal played a role in the subject's self-initiation of the safety protection mechanism. When the subject felt uncomfortable during the test, he or she could stop the robot by clenching his or her fist and activating the ulnar flexor.

Figure 2 shows the experimental setup. A 16-channel sEMG acquisition instrument with 1-kHz sampling frequency in each channel was used. Each channel was related to a three-channel differential electrode. After the muscle was disinfected by alcohol, the electrodes were placed along the direction of the muscle abdomen with an interval of 2 cm. **Figure 3** shows the raw sEMG signals of the eight muscles without preprocessing from one of the five healthy individuals.

Data Processing

Preprocessing

sEMG signals are easily disturbed by the external environment in the acquisition process. Motion artifacts, baseline offset, and power frequency interference may all lead to distortion of the sEMG signals, which leads to poor classification accuracy (Chen et al., 2015). Data preprocessing methods of baseline correction, 20–500 Hz bandpass filter, power frequency filter (50 Hz notch), full-wave rectification, and amplitude normalization were carried out to improve the signal-to-noise ratio (SNR) of the sEMG signal. All filters used in this paper are fourth-order Butterworth filters.

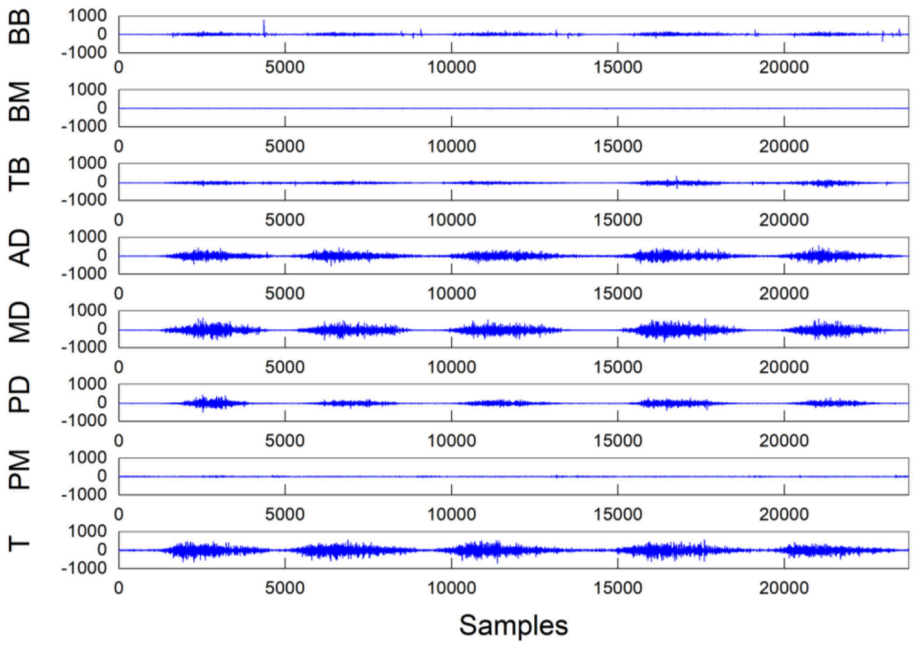


FIGURE 3 | Raw experimental data recorded during a single trial of shoulder abduction from one of the five individuals. The unit of the *Y* coordinate is μV. (T, trapezius; PM, pectoralis major; PD, posterior deltoid; MD, middle deltoid; AD, anterior deltoid; TB, triceps brachii; BM, brachioradial muscle; BB, biceps brachii).

Signal Segmentation

Although the intensity of the sEMG signal detected in each channel is different in the different movements, the signals show good synchronization (Zhang and Zhou, 2012): if the related muscles did not contract, then the sEMG signal showed a stable low-amplitude signal before the test; in contrast, the signal changed dramatically in the course of executing the action. The characteristic of this type of signal was that the signal could be segmented by a sample entropy algorithm (Liu and Zhou, 2013). The sample entropy algorithm is an efficient and time-consuming algorithm that can avoid the signal deviation caused by self-matching. Therefore, we adopted the sample entropy algorithm for data segmentation.

In the experiments, the action signals of eight channels were collected, and the muscle signal for the sample entropy analysis was from the sum of the eight-channel signals, which is:

$$\text{sEMG (t)} = \sum_{i=1}^{M} sEMG_i(t) \tag{1}$$

where M is the total number of channels, $\text{sEMG}_i(t)$ is the *tth* value of channel *i*, and *sEMG* is the sum of all channel signals.

The sample entropy can be calculated as follows:

$$\text{SampEn}\,(m, r, L) = -\ln\left[\frac{B^{m+1}(r)}{B^m(r)}\right] \tag{2}$$

where m is the dimension of the sEMG signal, r is the similar tolerance, L is the length of the muscle signal, and $B^m(r)$ is the probability of the two signal sequences matching m points. In this study, we set $m = 2$ and $r = 0.25^*\sigma$, where σ is the standard deviation of the sEMG signal.

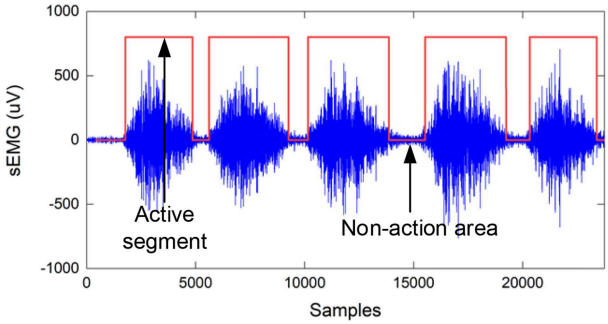


FIGURE 4 | Teacher sample labels based on the sample entropy algorithm, where the label value of the active segment is 1, and the non-action area is 0.

$$s(n) = \begin{cases} 0, & |SampEn < d \\ 1, & |SampEn \geq d \end{cases} \tag{3}$$

where d ($d = 0.6$) is the threshold and $s(n)$ is the judgment function of the EMG signal. When $s(n) = 1$, it is the effective part of the action; when $s(n) = 0$, it is the invalid part of the action, as shown in **Figure 4**.

Feature Extraction and Classification

Because of the short-term stationarity of sEMG signals, the signals need to be divided into frames. To prevent spectrum

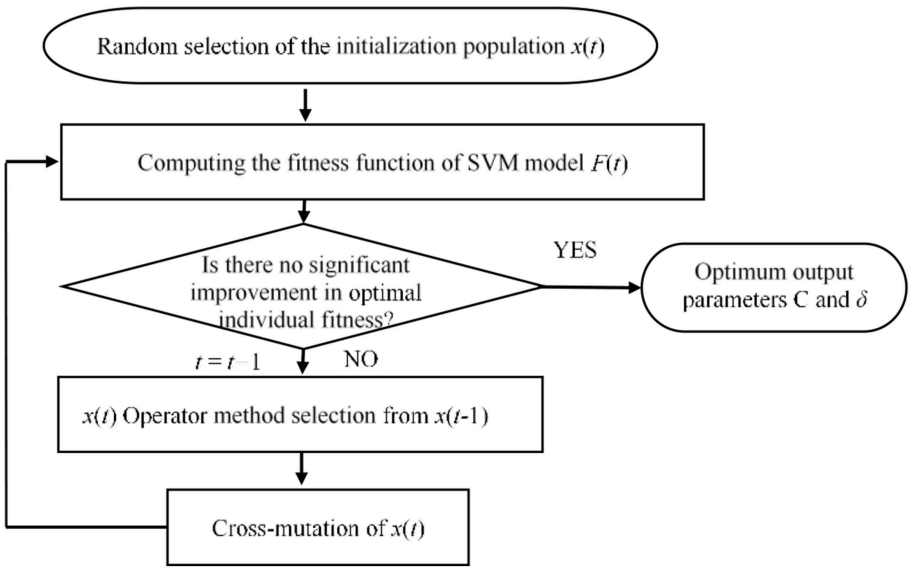


FIGURE 5 | Parameter optimization process based on a genetic algorithm.

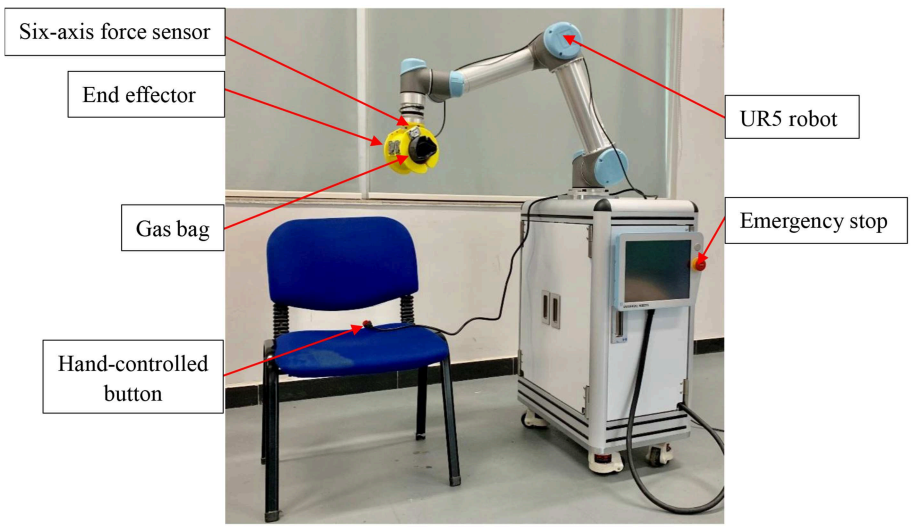


FIGURE 6 | The ReRobot system. The system is set up to support seated rehabilitation training in 3D space, and the UR5 robot is the main system used.

leakage, window functions should be used for interception. Compared with window functions such as the rectangular window and triangular window, the Hanning window has the characteristics of fast side lobe attenuation and is suitable for non-stationary signals. Therefore, this paper adopts a Hanning window for framing. Hanning windows with window lengths ranging from 30 to 300 ms (Chowdhury et al., 2013) were used to extract the characteristics of the sEMG signals. To ensure the implementation of the system and the stability of the classification, 128 ms was selected as the window length, and the sliding step size was 64 ms. The root mean square (RMS), fourth-order autocorrelation factor, wavelength, variance, absolute mean, and short-term energy of each window in each channel are calculated as follows:

$$\mathrm{RMS}_{kj} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(sEMG_{ij})^2} \tag{4}$$

$$\mathrm{VAR}_{kj} = \frac{1}{N}\sum_{i=1}^{N}\left(sEMG_{ij} - \overline{sEMG_j}\right) \tag{5}$$

$$\mathrm{MAV}_{kj} = \frac{1}{N}\sum_{i=1}^{N}\left|sEMG_{ij}\right| \tag{6}$$

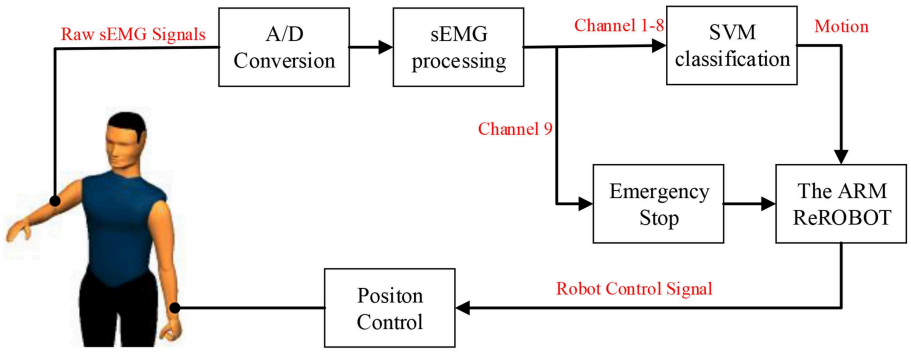


FIGURE 7 | Control structure for robot-assisted exercise training based on the SVM classification of sEMG signals.

TABLE 1 | Classification performance—five types of motions.

Motion	Precision	Recall	F1-score
Shoulder flexion	0.933	0.942	0.938
Shoulder external rotation	0.905	0.858	0.881
Shoulder internal rotation	0.906	0.937	0.921
Shoulder abduction	0.964	0.958	0.961
Elbow joint flexion	0.991	0.991	0.991

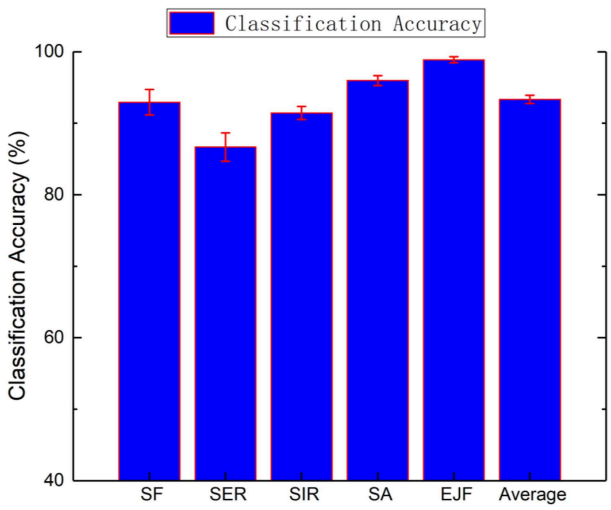


FIGURE 8 | Mean recognition accuracies for the five healthy subjects (SF, shoulder flexion; SER, shoulder external rotation; SIR, shoulder internal rotation; SA, shoulder abduction; and EJF, elbow joint flexion).

$$\mathrm{SSI}_{kj} = \sum_{i=1}^{N} (sEMG_{ij})^2 \tag{7}$$

where k represents the *kth* window and j is the jth channel.

Support Vector Machine

Support vector machine (SVM) is a machine learning method based on statistical learning theory. The characteristic behavior of SVM is to construct a high-dimensional hyperplane for small samples and non-linear models and to classify samples by calculating the maximum distance of training data points on the hyperplane (Ma et al., 2004). Due to the physical limitations of stroke patients, the sample size of the data that can be collected is small. In small-sample model training, SVM has advantages of higher stability and fewer training parameters (Raczko and Zagajewski, 2017). Therefore, SVM is a better choice than a neural network. The equation solved by the SVM algorithm after the Lagrange operator can be expressed as:

$$\begin{cases} \min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi i \\ s.t. y_k(\mathbf{w} \bullet \mathbf{x_k} + b) \geq 1 - \xi_i, k = 1...N \end{cases} \tag{8}$$

where $(\mathbf{x_k}, y_k)$ represents the training data of the *kth* window. ξ_i is a slack variable, which represents the magnitude of the classification error.

The radial basis kernel function can be expressed as (Chung et al., 2003):

$$\kappa(\mathbf{x_i}, \mathbf{x_j}) = \exp\left\{ \frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\delta^2} \right\} \tag{9}$$

The penalty factor C and kernel function parameter δ are the main parameters that affect the performance of the model. Therefore, while training the model, the penalty factor C and kernel function parameter δ should be optimized. To optimize parameters C and δ of the model, the genetic optimization algorithm is used. The accuracy of the time series prediction is selected as the fitness function. The optimization steps of SVM parameters based on the genetic algorithm are shown in **Figure 5**.

To test the feasibility of the genetic optimization algorithm, simulations were carried out using MATLAB. Five healthy subjects ($N = 5$, age 25 ± 4 years, body mass 70 ± 5 kg, height 174 ± 6 cm, all male and all right-handed) were selected for shoulder flexion, abduction, pronation, and elbow flexion. Each action was performed five times for data classification and recognition.

The default value of MATLAB is $C = 1$; $\sigma = 1/\mathrm{num}_{\mathrm{features}}$, where $\mathrm{num}_{\mathrm{features}}$ is the number of features. In this paper,

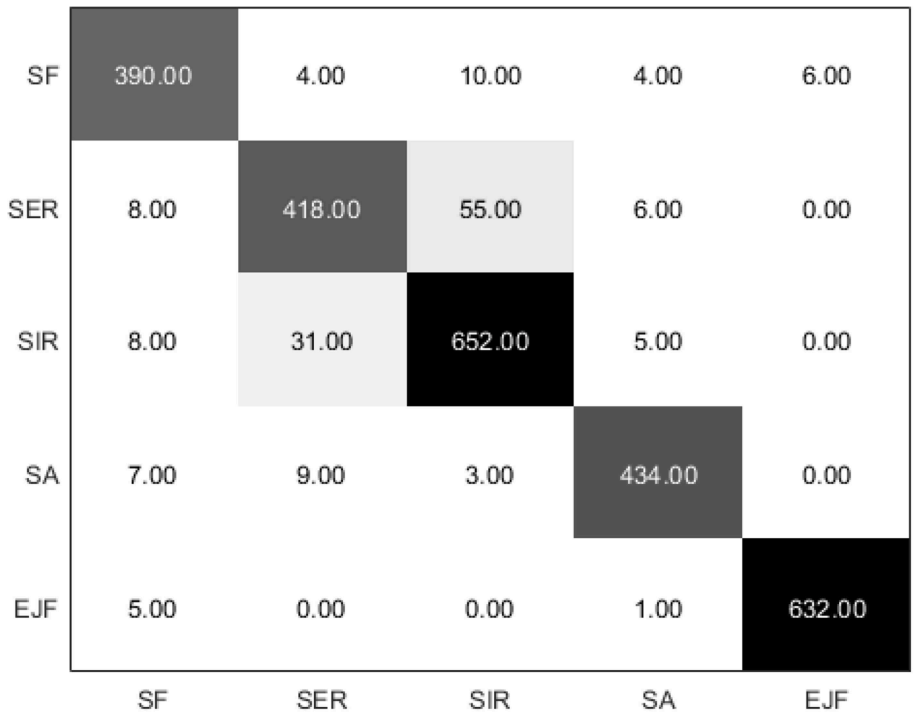


FIGURE 9 | Chaotic matrix of the five healthy subjects (the diagonal value of the matrix is the correct number of classifications, while the non-diagonal value is the wrong number of classifications).

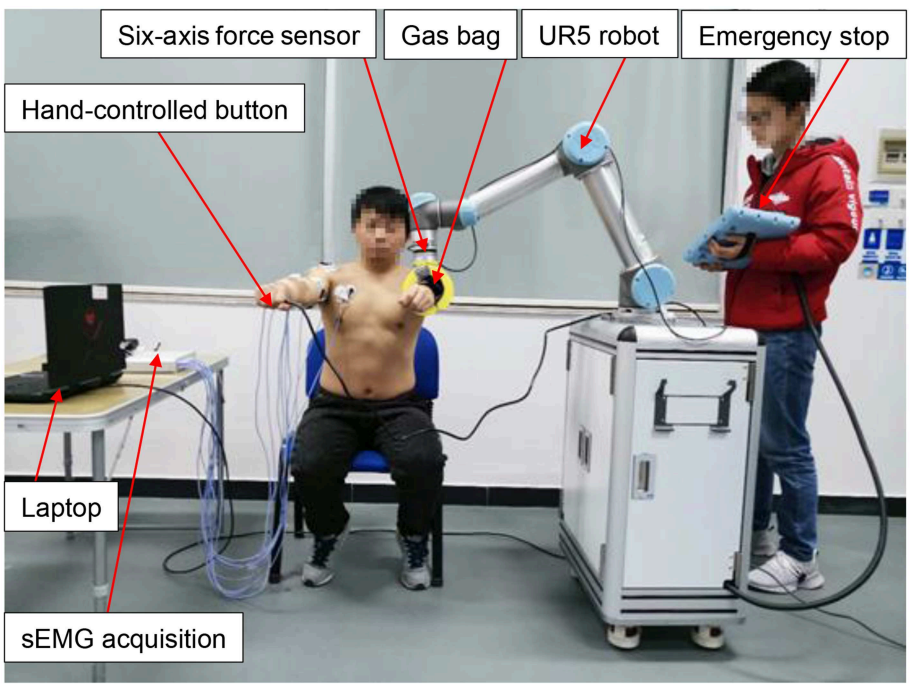


FIGURE 10 | Experiments for sEMG signal acquisition and SVM-based classification during rehabilitation exercises from the healthy side of the subject. Written informed consents were obtained from the individuals in this image for the publication.

$num_{features} = 40$. The value after optimization is $C = 0.4579$ and $\sigma = 371.6339$. The influence of the parameter optimization is significant. Compared with the default value, the optimized value has higher classification accuracy. When the default parameters are used, the classification accuracy is 78.53%. After parameter optimization, the classification accuracy reached 94.18%.

UPPER-LIMB REHABILITATION ROBOT PLATFORM

Robot System

The ReRobot is a rehabilitation robot platform developed to permit training of the upper limb in three-dimensional space, as shown in **Figure 6**. The platform is set up to support and guide the movement of the affected arm using a UR5 robot arm. UR robotic arms are lightweight, fast, easy to program, flexible, and safe robotic arms with 6 DOF (Kebria et al., 2017). The configuration can provide positioning and orientation to a patient's upper limb in the training tasks. The transmission control protocol/internet protocol (TCP/IP) was used to communicate with the robot and the MATLAB user interface. After attaching to the forearm of a stroke patient, ReRobot focuses on the rehabilitation exercise of the shoulder and elbow joints in accordance with the range of movement of the human arm, including shoulder flexion/extension, shoulder abduction/adduction, shoulder internal/external rotation, elbow joint flexion/extension, and forearm supination/pronation. The rehabilitation data, e.g., data that include the sEMG, forces, velocities, and positions, are used for analysis and ensure the safety of subject is collected in real time during the exercises. To ensure a comfortable fixation to the patient's arm, a gas bag is used as the buffer device at the attachment point with the human arm.

Safety is an important issue in the ReRobot therapy system. The safety system consists of several components, including an emergency stop switch, force sensor stops, a hand-controlled switch, and an sEMG signal stop. The emergency stop button, which is held by the experimenter, cuts power to the robot and shuts down all systems. The six-axis force sensor located at the robot arm measures interaction forces generated during the tasks. If the interaction force is abnormal, the power of the system would be automatically cut off. The hand-controlled switch, which is held by the subject, stops the movement of the robot. In the event of any abnormality, the robot stops and the patient's arm can be easily removed from the end of the ReRobot by a freely actuating mechanism and removable ends.

Control Scheme Based on the SVM Classification of sEMG Signals

The control system of the ReRobot system is intended to develop a human–machine interface (HMI) that is able to activate the device as soon as the patient's motion intention is detected. By using the SVM system, the upper-limb movements of the healthy side can be detected and classified automatically through sEMG signals, and the ReRobot will then assist the impaired arm with that movement. The control scheme based on the SVM classification of sEMG signals was thus established, as shown in **Figure 7**.

This platform acquires the test subject's sEMG signals of the upper limb on the healthy side, and the preprocessed EMG signals from one to eight channels are used for upper limb motion recognition using the SVM classification method. The recognized action label signals are sent to the robot for motion calculation, and then the robot actuates the subject's affected arm with the position control method to perform the corresponding actions based on a presupposed trajectory.

When the patient feels discomfort in his or her arm or muscle abnormalities during the test, the whole system can be safely stopped by patient through the first clenching motion, and the sEMG signal from channel 9 is sent to the manipulator. At the same time, there is an emergency stop button in the test subject's hand throughout the test process to ensure safety.

Experiment and Results

To test the feasibility of the upper-limb motion recognition method based on the SVM classification, simulations were carried out using MATLAB. Five movements of five normal people were collected 20 times. After processing feature extractions and label recognition of the collected data, 10 five-fold cross-validations were performed. The results are shown in **Figure 8**. The classification accuracy of each action is as follows: average recognition rate, 93.34 ± 0.59%; shoulder flexion, 92.95 ± 1.78%; shoulder external rotation, 91.44 ± 0.91%; shoulder internal rotation, 86.67 ± 1.98%; shoulder abduction, 95.98 ± 0.70%; and elbow flexion, 98.89 ± 0.42%. The chaotic matrix of one of the classifications is shown in **Figure 9**. The misclassification rate of shoulder internal rotation and shoulder external rotation is high. The main reason is that the muscles involved in the two movements have a high coincidence, so they are easily confused. However, the overall recognition rate is high, so the system can be used in the actual operation of the experimental platform.

Based on the chaotic matrix, three accuracy metrics (precision, recall, and F1-score) can be obtained (Sokolova and Lapalme, 2009). Precision describes the accuracy of the detection. Recall is the detection rate, which refers to how well the target objects are detected without being missed. The F1-score combines the precision and recall and provides a single measure of quality that is easy for end-users to understand. The precision, recall, and F1-score of SVM algorithm in classifying five different motions, including shoulder flexion, shoulder external rotation, shoulder internal rotation, shoulder abduction, and elbow joint flexion, were calculated to evaluate the performance, as shown in **Table 1**. The elbow joint flexion was detected with excellent performance (F1-score = 0.991), followed by shoulder abduction (F1-score = 0.961), shoulder flexion (F1-score = 0.921), and shoulder external rotation (F1-score = 0.881). The SVM-based classifier generally classified well and the average F1-score of five types of motions was 0.9368.

Five subjects ($N = 5$, age 25 ± 4 years, body mass 70 ± 5 kg, height 174 ± 6 cm, all male and all right-handed) participated in the rehabilitation training experiments, and all five subjects were able to complete robot-assisted voluntary exercises, as shown in **Figure 10**. Since these five subjects are all right-handed people, the sEMG signals of their right arms were acquired and processed. Based on the SVM classification method, their motion intentions were analyzed and used as input into the control schema of ReRobot. Thus, ReRobot can understand the desired movement of the subjects and facilitate its execution, thus providing active support to their left arms. A total of 100 actions were performed in the actual test, and 92 actions were correctly identified using the SVM-based method. The robot arm successfully assisted the subject's arm with recognized movements. Multiple security guarantees ensure the safety of the subjects in this process. The experimental results showed that SVM-based classification achieved good accuracy in rehabilitation training.

To verify the effectiveness of the safety switch based on the sEMG signal, a total of 25 tests of safety stops based on sEMG signals were performed. ReRobot had its cut power during each test, which proved the feasibility of using sEMG signals as an emergency stop in the rehabilitation system.

The experimental results showed that SVM-based classification can provide good accuracy in upper-limb motion pattern recognition and enabled patients to choose actions actively for rehabilitation. It is possible to use sEMG signals as an emergency stop button in the upper-limb rehabilitation system to ensure safety.

CONCLUSIONS

In this paper, we investigated the feasibility of SVM classifiers for intention identification of the upper limb from sEMG signals. A new human–machine interface for self-rehabilitation training with stroke patients was developed. The upper-limb rehabilitation robot, ReRobot, could adequately understand the desired upper-limb movement and facilitate its execution, thus providing active support to the impaired arm. Experiments with the ReRobot showed that the SVM classification based on sEMG signals can provide good accuracy in upper-limb motion pattern recognition when a time-dependent multifeature set was used.

In future research, this method to extract upper-limb intention from sEMG signals will be tested by experiments with stroke patients. The application of this classifier to upper-limb rehabilitation robots will be implemented to achieve successful clinical verification.

ETHICS STATEMENT

This study was carried out in accordance with the recommendations of SCUT Research Ethics Guidelines and Researcher's Handbook, Ethic Board of Medical school, South China University of Technology with written informed consent from all subjects. All subjects gave written informed consent in accordance with the Declaration of Helsinki. The protocol was approved by the Ethic Board of Medical school, South China University of Technology.

AUTHOR CONTRIBUTIONS

SC, YC, and LX contributed conception and design of the study. SH carried out the experiments. SC and YC performed the formal analysis and methodology part. SC wrote the first draft of the manuscript. YC, SH, YW, HZ, XL, and LX wrote sections of the manuscript. All authors contributed to manuscript revision, read and approved the submitted version.

FUNDING

This work was supported in part by the National Natural Science Foundation of China (Grant No. 51575188), National Key R&D Program of China (Grant No. 2018YFB1306201), Research Foundation of Guangdong Province (Grant No. 2016A030313492 and 2019A050505001), and Guangzhou Research Foundation (Grant No. 201903010028).

REFERENCES

Artz, E. J., Blank, A. A., and O'malley, M. K. (2015). "Proportional sEMG based robotic assistance in an isolated wrist movement," in *ASME 2015 Dynamic Systems and Control Conference*. V002T27A011 (Columbus, OH).

Aubin, P. M., Sallum, H., Walsh, C., Stirling, L., and Correia, A. (2013). "A pediatric robotic thumb exoskeleton for at-home rehabilitation: the Isolated Orthosis for Thumb Actuation (IOTA)," in *IEEE International Conference on Rehabilitation Robotics* (Seattle, WA). doi: 10.1109/ICORR.2013.6650500

Burton, J. K., Eec, F., Barugh, A. J., Walesby, K. E., Amj, M. L., Shenkin, S. D., et al. (2017). Predicting discharge to institutional long-term care after stroke: a systematic review and metaanalysis. *J. Am. Geriatr. Soc.* 66, 161–169. doi: 10.1111/jgs.15101

Chen, M., and Zhou, P. (2015). A novel framework based on FastICA for high density surface EMG decomposition. *IEEE Trans. Neural. Syst. Rehabil. Eng.* 24, 117–127. doi: 10.1109/TNSRE.2015.2412038

Chen, Y., Yang, Z., and Wang, J. (2015). Eyebrow emotional expression recognition using surface EMG signals. *Neurocomputing* 168, 871–879. doi: 10.1016/j.neucom.2015.05.037

Chowdhury, R. H., Reaz, M. B. I., Ali, M. A. B. M., Bakar, A. A. A., Chellappan, K., and Chang, T. G. (2013). Surface electromyography signal processing and classification techniques. *Sensors* 13, 12431–12466. doi: 10.3390/s130912431

Chung, K. M., Kao, W. C., Sun, C. L., Wang, L. L., and Lin, C. J. (2003). Radius margin bounds for support vector machines with the RBF kernel. *Neural Comput.* 15, 2643–2681. doi: 10.1162/089976603322385108

Coote, S., Murphy, B., Harwin, W., and Stokes, E. (2008). The effect of the GENTLE/s robot-mediated therapy system on arm function after stroke. *Clin. Rehabil.* 22, 395–405. doi: 10.1177/0269215507085060

Diftler, M., Ihrke, C. A., Bridgwater, L. B., Davis, D. R., Linn, D. M., Laske, E. A., et al. (2014). *RoboGlove—A Robonaut Derived Multipurpose Assistive Device*. Hong Kong: NASA National Aeronautics and Space Administration.

Domien, G., Ilse, L., Lore, K., Geert, A., Els, K., and Peter, F. (2011). The Armeo Spring as training tool to improve upper limb functionality in multiple sclerosis: a pilot study. *J. Neuroeng. Rehabil.* 8:5. doi: 10.1186/1743-0003-8-5

Gittler, M., and Davis, A. M. (2018). Guidelines for adult stroke rehabilitation and recovery. *JAMA* 319, 820–821. doi: 10.1001/jama.2017.22036

Kebria, P. M., Al-Wais, S., Abdi, H., and Nahavandi, S. (2017). "Kinematic and dynamic modelling of UR5 manipulator," in *IEEE International Conference on Systems, Man, and Cybernetics* (Budapest). doi: 10.1109/SMC.2016.7844896

Kiguchi, K., and Hayashi, Y. (2012). An EMG-based control for an upper-limb power-assist exoskeleton robot. *IEEE Trans. Syst. Man Cybern. B Cybern.* 42:1064. doi: 10.1109/TSMCB.2012.2185843

Krebs, H. I., Hogan, N., Volpe, B. T., Aisen, M. L., Edelstein, L., and Diels, C. (1999). Overview of clinical trials with MIT-MANUS: A robot-aided neuro-rehabilitation facility. *Technol. Health Care* 7, 419–423.

Liu, J., and Zhou, P. (2013). A novel myoelectric pattern recognition strategy for hand function restoration after incomplete cervical spinal cord injury. *IEEE Trans. Neural. Syst. Rehabil. Eng.* 21, 96–103. doi: 10.1109/TNSRE.2012.2218832

Lucas, M.-F., Gaufriau, A., Pascual, S., Doncarli, C., and Farina, D. (2008). Multi-channel surface EMG classification using support vector machines and signal-based wavelet optimization. *Biomed. Signal Process. Control* 3, 169–174. doi: 10.1016/j.bspc.2007.09.002

Lum, P. S., Burgar, C. G., Van, D. L. M., Shor, P. C., Majmundar, M., and Yap, R. (2006). MIME robotic device for upper-limb neurorehabilitation in subacute stroke subjects: a follow-up study. *J. Rehabil. Res. Dev.* 43, 631–642. doi: 10.1682/JRRD.2005.02.0044

Ma, J., Krishnamurthy, A., and Ahalt, S. C. (2004). *SVM training with duplicated samples and its application in SVM-based ensemble methods*. *Neurocomputing* 61, 455–459. doi: 10.1016/j.neucom.2004.04.004

Merletti, R., Roy, S. H., Kupa, E., Roatta, S., and Granata, A. (1999). Modeling of surface myoelectric signals—Part II: Model-based signal interpretation. *IEEE Trans. Biomed. Eng.* 46, 821–829. doi: 10.1109/10.771191

Peternel, L., Noda, T., Petri, T., Ude, A., Morimoto, J., and Babi, J. (2016). Adaptive control of exoskeleton robots for periodic assistive behaviours based on EMG feedback minimisation. *PLoS ONE* 11:e0148942. doi: 10.1371/journal.pone.0148942

Platt, J. C. (1999). *Fast Training of Support Vector Machines Using Sequential Minimal Optimization*. Cambridge, MA: MIT Press.

Raczko, E., and Zagajewski, B. (2017). Comparison of support vector machine, random forest and neural network classifiers for tree species classification on airborne hyperspectral APEX images. *Eur. J. Remote Sens.* 50, 144–154. doi: 10.1080/22797254.2017.1299557

Rosen, J., Brand, M., Fuchs, M. B., and Arcan, M. (2001). A myosignal-based powered exoskeleton system. *IEEE Trans. Syst. Man Cybern. A Syst. Hum.* 31, 210–222. doi: 10.1109/3468.925661

Singh, R. M., Chatterji, S., andKumar, A. (2014). "A review on surface EMG based control schemes of exoskeleton robot in stroke rehabilitation," in *International Conference on Machine Intelligence and Research Advancement* (Katra: Shri Mata Vaishno Devi Univ). 310–315. doi: 10.1109/ICMIRA.2013.65

Sokolova, M., and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* 45, 427–437. doi: 10.1016/j.ipm.2009.03.002

Song, Q., Sun, B., Lei, J., Gao, Z., Yu, Y., Liu, M., et al. (2007). "Prediction of human elbow torque from EMG using SVM based on AWR information acquisition platform," in *IEEE International Conference on Information Acquisition*. 1274–1278. doi: 10.1109/ICIA.2006.305933

Stefano, M., Mario, A., Gregorio, F., Giulio, R., and Aldo, R. (2014). Randomized trial of a robotic assistive device for the upper extremity during early inpatient stroke rehabilitation. *Neurorehabil. Neural Repair* 28, 377–386. doi: 10.1177/1545968313513073

Steven, L. W., Carolee, J. W., Philip, M., Edward, T., Gitendra, U., David, M., et al. (2006). Effect of constraint-induced movement therapy on upper extremity function 3 to 9 months after stroke: the EXCITE randomized clinical trial. *JAMA* 296, 2095–2104. doi: 10.1001/jama.296.17.2095

Stinear, C. M., Byblow, W. D., Ackerley, S. J., Smith, M. C., Borges, V. M., and Barber, P. A. (2017). Proportional motor recovery after stroke: implications for trial design. *Stroke* 48, 795–798. doi: 10.1161/STROKEAHA.116.016020

Suberbiola, A., Zulueta, E., Lopez-Guede, J. M., and Etxeberria-Agiriano, I. (2015). Arm orthosis/prosthesis movement control based on surface EMG signal extraction. *Int. J. Neural. Syst.* 25, 196–203. doi: 10.1142/S0129065715500094

Suykens, J. A. K., Brabanter, J. D., Lukas, L., and Vandewalle, J. (2002). Weighted least squares support vector machines: robustness and sparse approximation &z.star. *Neurocomputing* 48, 85–105. doi: 10.1016/S0925-2312(01)00644-0

Suykens, J. A. K., Gestel, T. V., Brabanter, J. D., Moor, B. D., and Vandewalle, J. (2015). Least squares support vector machines. *Int. J. Circuit Theory Appl.* 27, 605–615. doi: 10.1002/(SICI)1097-007X(199911/12)27:6<605::AID-CTA86>3.0.CO;2-Z

Taati, B., Wang, R., Huq, R., Snoek, J., and Mihailidis, A. (2012). "Vision-based posture assessment to detect and categorize compensation during robotic rehabilitation therapy," in *IEEE Ras and Embs International Conference on Biomedical Robotics and Biomechatronics* (Rome). 1607–1613. doi: 10.1109/BioRob.2012.6290668

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. New York, NY: Springer.

Veerbeek, J. M., Langbroek-Amersfoort, A. C., Van Wegen, E. E., Meskers, C. G., andKwakkel, G. (2017). Effects of robot-assisted therapy for the upper limb after stroke: a systematic review and meta-analysis. *Neurorehabil. Neural Repair* 31, 107–121. doi: 10.1177/1545968316666957

Yang, Z., Chen, Y., Tang, Z., and Wang, J. (2016). Surface EMG based handgrip force predictions using gene expression programming. *Neurocomputing* 207, 568–579. doi: 10.1016/j.neucom.2016.05.038

Yoo, D. H., and Kim, S. Y. (2015). Effects of upper limb robot-assisted therapy in the rehabilitation of stroke patients. *J. Phys Ther Sci.* 27, 677–679. doi: 10.1589/jpts.27.677

Young, A. J., Smith, L. H., Rouse, E. J., and Hargrove, L. J. (2013). Classification of simultaneous movements using surface EMG pattern recognition. *IEEE Trans. Biomed. Eng.* 60, 1250–1258. doi: 10.1109/TBME.2012.2232293

Zhang, X., and Zhou, P. (2012). Sample entropy analysis of surface EMG for improved muscle activity onset detection against spurious background spikes. *J. Electromyogr. Kinesiol.* 22, 901–907. doi: 10.1016/j.jelekin.2012.06.005

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 28 June 2019
doi: 10.3389/fnbot.2019.00040

Toward a Brain-Inspired System: Deep Recurrent Reinforcement Learning for a Simulated Self-Driving Agent

Jieneng Chen[1]*, *Jingye Chen*[2], *Ruiming Zhang*[1] and *Xiaobin Hu*[3]*

[1] Department of Computer Science, College of Electronics and Information Engineering, Tongji University, Shanghai, China, [2] School of Computer Science, Fudan University, Shanghai, China, [3] Department of Computer Science, Technical University of Munich, Munich, Germany

An effective way to achieve intelligence is to simulate various intelligent behaviors in the human brain. In recent years, bio-inspired learning methods have emerged, and they are different from the classical mathematical programming principle. From the perspective of brain inspiration, reinforcement learning has gained additional interest in solving decision-making tasks as increasing neuroscientific research demonstrates that significant links exist between reinforcement learning and specific neural substrates. Because of the tremendous research that focuses on human brains and reinforcement learning, scientists have investigated how robots can autonomously tackle complex tasks in the form of making a self-driving agent control in a human-like way. In this study, we propose an end-to-end architecture using novel deep-Q-network architecture in conjunction with a recurrence to resolve the problem in the field of simulated self-driving. The main contribution of this study is that we trained the driving agent using a brain-inspired trial-and-error technique, which was in line with the real world situation. Besides, there are three innovations in the proposed learning network: raw screen outputs are the only information which the driving agent can rely on, a weighted layer that enhances the differences of the lengthy episode, and a modified replay mechanism that overcomes the problem of sparsity and accelerates learning. The proposed network was trained and tested under a third-party OpenAI Gym environment. After training for several episodes, the resulting driving agent performed advanced behaviors in the given scene. We hope that in the future, the proposed brain-inspired learning system would inspire practicable self-driving control solutions.

Keywords: self-driving agent, brain-inspired learning, reinforcement learning, end-to-end architecture, recurrence

OPEN ACCESS

Edited by:
Caixia Cai,
*Agency for Science, Technology and Research (A*STAR), Singapore*

Reviewed by:
Keyu Wu,
Nanyang Technological University, Singapore
Jacques Kaiser,
Research Center for Information Technology, Germany

***Correspondence:**
Jieneng Chen
chenjn@tongji.edu.cn
Xiaobin Hu
xiaobin.hu@tum.de

Received: *25 February 2019*
Accepted: *28 May 2019*
Published: *28 June 2019*

Citation:
Chen J, Chen J, Zhang R and Hu X (2019) Toward a Brain-Inspired System: Deep Recurrent Reinforcement Learning for a Simulated Self-Driving Agent. Front. Neurorobot. 13:40. doi: 10.3389/fnbot.2019.00040

1. INTRODUCTION

Recently, research in brain science has gradually received the public's attention. Given the rapid progress in brain imaging technologies and in molecular and cell biology, much progress has been made in understanding the brain at the macroscopic and microscopic levels. Currently, the human brain is the only truly general intelligent system that can cope with different cognitive functions with extremely low energy consumption. Learning from the information processing mechanisms of the brain is clearly the key to building stronger and more efficient machine intelligence

(Poo et al., 2016). In recent years, some bio-inspired intelligent methods have emerged (Marblestone et al., 2016; Gershman and Daw, 2017; Hassabis et al., 2017; Botvinick et al., 2019), and they are clearly different from the classical mathematical programming principle. Bio-inspired intelligence has the advantages of strong robustness and an efficient, well distributed computing mechanism. It is also easy to combine with other methods.

The mammalian brain has multiple learning subsystems. Niv (2009) categorized major learning components into four classes: the neocortex, the hippocampal formation (explicit memory storage system), the cerebellum (adaptive control system), and the basal ganglia (reinforcement learning). Among these learning components, reinforcement learning is particularly attractive to research. Nowadays, converging evidence links reinforcement learning to specific neural substrates, thus assigning them to precise computational roles. Most notably, much evidence suggests that the neuromodulator known as dopamine provides basal ganglia target structures with phasic signals that convey a reward prediction error which can influence learning and action selection, particularly in stimulus-driven habitual instrumental behaviors (Rivest et al., 2005). Hence, many efforts have been made to investigate the capability of bio-inspired reinforcement learning by applying them to artificial intelligence-related tasks (Peters and Schaal, 2008; Mnih et al., 2015; Zhu et al., 2017; Gu et al., 2017).

In recent years, deep reinforcement learning has contributed to many of the spectacular success stories of artificial intelligence (Kober et al., 2013; Henderson et al., 2018). After the initial success of the deep Q network (DQN) (Mnih et al., 2013), a variety of improved models have been published successively. Later on and based on the former discoveries, Mnih et al. (2015) proposed the Nature DQN in 2015 and introduced the replay memory mechanism to break the strong correlations between the samples. Mnih et al. (2016) proposed a deep reinforcement learning approach, in which the parameters of the deep network are updated by multiple asynchronous copies of the agent in the environment. Van Hasselt et al. (2016) suggested the Double DQN to eliminate overestimation; they added a target Q network independent from the current Q network. It was shown to apply to large-scale function approximation (Van Hasselt et al., 2016). Wolf et al. (2017) applied a deep Q network to a driving scenario in a physics simulation based track. Newer techniques included deep deterministic policy gradients and mapping an observation directly to action, both of which could operate over continuous action spaces (Lillicrap et al., 2016). Schaul et al. (2016) suggested prioritized replay, adding priority to replay memory to relieve the sparse reward and slowly converge on the problem (Schaul et al., 2016). Reviewed in Hassabis et al. (2017), experience replay was inspired by theories that seek to understand how the memory system in the mammalian brain might interact, and thus has biological plausibility. In the case of partially observable states, the recurrent neural network (RNN) and long short-term memory (LSTM) have been proven to be effective in processing sequence data (Hochreiter and Schmidhuber, 1997). Hausknecht and Stone (2015) replaced the last fully connected layer in the network with an LSTM layer. They integrated information through time and replicated DQN's performance on standard Atari games and partially observed equivalents featuring flickering game screens. Also, Foerster et al. (2016) proposed to use multi-agent to describe a state distributively. A recent work (Kahn et al., 2018) adopted double Q learning with recurrency and computation graphs to tackle a robotics navigation task. Nevertheless, some previous studies, such as those with Atari games, focused on the simple environment and action space. Moreover, the previous studies do not provide comprehensive comparisons with supervised learning in a specific scenario. Because of these limitations, there is an urgent need to further improve the capability of deep reinforcement learning in a more challenging and complex scenario such as the simulated driving control problem.

To clarify the biological plausibility, Lake et al. (2017) state that there is indeed substantial evidence that the brain uses similar model-free RL learning algorithms in simple associative learning or discrimination learning tasks. In particular, the phasic firing of midbrain dopaminergic neurons is qualitatively and quantitatively consistent with the reward prediction error that drives updating of value estimates. In the process of reinforcement learning, the agent's attempt in each state was like the regulation process of dopamine in the brain (Dolan and Dayan, 2013). To the best of our knowledge, there is rare work studying the behavior difference between supervised learning and RL in a specific scenario. In the kart driving case in this work, the proposed learned agent shows stronger biological plausible learning capability than the supervised learned agent, in respect to dealing with specific situations and its adaptability.

One supervised learning-based study looked at the simulated self-driving game (Ho et al., 2017). However, three problems existed in their implementation. First, they created a handcrafted dataset. Obviously, one can never create this ideal benchmark dataset that includes all the bad situations encountered by the driving agent during training. At best, one can include the best behavior that the driving agent should implement in each step. The driving agent was reported to perform well when it had a good position in the driveway. However, the behavior deteriorated rapidly when the driving agent deviated from the driveway. Such behaviors indicated the dissimilar distribution and instability even though correctional measures were taken on the dataset. Second, they trained and supervised their network in a supervised way. As there are many possible scenarios, manually tackling all possible cases using supervised learning methods will likely yield a more simplistic policy (Shalev-Shwartz et al., 2016). Third, their experiments were built on ideal conditions; for example, they assumed that the brakes were ignored. In our experiments, we take the brakes into consideration. Moreover, to support autonomous capabilities, a robotic driven agent should adopt human driving negotiation skills when braking, taking left and right turns, and pushing ahead in unstructured roadways. It comes naturally that a trial-and-error way of study is more suitable for this simulated self-driving game. Hence, the bio-inspired reinforcement learning method in the study is a more suitable way for the driving agent to learn how to make decisions.

In our study, we proposed a deep recurrent reinforcement learning network to solve simulated self-driving problems.

Rather than creating a handcrafted dataset and training in a supervised way, we adopted a bio-inspired trail-and-error technique for the driving agent to learn how to make decisions. Furthermore, this paper provides three innovations. First, intermediate game parameters were completely abandoned, and the driving agent relied on only raw screen outputs. Second, a weighting layer was introduced in the network architecture to strengthen the intermediate effect. Third, a simple but effective experience recall mechanism was applied to deal with the sparse and lengthy episode.

The rest of this study is organized as follows: section 2 describes deep Q-learning, recurrent reinforcement learning, network architecture, and implementation details. Section 3 verifies experimental results. The conclusion of this study is drawn in section 4.

2. METHODOLOGY

Deep Q-learning is used to help AI agents operate in environments with discrete actions spaces. Based on the knowledge of Deep Q-learning, we proposed a modified DRQN model in order to infer the full state in partially observable environments.

2.1. Deep Q-Learning

Reinforcement learning manages learning policies for an agent interacting in an unknown environment. In each step, an agent observes the current states of the environment, makes decisions according to a policy π, and observes a reward signal r_t (Lample and Chaplot, 2017). Given the current states and a set of available actions, the main aim of the DQN is to approximate the maximum sum of discounted rewards. According to the Bellman equation, it gives the approximating form of Q-values by combining the reward obtained with the current state-action pair and the highest Q-value at the next state s_{t+1}, and the best action $a\prime$:

$$Q(s_t, a_t) \leftarrow r_t + \gamma * \arg\max_{a'} Q\left(s_{t+1}, a\prime\right) \tag{1}$$

We often use the form involving an iterative process:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma * \arg\max_{a'} Q\left(s_{t+1}, a'\right) - Q(s_t, a_t) \right) \tag{2}$$

In the assignments above, α stands for the learning rate and γ stands for the discounted factor.

The agent chooses the action following a ε-greedy exploration policy. The value of ε ranges from 0.0 to 1.0. In order to encourage the agent to explore the environment, the ε was set to 1.0 at first. During the training process, the value decreased gradually as the experience accumulated. Then, the agent could use experience to complete the task.

When we sample a sequence (s_t, a_t, r_t, s_{t+1}) from the replay memory unit, the target value y_t is calculated as:

$$y_t = \begin{cases} r_t & \text{for terminal } s_{t+1} \\ r_t + \gamma \arg\max_{a'} Q\left(s_{t+1}, a'|\theta\right) & \text{for non-terminal } s_{t+1} \end{cases} \tag{3}$$

The network was trained to approximate the expected Q-value, which led to the loss function, with parameters θ in the model:

$$\text{Loss}(\theta) = \sum \left(y_t - Q(s_t, a_t|\theta)\right)^2 \tag{4}$$

2.2. Recurrent Reinforcement Learning

For some special games which are three-dimensional and partially observable, the DQN lacks the ability to solve the problem. In partially observable environments, the agent only receives an observation o_t of the current environment, which is usually insufficient to infer the full state of the system. The real state s_t is the combination of the current observation o_t and an unfixed length of history states. Hence, we adopted the DRQN model on top of the DQN to deal with such conditions (see **Figure 1**). The last fully connected layer was replaced by the LSTM in the DRQN model in order to record former information. **Figure 2** shows the sequential updates in the recurrent network. When updating the DRQN model, a sequence S was randomly sampled from the replay memory unit, and the beginning time step t was also randomly chosen according to the maximum length l. Then the cut sequence $S_{t,t+1,\ldots,l-1,l}$ was sent to the DRQN model. An additional input h_{t-1} standing for the previous information was added to the recurrent model. At the zero time step, h_{t-1} was set to zero. The output of the LSTM $z\left(o_t, h_{t-1}\right)$, which combined the current observation o_t and the history information h_{t-1}, was used to approximate the Q-value $Q\left(o_t, h_{t-1}, a_t\right)$. The history information was updated and passed through the hidden state to the network in the next time step:

$$h_t = LSTM\left(h_{t-1}, o_t\right) \tag{5}$$

2.3. Network Architecture

In the beginning, we used the baseline DRQN model (Lample and Chaplot, 2017) to make an agent perform self-driving. However, we obtained unsatisfied results with the same model. Hence, we have made three improvements in our modified model to make things work better. The whole architecture is shown in **Figure 3**. First, the network was built on top of the NVIDIA's autopilot model (Krizhevsky et al., 2012). To reduce overfitting, the original model was modified by adding several batch normalization layers. We used a four-layer stronger CNN for feature extraction. The input size was resized to 320∗240. The first convolutional layer contained 32 kernels, with a size size of 8∗8 and a stride of 4. The second convolutional layer contained 64 kernels, with a size of 4∗4 and a stride of 4. The third layer contained 128 kernels, with a size of 3∗3 and a stride of 1. The last convolutional layer contained 256 kernels with a size of 7∗7 and a stride of 1. Relu was used as the activated function in the network, and the sizes of the pooling layers were all 2∗2. Second, we abandoned the fully connected layers before the LSTM layer in the original DRQN model and fed the LSTM directly with the

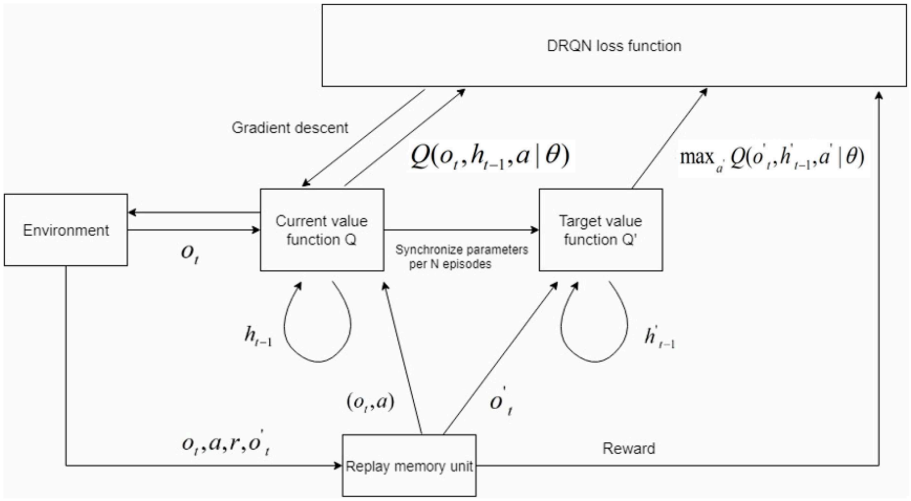


FIGURE 1 | The modified DRQN model. The value function was divided into two categories: the current value function Q and target value function $Q\prime$. The parameters in Q were assigned to Q' per N episodes. The state contained two elements: o_t gained from the current environment and h_{t-1} gained from former information. The agent performed action a using a specific policy, and the sequence $(o_t, a, r, o_t\prime)$ was stored in the replay memory unit. We used a prioritized experience replay memory unit here. During training, the sequence was randomly chosen from the replay memory unit. We trained the network using gradient descent to make the current value function Q approach $Q\prime$ given a specific sequence. The loss function was shown in Equation (4).

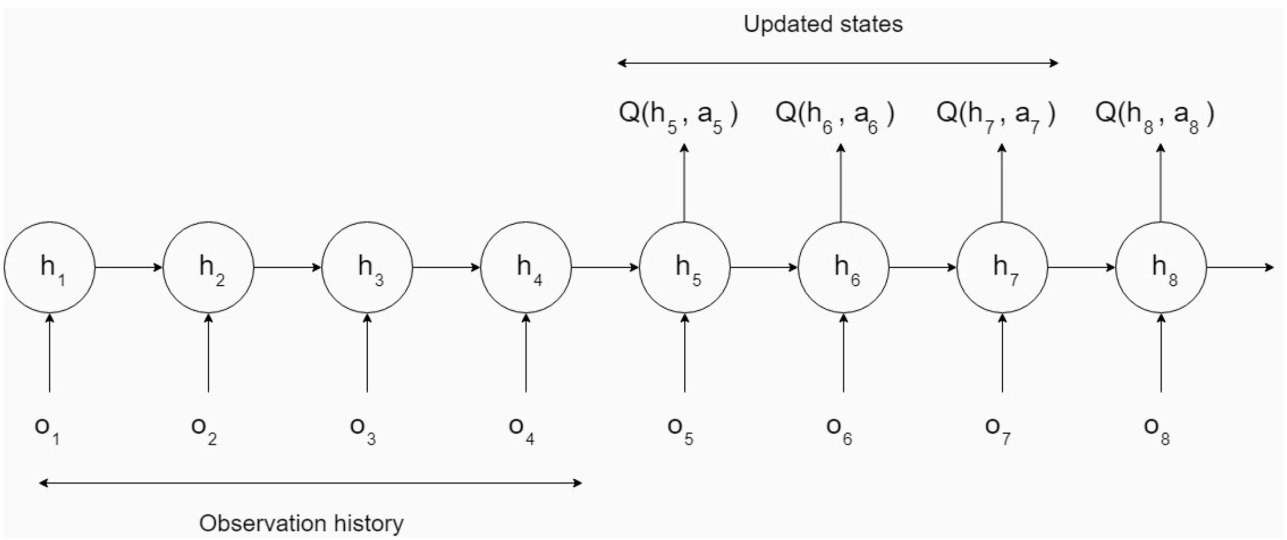


FIGURE 2 | Sequence updates in the recurrent network. Only the scores of the actions taken in states 5, 6, and 7 will be updated. The first four states provide a more accurate hidden state to the LSTM, while the last state provides a target for state 7.

high-level feature. The number of units in LSTM was set to 900. Third, the subsequent structure was divided into two groups for different purposes. One was mapped to the set of possible actions, and the other was a set of scalar values. The final action value function was value function was acquired using both of them. We will introduce their functions respectively.

We used two collateral layers rather than a single DRQN network to approximate the value function. The auxiliary layer also had a five-dimensional output as the action space layer. The original target was to balance the impact of the current and history. The agent could make a suitable action with a fully observing perspective. Nevertheless, we wanted to focus on the precise instantaneous changes of the current scene. The output of the auxiliary layer was mapped to [0, 1.0] using the *softmax* function, thus suggesting the correction for the raw approximations using the DRQN model. Because of this intervention, the network would not only learn the best action for a state but also understand the significance of taking actions. If an agent drove in a straight line and had an obstacle far ahead, an original DRQN model could learn that it was time to make

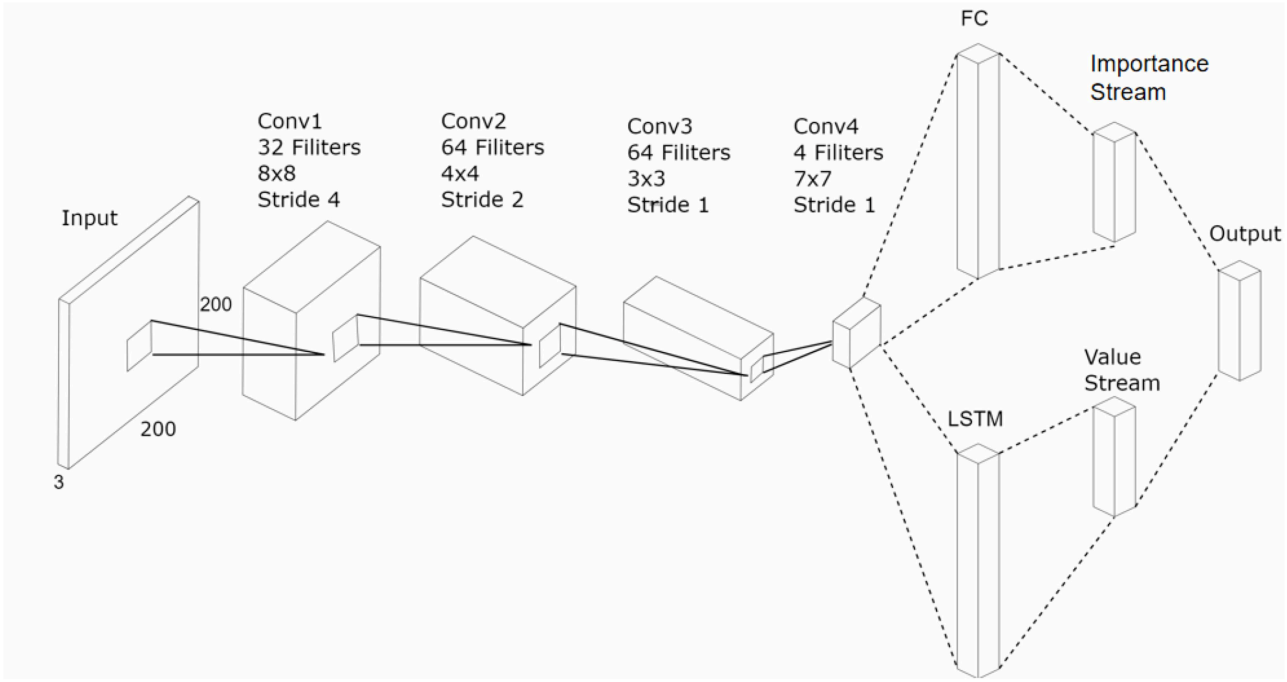


FIGURE 3 | An illustration of the architecture of our model. The input image is assigned to four convolution layers. The output of the convolution layers is split into two streams. The first stream (bottom) flattens the output and feeds it to an LSTM. The second one (top) flattens the output and feeds it to a fully connected layer. Then, we obtained an importance stream and value stream individually and multiplied them as the output. We stored the information in prioritized experience memory unit. As was shown in **Figure 1**, the network was trained using the DRQN loss function.

the driving agent move a bit to avoid hitting the obstacle. The modified model would also have insight into when it was the best time to move, with the knowledge that the danger of the obstacle increases as it gets closer. $V(s, a)$ was used to represent the original output of the DRQN, and $I(s, a)$ was used to represent the importance provided by the auxiliary layer. We used the formula to express the final strengthen of the Q-value (see **Figure 3**):

$$Q(s, a) = V(s, a)^T * I(s, a) \tag{6}$$

The result was stored in the prioritized experience memory unit. During training, the sequence in the memory unit was removed, and we used Equation (4) to calculate the loss.

2.4. Implementation Details

Reinforcement learning consists of two basic concepts: action and reward. Action is what an agent can do in each state. Given that the screen is the input, a robot can take steps within a certain distance. An agent can take finite actions. When a robot takes an action in a state, it receives a reward. Here, the term reward is an abstract concept that describes the feedback from the environment. A reward can be positive or negative. When the reward is positive, it corresponds to our normal meaning of reward. However, when the reward is negative, it corresponds to what we usually call punishment. We also describe the training details such as the hyperparameters, input size selection, frameskip, and prioritized replay.

2.4.1. Action Space

The agent could perform five actions, including Left, Right, Straight, Brake, and Backwards. The range of the joystick reflects the numerical value of the speed control, which is mapped into a region of −80 to 80. The speed section was discretized into a set of [0, 20, 40, 80] for speed control. We considered that only the turning control had a high requirement for precision to keep the model simple. Another three actions, including forward flag, backward flag and brake, were represented by 1/0 flag. Thus, we used 5-dimensional vectors to represent each action as shown below:

$$\begin{aligned} actions &= [40, 0, 1, 0, 0],\ left \\ &= [-40, 0, 1, 0, 0],\ right \\ &= [0, -80, 0, 1, 0],\ go\ backwards \\ &= [0, 0, 1, 0, 0],\ go\ straight \\ &= [0, 0, 0, 0, 1],\ brake \end{aligned}$$

The meaning of each dimension in the vector represented forward speed, backward speed, backward flag, forward flag, and brake. For forward speed, a positive value indicates to the left and a negative value indicates to the right. When the backward flag was set to 1, the agent would move backwards at a specific speed.

2.4.2. Reward

Mnih et al. (2015) states that end-to-end human-level RL control draws on neurobiological evidence that reward signals during perceptual learning may influence the characteristics of representations within the primate visual cortex. The AI trained in a supervised way would only respond to visual information. If the kart picks the wrong direction, it would likely drive straight since the scenes of the correct and wrong direction are mostly the same. In other words, the supervised learned agent does not understand risky situations that are likely to lead into error states during real-time play. In contrast, an agent receiving reward and punishment signals can avoid the noted situation efficiently. Under most circumstances, the driving agent cannot explore a path with big rewards initially. The driving agent often gets stuck somewhere in halfway through and waits for the time to elapse before resetting. We have to make the rewards of these cases variant in order to make these experiences meaningful. Hence, we set a series of checkpoints along the track. A periodical reward was given to the driving agent when each checkpoint was reached. The closer

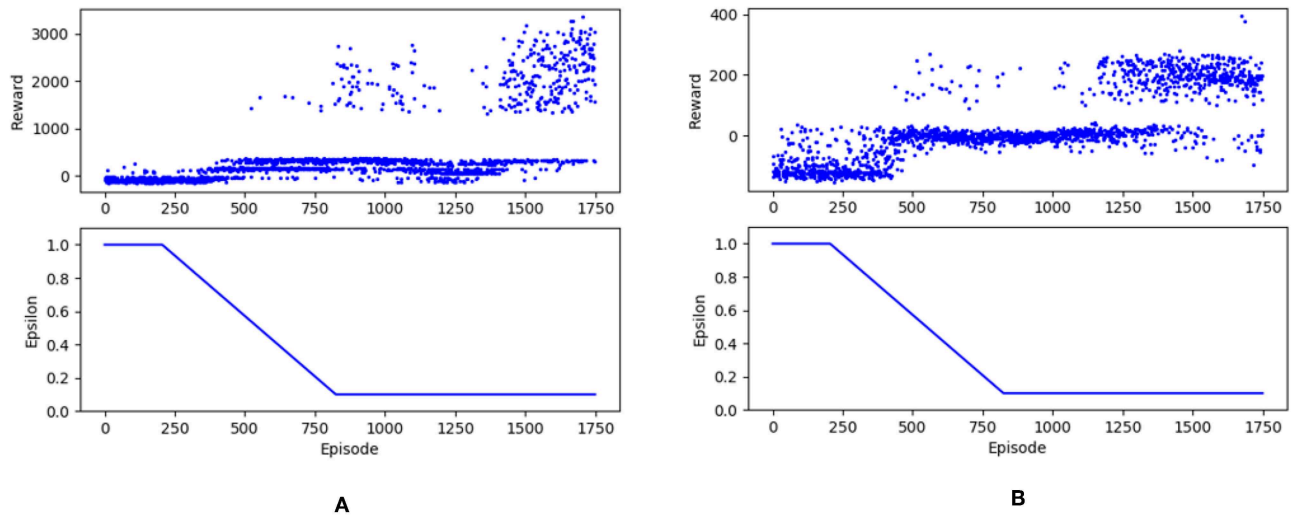


FIGURE 4 | There are reward trends for two different maps. **(A)** corresponded to Farm and **(B)** corresponded to Raceway. After training for 1,750 episodes, we obtained the reward tendency. At about 400 episodes, the stability of the driving agent began to increase. After 1,400 episodes, the reward stabilized at a high level.

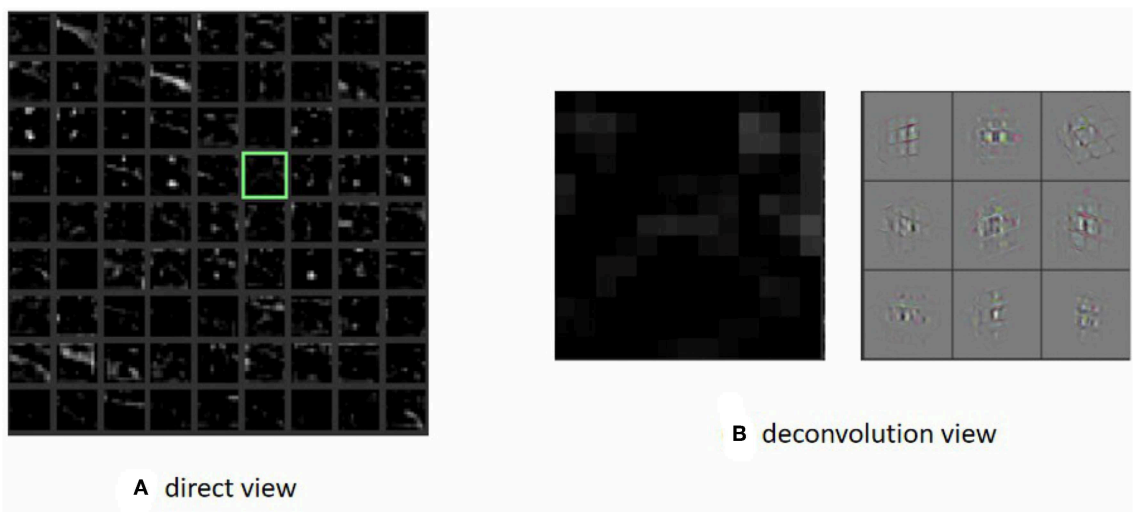


FIGURE 5 | We visualized the first layer and obtain a direct view (left). However, many grids contain unreadable information such as the grid marked with a green frame. Because the first layer's output was quite abstract, we then visualized the last layer using deconvolution and obtained the right picture. It seemed to represent the wall element in the original graph. That means the agent could pay attention to the wall and then take suitable actions. Panel **(A)** shows the direct view visualization of the first layer of CNN. Panel **(B)** shows the visualization of the last layer of CNN using deconvolution.

the distance between the checkpoint and the destination, the bigger the phased reward was given. We have established a more precise reward system to increase density, such as giving the driving agent a slight punishment when it moves backwards. At each step, the agent will also get a little punishment. The detailed component of the reward will be introduced in section 2.4.4.

2.4.3. Prioritized Replay

Hassabis et al. (2017) states that experience replay was directly inspired by theories that seek to understand how the memory systems in the mammalian brain might interact. According to the biological plausibility mentioned in Hassabis et al. (2017) and Schaul et al. (2016), we modified the original replay mechanism. In most cases, there was little replay memory with high rewards, which would be time-consuming with a huge replay table and many sparse rewards. As prioritized replay was a method that can make learning from experience replay more efficient, we simply chose the important experiences in proportion to their rewards and stored them into the replay memory. That way, memories with high rewards would have a greater opportunity to be recalled.

2.4.4. Hyperparameters

The original screen outputs were three channel RGB images. They were first transformed into gray-scale images and then fed to the network to train. The network was trained using the RMSProp algorithm. The size of minibatch was set to 40. The size of replay memory was set to contain 10,000 recent frames. The learning rate α was set to 1.0 in the beginning and followed a linear degradation and finally was fixed at 0.1. The exploration rate ε was set to 0 when we evaluated the model. When the agent finished the game, it would get 1,000 scores as reward. Whenever it crossed each lap, it would get 100 scores. The agent would get 0.5 scores as it got to a checkpoint. If the agent moved backwards, it would get –0.5 scores as punishment. At each time step, it would get –0.1 scores as punishment because we wished the agent to finish the game as quickly as possible.

2.4.5. Other Details

To accelerate training and save running memory, the original 640∗480 screen resolution was resized to 160∗120 in the beginning. After several hours of trials, the driving agent still got stuck in most cases and could not complete one lap. The resulting rewards oscillated for not finishing the game in the limited number of steps, thus indicating the resolution was too low for the model to recognize. To enrich the visual information and address the above problems, the input size was set to 320∗240. We also attempted to reduce the punishments to encourage positive rewards. After the observation of the same length of time, the distribution of the resulting rewards became steady and started to turn positive over the baseline. Hence, the input size of the resolution was finally set to 320∗240 in the experiment in spite of the memory consumption. The system started to learn successfully within the acceptable limit of time.

Since slight change occurs between adjacent frames, we utilized the frame-skip technique (Bellemare et al., 2013). We took out one frame as the network input by every k + 1 frame, and the same action was repeated over the skipped frames. When k became higher, the training speed became high as well. However, the information the agent got became imprecise as well. In order to achieve the balance of low computing resource consumption and smooth control, we finally choose a frame skip of k = 3 by relying on our experience.

3. EXPERIMENTS

The model was trained using three different tracks, which cover all the track that Stanford used for comparison. An individual set of weights was trained separately for each model because each track has different terrain textures. The rewards were low initially because it is equivalent to a random exploration at the beginning of training and because the driving agent would get stuck somewhere without making any significant progress. After about 1,400 episodes of training, the driving agent finished the race. Under most circumstances, the driving agent did not finish the race in given steps so the reward was positive but not as high as receiving the final big reward. We set this step limit because of a lack of a reset mechanism for dead situations, which was very useful in the early stage of training. In Stanford's report, they created a DNF flag to represent the driving agent getting stuck. In our experiment, the driving agent had learned better policy and displayed better behaviors, proving better robustness of the system. We also visualized the CNN in order to validate the ability of the model.

3.1. Experimental Environment

We chose the car racing game to carry out our simulated self-driving experiment. In order to play the car racing game autonomously, we used a third-party OpenAI Gym environment wrapper for the car racing game developed by Bzier[1]. With the assistance of the API, we accessed the game engine directly and ran our code while playing the game frame-by-frame. By means of the API, we can easily get the game information, whether it is screen output intermediate game parameters, such as its location in the small map. Our models proved to efficiently handle the observable gaming environments. To demonstrate the agent's driving status, we include a Youtube link at https://youtu.be/KV-hh8N5x3M.

3.2. Rewards Analysis

For comparison, the model was trained and tested using the same tracks like those used by Stanford in their supervised learning method. **Figure 4** shows the rewards trends for two different maps. From the rewards trends we can observe that at about 400 episodes, the stability of the driving agent began to increase. The reward stabilized at a high level after 1400 episodes, where the agent performed a good driving behavior and finished the tracks well. Each track has different terrain textures and difficulty routes. Therefore, an individual set of weights was

[1]https://github.com/bzier/gym-mupen64plus

TABLE 1 | Performance comparison.

Track	Our model	Stanford model (Ho et al., 2017)	Human
Farm	98.33	97.46	94.07
Raceway	166	129.09, 1 DNF*	125.30
Mountain	213	138.37, 2 DNF*	129.50

For each track, we run 10 races in real time and calculate the mean race times as the final result. The Stanford results were borrowed from the Stanford report. Human results are obtained by real human participants playing each track twice: The first time is to get used to the track, and the second time is to record the time they finish the game. DNF signifies that the autopilot got stuck and was unable to finish some number of races.*

TABLE 2 | System comparison between our work and Stanford's supervised model.

Aspect	Details	Our agent	Stanford's agent
Skills	Turning left/right, driving forward	✓	✓
	Driving backward	✓	
	Brake	✓	
	Turning around	✓	
Bad situation	Driving reverse		✓
	Getting stuck		✓
Performance comparison	Less time-consuming		✓
	Finishing the tracks	✓	✓
Data-consuming	Handcrafted dataset		✓
	Annotation		✓
	Unlabeled data	✓	
Self-adaptability		✓	
Online learning paradigm		✓	

trained separately for each model. The experiments were carried out on a common configured portable laptop, and all models converged after spending over 80 h each.

3.3. CNN Visualization

CNN layers were used to extract abundant information in the scene, and the result of the high-level feature was the critical measurement of the training process. In the traditional supervised learning, the network could be evaluated from many methods such as the validation accuracy and the loss function. However, in the reinforcement learning, we did not have this kind of method to provide a quantitative assessment of the model. Hence, we visualized the output of the first layer and the last layer (see **Figure 5**), with the aim of ascertaining the quantitative features that are captured by the network. The high-level low-level layer output seemed quite abstract through direct observation. Thus, we visualized the high-level features through deconvolution. Through the visualization procedure, we were sure that the network could capture the important element in the scene.

3.4. Result and Discussion

The test results are shown in **Table 1**. The driving agent was tested on several tracks in line with those of Stanford's experiment such as Farm, Raceway, and Mountain. We evaluated our model based on the time. For each track, we ran 10 races in real-time and calculated the mean race times as the final result. The human results (Ho et al., 2017) and the Stanford results were borrowed from the Stanford report. Human results are obtained by real human participants playing each track two times: The first time is to get used to the track, and the second time is to record the time they finish the game.

The proposed model shows some advantages via a comprehensive comparison in **Table 2**. Firstly, some of the actions such as braking and going backwards are important in the driving kart scenario. Stanford's paper reported that the agent is unable to handle situations where the agent may have to turn around or drive backward, and thus would lead to getting stuck. Secondly, their AI is more sensitive to positive visual information. If the kart picks the wrong direction, it would likely drive straight since the visual scenes of the true and wrong direction are mostly the same. In other words, their AI does not understand risky situations that are likely to lead to error states during real-time play. Thirdly, the Stanford model was trained in a handcrafted dataset collected from 18,658 training examples across four tracks, three of which were also used for testing. If they want to generalize their model to other tracks, they need to collect new data and annotations, which might be expensive and unfeasible. Intuitively, our proposed model attempts to learn actions by trial and error without a huge amount of labels and handcrafted datasets.

By analyzing the route the agent runs, we found that the route was not as smooth as that in Stanford's experiment, for which there are two reasons. On one hand, ϵ decayed too fast. As shown in **Figure 5**, the value of ϵ rapidly decreased to 0.1 while the rewards increased. Then, during the latter phase of training, the agent depended mainly on experiences even though there were still many better state-action sets to explore. On the other hand, the actions was discretized roughly. We used a set of [0; 20; 40; 80] as the choices for speed. Through observation, speeds of 20 and 40 both produce a tiny effect while a speed of 80 would make a radical change. More efforts should put on the selection of numerical value of the joystick parameter. However, compared with the experiment done by the Stanford group, our experiments performed well even if the driving agent deviated from the driveway. We considered the brake and trained the driving agent using a trial-and-error method, which was more in line with the real situation. Hence, the bio-inspired reinforcement learning method in the study was a more suitable approach for the driving agent to make decisions.

4. CONCLUSION

Brain-inspired learning has recently gained additional interest in solving control and decision-making tasks. In this paper, we propose an effective brain-inspired end-to-end learning method with the aim of controlling the simulated self-driving agent. Our modified DRQN model has proven to manage

plenty of error states effectively, thus indicating that our trial-and-error method using deep recurrent reinforcement learning could achieve better performance and stability. By using the screen pixels as the only input of the system, our method highly resembles the experience of human beings solving a navigation task from the first-person perspective. This resemblance makes this research inspirational for real-world robotics applications. Hopefully, the proposed brain-inspired learning system will inspire real-world self-driving control solutions.

DATA AVAILABILITY

No datasets were generated or analyzed for this study.

REFERENCES

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: an evaluation platform for general agents. *J. Artif. Intell. Res.* 47, 253–279. doi: 10.1613/jair.3912

Botvinick, M., Ritter, S., Wang, J., and Hassabis, D. (2019). Reinforcement learning, fast and slow. *Trends Cogn. Sci.* 23, 408–422. doi: 10.1016/j.tics.2019.02.006

Dolan, R. J., and Dayan, P. (2013). Goals and habits in the brain. *Neuron* 80, 312–325. doi: 10.1016/j.neuron.2013.09.007

Foerster, J. N., Assael, Y. M., de Freitas, N., and Whiteson, S. (2016). "Learning to communicate to solve riddles with deep distributed recurrent q-networks," in *Advances in Neural Information Processing Systems (NeurIPS)* (Barcelona: Curran Associates, Inc), 2137–2145.

Gershman, S. J., and Daw, N. D. (2017). Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annu. Rev. Psychol.* 68, 101–128. doi: 10.1146/annurev-psych-122414-033625

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (Singapore: IEEE), 3389–3396.

Hassabis, D., Kumaran, D., Summerfield, C., and Botvinick, M. (2017). Neuroscience-inspired artificial intelligence. *Neuron* 95, 245–258. doi: 10.1016/j.neuron.2017.06.011

Hausknecht, M., and Stone, P. (2015). "Deep recurrent q-learning for partially observable mdps," in *AAAI* (Austin, TX).

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). "Deep reinforcement learning that matters," in *AAAI* (New Orleans, LA).

Ho, H., Ramesh, V., and Montano, E. T. (2017). *Neuralkart: A Real-Time Mario Kart 64 AI*. Available online at: http://cs231n.stanford.edu/reports/2017/pdfs/624.pdf (accessed June 20, 2019).

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Kahn, G., Villaflor, A., Ding, B., Abbeel, P., and Levine, S. (2018). "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD: IEEE), 1–8.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: a survey. *Int. J. Robot. Res.* 32, 1238–1274. doi: 10.1177/0278364913495721

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)* (Lake Tahoe, NV), 1097–1105.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. (2017). Building machines that learn and think like people. *Behav. Brain Sci.* 40:e253. doi: 10.1017/S0140525X16001837

Lample, G., and Chaplot, D. S. (2017). "Playing fps games with deep reinforcement learning," in *AAAI* (San Francisco, CA), 2140–2146.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2016). "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)* (San Jun).

Marblestone, A. H., Wayne, G., and Kording, K. P. (2016). Toward an integration of deep learning and neuroscience. *Front. Comput. Neurosci.* 10:94.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., et al. (2016). "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning* (San Juan), 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518:529. doi: 10.1038/nature14236

Niv, Y. (2009). Reinforcement learning in the brain. *J. Math. Psychol.* 53, 139–154. doi: 10.1016/j.jmp.2008.12.005

Peters, J., and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural Netw.* 21, 682–697. doi: 10.1016/j.neunet.2008.02.003

Poo, M.-M., Du, J.-L., Ip, N. Y., Xiong, Z.-Q., Xu, B., and Tan, T. (2016). China brain project: basic neuroscience, brain diseases, and brain-inspired computing. *Neuron* 92, 591–596. doi: 10.1016/j.neuron.2016.10.050

Rivest, F., Bengio, Y., and Kalaska, J. (2005). "Brain inspired reinforcement learning," in *Advances in Neural Information Processing Systems (NeurIPS)* (Vancouver, BC), 1129–1136.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). "Prioritized experience replay," in *International Conference on Learning Representations (ICLR)* (San Jun).

Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double q-learning," in *AAAI*, Vol. 2 (Phoenix, AZ), 5.

Wolf, P., Hubschneider, C., Weber, M., Bauer, A., Härtl, J., Dürr, F., et al. (2017). "Learning how to drive in a real world simulation with deep q-networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)* (Redondo Beach, CA: IEEE), 244–250.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., et al. (2017). "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (Singapore: IEEE), 3357–3364.

AUTHOR CONTRIBUTIONS

JieC, JinC, RZ, and XH carried out the conception and design of the study, the analysis and interpretation of the data, and drafted and revised the article.

FUNDING

This work was financially supported by the German Research Foundation (DFG) and the Technical University of Munich (TUM) in the framework of the Open Access Publishing Program. This research was also funded by the Chinese Ministry of Education's National University Student Innovation and Entrepreneurship Training Program (2018).

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 04 July 2019
doi: 10.3389/fnbot.2019.00047

Deep Recurrent Neural Networks Based Obstacle Avoidance Control for Redundant Manipulators

Zhihao Xu[1], *Xuefeng Zhou*[1] *and Shuai Li*[2]*

[1] Guangdong Key Laboratory of Modern Control Technology, Guangdong Institute of Intelligence Manufacturing, Guangzhou, China, [2] School of Engineering, Swansea University, Swansea, United Kingdom

Obstacle avoidance is an important subject in the control of robot manipulators, but is remains challenging for robots with redundant degrees of freedom, especially when there exist complex physical constraints. In this paper, we propose a novel controller based on deep recurrent neural networks. By abstracting robots and obstacles into critical point sets respectively, the distance between the robot and obstacles can be described in a simpler way, then the obstacle avoidance strategy is established in form of inequality constraints by general class-K functions. Using minimal-velocity-norm (MVN) scheme, the control problem is formulated as a quadratic-programming case under multiple constraints. Then a deep recurrent neural network considering system models is established to solve the QP problem online. Theoretical conduction and numerical simulations show that the controller is capable of avoiding static or dynamic obstacles, while tracking the predefined trajectories under physical constraints.

Keywords: recurrent neural network, redundant manipulator, obstacle avoidance, zeroing neural network, motion plan

OPEN ACCESS

Edited by:
Changhong Fu,
Tongji University, China

Reviewed by:
Haifei Zhu,
Guangdong University of Technology, China
Yongping Pan,
National University of Singapore, Singapore

***Correspondence:**
Shuai Li
shuaili@ieee.org

Received: *15 April 2019*
Accepted: *17 June 2019*
Published: *04 July 2019*

Citation:
Xu Z, Zhou X and Li S (2019) Deep Recurrent Neural Networks Based Obstacle Avoidance Control for Redundant Manipulators. Front. Neurorobot. 13:47. doi: 10.3389/fnbot.2019.00047

1. INTRODUCTION

As industrial automation develops, robot manipulators have been used in a wide range of applications such as painting, welding, assembly, etc., (Cheng et al., 2009; Yang et al., 2018a). With the evolution of intelligent manufacturing, the way robot works is also changing. In order to fulfill more difficult tasks in complex environment, the robot is required to have better execution capabilities (Pan et al., 2018). Therefore, robots with redundant DOFs have attracted much attention in the field of robotic control since its wonderful flexibility (Chan and Dubey, 1995; Zhang, 2015).

Obstacle avoidance is a core problem in the control of redundant manipulators, in order to realize human-machine collaboration and integration, robots no longer work in a separate environment that is completely isolated (Ge and Cui, 2000; Sugie et al., 2003; Lee and Buss, 2007). Instead, collaboration is required between human or other robots, as a result, the obstacle avoidance control is becoming a matter of urgency: robots need to achieve real-time avoidance of static or dynamic obstacles while completing given motion tasks.

Many obstacle avoidance methods for robot manipulators haven been reported, which are designed online or off-line. Based on stochastic sampling algorithm, a series of obstacle avoidance methods are proposed, these methods could obtain effective solutions even in ultra-redundant systems. In Wei and Ren (2018), Wei et al. propose a modified RRT based method, namely Smoothly RRT, in which a maximum curvature constraint is built to obtain a smooth curve when avoiding obstacles, simulation results also show that the method achieves faster convergence than

traditional RRT based ones. In Hsu et al. (2006), Hsu discusses the probabilistic foundations of PRM based methods, a conclusion is drew that the visibility properties rather than dimensionality of C has a greater impact on the probability, and the convergence would be faster if extract partial knowledge could be introduced. However, due to the large computational costs, those methods can be hardly used online.

Different from stochastic results obtained by above mentioned methods, artificial potential field methods plan the same path each time in the same environment, which is important in industrial applications (Khatib, 1986). The basic idea of artificial potential field methods is that the target bears as an attractive pole while the obstacle creates repulsion on the robot, then the robot will be controlled to converge to the target without colliding with obstacles. At the same time, artificial potential field methods have shown great ability in tracking dynamic targets as well as avoiding dynamic obstacles. In Csiszar et al. (2011), a modified method is proposed, which describes the obstacles by different geometrical forms, both theoretical conduction and experimental tests validate the proposed method. Considering the local minimum problem that may be caused by multi-link structures, in Badawy (2016), a two minima is introduced to construct potential field, such that a dual attraction between links enables faster maneuvers comparing with traditional methods. Other improvements to artificial potential field method can be found in Tsai et al. (2001); Tsuji et al. (2002); Wen et al. (2017). Taking advantage of redundant DOFs, obstacles can be avoided by the self-motion in the null space, by calculating pseudo-inverse of Jacobian matrix, the solution can be formulated as the sum of a minimum-norm particular solution and homogeneous solutions (Cao et al., 1999; Moosavian and Papadopoulos, 2007; Krzysztof and Joanna, 2016).

The application of artificial intelligence algorithms based on neural networks provide a new idea for robotic control, these methods are considered to be very promising since its excellent learning ability (Jung and Kim, 2007). For instance, in Pan et al. (2017), the authors propose a command-filtered back-stepping method, in which a neural network based learning scheme is introduced to deal with functional uncertainties. In Pan and Yu (2017), a biomimetic hybrid controller is established, in which the control strategy consist of a feed-forward predictive machine based on a RBF Neural Network and a feedback servo machine based on a proportional-derivative controller. In Fu et al. (2018), a fuzzy logic controller is proposed for long-term navigation of quad-rotor UAV systems with input uncertainties. Experiment results show that the controller can achieve better control performance when compared to their singleton counterparts. In Fu et al. (2019), an online learning mechanism is built for visual tracking systems. The controller uses both positive and negative sample importances as input, and it is shown that the proposed weighted multiple instance learning scheme achieves wonderful tracking performance in challenging environments. Typically, the structure of a neural network may be complex in order to achieve better performance. Although the model of robot manipulator is highly nonlinear, by introducing the priori information of the system model, the neural network can be optimized, i.e., the number of nodes in neural networks can be reduced effectively while maintaining the learning efficiency (Fontaine and Germain, 2001). Inspired by this, a series dynamic neural networks are proposed to realize robotic control in realtime (Zhang et al., 2004; Li et al., 2017; Yang et al., 2018b). Based on the idea of constraint-optimization, quadratic-programming approaches haven been introduced for kinematic control of redundant manipulators. The designed outer-loop controller is described as equality constraints, and objective functions are established to describe certain performance of the system. Using the learning and parallel calculation ability, dynamic neural networks are established to solve the quadratic-programming problem online. The kinematic control is thus achieved by ensuring the equality constraints, and the flexibility is used by optimizing the objective functions. On the other hand, these methods is capable of handling inequality constraints and model uncertainties (Zhang et al., 2018; Li et al., 2019; Xu et al., 2019b). In Cheng et al. (1993), the obstacle avoidance strategy is described as equality constraints, but the parameters of escape velocity is difficult to obtain. In Zhang and Wang (2004), Zhang et al. propose an inequality based method, in which the distance between the robot and obstacles are formulated as a group of distances from critical points and robot links. On this basis, an improved method is proposed by Guo et al. in Guo and Zhang (2012), which is capable of suppressing undesirable discontinuity in the original solutions.

Motivated by the above observations, in this paper, we proposed a novel obstacle avoidance strategy based on deep recurrent neural networks. By abstracting robot and obstacles as a set of critical points, the distances between the robot and obstacles are approximately described by a group of point-to-point distances. And the obstacle avoidance is realized by inequality constraint described by class-K functions. Then the obstacle avoidance problem is reformulated as a QP problem in the speed level, and a deep recurrent neural network is designed to solve the QP online. Numerical results show that the robot is capable of avoiding the obstacles while tracking the predefined trajectories. Before ending this section, the main contributions of this paper are summarized as below

- The proposed deep RNN based controller is able to achieve both path tracking and obstacle avoidance, at the same time, physical constraints such as angular joints and velocities are satisfied.
- In this paper, we propose a class-K function based obstacle avoidance strategy, which has a more general form of description than traditional linear escape velocity methods.
- By abstracting robots and obstacles into critical point sets respectively, the distance between the robot and the obstacle can be described in a simpler way. Besides, numerical results show that the control algorithm can realize the avoidance of static and dynamic obstacles.

2. PROBLEM FORMULATION

2.1. Basic Description

When a redundant robot is controlled to track a particular trajectory in the cartesian space, the positional description of the

end-effector can be formulated as:

$$x = f(\theta), \tag{1}$$

where $x \in \mathbb{R}^m$ and $\theta \in \mathbb{R}^n$ are the end-effector's positional vector and joint angles, respectively. In the velocity level, the kinematic mapping between $\dot{x}$ and $\dot{\theta}$ can be described as:

$$\dot{x} = J(\theta)\dot{\theta}, \tag{2}$$

where $J(\theta) \in \mathbb{R}^{m\times n}$ is the Jacobian matrix from the end-effector to joint space.

In engineering applications, obstacles are inevitable in the workspace of a robot manipulator. For example, robot manipulators usually work in a limited workspace restricted by fences, which are used to isolated robots from humans or other robots. This problem could be even more acute in tasks which requires collaboration of multiple robots. Let C_1 be the set of all the points on a robot body, and C_2 be the set of all the points on the obstacles, then the purpose of obstacle avoidance of a robot manipulator is to ensure $C_1 \cup C_2 = \emptyset$ at all times. By introducing d as a safety distance between the robot and obstacles, the obstacle avoidance is reformulated as

$$|O_jA_i| \geq d, \qquad \forall A_i \in C_1, \forall O_i \in C_2. \tag{3}$$

where $|O_jA_i| = \sqrt{(A_i - O_j)^{\mathrm{T}}(A_i - O_j)}$ is the Euclidean norm of the vector A_iO_j.

Equation (3) gives a basic description of obstacle avoidance problem in form of inequalities. However, there are too many elements in sets C_1 and C_2, the vast majority of which are actually unnecessary. Therefore, by uniformly selecting points of representative significance from C_1 and C_2, and increasing d properly, Equation (3) can be approximately described as below:

$$|O_jA_i| \geq d, \tag{4}$$

with $A_i, i = 1, \ldots, a$ and $O_j, j = 1, \ldots, b$ being the representative points of the robot and obstacles, respectively. The schematic diagram of Equation (4) in shown in **Figure 1**.

Remark. 1 In real implementations, there are many ways to measure $|O_jA_i|$. For instance, since physical structure of the a manipulator is known, the key points A_i are available in advance, both positions and velocities of those points can be calculated directly using the feedback of robot joints. The real-time measurement of obstacles can be achieved through industrial cameras. Therefore, the information of A_i and B_j are all available. As to measurement noise, by introducing a bigger safety distance d, e.g., $d = 1.5(d_1 + d_2)$, the safety can be ensured.

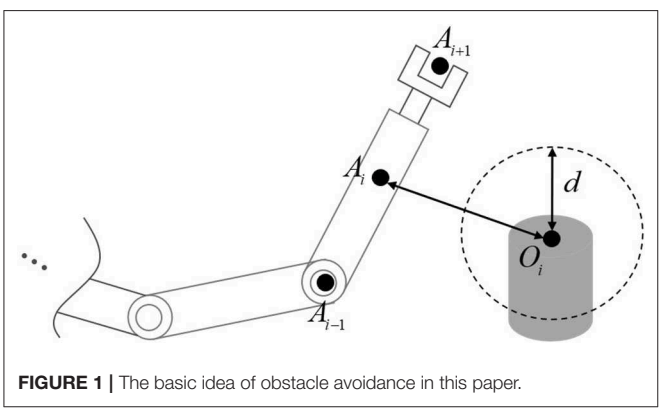


FIGURE 1 | The basic idea of obstacle avoidance in this paper.

2.2. Reformulation of Inequality in Speed Level

In order to guarantee the inequality (4), by defining $D = |O_jA_i| - d$, an inequality is rebuilt in speed level as:

$$\mathrm{d}(|O_jA_i|)/\mathrm{d}t \geq -\mathrm{sgn}(D)g(|D|), \tag{5}$$

in which $g(\bullet)$ belongs to class-K. Remarkable that the velocities of critical points A_i can be described by joint velocities:

$$\dot{A}_i = J_{ai}(\theta)\dot{\theta}, \tag{6}$$

where $J_{ai} \in \mathbb{R}^{m\times n}$ is the Jacobian matrix from the critical point A_i to joint space. If O_j is prior known, the left-side of Equation (5) can be reformulated as:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(|O_jA_i|) &= \frac{\mathrm{d}}{\mathrm{d}t}(\sqrt{(A_i - O_j)^{\mathrm{T}}(A_i - O_j)}) \\ &= \frac{1}{|O_jA_i|}(A_i - O_j)^{\mathrm{T}}(\dot{A}_i - \dot{O}_j) \\ &= \overrightarrow{|O_jA_i|}^{\mathrm{T}} J_{ai}(\theta)\dot{\theta} - \overrightarrow{|O_jA_i|}^{\mathrm{T}}\dot{O}_j,\end{aligned} \tag{7}$$

where $\overrightarrow{|O_jA_i|} = (A_i - O_j)^{\mathrm{T}}/|O_jA_i| \in \mathbb{R}^{1\times m}$ is the unit vector of $\overrightarrow{A_i - O_j}$. Therefore, the collision between critical point A_i and object O_j can be obtained by ensuring the following inequality:

$$J_{oi}\dot{\theta} \leq \mathrm{sgn}(D)g(|D|) - \overrightarrow{|O_jA_i|}^{\mathrm{T}}\dot{O}_j, \tag{8}$$

where $J_{oi} = -\overrightarrow{|O_jA_i|}^{\mathrm{T}} J_{ai} \in \mathbb{R}^{1\times n}$. Based on the inequality description (8), the collision between robot and obstacle can be avoided by ensuring:

$$J_o\dot{\theta} \leq B, \tag{9}$$

where $J_o = [\underbrace{J_{o1}^{\mathrm{T}}, \cdots, J_{o1}^{\mathrm{T}}}_{b}, \cdots, \underbrace{J_{oa}^{\mathrm{T}}, \cdots, J_{oa}^{\mathrm{T}}}_{b}]^{\mathrm{T}} \in \mathbb{R}^{ab\times n}$ is the concatenated form of J_{oi} considering all pairs between A_i and O_j, $B = [B_{11}, \cdots, B_{1b}, \cdots, B_{a1}, \cdots, B_{ab}]^{\mathrm{T}} \in \mathbb{R}^{ab}$ is the vector of upper-bounds, in which $B_{ij} = \mathrm{sgn}(D)g(|D|) - \overrightarrow{|O_jA_i|}^{\mathrm{T}}\dot{O}_j$.

Remark 2: According to 5 the definition of class-K functions, the original escape velocity based obstacle avoidance methods in Zhang and Wang (2004); Guo and Zhang (2012) can be regarded as a special case of 5, in which $G(|D|)$ is selected as $G(|D|) = k|D|$. Therefore, in this paper, the proposed obstacle avoidance strategy is more general than traditional methods.

2.3. QP Type Problem Description

As to redundant manipulators, in order to take full advantage of the redundant DOFs, the robot can be designed to fulfill a secondary task when tracking a desired trajectory. In this paper, the secondary task is set to minimize joint velocity while avoiding obstacles. In real implementations, both joint angles and velocities are limited because of physical limitations such as mechanical constraints and actuator saturation. Because of the fact that rank $(J) < n$, there might be infinity solutions to achieve kinematic control. In this paper, we aim to design a kinematic controller which is capable of avoiding obstacles while tracking a pre-defined trajectory in the cartesian space. For safety's sake, the robot is wished to move at a low speed, on the other hand, lower energy consumption is guaranteed. By defining an objective function scaling joint velocity as $\dot{\theta}^{\mathrm{T}}\dot{\theta}/2$, the tracking control of a redundant manipulator while avoiding obstacles can be described as:

$$\begin{aligned} \min \quad & \dot{\theta}^{\mathrm{T}}\dot{\theta}/2, && (10a)\\ s.t. \quad & x = x_{\mathrm{d}}, && (10b)\\ & \theta^{-} \le \theta \le \theta^{+}, && (10c)\\ & \dot{\theta}^{-} \le \dot{\theta} \le \dot{\theta}^{+}, && (10d)\\ & J_o\dot{\theta} \le B. && (10e) \end{aligned}$$

It is remarkable that the constraints 10b–10e and the objective function 10a to be optimized are built in different levels, which is very difficult to solve directly. Therefore, we will transform the original QP problem (10) in the velocity level. In order to realize precise tracking control to the desired trajectory x_{d}, by introducing a negative feedback in the outer-loop, the equality constraint can be ensured by letting the end-effector moves at a velocity of $\dot{x} = \dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x)$. In terms with (10c), according to escape velocity method, it can be obtained by limiting joint speed as $\alpha(\theta^{-} - \theta) \le \dot{\theta} \le \alpha(\theta^{+} - \theta)$, where α is a positive constant. Combing the kinematic property (2), the reformulated QP problem is:

$$\begin{aligned} \min \quad & \dot{\theta}^{\mathrm{T}}\dot{\theta}/2, && (11a)\\ s.t. \quad & J(\theta)\dot{\theta} = \dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x), && (11b)\\ & \max(\alpha(\theta^{-} - \theta), \dot{\theta}^{-}) \le \dot{\theta} \le \min(\dot{\theta}^{+}, \alpha(\theta^{+} - \theta)), && (11c)\\ & J_o\dot{\theta} \le B. && (11d) \end{aligned}$$

It is noteworthy that both the formula (11a) and (11d) are nonlinear. The QP problem cannot be solved directly by traditional methods. Using the parallel computing and learning ability, a deep RNN will be established later.

3. DEEP RNN BASED SOLVER DESIGN

In this section, a deep RNN is proposed to solve the obstacle avoidance problem (11) online. To ensure the constraints (11b), (11c), and (11d), a group of state variables are introduced in the deep RNN. The stability is also proved by Lyapunov theory.

3.1. Deep RNN Design

Firstly, by defining a group of state variables $\lambda_1 \in \mathbb{R}^m$, $\lambda_2 \in \mathbb{R}^{ab}$, a Lagrange function is selected as:

$$L = \dot{\theta}^{\mathrm{T}}\dot{\theta}/2 + \lambda_1^{\mathrm{T}}(\dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x) - J(\theta)\dot{\theta}) + \lambda_2^{\mathrm{T}}(J_o\dot{\theta} - B), \quad (12)$$

λ_1 and λ_2 are the dual variables corresponding to the constraints (11b) and (11d). According to Karush-Kuhn-Tucker conditions, the optimal solution of the optimization problem (12) can be equivalently formulated as:

$$\dot{\theta} = P_{\Omega}(\dot{\theta} - \frac{\partial L}{\partial \dot{\theta}}), \quad (13a)$$

$$J(\theta)\dot{\theta} = \dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x), \quad (13b)$$

$$\begin{cases} \lambda_2 > 0 & \text{if } J_o\dot{\theta} = B, \\ \lambda_2 = 0 & \text{if } J_o\dot{\theta} \le B, \end{cases} \quad (13c)$$

where $P_{\Omega}(x) = \mathrm{argmin}_{y\in\Omega}||y - x||$ is a projection operator to a restricted interval Ω, which is defined as $\Omega = \{\dot{\theta} \in \mathbb{R}^n | \max(\alpha(\theta^{-} - \theta), \dot{\theta}^{-}) \le \dot{\theta} \le \min(\dot{\theta}^{+}, \alpha(\theta^{+} - \theta))\}$. Notable that Equation (13c) can be simply described as:

$$\lambda_2 = (\lambda_2 + J_o\dot{\theta} - B)^{+}, \quad (14)$$

where $(\bullet)^{+}$ is a projection operation to the non-negative space, in the sense that $y^{+} = \max(y, 0)$.

Although the solution of (13) is exact the optimal solution of the constrained-optimization problem (11), it is still a challenging issue to solve (13) online since its inherent nonlinearity. In this paper, in order to solve (13), a deep recurrent neural network is designed as:

$$\epsilon\ddot{\theta} = -\dot{\theta} + P_{\Omega}(J^{\mathrm{T}}\lambda_1 - J_o^{\mathrm{T}}\lambda_2), \quad (15a)$$

$$\epsilon\dot{\lambda}_1 = \dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x) - J(\theta)\dot{\theta}, \quad (15b)$$

$$\epsilon\dot{\lambda}_2 = -\lambda_2 + (\lambda_2 + J_o\dot{\theta} - B)^{+}, \quad (15c)$$

with ϵ is a positive constant scaling the convergence of (15).

Remark. 3 As to the established deep RNN (15), the first dynamic equation is also the output dynamics, which combines the dynamics of state variables λ_1 and λ_2, as well as the physical limitations including joint angles and velocities. State variable λ_1 is used to ensure the equality constraint (11b), as shown in (15b), λ_1 is updated according to the difference between reference speed $\dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x)$ and actually speed $J(\theta)\dot{\theta}$. Similarly, λ_2 is used to ensure the inequality constraint 11d, which will be further discussed later. It is remarkable that ϵ plays an important role in the convergence of the deep RNN. The smaller ϵ, the faster the deep RNN converges.

Remark. 4 By introducing the model information such as J, J_o, etc., the established deep RNN consists of three class of nodes, namely $\dot{\theta}$, λ_1 and λ_2, and the total number of nodes is $n+m+ab$. Comparing to traditional neural networks in Jung and Kim (2007), the complexity of neural networks is greatly reduced.

3.2. Stability Analysis

In this subsection, we offer stability analysis of the obstacle avoidance method based on deep RNN based. First of all, some basic Lemmas are given as below.

Definition 1: A continuously differentiable function $F(\bullet)$ is said to be monotone, if $\nabla F + \nabla F^{\mathrm{T}}$ is positive semi-definite, where ∇F is the gradient of $F(\bullet)$.

Lemma 1: A dynamic neural network is said to converge to the equilibrium point if it satisfies:

$$\kappa \dot{x} = -x + P_S(x - \varrho F(x)), \tag{16}$$

where $\kappa > 0$ and $\varrho > 0$ are constant parameters, and $P_S = \mathrm{argmin}_{y \in S}||y - x||$ is a projection operator to closed set S.

Lemma 2: (Slotine and Li, 2004) Let $V : [0, \infty) \times B_d \to \mathbb{R}$ be a C^1 function, α_1, α_2 be class-K functions defined on $[0, d)$ which satisfy $\alpha_1(||x||) \leq V(t, x) \leq \alpha_2(||x||), \forall (t, x) \in [0, d) \times B_d$, then $x = 0$ is a uniformly asymptotically stable equilibrium point if there exist some class-K function g defined on $[0, d)$, to make the following inequality hold:

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t, x) \leq -\alpha(||x||), \forall (t, x) \in [0, \infty) \times B_d, \tag{17}$$

About the stability of the closed-loop system, we offer the following theorem.

Theorem 1: Given the obstacle avoidance problem for a redundant manipulator in kinematic control tasks, the proposed deep recurrent neural network is stable and will globally converge to the optimal solution of (10).

Proof: The stability analysis consists of two parts: firstly, we will show that the equilibrium of the deep RNN is exactly the optimal solution of the control objective described in (11), which prove that the output of deep RNN will realize obstacle avoidance while tracking a given trajectory, and then we will prove that the deep recurrent neural network is stable in sense of Lyapunov.

Part I. As to the deep recurrent neural network (15), let $(\dot{\theta}^*, \lambda_1^*, \lambda_2^*)$ be the equilibrium of the deep RNN, then $(\dot{\theta}^*, \lambda_1^*, \lambda_2^*)$ satisfies:

$$-\dot{\theta}^* + P_\Omega(J^{\mathrm{T}}\lambda_1^* - J_o^{\mathrm{T}}\lambda_2^*) = 0, \tag{18a}$$

$$\dot{x}_{\mathrm{d}} + k(x_{\mathrm{d}} - x) - J(\theta)\dot{\theta}^* = 0, \tag{18b}$$

$$-\lambda_2^* + (\lambda_2^* + J_o\dot{\theta}^* - B)^+ = 0, \tag{18c}$$

with $\dot{\theta}^*$ be the output of deep RNN. By comparing Equation (18) and (13), we can readily obtain that they are identical, i.e., the equilibrium point satisfies the KKT condition of (10).

Then we will show that the equilibrium point(output of the proposed deep RNN) is capable of dealing with kinematic tracking as well as obstacle avoidance problem. Define a Lyapunov function V about the tracking error $e = x_{\mathrm{d}} - x$ as $V = e^{\mathrm{T}}e/2$, by differentiating V with respect to time and combining (11b), we have:

$$\begin{aligned} \dot{V} &= e^{\mathrm{T}}\dot{e} = e^{\mathrm{T}}(\dot{x}_{\mathrm{d}} - J(\theta)\dot{\theta}^*) \\ &= -ke^{\mathrm{T}}e \leq 0, \end{aligned} \tag{19}$$

in which the dynamic Equation 18b is also used. It can readily obtained that the tracking error would eventually converge to 0.

It is notable that the dynamic (Equation 18c) satisfies:

$$\lambda_2^* + J_o\dot{\theta}^* - B - (\lambda_2^* + J_o\dot{\theta}^* - B)^+ = J_o\dot{\theta}^* - B. \tag{20}$$

According to the property of projection operator $(\bullet)^+, y - (y)^+ \leq 0$ holds for any y, then we have $J_o\dot{\theta}^* - B \leq 0$, together with (5), the inequality (5) is satisfied. Notable that (5) can be rewritten as:

$$\dot{D} \geq -\mathrm{sgn}(D)g(|D|). \tag{21}$$

As to (21), we first consider the situation when equality holds. Since $g(|D|)$ is a function belonging to class K, it can be easily obtained that $D = 0$ is the only equilibrium of $\dot{D} = -\mathrm{sgn}(D)g(|D|)$. Define a Lyapunov function as $V_2(t, D) = D^2/2$, and select two functions as $\alpha_1(|D|) = \alpha_2(|D|) = D^2/2$. It is obvious that $\alpha_1(|D|) = \alpha_2(|D|)$ belong to class-K, and the following inequality will always hold:

$$\alpha_1(|D|) \leq V_2(t, D) \leq \alpha_2(|D|). \tag{22}$$

Taking the time derivative of $V_2(t, D)$, we have:

$$\frac{\partial V_2}{\partial \mathrm{t}} + \frac{\partial V}{\partial D}\dot{D} = -|D|g(|D|) \leq 0. \tag{23}$$

According to Lemma 2, the equilibrium $x = 0$ is uniformly asymptotically stable. Then we arrive at the conclusion that if the equality $\mathrm{d}(|O_jA_i|)/\mathrm{d}t = -\mathrm{sgn}(D)g(|D|)$ holds, $|D| = 0$ will be guaranteed, i.e., $|O_jA_i| - d$ for all $i = 1 \cdots a, = 1 \cdots b$. Based on comparison principle, we can readily obtain that $|O_jA_i| \geq d$ when $\mathrm{d}(|O_jA_i|)/\mathrm{d}t \geq -\mathrm{sgn}(D)g(|D|)$.

Part II. Then we will show the stability of the deep RNN (15). Let $\xi = [\dot{\theta}^{\mathrm{T}}, \lambda_1^{\mathrm{T}}, \lambda_2^{\mathrm{T}}]^{\mathrm{T}}$ be the a concatenated vector of state variables of the proposed deep RNN, then (15) can be rewritten as:

$$\epsilon\dot{\xi} = -\xi + P_{\bar{\Omega}}[\xi - F(\xi)], \tag{24}$$

where $P_S(\bullet)$ is a projection operator to a set S, and $F(\xi) = [F_1(\xi), F_2(\xi), F_3(\xi)]^{\mathrm{T}} \in \mathbb{R}^{n+m+ab}$, in which:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} \dot{\theta} - J^{\mathrm{T}}\lambda_1 + J_o^{\mathrm{T}}\lambda_2 \\ J\dot{\theta} - \dot{x}_{\mathrm{d}} - k(x_{\mathrm{d}} - x) \\ -J_o\dot{\theta}^* - B \end{bmatrix}.$$

Let $\nabla F = \partial F/\partial \xi$, we have:

$$\nabla F(\xi) = \begin{bmatrix} I & -J^{\mathrm{T}} & J_o^{\mathrm{T}} \\ J & 0 & 0 \\ -J_o^{\mathrm{T}} & 0 & 0 \end{bmatrix}. \tag{25}$$

According to the definition of monotone function, we can readily obtain that $F(\xi)$ is monotone. From the description of (24), the projection operator P_S can be formulated as $P_S = [P_\Omega; P_R; P_\Lambda]$, in which P_Ω is defined in (13), P_R can be regarded as a projection

operator of λ_1 to R, with the upper and lower bounds being $\pm\infty$, and $P_\Lambda = (\bullet)^+$ is a special projection operator to closed set $\mathbb{R}_+^{ab}$. Therefore, P_S is a projection operator to closed set $[\Omega; \mathbb{R}^m; \mathbb{R}_+^{ab}]$. Based on Lemma 1, the proposed neural network (15) is stable and will globally converge to the optimal solution of (10). The proof is completed.

4. NUMERICAL RESULTS

In this section, the proposed deep RNN based controller is applied on a planar 4-DOF robot. Firstly, a basic case where the obstacle is described as a single point is discussed, and then the controller is expanded to multiple obstacles and dynamic ones. Comparisons with existing methods are also listed to indicate the superiority of the deep RNN based scheme.

4.1. Simulation Setup

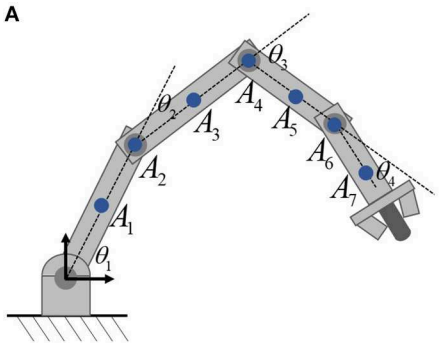
The physical structure of the 4-link planar robot to be simulated in shown in **Figure 2**, in which the critical points of the robot are also marked. As shown in **Figure 2A**, critical points A_2, A_4, A_6 are selected at the joint centers, and A_1, A_3, A_5, A_7 are selected at the center of robot links. The D-H parameters are given in **Figure 2B**. It is notable that A_i and the Jacobian matrix J_oi are essential in the proposed control scheme. Based on the above description of A_i, the D-H parameters of A_1 is $a_1 = 0.15$, $a_2 = a_3 = 0$, $\alpha_1 = \alpha_2 = \alpha_3 = 0$, $d_1 = d_2 = d_3 = 0$, then both the position and Jacobian matrix J_{a1} of A_1 can be calculated readily. Based on the definition in Equation 8, J_{o1} can be obtained. A_i and J_{oi} can be calculated similarly. The control parameters are set as $\epsilon = 0.001$, $\alpha = 8$, $k = 8$. As to the physical constraints, the limits of joint angles and velocities are selected as $\theta_i^- = -3$rad, $\theta_i^+ = 3$rad, $\dot{\theta}_i^- = -1$rad/s, $\dot{\theta}_i^+ = 1$rad/s for $i = 1 \ldots 4$. The safety distance d is set to be 0.1m.

4.2. Single Obstacle Avoidance

In this simulation, the obstacle is assumed to be centered at $[-0.1, 0.2]^{\mathrm{T}}$m, the desired path is set as $x_{\mathrm{d}} = [0.4 + 0.1cos(0.5t), 0.4 + 0.1sin(0.5t)]^{\mathrm{T}}$m, and the initial joint angles are set to be $\theta_0 = [\pi/2, -\pi/3, -\pi/4, 0]^{\mathrm{T}}$rad. The class-K function is selected as $G(|D|) = K_1|D|$ with $K_1 = 200$. In order to show the effectiveness of the proposed obstacle avoidance method, contrast simulations with and without inequality constraint (10e) are conducted. Simulation results are shown in **Figure 3**. When ignoring the obstacle, the end-effector trajectories and the corresponding incremental configurations are shown in **Figure 3A**, although the robot achieves task space tracking to x_{d}, obviously the first link of the robot would collide with the obstacle. After introducing obstacle avoidance scheme, the robot moves other joints rather than the first joint, and then avoids the obstacle effectively (**Figure 3B**). Simultaneously, the tracking errors when tracking the given circle are shown in **Figure 3C**. From the initial state, the end-effector moves toward the circle quickly and smoothly, after that, the tracking error in stable state keeps $< 1 \times 10^{-4}$m, showing that the robot could achieve kinematic control as well as obstacle avoidance tasks. To show more details of the proposed deep RNN based method, some important process data is given. As the obstacle is close to the first joint, critical points A_1 and A_2 are more likely to collide with obstacle, therefore, as a result, the distances between the obstacle O_1 and A_1, A_2 are shown in **Figure 3D**, from $t = 2$s to $t = 6.5$s, $||A_1O_1||$ remains at the minimum value $d = 0.1$, that is to say, using the proposed obstacle avoidance method, the robot maintains minimum distance from the obstacle. The profile of joint velocities are shown in **Figure 3E**, at the beginning of simulation, the robot moves at maximum speed, which leads to the fast convergence of tracking errors. The curve of joint angles change over time is shown in **Figure 3F**.

4.3. Discussion on Class-K Functions

In this part, we will discuss the influence of different class-K functions in the avoidance scheme (5). Four functions are selected as $G_1(|D|) = K|D|^2$, $G_2(|D|) = K|D|$, $G_3(|D|) = K\tanh(5|D|)$, $G_4(|D|) = K\tanh(10|D|)$, **Figure 4A** shows the comparative curves the these functions. Other simulation settings are the same as the previous one. Simulation results are shown in **Figure 4B**. When selecting the same positive gain K, the minimum distance is about 0.08m, which shows the robot can avoid colliding with the obstacle using the avoidance scheme (5). The close-up graph of the tracking error is also shown, it is remarkable that the minimum distance deceases, as the gradient of the class-K function increases near 0.



B

Joint	a_i(m)	α_i(rad)	d_i(m)
1	0.3	0	0
2	0.3	0	0
3	0.1	0	0
4	0.2	0	0

FIGURE 2 | The planar robot to be simulated in this paper. **(A)** is the physical structure and critical points, **(B)** is the corresponding Dh parameters.

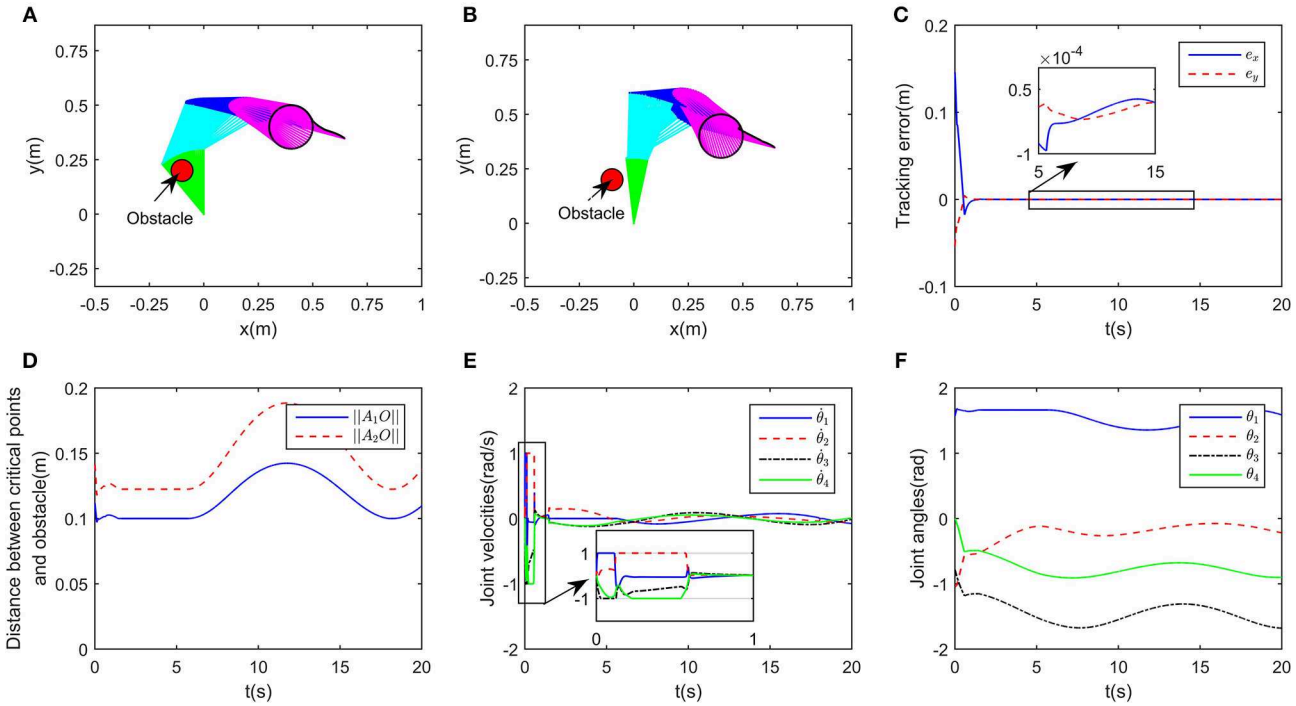


FIGURE 3 | Numerical results of single obstacle avoidance. **(A)** is the motion trajectories when ignoring obstacle avoidance scheme, **(B)** is the motion trajectories when considering obstacle avoidance scheme, **(C)** is the profile of tracking errors when considering obstacle avoidance scheme, **(D)** is the profile of distances between critical points and obstacle, **(E)** is the profile of joint velocities, **(F)** is the profile of joint angles.

Therefore, one conclusion can be drawn that the function can be more similar with Sign function, to achieve better obstacle avoidance.

4.4. Multiple Obstacles Avoidance

In this part, we consider the case where there are two obstacles in the workspace. The obstacles are set at $[0.1, 0.25]^{\mathrm{T}}$m and $[0, 0.4]^{\mathrm{T}}$m, respectively. Simulation results are shown in **Figure 5**. The desired path is defined as $x_{\mathrm{d}} = [0.45 + 0.1cos(0.5t), 0.4 + 0.1sin(0.5t)]^{\mathrm{T}}$. The initial joint angle of the robot is selected as $\theta_0 = [1.5, -1 - 1, 0]^{\mathrm{T}}$. To further show the effectiveness of the proposed obstacle avoidance strategy 5, $g|D|$ is selected as $g|D| = K_1/(1 + e^{-|D|}) - K_1/2$ with $K_1 = 200$. When λ_2 is set to 0, as shown in **Figure 5A**, the inequality constraint (11d) will not work, in other words, only kinematic tracking problem in considered rather than obstacle avoidance, in this case, the robot would collide with the obstacles. After introducing online training of λ_2, the simulation results are given in **Figures 5B–H**. The tracking errors are shown in **Figure 5C**, with the transient time being about 4s, and steady state error $< 1 \times 10^{-3}$m. Correspondingly, the robot moves fast in the transient stage, ensuring the quick convergence of the tracking errors. It is remarkable that the distances between the critical points and obstacle points are kept larger than 0.1m at all times, showing the effectiveness of the proposed method. At $t =$ 14s, from **Figures 5D,G**, when the distance between A_3 and O_1 is close to 0.1m, the corresponding dual variable λ_2 becomes positive, making the inequality constraint (11d) hold, and the boundedness between the robot and obstacle is thus guaranteed. After $t = 18$s, $||A_3O_1||$ becomes greater, then λ_2 converges to 0. Notable that although λ_1 and λ_2 do not converge to certain values, the dynamic change of λ_1 and λ_2 ensures the regulation of the proposed deep RNN.

4.5. Enveloping Shape Obstacles

In this part, we consider obstacles of general significance. Suppose that there is a rectangular obstacle in the workspace, with the vertices being $[0, 0.5]^{\mathrm{T}}$, $[0.4, 0.5]^{\mathrm{T}}$, $[0.4, 0.6]^{\mathrm{T}}$ and $[0.5, 0.6]^{\mathrm{T}}$, respectively. By selecting the safety distance $d =$ 0.1m, and obstacle points as $O_1 = [0.05, 0.55]^{\mathrm{T}}$, $O_2 = [0.15, 0.55]^{\mathrm{T}}$, $O_3 = [0.25, 0.55]^{\mathrm{T}}$ and $O_4 = [0.35, 0.55]^{\mathrm{T}}$. It can be readily obtained that the rectangular obstacle is totally within the envelope defined by O_i and d. The incremental configurations when tracking the path while avoiding the obstacle are shown in **Figure 6B**, in which a local amplification diagram is also given. showing that the critical points A_3 is capable of avoiding O_2 and O_3. It is worth noting that by selecting proper point group and safety distance, the obstacle can be described by the envelope shape effectively. While in **Figure 6A**, when obstacle avoidance is ignored, the collision emerges. **Figures 6C–H** also give important process data of the system under the proposed controller, including tracking errors, joint angles, angular velocities, and state variables of deep RNNs. We can observe that the physical constraints as well as kinematic control task are realized using the controller.

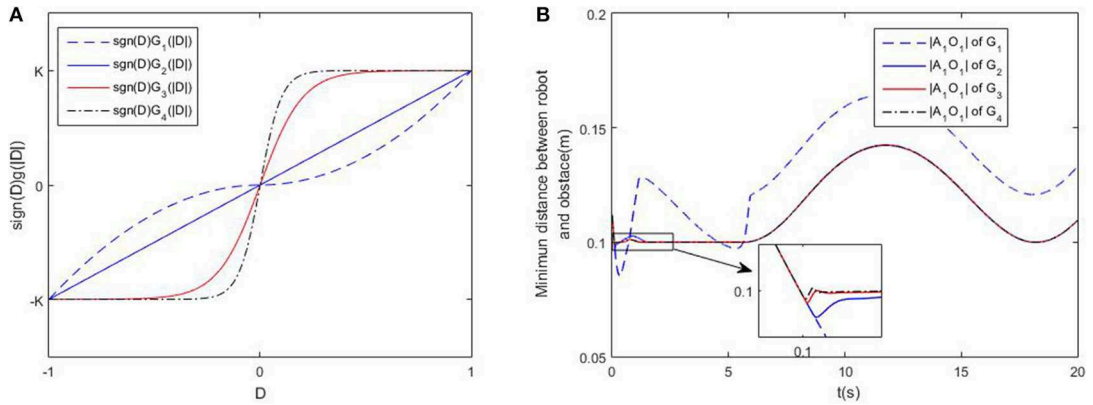


FIGURE 4 | Discussions on different obstacle avoidance functions. **(A)** is the comparative curves of different obstacle avoidance functions. **(B)** is the profile of minimum distance of the robot and obstacle using different obstacle avoidance functions.

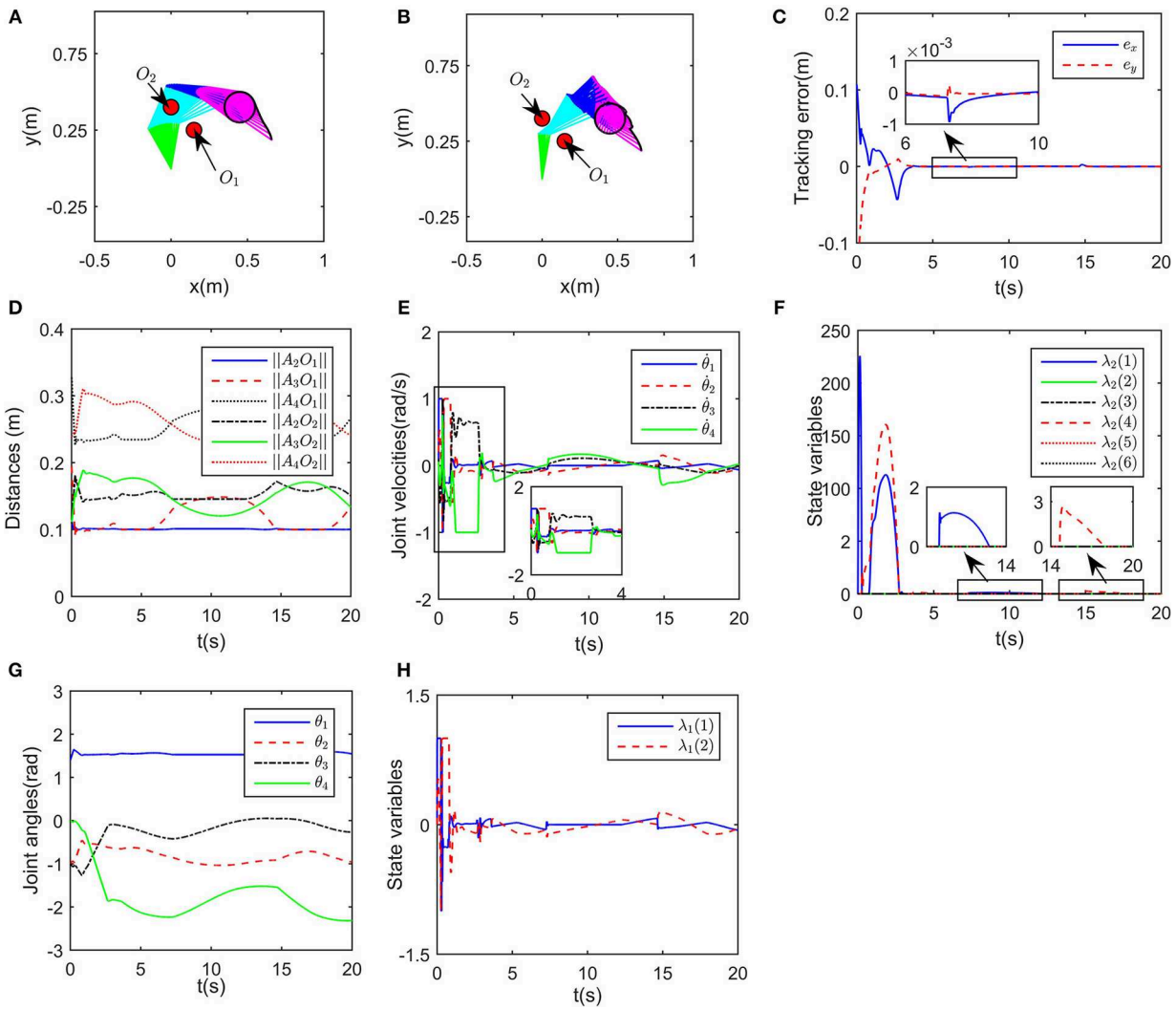


FIGURE 5 | Numerical results of multiple obstacle avoidance. **(A)** is the motion trajectories when ignoring obstacle avoidance scheme, **(B)** is the motion trajectories when considering obstacle avoidance scheme, **(C)** is the profile of tracking errors when considering obstacle avoidance scheme, **(D)** is the profile of distances between critical points and obstacles, **(E)** is the profile of joint velocities, **(F)** is the profile of λ_2, **(G)** is the profile of joint angles, **(H)** is the profile of λ_1.

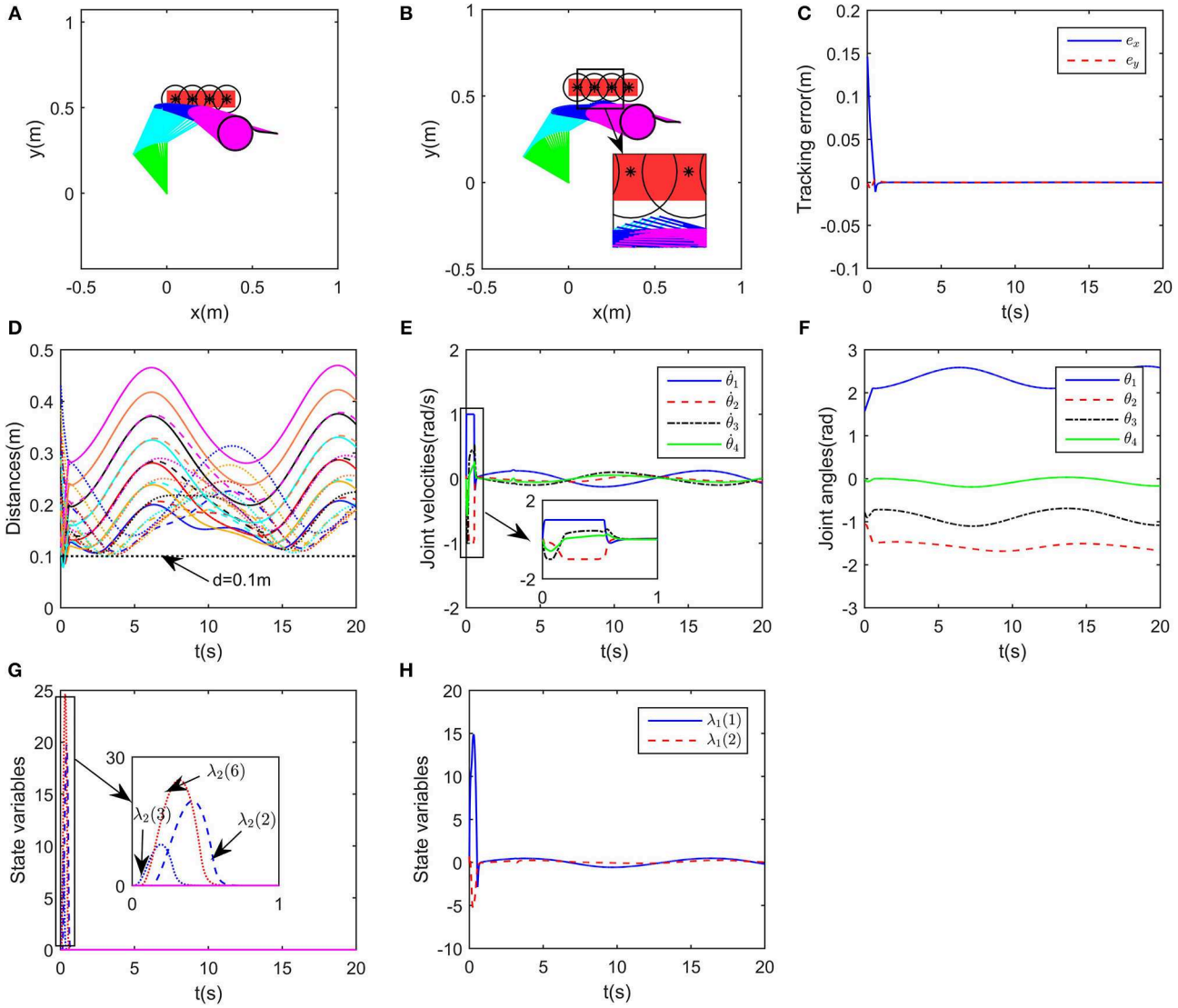


FIGURE 6 | Numerical results of enveloping shape obstacles. **(A)** is the motion trajectories when ignoring obstacle avoidance scheme, **(B)** is the motion trajectories when considering obstacle avoidance scheme, **(C)** is the profile of tracking errors when considering obstacle avoidance scheme, **(D)** is the profile of distances between critical points and obstacles, **(E)** is the profile of joint velocities, **(F)** is the profile of joint angles, **(G)** is the profile of λ_2, **(H)** is the profile of λ_1.

4.6. Dynamic Obstacles

In this part, we consider dynamic obstacles moving in the workspace. In real applications, pedestrian or other obstacles always tend to be mobile. Obstacle avoidance for dynamic obstacles is of more general significance. In real time, static obstacles can be considered a special case. In this simulation, the simulation duration is selected as 20s, and the trajectory of a dynamic obstacle is defined as $x_{\mathrm{d}} = [-0.1 + 0.01t, 0.3]^{\mathrm{T}}$. The snapshots in the control process are shown in **Figure 8**. While ensuring effective tracking of the defined path, the robot is able to use its self-motion to avoid the dynamic obstacle effectively, and maintain a safe distance. The distances between critical points and the dynamic O is shown in **Figure 7B**. At the beginning of simulation, the tracking error is big, in order to ensure the convergence of tracking error, the joints move a big range except $J1$. It is worth noting that since the critical point A_2 is next to O, joint 1 rotates in the direction which conforms to the movement of O. In the stable state, tracking error is $< 5 \times 10^{-4}$m (**Figure 7A**). At about $t = 14$s, A_2O decreases to 0.1m, accordingly, the joint velocities change obviously (as shown the significant change of joint velocities in **Figure 7C**, the tracking error changes to 10^{-3}m, and then converges quickly. At $t = 18$s, although A_2 and A_3 are near O, the robot is still capable of avoiding the dynamic obstacle. During the control process, joint angles are ensured not to exceed the limits, as shown in **Figure 7D**.

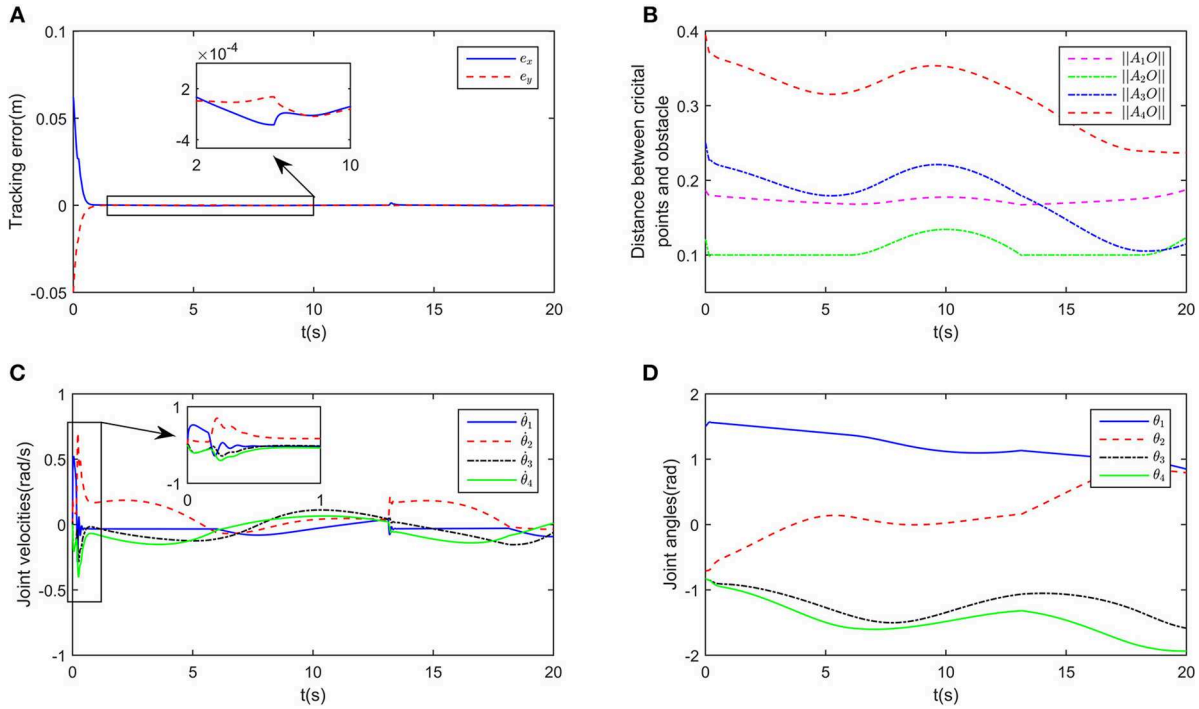


FIGURE 7 | Numerical results of enveloping shape obstacles. **(A)** is the profile of tracking errors when considering obstacle avoidance scheme, **(B)** is the profile of distances between critical points and obstacles, **(C)** is the profile of joint velocities, **(D)** is the profile of joint angles.

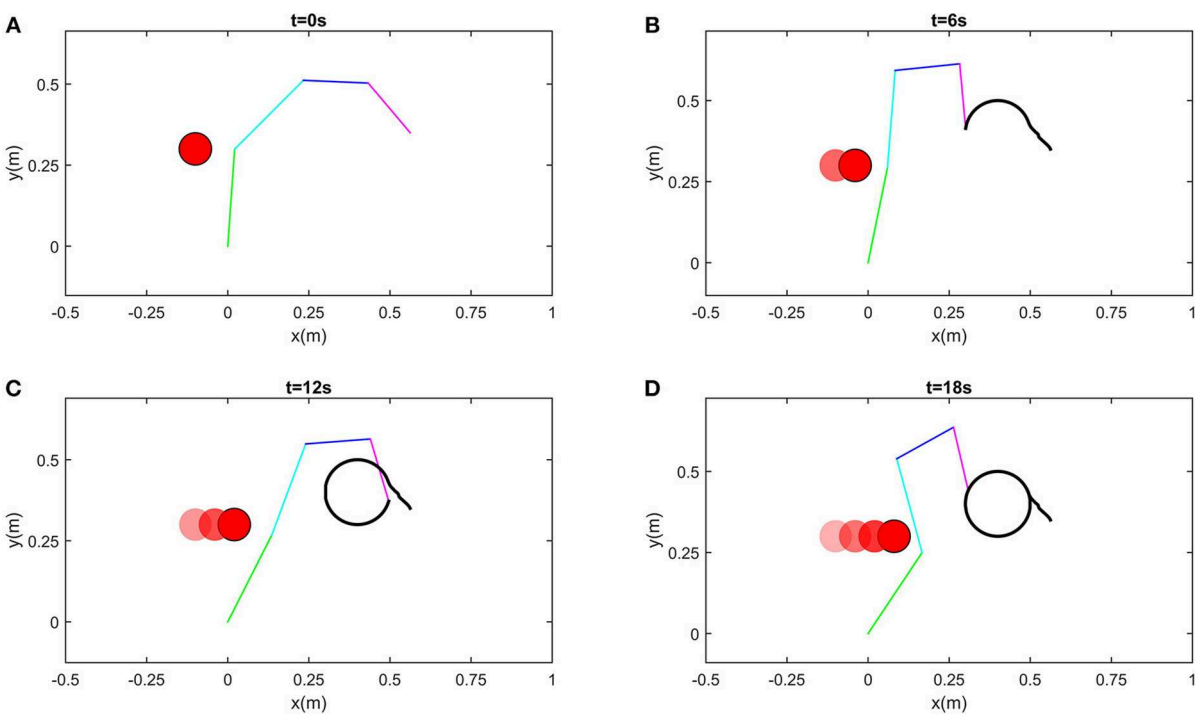


FIGURE 8 | Snapshots when robot avoiding a dynamic obstacle. **(A)** is the snapshot when $t = 0$s, **(B)** is the snapshot when $t = 6$s, **(C)** is the snapshot when $t = 12$s, **(D)** is the snapshot when $t = 18$s.

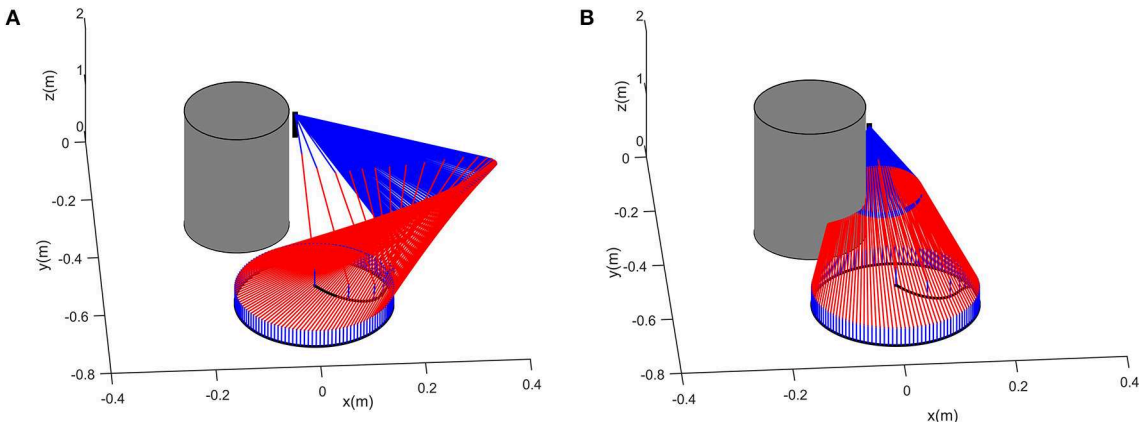


FIGURE 9 | Comparative results when the proposed controller is used on a 7-DOF manipulator iiwa in 3-dimensional space. **(A)** is the tracking trajectory and the corresponding joint configurations when obstacle avoidance scheme is introduced. **(B)** is the tracking trajectory and the corresponding joint configurations when obstacle avoidance scheme is not introduced.

4.7. Obstacle Performance on 7-DOF Manipulator in 3-Dimensional Space

To further verify the effectiveness of the control scheme, another simulation on a 7DOF manipulator iiwa is carried out. The desired path to be tracked is also a planar circular, which is centered at $[0, -0.6, 0.1]^{\mathrm{T}}$m with radius being 0.15m. The physical parameters can be found in Xu et al. (2019a). Suppose that there exist a cylinder obstacle in the workspace, the obstacle is centered as $[-0.13, -0.3, 0]^{\mathrm{T}}$m, with the radius and height being 0.15m and 2m, respectively. Simulation results are shown in **Figure 9**. It can be readily found that the proposed schemes can obtain satisfying performance in 3-dimensional spaces.

4.8. Comparisons

To illustrate the priority of the proposed scheme, a group of comparisons are carried out. As shown in **Table 1**, all the controllers in Zhang and Wang (2004); Csiszar et al. (2011); Guo and Zhang (2012); Krzysztof and Joanna (2016) achieve the avoidance of obstacles. Comparing to APF method in Csiszar et al. (2011); Krzysztof and Joanna (2016) of JP based method in Csiszar et al. (2011); Krzysztof and Joanna (2016), the proposed controller can realize a secondary task, at the same time, we present a more general formulation of the obstacle avoidance strategy, which is helpful to gain a deeper understanding of the mechanism for avoidance of obstacles. Moreover, in this paper, both dynamic trajectories and obstacles are considered. The comparisons above also highlight the main contributions of this paper.

TABLE 1 | Comparisons among different obstacle avoidance controllers on manipulators.

Method	Convergence	Secondary task	Handling physical constraints	Dynamic obstacles	obstacle avoidance description
This paper	Yes	Yes	Yes	Yes	Inequalities
Guo and Zhang, 2012	Yes	Yes	Yes	*	Inequalities**
Zhang and Wang, 2004	Yes	Yes	Yes	*	Inequalities**
Csiszar et al., 2011	Yes	No	No	Yes	Repulsion
Krzysztof and Joanna, 2016	Yes	No	No	*	Null space

In Zhang and Wang (2004); Guo and Zhang (2012); Krzysztof and Joanna (2016), dynamic obstacles are not considered.
***Regular escape velocity method is used, which is only a special case of 5.*

5. CONCLUSIONS

In this paper, a novel obstacle avoidance strategy is proposed based on a deep recurrent neural network. The robots are obstacles are presented by sets of critical points, then the distance between the robot and obstacle can be approximately describes as point-to-points distances. By understanding the nature escape velocity methods, a more general description of obstacle avoidance strategy is proposed. Using minimum-velocity-norm (MVN) scheme, the obstacle avoidance together with path tracking problem is formulated as a QP problem, in which physical limits are also considered. By introducing model information, a deep RNN with simple structure is established to solve the QP problem online. Simulation results show that the proposed method can realize the avoidance of static and dynamic obstacles.

DATA AVAILABILITY

All datasets analyzed for this study are included in the manuscript and the supplementary files.

AUTHOR CONTRIBUTIONS

Theoretical derivation was done by ZX. Simulation research was performed by ZX, with important advice from XZ. The paper was revised by XZ and SL. All authors approved the manuscript.

FUNDING

This work is supported by the GDAS′ Foundation of National and Provincial Science and Technology Talent (Grant No. 2018GDASCX-0603), Guangdong Province Applied Science and Technology Research Project (Grant No.2015B090922010), Guangzhou Science and Technology Planning Project(Grant NO.201803010106), Guangdong Province Science and Technology Major Projects (Grant No. 2016B090910003), Guangdong Province Science and Technology Major Projects (Grant No. 2017B010116005), Guangdong Province Key Areas R&D Program (Grant No. 2019B090919002).

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their valuable comments and suggestions.

REFERENCES

Badawy, A. (2016). Dynamic and interactive path planning and collision avoidance for an industrial robot using artificial potential field based method. *Alexandria Eng. J.* 55, 1235–1241. doi: 10.1016/j.aej.2016.03.042

Cao, B., Dodds, G., and Irwin, G. (1999). Redundancy resolution and obstacle avoidance for cooperative industrial robots. *J. Robot. Syst.* 16, 405–417.

Chan, T., and Dubey, R. (1995). A weighted least-norm solution based scheme for avoiding joint limits for redundant joint manipulators. *IEEE Trans. Robot. Automat.* 11, 286–292.

Cheng, F., Chen, T., Wang, Y., and Sun, Y. (1993). "Obstacle avoidance for redundant manipulators using the compact qp method," in *IEEE International Conference on Robotics and Automation* (Atlanta, GA: IEEE), 41–50. doi: 10.1109/ROBOT.1993.292186

Cheng, L., Hou, Z. G., and Tan, M. (2009). Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. *Automatica* 45, 2312–2318. doi: 10.1016/j.automatica.2009.06.007

Csiszar, A., Drust, M., Dietz, T., Verl, A., and Brisan, C. (2011). Dynamic and interactive path planning and collision avoidance for an industrial robot using artificial potential field based method. *Mechatronics* 1, 413–421. doi: 10.1007/978-3-642-23244-2-50

Fontaine, J., and Germain, A. (2001). Model-based neural networks. *Comput. Chem. Eng.* 25, 1045–1054. doi: 10.1016/S0098-1354(01)00679-2

Fu, C., Duan, R., and Kayacan, E. (2019). Visual tracking with online structural similarity-based weighted multiple instance learning. *Informat. Sci.* 481, 292–310. doi: 10.1016/j.ins.2018.12.080

Fu, C., Sarabakha, A., Kayacan, E., Wagner, C., John, R., and Garibaldi, J. (2018). Input uncertainty sensitivity enhanced non-singleton fuzzy logic controllers for long-term navigation of quadrotor uavs. *IEEE/ASME Trans. Mech.* 23, 725–734. doi: 10.1109/TMECH.2018.2810947

Ge, S., and Cui, Y. (2000). New potential functions for mobile robot path planning. *IEEE Trans. Robot. Automat.* 16, 615–620. doi: 10.1109/70.880813

Guo, D., and Zhang, Y. (2012). A new inequality-based obstacle-avoidance mvn scheme and its application to redundant robot manipulators. *IEEE Trans. Syst. Man Cybernet. Part C* 42, 1326–1340. doi: 10.1109/TSMCC.2012.2183868

Hsu, D., Latombe, J., and Kurniawati, H. (2006). On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.* 25, 627–643. doi: 10.1177/0278364906067174

Jung, S., and Kim, S. (2007). Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems. *IEEE Trans. Indust. Electr.* 54, 265–271. doi: 10.1109/TIE.2006.888791

Khatib, O. (1986). Real-time obstacle avoidance system for manipulators and mobile robots. *Int. J. Robot. Res.* 5, 90–98.

Krzysztof, T., and Joanna, R. (2016). Dynamically consistent jacobian inverse for non-holonomic robotic systems. *Nonlinear Dynam.* 85, 107–122. doi: 10.1007/s11071-016-2672-x

Lee, K. K., and Buss, M. (2007). "Obstacle avoidance for redundant robots using jacobian transpose method," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (San Diego, CA: IEEE), 3509–3514.

Li, S., Zhang, Y., and Jin, L. (2017). Kinematic control of redundant manipulators using neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 2243–2254. doi: 10.1109/TNNLS.2016.2574363

Li, Y., Li, S., and Hannaford, B. (2019). A model based recurrent neural network with randomness for efficient control with applications. *IEEE Trans. Indust. Informat.* 15, 2054–2063. doi: 10.1109/TII.2018.2869588

Moosavian, S. A. A., and Papadopoulos, E. (2007). Modified transpose jacobian control of robotic systems. *Automatica* 43, 1226–1233. doi: 10.1016/j.automatica.2006.12.029

Pan, Y., Sun, T., Liu, Y., and Yu, H. (2017). Composite learning from adaptive backstepping neural network control. *Neural Netw.* 95, 134–142. doi: 10.1016/j.neunet.2017.08.005

Pan, Y., Yang, C., Pan, L., and Yu, H. (2018). Integral sliding mode control: Performance, modification, and improvement. *IEEE Trans. Indust. Informat.* 14, 3087–3096. doi: 10.1109/TII.2017.2761389

Pan, Y., and Yu, H. (2017). Biomimetic hybrid feedback feedforward neural-network learning control. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 1481–1487. doi: 10.1109/TNNLS.2016.2527501

Slotine, J., and Li, W. (2004). *Applied Nonlinear Control*. Beijing: China Machine Press.

Sugie, T., Kito, Y., and Fujimoto, K. (2003). Obstacle avoidance of manipulators with rate constraints. *IEEE Trans. Robot. Automat.* 19, 168–174. doi: 10.1109/TRA.2002.807554

Tsai, C., Lee, J., and Chuang, J. (2001). Path planning of 3-d objects using a new workspace model. *IEEE Trans. Syst. Man. Cybernet. Part C* 31, 405–410. doi: 10.1109/5326.971669

Tsuji, T., Tanaka, Y., Morasso, P., Sanguineti, V., and Kaneko, M. (2002). Bio-mimetic trajectory generation of robots via artificial potential field with time base generator. *IEEE Trans. Syst. Man. Cybernet. Part C* 32, 426–439. doi: 10.1109/TSMCC.2002.807273

Wei, K., and Ren, B. (2018). A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm. *Sensors* 18, 571–578. doi: 10.3390/s18020571

Wen, G., Ge, S., Tu, F., and Choo, Y. (2017). Artificial potential based adaptive h∞ synchronized tracking control for accommodation vessel. *IEEE Trans. Indust. Elect.* 64, 5640–5647. doi: 10.1109/TIE.2017.2677330

Xu, Z., Li, S., Zhou, X., and Cheng, T. (2019a). Dynamic neural networks based adaptive admittance control for redundant manipulators with model uncertainties. *Neurocomputing* 1, 1–22. doi: 10.1016/j.neucom.2019.04.069

Xu, Z., Li, S., Zhou, X., Yan, W., Cheng, T., and Huang, D. (2019b). Dynamic neural networks based kinematic control for redundant manipulators with model uncertainties. *Neurocomputing* 329, 255–266. doi: 10.1016/j.neucom.2018.11.001

Yang, C., Jiang, Y., He, W., Na, J., Li, Z., and Xu, B. (2018a). Adaptive parameter estimation and control design for robot manipulators with finite-time convergence. *IEEE Trans. Indust. Electr.* 65, 8112–8123. doi: 10.1109/TIE.2018.2803773

Yang, C., Peng, G., Li, Y., Cui, R., Cheng, L., and Li, Z. (2018b). Neural networks enhanced adaptive admittance control of optimized robot-environment interaction. *IEEE Trans. Cybern.* 49, 2568–2579. doi: 10.1109/TCYB.2018.2828654

Zhang, Y. (2015). Singularity-conquering tracking control of a class of chaotic systems using zhang-gradient dynamics. *IET Control Theory Appl.* 9, 871–881. doi: 10.1049/iet-cta.2014.0931

Zhang, Y., Chen, S., Li, S., and Zhang, Z. (2018). Adaptive projection neural network for kinematic control of redundant manipulators with

unknown physical parameters. *IEEE Trans. Indust. Electr.* 65, 4909–4920. doi: 10.1109/TIE.2017.2774720

Zhang, Y., Ge, S., and Lee, T. (2004). A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators. *IEEE Trans. Syst. Man Cybernet. Part B* 34, 2126–2132. doi: 10.1109/TSMCB.2004.830347

Zhang, Y. and Wang, J. (2004). Obstacle avoidance for kinematically redundant manipulators using a dual neural network. *IEEE Trans. Syst. Man Cybernet. Part B* 34, 752–759. doi: 10.1109/TSMCB.2003.811519

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 11 July 2019
doi: 10.3389/fnbot.2019.00052

A Novel Model for Arbitration Between Planning and Habitual Control Systems

*Farzaneh Sheikhnezhad Fard and Thomas P. Trappenberg**

Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada

It is well-established that human decision making and instrumental control uses multiple systems, some which use habitual action selection and some which require deliberate planning. Deliberate planning systems use predictions of action-outcomes using an internal model of the agent's environment, while habitual action selection systems learn to automate by repeating previously rewarded actions. Habitual control is computationally efficient but are not very flexible in changing environments. Conversely, deliberate planning may be computationally expensive, but flexible in dynamic environments. This paper proposes a general architecture comprising both control paradigms by introducing an arbitrator that controls which subsystem is used at any time. This system is implemented for a target-reaching task with a simulated two-joint robotic arm that comprises a supervised internal model and deep reinforcement learning. Through permutation of target-reaching conditions, we demonstrate that the proposed is capable of rapidly learning kinematics of the system without *a priori* knowledge, and is robust to (A) changing environmental reward and kinematics, and (B) occluded vision. The arbitrator model is compared to exclusive deliberate planning with the internal model and exclusive habitual control instances of the model. The results show how such a model can harness the benefits of both systems, using fast decisions in reliable circumstances while optimizing performance in changing environments. In addition, the proposed model learns very fast. Finally, the system which includes internal models is able to reach the target under the visual occlusion, while the pure habitual system is unable to operate sufficiently under such conditions.

Keywords: machine learning, reinforcement learning, supervised learning, habitual controller, planning, internal models, decision making

OPEN ACCESS

Edited by:
Florian Röhrbein, Technical University of Munich, Germany

Reviewed by:
Eiji Uchibe, Advanced Telecommunications Research Institute International (ATR), Japan
*Yangwei You, Institute for Infocomm Research (A*STAR), Singapore*

****Correspondence:***
Thomas P. Trappenberg
tt@cs.dal.ca

Received: *27 February 2019*
Accepted: *28 June 2019*
Published: *11 July 2019*

Citation:
Sheikhnezhad Fard F and Trappenberg TP (2019) A Novel Model for Arbitration Between Planning and Habitual Control Systems. Front. Neurorobot. 13:52. doi: 10.3389/fnbot.2019.00052

1. INTRODUCTION

Much of the current reinforcement learning (RL) literature implements model-free control. Such a learning agent learns a value function from interacting with the environment, usually updating a proposed value function from a temporal difference between the previous expectation and a new experience (Mnih et al., 2013, 2015). The value function is like a big lookup-table that can quickly supply evaluations for possible actions and hence provide guidance for actions in a fast and somewhat automated way. Such a decision system can be characterized as habitual. While habitual action selection takes time to learn and requires that similar previous situations have been encountered sufficiently, the advantage is that decisions and corresponding actions can be generated in a timely manner.

In contrast, a system that has some internal models of the environment can be used to derive a value function on demand for a specific situation. A prime example is a Markov decision problem where the reward function and transition function of the agent are known so that the Bellman equations can be used to calculate the optimal value function for every state action pair without the need to explore the environment. Of course, this system requires learning of the internal models, which requires previous interactions with the environment. The learning of internal models can be achieved through some form of supervised learning. Once the models have been learned, the model-based system is able to calculate a value function on the fly. This resembles some form of internal deliberation. The advantage of such a system is its flexibility to new situations. However, deliberations take time so that a habitual system is preferable when it comes to situations that benefit from a higher degree of automation.

In this paper, we introduce a learning system that we call the Arbitrated Predictive Actor-Critic (APAC) that combines a habitual reinforcement learning system with a supervised learning system of internal models. Most importantly, we introduce an arbitration system that mediates between their usage. We specifically discuss a situation in which both systems alone can solve an exemplary task so that we can study the consequences of their direct interactions in relation to their exclusive use. We show that this system is responsive to changes in the environment and that it can learn the reward function very fast. Our results demonstrate how the learning paradigm tend to rely on habits after learning the reward function. Our results are in line with evidence of human behavior mentioned above.

2. THEORETICAL PREMISES

There is a lot of behavioral and neurophysicological evidence for different types of control systems in the brain that are usually termed habitual or model-free and goal-directed or model-based (Balleine and Dickinson, 1998; Gläscher et al., 2010; Daw et al., 2011). In particular, one control system associated with the prefrontal cortex (Miller and Cohen, 2001) predicts action-outcomes using an internal model of the agent's environment and hence can be associated with a control system that uses deliberative planning. We will use in this paper the term deliberative planning instead of goal directed model-based control to minimize the possible confusion between the models of the environment from the models of the value function. Another control pathway in the brain is associated with the dorsolateral basal ganglia (Houk and Barto, 1995) learns to repeat previously rewarded actions that resemble a habitual system.

Some research showed that the two different control systems are used in different situations and can be simultaneously active (Poldrack et al., 2001; Lengyel and Dayan, 2008). For example, in the brain, the cortical system represents a generalized mapping of input distributions while hippocampal learning is an instance-based system (Lengyel and Dayan, 2008). Moreover, when the model of the environment is known and there is sufficient time to plan, the best strategy is deliberate planning (Daw et al., 2005), but when the decision should be taken very fast the habitual controller is used (Blundell et al., 2016). Other work shows that cooperation and competition between different control systems in the brain happens especially when outcomes of each control system disagree, that is, if a deliberated planning task does not align with a habitual skill (Daw et al., 2005, 2011; Daw and O'Doherty, 2013; Lee et al., 2014).

Moreover, feedback to learning systems can differ in different situations and can be provided from different modalities such as vision or auditory input. In machine learning, it is common to distinguish different learning paradigms. One is supervised learning where a teacher gives feedback from the knowledge of a desired behavior. The system can be trained by comparing the actual output of a leaner to the desired output provided by the teacher. The teacher is basically a critic who can quantify an objective function that a leaner needs to optimize. Another slightly more general learning paradigm is reinforcement learning where the environment only provides some indication of value, often only after a series of actions have been chosen by a learning agent. Reinforcement learning is thus more general in that it can be applied to a lot of more typical learning situations of an agent in an environment. The subsystems in our model align in our implementation with a supervised paradigm to learn internal models and a reinforcement learning paradigm to learn habitual control.

Habitual reinforcement learning which is based on TD learning (Sutton, 1985) has been very successful in explaining experimental evidence from the animal learning literature and dopamine-based learning in the brain (Barto, 1995; Schultz et al., 1997). Such models which have originally been formulated with tubular methods based on discrete state action spaces are now commonly combined with neural networks as a function approximator that broadens the range of practical applications to be solved using RL, especially for control problems with continuous states/actions spaces (Waltz and Fu, 1965; Barto et al., 1990). Barto, Sutton and Anderson introduced the Actor-Critic architecture that was implemented by neural networks (Barto et al., 1983). Later, Barto (1995) represented an adaptive critic which has similar behavior to the dopamine neurons projection to the Striatum and frontal cortex. The adaptive critic uses the internal sensory information to learn an effective reinforcement signal.

It has long been hypothesized that the brain builds an internal representation of the world and its body (Miall et al., 1993; Miall and Wolpert, 1996; Wolpert et al., 1998; Kawato et al., 2003), and evidence shows that "forward" and "inverse models" exist in the brain (Miall et al., 1993; Kawato et al., 2003). The internal model is used to perform in the environment and learn a new task. Flanagan and Wing (1997) showed that the internal model can predict the load force and the kinematics of a hand movement that depends on the load. Moreover, when learning how to use a new tool, humans make a transient change in the internal model of the arm as well as making an internal model of the tool (Kluzik et al., 2008). Furthermore, imitation experiments show that a direct mapping develops between observation and the internal model (Iacoboni et al., 1999). Another advantage of having an internal representation

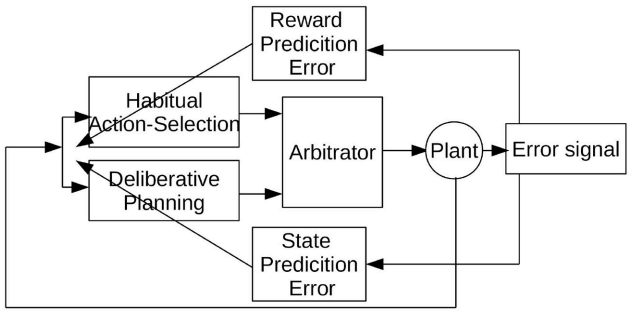


FIGURE 1 | Overview of the Arbitrated Predictive Actor-Critic (APAC) model.

is obtaining a reliable source of information for the agent to perform accurately even if there are no other sources of information (e.g., visual information) available (Wolpert et al., 1998; Kawato et al., 2003).

In this paper, we propose a model to study the cooperation and competition between a habitual and planning-based control components with an arbitrator component. The general architecture of the proposed *Arbitrated Predictive Actor-Critic* (APAC) is shown in **Figure 1**. In this model, each control paradigm implies a specific type of teaching feedback. The deliberative planning controller incorporates internal models that are usually trained with supervised errors so that we consider here an explicit state predictions error. In contrast, the habitual action selection system is a common deep reinforcement learner which learns from reward prediction errors. The new component here is an arbitrator that mediates between these systems that can select the command given to the controlled system, the agent or in the plant in the common language of control theory. Of course, it is possible that both decision systems are trained with a combination of supervised and reinforcement learning, but this is not the crucial point in this paper. The model is designed to study how a combined control system behaves in different environmental situations. In the following section we apply this general model to a specific motor control task in which both systems can be trained on the same feedback signal, but in which the execution would follow a habitual or planning implementation.

3. APAC FOR TARGET REACHING

In this section, we apply the APAC model to the motor control task of target reaching. We choose this task as it is a good example of a minimal control task while being complex enough to highlight the advantages and disadvantages of the two principle control architectures discussed in this paper. Target reaching lives in a continuous state and action space with 6 degrees of freedom when considering a shoulder and elbow yaw, pitch and roll, although we simplify this here even more to a 2-dimensional system with only one angle for the elbow and one the shoulder. Learning the reaching task in this 2D environment is learning a non-linear mapping function that

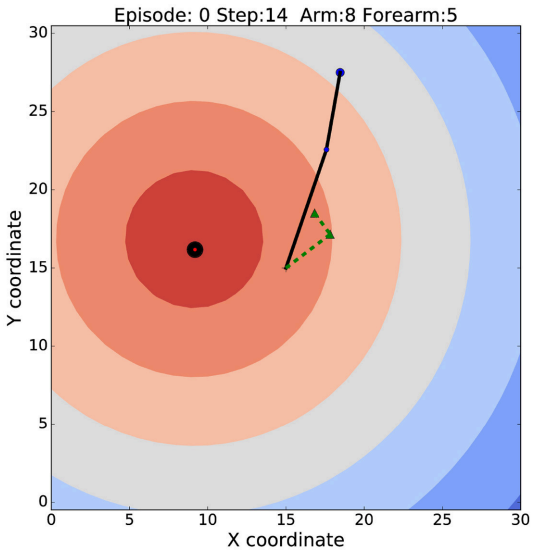


FIGURE 2 | The robotic arm set up in the environment. The solid black line indicates the robot arm. Blue and red circles display the end-effector and the target locations respectively. Colored contours illustrates reward function. Black circle shows the target zone. The dashed green line represents the internal model of the arm at very early stages of learning.

maps joint angles of the robot arm onto a location of the end-effector in the environment. An example image of our simulated robot arm is shown in **Figure 2** with the black line. The contour plot shows the distance to the target while the dotted green line shows an internal model of the robot arm early in learning.

The refined control architecture of our APAC model for the reaching tasks is shown in **Figure 3**. For this application, the state is defined as the position of the elbow, the end-effector, and the target. The planning component is now implemented as a combination of deep forward and inverse models, while the habitual system is implemented as a deep actor-critic model. An integrator is used to derive the training signals that are used for the feedback. In the following, we specify each subsystem in detail.

3.1. Habit Learning Control System

The habitual controller is implemented as a deep deterministic policy gradient model (DDPG) following the work of Lillicrap et al. (2015). The arm position is given by the vector X with two vector components, the position of the end-effector X^{end} and the position of the elbow X^{elbow}. The arm position together with the target location X_{target} defines the current state $s_t = [X^{end}, X^{elbow}, X_{target}]$ of the agent.

The critic $\mathcal{Q}(s_t, a_t; \theta^{\mathcal{Q}})$ is implemented as a deep neural network, where s_t is the current state at time t, a_t is the action taken at time t, and $\theta^{\mathcal{Q}}$ are the parameters of the critic network. The goal of the critic is to approximate the accumulation of the environmental reward (sometimes called return) that can be expected from a certain state action

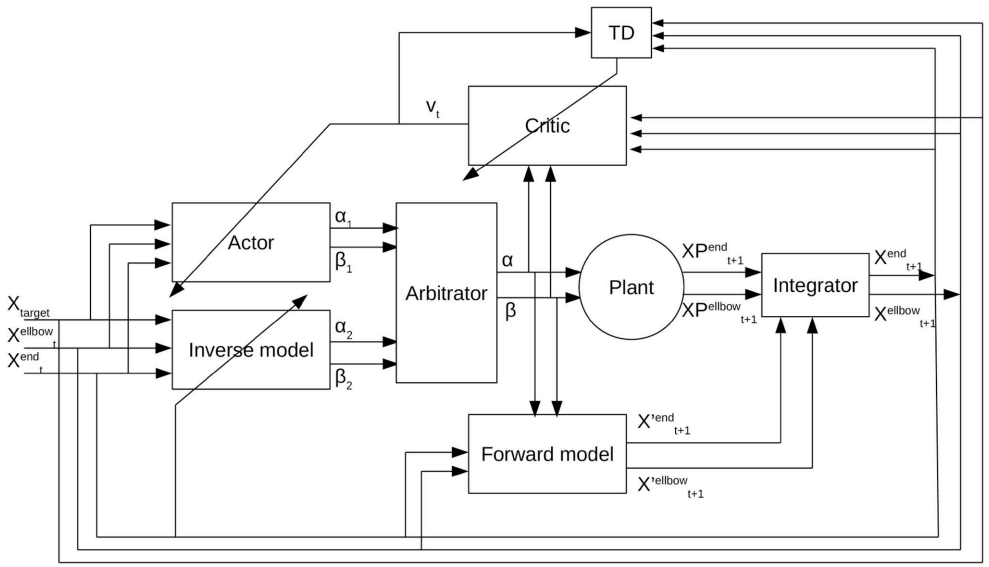


FIGURE 3 | Arbitrated predictive actor critic: The actor receives the current state defined as the current location of the robot arm and the target location, and provides an action that is the change in angles for shoulder and elbow. The inverse model takes the current state and predicts another action. The output of the inverse model and the actor goes to the arbitrator. Then the arbitrator selects one of these actions. The output of the arbitrator along with the current state is the input to the critic from which the critic predicts a reward. The forward model receives the selected action and the current position of the agent and predicts the future state of the agent. The agent takes the selected actions and transfers to the next state. The predicted future state from the forward model is integrated with the estimation of the actual state of the plant after taking the action and obtains a new current state for the system.

combination. The critic is learned through temporal difference (TD) learning (Sutton, 1988; Schultz et al., 1997).

$$\mathcal{Q}(s_t, a_t; \theta^{\mathcal{Q}}) \leftarrow \mathcal{Q}(s_t, a_t; \theta^{\mathcal{Q}}) + l_1\delta, \tag{1}$$

$$\delta = \underbrace{r_t + \gamma \max_{a'} \mathcal{Q}'(s_{t+1}, a'; \theta^{\mathcal{Q}})}_{\text{estimated reward}} - \underbrace{\mathcal{Q}(s_t, a_t; \theta^{\mathcal{Q}})}_{\text{actual reward}}, \tag{2}$$

where l_1 is the learning rate of the critic network, r_t is the actual immediate reward received from the environment at time t, γ is a discount factor, and $\mathcal{Q}'$ represents the estimation of the value of a state-action pair. As in DDPG, we use the main network for training but we use a *target network* for predicting, which is a less frequently updated copy of the main network to avoid oscillation. More precisely, DDPG actually has two target networks, one for the critic network and one for the actor network. We follow directly the smooth update for the target networks as in DDPG (Lillicrap et al., 2015),

$$\theta' = \theta' \times (1 - \tau) + \theta \times \tau, \tag{3}$$

with change parameter $\tau \ll 1$. The parameters θ' represents the target network parameters, and θ is the parameter of the main network, either the actor or the critic.

The computation in Equation (2) is done in the TD component. To train the critic using the TD rule, the error needs to be back-propagated through the critic. The error between estimated value and the actual value is used to compute the loss function of the critic (Equation 4),

$$L^{\mathcal{Q}} = 1/N \sum(\delta)^2. \tag{4}$$

DDPG takes advantage of the *experience memory replay* which is a memory to store and reuse past experiences. The memory replay is in form of $R(s_t, a_t, r_t, s_{t+1}, T_t)$, where s_t is the current state at time t, a_t is the action taken at time t, s_{t+1} is the next state, r_t is the reward received at time t, and T_t indicates whether the state at time $t + 1$ is a terminal or not. The replay memory is a queue-like buffer with a finite size. The agent will forget older experiences and it will update its parameters based on its recent experiences. At each time step, a random batch of N samples is selected from the experience memory replay, and this batch is used to train both the actor and the critic.

The actor (π) receives the current state (s_t) and predicts future actions to be taken (a_t).

$$\pi(s_t; \theta^{\pi}) = a_t, \tag{5}$$

where $a_t = [\alpha_1, \beta_1]$ and α_1 and β_1 are motor commands sent to the shoulder and elbow, respectively. The actor is implemented as a deep network where θ^{π} indicates the parameter of the actor network and is trained using the deterministic policy gradient method (Silver et al., 2014). Note that the main actor network is used for training, however, the target network of the actor is used for the action prediction.

The changes of the weights of the actor corresponded to the changes in expected reward with respect to the actor's parameters,

$$\theta_t^{\pi} \leftarrow \theta_t^{\pi} + l_2 \frac{\partial \mathcal{Q}(s_t, a_t; \theta^{\mathcal{Q}})}{\partial \pi(s_t; \theta_t^{\pi})} \frac{\partial \pi(s_t; \theta_t^{\pi})}{\partial \theta_t^{\pi}}, \tag{6}$$

where l_2 here is the learning rate of the actor. The plant, which is the simulated arm in our example, takes the action and

transitions to its new position $[Z^{end}_{t+1}, Z^{elbow}_{t+1}]$, which forms the new state s_{t+1} when combined with the target location. Like DDPG we apply noise to the environment using an Ornstein-Uhlenbeck process (Uhlenbeck and Ornstein, 1930) that results in new samples.

3.2. Internal Models for Planning

For the planning controller, we need to learn the transition function of the plant to build the model of the environment. Here we use supervised learning to learn the internal representation of the agent. More specifically, we used a supervised learning controller that uses past experiences to generalize an inverse model of the arm and a forward model of the arm. The training examples used in our implementation are obtained from the same experience replay memory that is used for the habitual controller.

A combination of a forward and an inverse model is used for planning the next actions. The forward model f_F is a neural network that receives the current position of the arm $[X^{end}_t, X^{elbow}_t]$ and the action a_t and predicts the future position of the arm $[X'^{end}_{t+1}, X'^{elbow}_{t+1}]$. We can train the network from the discrepancy between the predicted future position $[Z^{end}_{t+1}, Z^{ellbow}_{t+1}]$ and the actual position from visual feedback. For training we use the loss function

$$L^{f_F} = \frac{1}{N}\sum_t ([X'^{end}_{t+1}, X'^{ellbow}_{t+1}] - [Z^{end}_{t+1}, Z^{ellbow}_{t+1}])^2, \qquad (7)$$

where N is the number of selected samples in a batch of experiences stored in the replay memory. The size of the batch to train the forward model and the inverse model is the same as the one used for the actor and the critic.

An inverse model is another deep network, $f_I(s_t; \theta^{f_I})$. The aim of the inverse model is to provide a proper action to reach the target by minimizing the error between predicted action (a'_t) with the actual action taken (a_t) that transfers the agent from the current position to its next position. This network is then trained on the loss function:

$$L^{f_I} = \frac{1}{N}\sum_t (a_t - a'_t)^2. \qquad (8)$$

The aim of having the forward model is learning to predict future positions of the agent by taking specific actions. Such a model enables the agent to perform the task even with occluded vision. When the inverse model has been trained well, it can be used to produce a suitable action to transfer the agent from its current state to the target location by replacing X_{target} with Z^{end}_t. Hence, the inverse model can be trained with the input $[X^{end}_{t-1}, X^{elbow}_{t-1}, Z^{end}_t]$ and predicting the proper actions on $[X^{end}_{t-1}, X^{elbow}_{t-1}, X_{target}]$. Note that $[X^{end}_{t-1}, X^{elbow}_{t-1}]$ are part of states s_t in the replay memory while Z^{end}_t is taken from s_{t+1} in the replay memory.

Another component of the overall system is "the integrator" module. In general, the integrator could be a Bayes filter such as a Kalman filter which estimates the best estimated position from the available information that combines an internal model prediction with external sensory feedback. Since we use a reliable visual feedback in our case study we simplify this to an integrator that passes the actual location of the plant in case visual information is available. With occluded vision, the prediction of the forward model is used as the estimated actual position of the agent. In our previous work (Fard et al., 2015), we showed how to implement a Kalman Filter with Dynamic Neural Fields (Amari, 1977). The integrator is the explicit critic in this example, which provides the state prediction error for the forward model (see **Figure 1**).

A training session of the system includes an infant phase that uses "motor babbling" (Iverson and Fagan, 2004; von Hofsten, 2004; Demiris and Dearden, 2005; Iverson et al., 2007; Caligiore et al., 2008). During the babbling phase, the plant produces random movements with random actions to produce actual samples to be stored in the experience memory. In the babbling phase, the actual position of the arm after taking an action is considered the target location. Therefore, all samples in this case reach the terminal state and will gain the maximum reward value. The babbling phase is used to provide valid examples to train both control systems.

3.3. Arbitration Between Habitual and Planning Controllers

A novel component of APAC is an arbitrator. The arbitrator receives action predictions from the deliberative planning module (the inverse model), and the habitual action selection module (the actor), and makes the final decision of which action to use. This selected action is transferred to the actuators of the plant to bring the agent into its new position resulting in a new state when combined with the target location. The arbitrator's decision is also fed into the forward model and the critic for training purposes. As in DDPG, noise from an Ornstein-Uhlenbeck process is added to both proposal actions provided by the inverse model and the actor.

In our implementation of the APAC, we consider discrete action steps so that both controllers (habitual and planning) create actions at each step. However, it is known that the habitual controller is faster than deliberative planning. Therefore, to imply the time constraint we set the arbitrator to give priority to the habitual controller. Moreover, the arbitrator is set to always take the action that is provided by the habitual system for the first two steps of every episode. However, from the third step on, the actor's prediction is taken if the habitual controller is reliable, meaning that the reward prediction error for the last experience is smaller than a threshold. We use $abs(\delta) < 1$ in the following experiments. Otherwise, the action from the inverse model is selected.

The implementation of the arbitrator here is somewhat a minimal model suitable for our experimental setting and to highlight the consequences of such arbitration. Of course, it is possible to implement a more dynamic realization of such an arbitrator. For example, the threshold could itself be modulated according to the tasks and in this way produces a more rich speed-accuracy trade-off (Satel et al., 2005). Indeed, such modeling will open the possibility to discuss behavioral consequences with different system settings a ultimately compare them to variations in populations or psychiatric disorders such

as eduction and eating disorders (Huys et al., 2016). However, the simple implementation discussed in this paper captures the minimal assumptions as outlined above and is sufficient for the following simulations.

3.4. Experimental Conditions and Environment

To test the APAC model on a simulated robot arm with a target reaching task (**Figure 2**), we simulated a two-joint robotic arm whose range of motion at each joint was constrained to 180 degrees. The arm's motion was limited to a 2D plane of width 30 and height 30, upon which the arm's "shoulder" was fixed in the center (15,15). The initial length segment from the shoulder to the elbow was set to $l_1 = 5$, and the initial length of the lower segment ("hand" to "elbow") is $l_2 = 8$.

All experiments described herein had an episodic trial structure. At the beginning, the arm's position was set to zero-degree angle at the shoulder and 180-degree angle at the elbow. Time was discretized in the simulations, and the learning agent was given only 30 action-steps per trial to achieve the designated goal. We define a "target zone" as a circle centered at the target location with a radius $r_{\text{target}} = 0.5$. The target is defined as "reached" once the robot arm is inside the target zone. If the goal was not reached within 30 time-steps, the trial was aborted and a new trial was started.

Importantly, the reward function is defined as the negative Euclidean distance between the end-effector and the center of the target area. This is for this example tasks the same information as is given to the supervised learner. This was deliberatively done so that the different systems are compared on the same feedback situation. It is possible to learn this task from much simpler feedback such as some reward if the target area is reached vs. no reward otherwise, although this would then also need more time to train the habitual system. The point of our study here is rather the direct comparison of decision components based on a value lookup vs. learning internal models.

Within this environment we define several conditions that defined the variety of the different target-reaching tasks studied here. These conditions include the target position (**static target** vs. **changing target** at each episode), kinematics (arm dimensions as **static kinematics** vs. **changing kinematics**), and vision (**occluded vision** vs. **perfect vision**). We tested all combinations of these factors.

Each experiment consisted of 1,000 episodes of maximal 30 action steps each. In the static target condition, the target is initialized randomly and stays unchanged for all 1,000 episodes. For the changing target condition, the target is located at a random location at the beginning of every episode. As discussed above, each episode was terminated when either (A) the target was reached, or (B) 30 time steps had elapsed. Targets were only placed within a reachable distance for the arm. The arm dimensions were kept fixed in the case of static kinematics; however, the length of both the upper and lower arm segments were increased by 0.001 at each time step for the changing kinematics condition. These changes were only started after the 100th episode of target reaching to provide some time for basic training. As already mentioned, an environmental noise was included in all experiments. In the occluded vision condition, the location of the arm and the target was unavailable for the agent during the movement. This task is also known as memory guided target reaching (Westwood et al., 2003; Heath et al., 2004). We repeated all static/changing kinematics and static/changing target conditions with our proposed models for the reaching-target task in the occluded vision condition. To examined the generalization of the models under each condition, we trained each model when targets are located only in a specific area that represents 2/3 of whole reaching area, and tested with targets located in the other part of the environment, which is the rest 1/3 of the reaching area.

4. RESULTS

We considered three versions of APAC that represent (a) exclusive habits, (b) exclusive deliberate planning, and (c) arbitration between habit and planning. Exclusive habit is when the arbitrator is set to always pick the action from the habitual system. In this case, the APAC behaves exactly like DDPG. If the arbitrator always selects the action from the inverse model for each step, then the APAC becomes an exclusive deliberate planning controller which we call supervised predictive actor-critic (SPAC) (Fard et al., 2017). The third model is when the APAC is able to arbitrate between the actions provided by the inverse model and the actor.

For each condition, we trained 50 independent instances of each model for a total of 1,000 episodes. At the end of the 1000th episode, all network parameters were frozen and no more training was applied. Subsequently, each of the independent model instances performed 100 target reaching episodes under the respective training conditions. In the case of the occluded vision, sensory input (i.e., visual target position) was initially presented at time step 0, and subsequently rendered unavailable. Since DDPG has no internal model and requires visual input throughout the task, we only compare APAC with SPAC in the experiments with occluded vision. All experiments were implemented and tested in Python (3.5) using the TensorFlow (1.3) package (Abadi et al., 2016) on an NVIDIA GeForce GTX 960 graphical processing unit.

Figure 4 illustrate the percentage of trials that reach the target within 30 action steps during training under different conditions for 1,000 episodes each. The pure deliberate planning model SPAC reaches almost near perfect performance very fast after 100 episodes. Furthermore, SPAC's performance is very robust under different conditions and neither changes in reward function nor in kinematics effects the performance of SPAC very much. DDPG learns to reach almost 100% of the targets only under static target/static kinematics condition. Performance of DDPG drops slightly under changing kinematics compared to static kinematics; however, its performance drops dramatically (about 20%) under changing target conditions. This is of course expected as habits become invalid solutions under changing environments. Our point here is that APAC can reach almost all of the targets both under static and changing targets as good as SPAC, although it tends to use more habits than planning after a few trials of

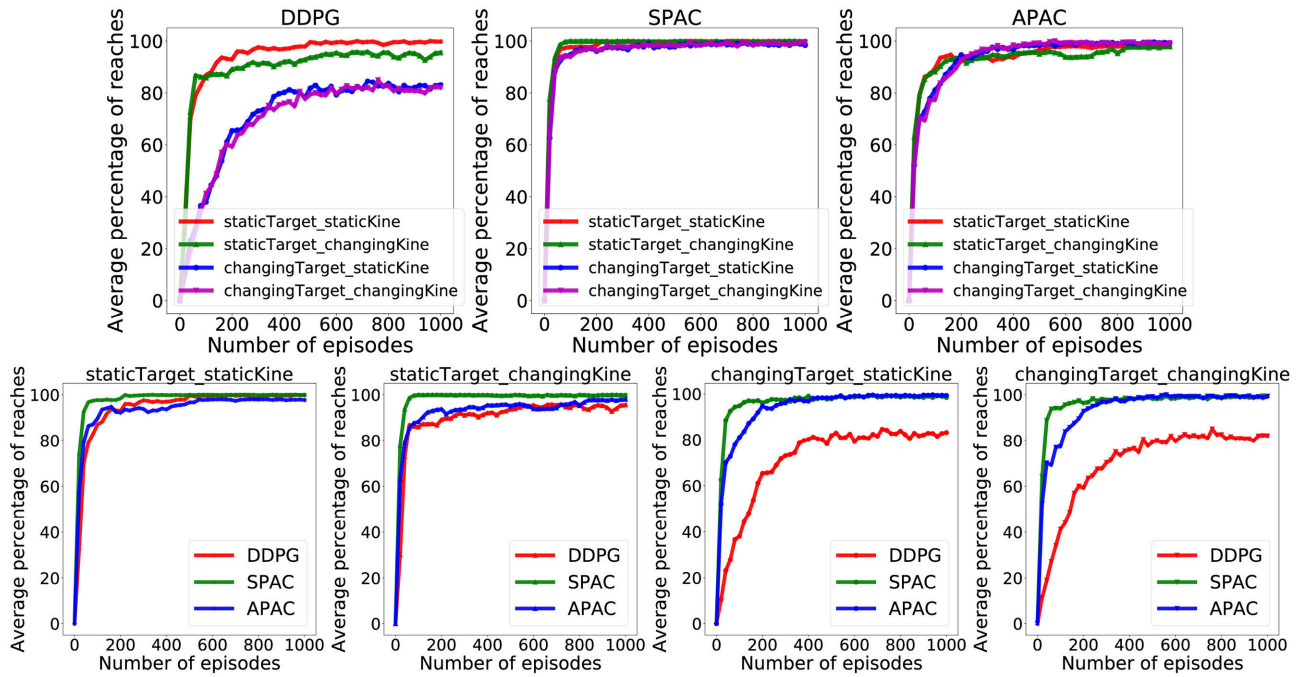


FIGURE 4 | Comparison of the proportion of trials in which the target was successfully reached during training between DDPG, SPAC, and APAC. **(Top)** highlights differences in each respective model's performance across the Target/Kinematic experimental combinations. APAC that arbitrates between planning and habits can resist to changes in the target position and kinematic changes, while DDPG is not flexible under either changing condition. **(Bottom)** plots highlight differences between models within each experimental combination of target position and kinematics. Performance of DDPG (pure habits) drops when target location is changing, while this has no effect on SPAC (exclusive planning) and APAC (arbitrated).

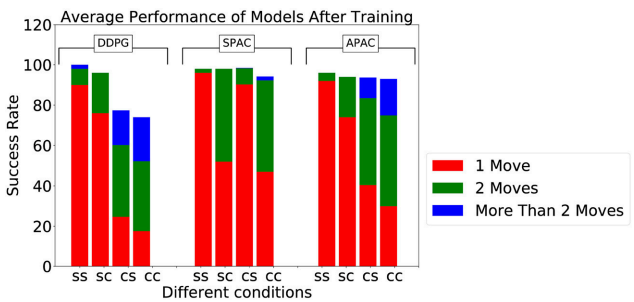


FIGURE 5 | Success rate to reach targets during testing by three models under different conditions, where "ss," "sc," "cs," and "cc" stand for static target/static kinematics, static target/changing kinematics, changing target/static kinematics, and changing target/ changing kinematics, respectively. The plot also displays the average number of steps to reach 100 targets after training.

learning(see **Figure 6**). The speed of learning in APAC is also very high and comparable to SPAC. In this sense it combines the benefits of DDPG and SPAC.

The above curves give an example of behavior of the models during one learning trials. To study how these results generalize we tested the performance of all three models after learning over 50 different learning trials with random initial conditions for the networks. For the static target location we tested on the target location, that was randomly chosen for each learning trial. However, with the changing target location we decided to cover the possible target locations more systematically and set target points on a regular grid in angle space. **Figure 5** displays average success rate over the 50 learning trials to reach these targets. All three models under the static target/static kinematics condition reach 100% of the targets. DDPG and APAC have slightly less success under static target/changing kinematics, while SPAC stays flexible under this condition. The major difference between DDPG and APAC become clear under changing target conditions, where DDPG's performance drops dramatically, while APAC obtains very good performance. SPAC is still very flexible to reach targets under changing target conditions. Overall it is remarkable how close APAC stays to the overall performance of SPAC in a situation where deliberative planning is the better choice.

The overall success rate does show the entire range of the solutions. We thus included the individual performances in terms of the average number of steps to reach the target. As can be seen, APAC needs to take sometimes more corrective steps to reach the target while an exclusive planning system can optimize the number of steps. This is interesting as this allows for different strategies in solving the task, that of relying somewhat on habitual control when the cost of the movement initiation might be small vs. more deliberate planning when the number of action steps might matter. This can explain a form of speed-accuracy trade-off.

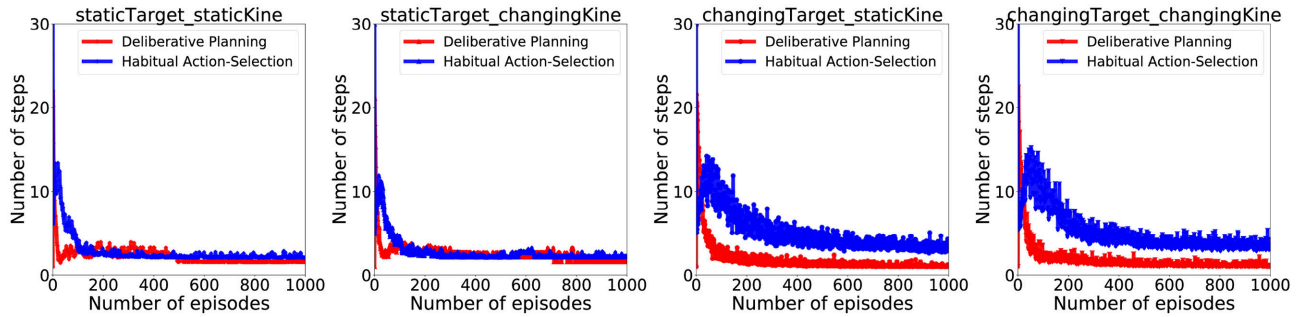


FIGURE 6 | Arbitration between habitual and planning by the APAC: For each condition, the blue line indicates the average number of actions which are selected by the habitual controller (i.e., the actor), while the red line demonstrates the average number of actions selected by the planning controller (i.e., inverse model). Results illustrate that planning controller is used early in the training, while later agent tends to use the habits more.

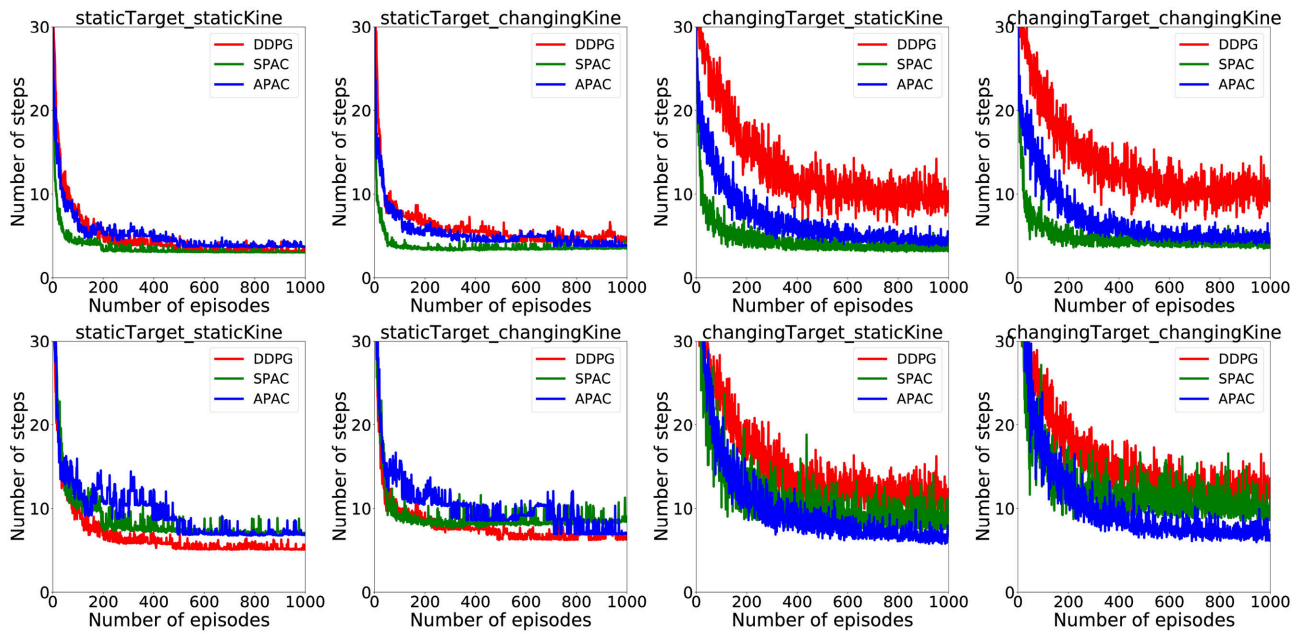


FIGURE 7 | (Top) Average action steps to complete the reaching task by each model under different conditions. **(Bottom)** Average time steps to complete the reaching task at each episode under different conditions when a three-times higher cost for a planning control compared to the habitual control is taken into account.

Figure 6 illustrates how APAC gradually shifts from a planning to a habitual control approach with increasing experience. After around 300 episode, more than 80% of APAC actions were taken from the habitual controller.Of course, SPAC uses planning control throughout the entirety of the task, so no commensurate figure was generated for it.

Since a habitual system should be faster than deliberate planning, this figure also illustrates that APAC would be less time-consuming than the SPAC at the same task and under the same condition. To visualize average time consumption by each model under different conditions, we assumed that each action selected by the deliberative planning takes three times longer than an action selected by the habitual controller. The number here is arbitrary and only chosen visualize the general effect. The top row of plots in **Figure 7** show the average number of action steps that each of the three models need to complete the task at each episode under different conditions. These plots demonstrate that the number of steps are almost the same under static target/static kinematics conditions. Under changing target conditions DDPG needs more steps to complete the task than APAC and SPAC. However, when including a higher cost for deliberative planning in the plots shown in the bottom row of the **Figure 7**, the picture for the average number of time that is needed to complete the task changes. In this case, DDPG needs shorter time under static target conditions. However, under changing target conditions, APAC completes the task of reaching targets faster. DDPG takes longer to finish the task as it needs more corrective actions.

Figure 8 shows all 50 runs to reach 100 targets of a reaching test under changing target conditions, with (bottom row) and without (top row) changing kinematics, for all three models.

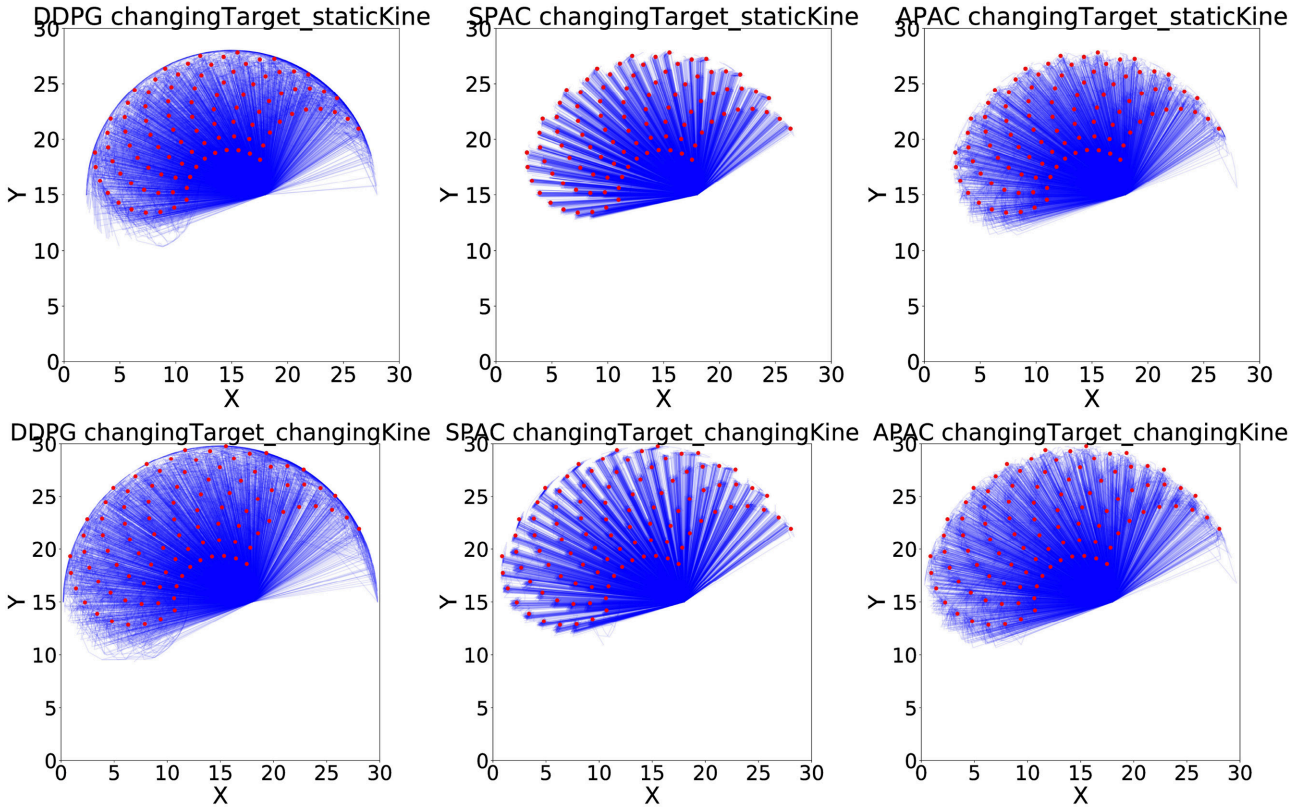


FIGURE 8 | Reaching examples of three models under changing target conditions. Blue lines show position of the end-effector at each time step toward the target, while red dots are target locations during the testing phase.

The plots illustrate that DDPG has some difficulty reaching the locations at the edges of the possible target area due to non-linearity of the mapping between angles and locations. SPAC can learn the mapping function much better, and the quality of APAC is similar to SPAC. Interestingly, although APAC tends to use more habits than deliberate planning, this model can still reach many more targets than DDPG, almost as good as SPAC.

We also tested a form of generalization of each model where a whole area of the target zone was not seen during training. This is a case of extrapolation compared to the interpolation trials in the previous generalization experiments. More specifically, we trained each model to reach the target located at a specific region in the environment and we tested each model to reach targets that are located on the unseen area of the environment (see the left most plot in the **Figure 9**). The same distribution of target locations has been used here and only those that are located in the blue area are set as targets for this experiments. Therefore, there are about 39 targets under static kinematics conditions and 42 targets under changing kinematics conditions (because of changing kinematics more targets will locate in the testing area). The results show that under static target training, neither model can reach even half of the targets. Their performance is worse under static target and changing kinematics. However, under changing target conditions, all models have obtained a good generalization but they need to take more than one step to reach any target. SPAC has again the best performance among other models under all conditions, while DDPG has the worst performance compared to other two models. These results indicate that learning with a static target hinders generalization as the learner overfits this specific target location.

Finally, we want to show results with occluded vision. Since the habitual controller (DDPG) requires sensory input at all times, only the SPAC and APAC models were compared under this condition. In these experiments, the arm moves toward the target when the target location is only visible at the first step. When the forward model indicates that the agent has reached the target it stops and the actual distance from the agent (here the arm) is measured. Results of these occluded vision test are summarized in **Figure 10** under changing target conditions, since the performance of both models under static target conditions are near perfect.

The target zone is marked by the red line in each plot, while the average distance from the agent to the target location is drawn as a black line. The blue area shows the range of the distances that the arm has experienced during the occluded trials when the forward model thinks that it has reached the target. SPAC shows better performance under occluded vision compared to APAC under all conditions. The average actual distance of the end-effector to the target location under changing target/ static kinematic with SPAC is only about a

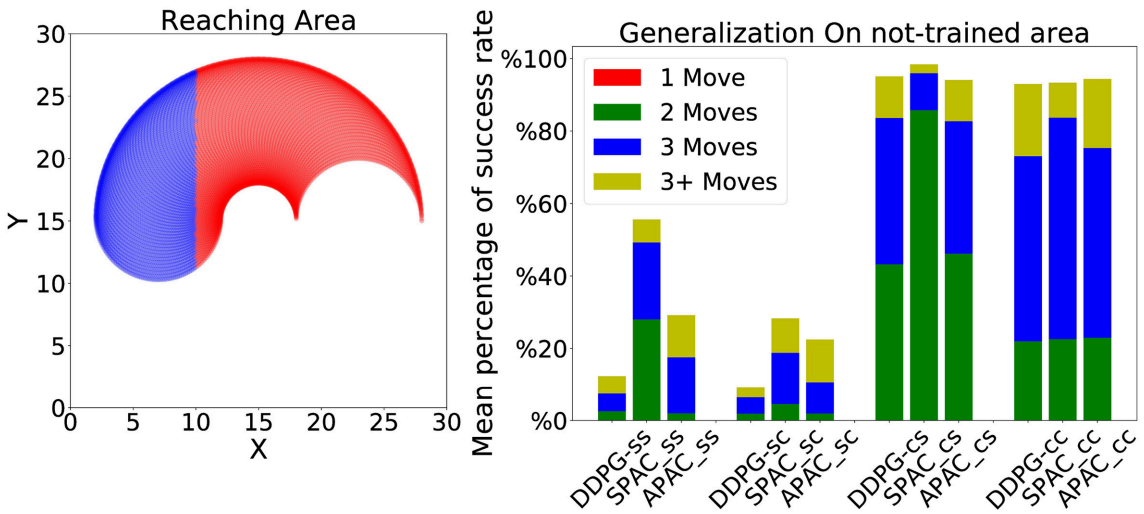


FIGURE 9 | (Left) The reaching area for arm under static kinematics conditions. The red area is used for training while the blue region is for testing. Around edges are more colored since these points can be reached with many more sets of angles. **(Right)** Performance of each model is shown under different conditions to reach about 40 targets located in the blue region. All models obtain a good generalization under changing target conditions.

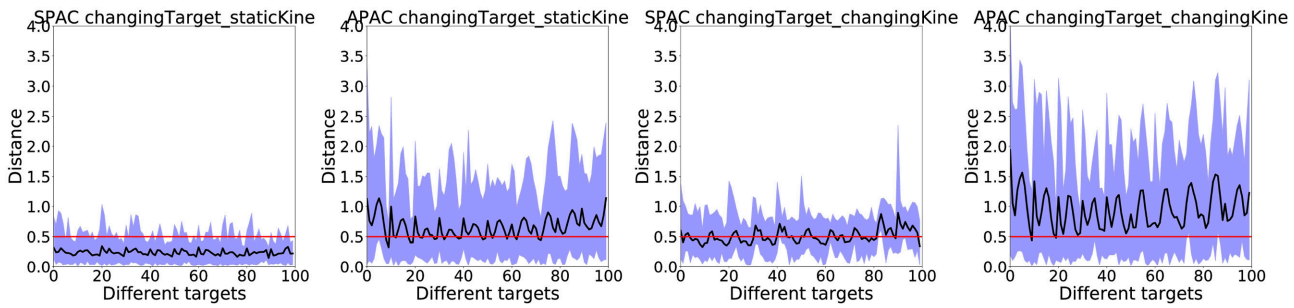


FIGURE 10 | The actual distance from the agent to the target location under changing target conditions are shown for SPAC and APAC to reach 100 targets after training over 50 runs. The black line shows the average distance to the target. The area between minimum distance to the maximum distance is colored in blue. The red line indicates the target zone. In general, SPAC performs better under the dark compared to APAC.

distance of 0.4, which is less than the target zone radius. However, the APAC model under the same conditions stays about a distance of 1 away from the target. Under changing target/changing kinematics, this average actual distance from end-effector to the target is around 0.7 for SPAC and about 1.5 for APAC. It thus seems that any form of habit should be suppressed in such situations, which could be achieved by a more advanced arbitrator.

5. CONCLUSION

This paper is about the study of a hybrid system with deliberate planning system and habitual control. Habitual control will, of course, be very good after long training in static environments. It was hence important to study the model in changing environments. We investigated the behavior of our proposed model (APAC) under changing target conditions (to manipulate the environmental reward function), changing kinematics of the agent (to manipulate the learned transitional model), and with and without vision. We also tested the model under various generalization conditions to see how good they can interpolate and extrapolate. The main results are classified as below:

Adaptive to changes: Results show significant improvement in performance when planning is available compared to the pure habitual system under the conditions when of changing environments that includes changing reward conditions and changing kinematics of the agent. In comparison, SPAC and APAC are flexible under these changes. These experiment shows that having an internal model is a key to robustness on changing environment.

Moving from planning to habits: By considering that planning is costly, having another control system that is able to provide less costly solutions can be useful. In this paper, there is no inherent time constraint or computation time difference between the habitual and planning controller. However, if we

take this time constraint into account, the APAC is a better model than pure planning (SPAC). Indeed, the overall number of actions taken based on a planned action takes less time than the arbitrated model than the number of planned actions taken by the non-arbitrated model when considering some cost of planning.

Reaching under occluded vision: Since DDPG has no internal model, it can not use any sort of planning to move under occlusion. However, systems like SPAC and APAC build an internal model of the environment that enables them to anticipate and plan a target even when there is no visual information available. Results show that SPAC can perform better in the dark. This is a good example to show that a more sophisticated arbitrator could take different circumstances into account. For occluded vision, such an arbitrator should suppress habitual actions.

Our application example focused on the implementation of the habitual system as an actor-critic for reinforcement learning and a planning system with a forward and inverse model using supervised learning. There have been previous examples of combining both, some form of supervised learning with RL systems and the use of the internal model. In particular, Dyna-Q (Sutton, 1991) and supervised actor-critic (Rosenstein and Barto, 2002; Barto and Rosenstein, 2004) are examples of models that bring the capacity of planning into a model-free reinforcement learning space. For example, the supervised actor-critic (Rosenstein and Barto, 2002; Barto and Rosenstein, 2004) is able to tune the actor manually and very fast when it is needed. This solution is beneficial when dynamics of the system changes dramatically or a new policy is needed to be learned in a very short time. These authors used a gain scheduler that weights the control signal provided by the actor from the reinforcement learner and the supervised actor. In contrast to supervised actor-critic, our proposed model autonomously learns the internal model and arbitrates between the two controllers automatically and not manually. The intention of our model is to study the interaction of habitual and planning systems in form of an arbitrator and ultimately to understanding human behavior.

Sutton proposed Dyna-Q that is an integrated model for learning and planning (Sutton, 1991). With respect to our model, Dyna-Q is also a blend of model-free and model-based reinforcement learning algorithms. Dyna-Q can build a transition function and the reward function by hallucinating random samples. Therefore, although it uses a model-free paradigm at the beginning it becomes a model-based solution by learning the model of the world using the hallucination. In our model, the internal model is used to predict the future state of the agent unlike Dyna-Q that uses the model to train the critic and anticipate the future reward. Moreover, Dyna-Q starts from model-free controller and becomes a model-based controller. Hence, while Dyna-Q has focused on the utilization of internal models to learn a reinforcement controller, our model and study here is concerned with the arbitration of two control systems. However, since APAC tend to select actions from the planning controller that it learns very fast, it provides more accurate samples in the experience replay memory. Therefore, similar to the Dyna-Q, the habitual system takes advantage of learning from more valuable samples that lead the habitual controller to a better performance compared to the time that it is trained stand alone (in pure habitual paradigm).

Another interesting approach by Uchibe and Doya (2005) explores a collection of different controllers. Their work takes also different times constrains into account. In contrast to our model, the controllers are combined probabilistically in a more collaborative way while our approach focuses more specifically on understanding the competitive decision making of a deliberative vs. habitual systems. The experiments are also somewhat different. Even though the barriers in their experiment are static, it seems that their habitual controller can not learn this task. In our experiment we made sure that the experimental task is learnable by both controllers in the static case. In addition we study the performance with changing kinematics and changing targets.

Not only have human decision-making studies supported the notion that both habitual and planning controls are used during decision-making (Daw et al., 2011; O'Doherty et al., 2015), there is evidence that arbitration may be a dynamic process involving specific brain regions Lee et al. (2014). Our APAC model results suggest that such an arbitration strategy, wherein the planning paradigm is used until the habitual system's predictions become reliable can result in performance that is non-inferior to exclusive planning control in most cases. Thus, our APAC model supports (A) the importance and value of implementing predominantly planning control early in behavioral learning and (B) the diminishing importance of planning control with greater experience in a relatively static environment.

DATA AVAILABILITY

All datasets generated for this study are included in the manuscript and/or the **Supplementary Files**.

AUTHOR CONTRIBUTIONS

All authors listed have made a substantial, direct and intellectual contribution to the work, and approved it for publication.

ACKNOWLEDGMENTS

The authors would like to thank Abraham Nunes for valuable input and acknowledge funding from Natural Science and Engineering Research Council of Canada (NSERC, Grant number 249885) and NS graduate scholarship.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbot.2019.00052/full#supplementary-material

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., et al. (2016). Tensorflow: large-scale machine learning on heterogeneous distributed systems. *arXiv [Preprint]*. arXiv:1603.04467.

Amari, S.-I. (1977). Dynamics of pattern formation in lateral-inhibition type neural fields. *Biol. Cybern.* 27, 77–87. doi: 10.1007/BF00337259

Balleine, B. W., and Dickinson, A. (1998). Goal-directed instrumental action: contingency and incentive learning and their cortical substrates. *Neuropharmacology* 37, 407–419. doi: 10.1016/S0028-3908(98)00033-1

Barto, A. G. (1995). *Adaptive Critics and the Basal Ganglia.* Computer Science Department Faculty Publication Series. 7. Amherst, MA: University of Massachusetts. Retrieved from: https://scholarworks.umass.edu/cs_faculty_pubs/7

Barto, A. G., and Rosenstein, M. T. (2004). "J. 4 supervised actor-critic reinforcement learning," in *Handbook of Learning and Approximate Dynamic Programming,* Vol. 2, eds S. Jennie, A. G. Barto, and B. Warren (Powell, OH), 359.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man. Cybern.* 5:834–846. doi: 10.1109/TSMC.1983.6313077

Barto, A. G., Sutton, R. S., and Watkins, C. J. (1990). "Sequential decision problems and neural network," in *Advances in Neural Information Processing Systems* (Denver, CO), 686–693.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., et al. (2016). Model-free episodic control. *arXiv [Preprint]*. arXiv:1606.04460.

Caligiore, D., Ferrauto, T., Parisi, D., Accornero, N., Capozza, M., and Baldassarre, G. (2008). "Using motor babbling and hebb rules for modeling the development of reaching with obstacles and grasping," in *International Conference on Cognitive Systems*, E1–E8.

Daw, N. D., Gershman, S. J., Seymour, B., Dayan, P., and Dolan, R. J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron* 69, 1204–1215. doi: 10.1016/j.neuron.2011.02.027

Daw, N. D., Niv, Y., and Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nat. Neurosci.* 8, 1704–1711. doi: 10.1038/nn1560

Daw, N. D., and O'Doherty, J. P. (2013). "Multiple systems for value learning," in *Neuroeconomics: Decision Making and the Brain, 2nd Edn*, eds W. P. Glimcher and E. Fehr (San Diego, CA: Elsevier Inc.), 393–410. doi: 10.1016/B978-0-12-416008-8.00021-8

Demiris, Y., and Dearden, A. (eds.) (2005). *From Motor Babbling to Hierarchical Learning by Imitation: A Robot Developmental Pathway*. Lund: Lund University Cognitive Studies.

Fard, F. S., Hollensen, P., Heinke, D., and Trappenberg, T. P. (2015). Modeling human target reaching with an adaptive observer implemented with dynamic neural fields. *Neural Netw.* 72, 13–30. doi: 10.1016/j.neunet.2015.10.003

Fard, F. S., Nunes, A., and Trappenberg, T. (2017). "An actor critic with an internal model," in *Inaugural Conference on Cognitive Computational Neuroscience (CCN)* (New York, NY).

Flanagan, J. R., and Wing, A. M. (1997). The role of internal models in motion planning and control: evidence from grip force adjustments during movements of hand-held loads. *J. Neurosci.* 17, 1519–1528. doi: 10.1523/JNEUROSCI.17-04-01519.1997

Gläscher, J., Daw, N., Dayan, P., and O'Doherty, J. P. (2010). States versus rewards: dissociable neural prediction error signals underlying model-based and model-free reinforcement learning. *Neuron* 66, 585–595. doi: 10.1016/j.neuron.2010.04.016

Heath, M., Westwood, D. A., and Binsted, G. (2004). The control of memory-guided reaching movements in peripersonal space. *Motor Control* 8, 76–106. doi: 10.1123/mcj.8.1.76

Houk, J. C., Adams, J. L., and Barto, A. G. (1995). "A model of how the basal ganglia generates and uses neural signals that predict reinforcement," in *Models of Information Processing in the Basal Ganglia*, 249–274.

Huys, Q. J., Maia, T. V., and Frank, M. J. (2016). Computational psychiatry as a bridge from neuroscience to clinical applications. *Nat. Neurosci.* 19, 404–413. doi: 10.1038/nn.4238

Iacoboni, M., Woods, R. P., Brass, M., Bekkering, H., Mazziotta, J. C., and Rizzolatti, G. (1999). Cortical mechanisms of human imitation. *Science* 286, 2526–2528. doi: 10.1126/science.286.5449.2526

Iverson, J. M., and Fagan, M. K. (2004). Infant vocal–motor coordination: precursor to the gesture–speech system? *Child Dev.* 75, 1053–1066. doi: 10.1111/j.1467-8624.2004.00725.x

Iverson, J. M., Hall, A. J., Nickel, L., and Wozniak, R. H. (2007). The relationship between reduplicated babble onset and laterality biases in infant rhythmic arm movements. *Brain Lang.* 101, 198–207. doi: 10.1016/j.bandl.2006.11.004

Kawato, M., Kuroda, T., Imamizu, H., Nakano, E., Miyauchi, S., and Yoshioka, T. (2003). Internal forward models in the cerebellum: fmri study on grip force and load force coupling. *Prog. Brain Res.* 142, 171–188. doi: 10.1016/S0079-6123(03)42013-X

Kluzik, J., Diedrichsen, J., Shadmehr, R., and Bastian, A. J. (2008). Reach adaptation: what determines whether we learn an internal model of the tool or adapt the model of our arm? *J. Neurophysiol.* 100, 1455–1464. doi: 10.1152/jn.90334.2008

Lee, S. W., Shimojo, S., and O'Doherty, J. P. (2014). Neural computations underlying arbitration between model-based and model-free learning. *Neuron* 81, 687–699. doi: 10.1016/j.neuron.2013.11.028

Lengyel, M., and Dayan, P. (2008). "Hippocampal contributions to control: the third way," in *NIPS*, Vol. 20 (Denver, CO), 889–896.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. *arXiv [Preprint]*. arXiv:1509.02971.

Miall, R., Weir, D., Wolpert, D. M., and Stein, J. (1993). Is the cerebellum a smith predictor? *J. Motor Behav.* 25, 203–216. doi: 10.1080/00222895.1993.9942050

Miall, R. C., and Wolpert, D. M. (1996). Forward models for physiological motor control. *Neural Netw.* 9, 1265–1279. doi: 10.1016/S0893-6080(96)00035-4

Miller, E. K., and Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. *Annu. Rev. Neurosci.* 24, 167–202. doi: 10.1146/annurev.neuro.24.1.167

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. *arXiv [Preprint]*. arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

O'Doherty, J. P., Lee, S. W., Mcnamee, D., O'Doherty, J. P., Lee, S. W., and Mcnamee, D. (2015). The structure of reinforcement-learning mechanisms in the human brain. *Curr. Opin. Behav. Sci.* 1, 1–7. doi: 10.1016/j.cobeha.2014.10.004

Poldrack, R. A., Clark, J., Pare-Blagoev, E., Shohamy, D., Moyano, J. C., Myers, C., et al. (2001). Interactive memory systems in the human brain. *Nature* 414, 546–550. doi: 10.1038/35107080

Rosenstein, M. T., and Barto, A. G. (2002). *Supervised Learning Combined with an Actor-Critic Architecture*. Department of Computer Science, University of Massachusetts, Tech. Rep, 02–41.

Satel, J., Trappenberg, T., and Klein, R. (2005). "Motivational modulation of endogenous inputs to the superior colliculus," in *IJCNN'05, Proceedings of 2005 IEEE International Joint Conference on Neural Networks, 2005* (IEEE), Vol. 1 (Montreal, QC: IEEE), 262–267.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science* 275, 1593–1599. doi: 10.1126/science.275.5306.1593

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 387–395.

Sutton, R. S. (1985). *Temporal credit assignment in reinforcement learning* (Ph.D. dissertation). University of Massachusetts, Amherst.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Mach. Learn.* 3, 9–44. doi: 10.1007/BF00115009

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bull.* 2, 160–163. doi: 10.1145/122344.122377

Uchibe, E., and Doya, K. (2005). "Reinforcement learning with multiple heterogeneous modules: a framework for developmental robot learning," in *Proceedings of the 4th International Conference on Development and Learning* (Osaka: IEEE), 87–92.

Uhlenbeck, G. E., and Ornstein, L. S. (1930). On the theory of the brownian motion. *Phys. Rev.* 36:823. doi: 10.1103/PhysRev.36.823

von Hofsten, C. (2004). An action perspective on motor development. *Trends Cogn. Sci.* 8, 266–272. doi: 10.1016/j.tics.2004.04.002

Waltz, M., and Fu, K. (1965). A heuristic approach to reinforcement learning control systems. *IEEE Trans. Autom. Control* 10, 390–398. doi: 10.1109/TAC.1965.1098193

Westwood, D. A., Heath, M., and Roy, E. A. (2003). No evidence for accurate visuomotor memory: systematic and variable error in memory-guided reaching. *J. Motor Behav.* 35, 127–133. doi: 10.1080/0022289030 9602128

Wolpert, D. M., Miall, R. C., and Kawato, M. (1998). Internal models in the cerebellum. *Trends Cogn. Sci.* 2, 338–347. doi: 10.1016/S1364-6613(98) 01221-2

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 31 July 2019
doi: 10.3389/fnbot.2019.00061

From Rough to Precise: Human-Inspired Phased Target Learning Framework for Redundant Musculoskeletal Systems

Junjie Zhou[1,2,3], *Jiahao Chen*[2,3,4], *Hu Deng*[1,3] *and Hong Qiao*[1,2,5]*

[1] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China, [2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China, [3] Beijing Key Laboratory of Research and Application for Robotic Intelligence of "Hand–Eye–Brain" Interaction, Beijing, China, [4] Research Center for Brain-Inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing, China, [5] CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai, China

Redundant muscles in human-like musculoskeletal robots provide additional dimensions to the solution space. Consequently, the computation of muscle excitations remains an open question. Conventional methods like dynamic optimization and reinforcement learning usually have high computational costs or unstable learning processes when applied to a complex musculoskeletal system. Inspired by human learning, we propose a phased target learning framework that provides different targets to learners at varying levels, to guide their training process and to avoid local optima. By introducing an extra layer of neurons reflecting a preference, we improve the Q-network method to generate continuous excitations. In addition, based on information transmission in the human nervous system, two kinds of biological noise sources are introduced into our framework to enhance exploration over the solution space. Tracking experiments based on a simplified musculoskeletal arm model indicate that under guidance of phased targets, the proposed framework prevents divergence of excitations, thus stabilizing training. Moreover, the enhanced exploration of solutions results in smaller motion errors. The phased target learning framework can be expanded for general-purpose reinforcement learning, and it provides a preliminary interpretation for modeling the mechanisms of human motion learning.

Keywords: musculoskeletal system, human-inspired motion learning, noise in nervous system, reinforcement learning, phased target learning

OPEN ACCESS

Edited by:
Changhong Fu,
Tongji University, China

Reviewed by:
Chenguang Yang,
University of the West of England, United Kingdom
Zhihao Xu,
Guangdong Institute of Intelligent Manufacturing, Guangdong Academy of Sciences, China

***Correspondence:**
Hong Qiao
hong.qiao@ia.ac.cn

Received: *12 April 2019*
Accepted: *15 July 2019*
Published: *31 July 2019*

Citation:
Zhou J, Chen J, Deng H and Qiao H (2019) From Rough to Precise: Human-Inspired Phased Target Learning Framework for Redundant Musculoskeletal Systems. Front. Neurorobot. 13:61. doi: 10.3389/fnbot.2019.00061

1. INTRODUCTION

Research on human-like musculoskeletal robots has become multidisciplinary in recent years, as it involves fields such as neuroscience and materials science for modeling and implementing musculoskeletal motor systems. In fact, this branch of robotics mainly comprises muscle models (actuators), skeletal systems (supporting structure), and methods for motion control and learning (control systems). Related work can roughly be divided into two types, namely, muscle dynamics modeling along with hardware design (Jäntsch et al., 2013; Kurumaya et al., 2016; Asano et al., 2017) and musculoskeletal robot control (Pennestrì et al., 2007; Jagodnik and van den Bogert, 2010; Tahara and Kino, 2010). Although most studies have been focused on the first type, the development of neuroscience has gradually increased the research on human-inspired control.

As a multibody mechanical system (Stoianovici and Hurmuzlu, 1996; Shi and McPhee, 2000) comprising muscles and joints, the human musculoskeletal system has several advantages. For instance, muscle redundancy maintains the reliable operation of the musculoskeletal system when some muscles are fatigued or even damaged. Under control of the central nervous system, the musculoskeletal system can accomplish accurate and fine manipulation (Rasmussen et al., 2001; Chen et al., 2018). To unveil the mechanisms that provide such advantages, Hill studied the contraction properties of muscles, establishing the Hill model (Hill, 1938). From this fundamental work, a series of muscle dynamic models have been proposed (Huxley and Niedergerke, 1954; Eisenberg et al., 1980; Zahalak and Ma, 1990), but all of them present specific limitations. For instance, the simple second-order model (Cook and Stark, 1968; Agarwal et al., 1970) lacks independent nodal locations for external input signals, which indirectly affect the output. The Huxley contraction model (Huxley, 1957) is highly complex and no general-purpose method has been developed to obtain its parameters (Winters and Stark, 1987). The Hill model presents difficulties in measuring the fiber length during motion (Arnold and Delp, 2011).

Research has also been devoted to design hardware for emulating muscle characteristics. The Anthrob muscle unit (Jäntsch et al., 2013) and the sensor–driver integrated muscle module (Asano et al., 2015) try to resemble muscular structures. However, the weight and size of motors make hardware models notably diverge from biological muscles. Furthermore, resembling tiny human muscles through hardware design is difficult, thus undermining their applicability. In materials science, the synthesis of ideal materials for artificial muscles is being pursued to achieve the characteristics of biological muscles regarding size, weight, stiffness, and dynamic behavior. New materials for artificial muscles usually share some problems, including unsafe voltages and low strain. Accessory equipment can partly adjust the characteristics of materials. For instance, liquid-vapor transition has been used on a soft composite material (Miriyev et al., 2017) for implementation as an actuator in a variety of robotic applications. In addition, a coiled polymer muscle (Haines et al., 2014) controlled by varying water temperature prevents dependence on electricity. Hence, advanced design methods and materials seem promising to develop artificial muscles that closely reflect the dynamics of their biological counterparts.

Based on the abovementioned models, control systems developed for musculoskeletal robotics also face challenges. Redundant muscles and extremely complex tendon forces impose several barriers for direct solutions of muscle excitation. Widely used methods, such as inverse dynamics with static optimization (Crowninshield and Brand, 1981), computed muscle control (Thelen et al., 2003), proportional-derivative control (Jagodnik and van den Bogert, 2010), and PI-type iterative learning control (Tahara and Kino, 2010), are used to regulate musculoskeletal systems. Although some conventional methods, such as computed muscle control, theoretically compute muscle excitation signals, they also demand intensive computations for sophisticated processes (Chen et al., 2018). In addition, these control strategies are hardly supported by biological evidence showing that they resemble the approach of human motion learning.

In recent years, reinforcement learning has become a popular control method in robotics as it provides a natural-like approach to learn from the environment. In fact, as a method that fosters interaction with uncertain environments, reinforcement learning allows a learner to observe the environment and then execute appropriate actions. The environment provides rewards for each action, and the learner aims to maximize its rewards during decision-making. This learning process is similar to that of humans and animals (Sutton and Barto, 2018). Studies in neuroscience (Schultz et al., 1997; Law and Gold, 2009) verify this principle, and hence it is reasonable to consider human-like learning from the viewpoint of reinforcement learning (Tesauro, 1995; Diuk et al., 2008; Riedmiller et al., 2009). Deep neural networks are adopted to implement reinforcement learning. Specifically, the deep Q-network (Mnih et al., 2015) uses a deep convolutional neural network to estimate the action-value function, making deep reinforcement learning a powerful weapon for a myriad of applications (Van Hasselt et al., 2016; Wang et al., 2016; Hou et al., 2017). However, when applied to the musculoskeletal system, the performances of deep neural networks can be unstable. Given muscle redundancy in the musculoskeletal system, the additional dimensions expand the solution space, hindering optimization through reinforcement learning.

In this study, we focused on the unstable training of musculoskeletal systems and the expanded solution space of excitations to provide three contributions. (1) The learning goal of humans, changes stepwise as learning proceeds over advancing levels. For example, running requires higher physical coordination than walking, and one cannot run before learning to walk. Thereby, the learner target evolves from walking to running during this process. Based on this principle, we propose the phased target learning (PTL) framework that reduces the computational cost for exploration in a high-dimensional solution space. In addition, phased targets guide the convergence of excitations to the expected value during training. (2) As sensory information may be encoded by opposite tuning neurons (Romo and de Lafuente, 2013), we improve an MLP-based Q-network by introducing an extra layer of neurons reflecting preference and using various relative action probabilities from value functions for obtaining continuous outputs to control a musculoskeletal arm model. (3) As noise exists in the nervous system (A Aldo et al., 2008) and based on information transmission in the human nervous system (Dhawale et al., 2017), we introduce two noise sources at the sensor and execution levels into the proposed PTL framework. These noise sources increase the exploration capacity in the solution space during training and strengthen the control robustness.

In this paper, in section 2, we introduce the muscle dynamics, the structure of the arm model, and detail the musculoskeletal system considered in this study. Moreover, optimization of the proposed PTL framework is outlined. Then, the PTL framework with the biological noise sources is introduced in section 3.

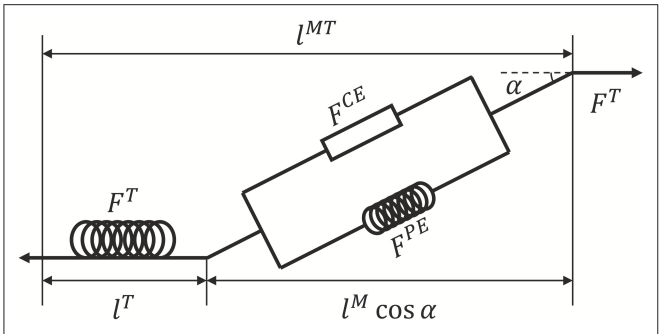


FIGURE 1 | Structure of a Hill-type muscle model. F^{CE} and F^{PE} are the active and passive forces, respectively, and F^T is the tendon force (Hill, 1938; Thelen, 2003).

Experimental results and conclusions are presented in sections 4 and 5, respectively.

2. MUSCULOTENDON MODEL AND MUSCULOSKELETAL ARM MODEL

Modeling muscles is difficult because most parameters cannot be measured precisely in real time (Arnold and Delp, 2011). According to the Hill model (Hill, 1938), which defines that a muscle is made up of separate elements, such as contractile elements (CE), passive elements (PE), and series elastic elements (SEE) (Zajac, 1989; Thelen et al., 2003), we design a control framework for musculoskeletal systems.

2.1. Musculotendon Model

To determine the way a human can control complex muscle systems, a muscle dynamic model is necessary. Let $u \in [0, 1]$ denote an idealized muscle excitation signal. According to a nonlinear first-order differential Equation (1), muscle activation signal a can be computed (Thelen, 2003):

$$\frac{da}{dt} = \frac{u - \hat{a}}{\tau(u, a)}, \tag{1}$$

where τ varies according to idealized muscle excitation signal u and activation signal a (Winters, 1995), $\hat{a}$ is the activation signal after normalization, and a is transmitted to the muscle contraction dynamic model as a final control signal.

Before introducing the muscle contraction dynamics, the structure of a Hill-type muscle model is shown in **Figure 1**, where l^T and l^M are the lengths of the tendon and muscle fiber, respectively, and α is the muscle pennation angle (Garner and Pandy, 2003). When the activation signal a is transmitted to the muscle, the corresponding muscle force is generated by contraction. Then, the muscle force pulls the skeletons to generate motion or to maintain the balance of forces.

Suppose that signal u is known. To calculate tendon force F^T, some assumptions are required. First, $F^T, F^{CE}, F^{PE} > 0$ because muscles move the skeleton by tension instead of thrusting. Second, the change of muscle width can be ignored during muscle contraction (Matthew et al., 2013). Third, muscle mass can be ignored. Using these assumptions, the dynamics of muscles can be described. Specifically, a pennation angle α can be obtained from

$$l_s^M \sin(\alpha_0) = l^M(t) \sin(\alpha(t)), \tag{2}$$

where l_s^M and α_0 are the slack length of a muscle fiber and initial pennation angle, respectively, which also define the initial muscle width, $l^M(t)$ and $\alpha(t)$ are the length of the muscle fiber and pennation angle at time t, respectively. From $\alpha(t)$, tendon force F^T can be computed by a piecewise nonlinear equation (Proske and Morgan, 1987; Thelen, 2003). In addition, the contraction velocity of a muscle fiber is necessary for the model. To determine this velocity, active force F^{CE} produced by the contractile element should be obtained first. According to the geometric relationship between tendon and muscle fiber (**Figure 1**), F^{CE} can be calculated indirectly as follows:

$$F^{CE} = \frac{F^T}{\cos(\alpha)} - F^{PE}, \tag{3}$$

where F^{PE} is the passive force of the muscle fiber. During simulations, the muscle length sometimes causes numerical problems that result in $F^{CE} < 0$, which clearly violate the first assumption about muscles. Therefore, a constraint should be added to avoid exceptional cases:

$$F^{CE} = \max\{F^{CE}, 0\}. \tag{4}$$

Then, contraction velocity v^M can be computed by another piecewise non-linear equation (Matthew et al., 2013):

$$v^M = f_v^{-1}\left(\frac{F^{CE}}{a f_l(l^M)}\right), \tag{5}$$

where f_v is the force–velocity function, f_v^{-1} is its inverse function, and f_l is a Gaussian function with variable l^M (Winters, 1990). As a key variable in the muscle dynamics model, $v^M(t)$ affects $l^M(t+1)$ at every timestep. Variable l^M is the fiber length and l^{MT} is the muscle length, which comprises fiber and tendon. Length l^M can be calculated directly using v^M and F^T, whereas l^{MT} can be measured. Consequently, if signal $u(t)$ is known, the contraction states of the muscle and tendon force $F^T(t)$ can be computed.

2.2. Musculoskeletal Arm Model

In the remainder of this section, we first establish a simplified arm model to connect muscles and bones. Then, we analyze the kinematic relationship between the arm model and muscle model. Finally, a control framework is outlined using this relationship.

According to the Newton–Euler equation (Zixing, 2000; Hahn, 2013), we establish a two degree-of-freedom model (**Figure 2**) that consists of two segments and four muscles. Then, expected torque τ_n at the joints can be calculated as

$$\tau_n = \frac{\partial W}{\partial \theta} = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta), \tag{6}$$

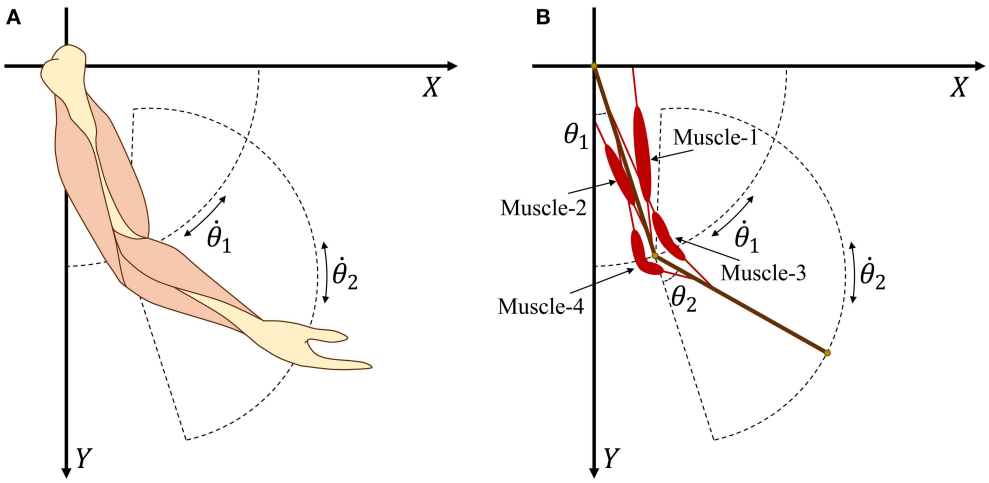


FIGURE 2 | (A) Human arm model with four main muscles. **(B)** Simplified arm model with four muscles.

where W is the work from external forces, $\dot{\theta}$ is the vector of rotational velocity, $\ddot{\theta}$ is the vector of rotational acceleration, $M(\theta) \in \mathbb{R}^{n\times n}$ and $C(\theta,\dot{\theta})\dot{\theta} \in \mathbb{R}^n$ is the inertia matrix and the centripetal and Coriolis force, respectively, and $G(\theta) \in \mathbb{R}^n$ is the gravitational force vector of our model. During forward calculation, Equation (6) provides a way to compute expected torques for known motion states. During inverse calculation, it can be used to compute actual angular acceleration.

2.3. Musculotendon Model Into Arm Model

In this section, we obtain the relationship between torques and motion states and define the adopted learning approach.

Unlike conventional robots that use a single joint motor to generate torque, each joint in a musculoskeletal system is usually affected by more than one muscle. Let τ_i be the muscle torque generated by muscle i:

$$\tau_i = F_i^T l_{i2} \sin\gamma_i,\ i = 1, 2, ..., n\,, \tag{7}$$

where F_i^T is the tendon force of muscle i and γ_i is the angle between the muscle and related bone. **Figure 3** provides geometric details of the muscles and bones. We set $m_1 = 2$ and $d_1 = 0.3$ as the mass and length of the upper arm, respectively, whereas $m_2 = 1.8$ and $d_2 = 0.3$ are the mass and length of the forearm, respectively. For the given geometry of the musculoskeletal model, the muscle torque can be written as

$$\begin{cases} \tau'_{n1} = \tau_1 - \tau_2 = F_1^T l_{12}\sin\gamma_1 - F_2^T l_{22}\sin\gamma_2 \\ \tau'_{n2} = \tau_3 - \tau_4 = F_3^T l_{32}\sin\gamma_3 - F_4^T l_{42}\sin\gamma_4 \end{cases}. \tag{8}$$

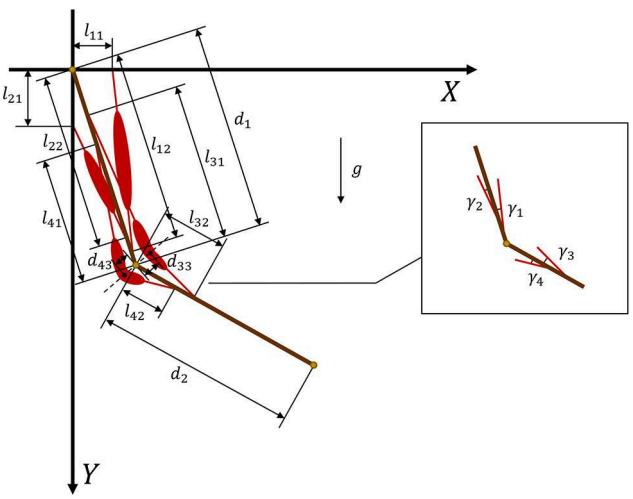


FIGURE 3 | Geometry and parameters of musculoskeletal arm model. Muscles 1 and 2 are defined by straight lines, whereas muscles 3 and 4 are defined by polylines.

In addition, the geometric parameters can be used to compute $\sin\gamma_i$:

$$\begin{cases} \sin\gamma_1 = \dfrac{l_{11}\cos\theta_1}{\sqrt{l_{11}^2+l_{12}^2-2l_{11}l_{12}\sin\theta_1}} \\ \sin\gamma_2 = \dfrac{l_{21}\sin\theta_1}{\sqrt{l_{21}^2+l_{22}^2-2l_{21}l_{22}\cos\theta_1}} \\ \sin\gamma_3 = \dfrac{d_{33}\cos\frac{\theta_2}{2}}{\sqrt{l_{32}^2+d_{33}^2-2l_{32}d_{33}\sin\frac{\theta_2}{2}}} \\ \sin\gamma_4 = \dfrac{d_{43}\cos\frac{\theta_2}{2}}{\sqrt{l_{42}^2+d_{43}^2+2l_{42}d_{43}\sin\frac{\theta_2}{2}}} \end{cases}. \tag{9}$$

In muscles 3 and 4 (**Figure 2**), we introduce two turning points at the angular bisector of the elbow to design polyline muscles, where d_{33} and d_{43} are the distances from the elbow to the

turning points of muscles 3 and 4, respectively. From Equation (9), it is clear that $\sin\gamma_i$ is a nonlinear function of θ_i. By substituting Equation (9) into Equation (8), we obtain muscle torque functions $\tau'_{n1}(F_1^T, F_2^T, \theta_1)$ and $\tau'_{n2}(F_3^T, F_4^T, \theta_2)$.

For the muscle description in our arm model, it is difficult to determine its inverse function, because F^T and Equation (5) are piecewise functions with complicated expressions. Therefore, we usually cannot calculate u_i by directly using muscle force, but instead we adopt an indirect method.

We assume that expected states $\theta_i, \dot{\theta}_i$ and $\ddot{\theta}_i$ are given. Expected torque τ_n can be calculated by Equation (6) as a learning target. On the other hand, actual tendon force F^T is known when corresponding excitation signals u are generated, and hence actual torque τ'_n is calculable. To obtain actual angular accelerations $\ddot{\theta}_i$, Equation (6) can be computed reversely. In general, $\ddot{\theta}$ can be rewritten as $\ddot{\theta}(\tau_n, \theta, \dot{\theta})$. Considering $\dot{\theta} = \frac{d\theta}{dt}$ and $\ddot{\theta} = \frac{d\dot{\theta}}{dt}$, joint angle θ at time $(t+1)$ can be obtained as

$$\theta_{t+1}(\tau_n, \theta_t(\ddot{\theta}_{t-1}), \dot{\theta}_t(\ddot{\theta}_{t-1}), \ddot{\theta}_t)\,. \tag{10}$$

If tendon force vector F^T satisfies

$$\tau_n(\theta, \dot{\theta}, \ddot{\theta}) = \tau'_n(F^T, \theta)\,, \tag{11}$$

we can rewrite Equation (11) as

$$\theta_{t+1}(\tau'_n(F_t^T, \theta_t), \theta_t(\ddot{\theta}_{t-1}), \dot{\theta}_t(\ddot{\theta}_{t-1}), \ddot{\theta}_t)\,. \tag{12}$$

The purpose of our framework is to find appropriate excitation signals u to generate tendon forces that satisfy Equation (11). As a result, the expected motions will be generated during exploration. Based on Equation (12), we establish a training framework for the musculoskeletal arm model. When excitation signal u is given, corresponding activation signal a and tendon force F^T can be calculated by muscle dynamics. Then, new motion states can be solved using the arm model. If excitation signal u is unknown, we should explore candidate solutions to generate F^T satisfying (Equation 11).

3. HUMAN-INSPIRED PHASED TARGET LEARNING FRAMEWORK

We design a learning framework to solve signal u_i. Conventional learning frameworks use expected states as the learning target. However, these targets can cause unforeseen problems during the solving process, and solutions can fall into local optima. In contrast, the proposed PTL framework can avoid local optima by guiding the learning process. Specifically, different learning targets are designed according to the learner's level, additionally providing high efficiency during training. We consider the musculoskeletal system, optimization model, and expected target state as the most essential aspects in our framework (**Figure 4**) and detail the last two parts in the sequel.

3.1. Phased Target Learning

3.1.1. Simplified Target Setup

Consider a beginner who starts to learn dancing or practicing a sport. It is difficult for him to acquire all the professional postures and skills at once. Instead of trying to enhance memory or learning skills, the simplest solution is reducing the quality requirements and perform intensive practice through gradual improvement. This way, the beginner will easily improve by establishing simple learning targets that are gradually set at different levels as learning proceeds. In this study, we calculated precise motion states to be expected targets. Then, we designed different simplified states as easier targets for learning. Formally, let $s \in \mathcal{S}$ be the expected states of the arm model, and $s_T \in \mathcal{S}_T$ be the simplified states. s_T can be calculated by simplifying s:

$$s_T(t) = s(ceil(\frac{t}{d}) \cdot d) \cdot \delta(0)\,, t = 1, 2, ..., T\,, \tag{13}$$

where $\delta(t)$ is an impulse function, and $d \in \mathbb{N}_+$ satisfying $\frac{d}{T} \in [\frac{1}{T}, 1]$ is a forgetting factor. When $d = T$, s_T only reflects the endpoint state of expected state s, and when $d = 1$, $s_T = s$, indicating that s_T reflects all the states of $\mathcal{S}$.

Obviously, simplification induces errors with respect to expected states. Suppose that $\theta \in \mathcal{S}$ is the expected joint angle of the arm model, and $\theta_T \in \mathcal{S}_T$ is the simplified joint angle. Then, we define the average allowed error between s and s_T as

$$\bar{e}_T = \frac{1}{T}\sum_{t=1}^{T} |\theta(t) - \theta_T(t)|\,. \tag{14}$$

According to Equations (13) and (14), average allowed error $\bar{e}_T$ depends only on the forgetting factor d. Geometrically, $\bar{e}_T$ can be considered as the width of the equivalent error region. **Figure 5** shows the width and effect of d on simplified joint angle curve θ_T.

PTL provides different simplified targets for learning at varying training phases. When the motion accuracy achieves the average allowed error range, $\bar{e}_T$, a new and smaller average allowed error range is given to guide training. Then, we define actual average error $\bar{e}_R$ of motion as

$$\bar{e}_R = \frac{1}{T}\sum_{t=1}^{T} |\theta_T(t) - \theta_R(t)|\,. \tag{15}$$

Unlike Equation (14), Equation (15) uses the actual joint angle, θ_R. In addition, $\bar{e}_T$ is updated after each training iteration. A new average allowed error is computed only when

$$\bar{e}_T - \bar{e}_R > 0\,. \tag{16}$$

$$d = \begin{cases} D(d), & \bar{e}_T - \bar{e}_R > 0, D(d) \geqslant 1; \\ 1, & \bar{e}_T - \bar{e}_R > 0, D(d) < 1; \\ d, & \bar{e}_T - \bar{e}_R \leqslant 0; \end{cases} \tag{17}$$

Equation (17) is the update rule of forgetting factor d, where $D(d)$ is a function that satisfies $D(d) < d$. It is convenient to maintain the value of $|d - D(d)|$ small, because a large difference between adjacent simplified states vanishes the gradual learning effect.

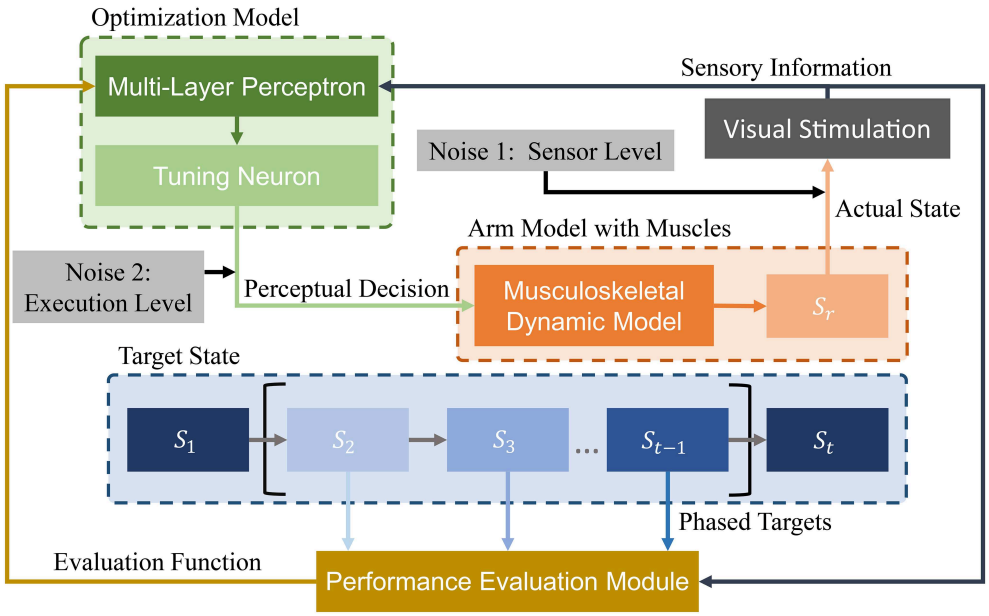


FIGURE 4 | Schematic of proposed PTL framework for motion control of musculoskeletal robots. A vision sensor collects motion information. Then, visual stimuli are transmitted to the optimization model and performance evaluation module. During optimization, state information is processed by a multi-layer perceptron. Then, perceptual decisions (excitation signals) are transmitted to the arm model as optimization results. During performance evaluation, different phased targets are designed to guide arm motion states. Finally, the evaluation results are transmitted to the optimization model for improved decision-making. In addition, two biological noise sources are considered during learning for improved exploration ability.

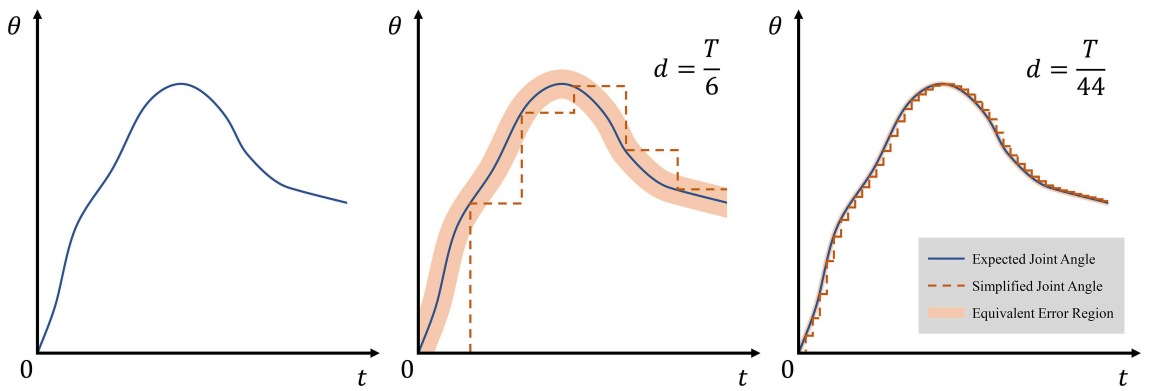


FIGURE 5 | Effect of forgetting factor d on equivalent error region. The thin blue line represents expected angles θ, and the dotted orange line represents simplified state θ_T. The equivalent error region is depicted with light orange and obtained as $\sum_{t=1}^{T} |\theta(t) - \theta_T(t)|$.

3.1.2. Performance Evaluation Function

Conventional temporal-difference learning methods are highly suitable for model-free learning. Considering Equation (11), the inverse function of τ_n' should be determined and can be set as a model-free problem. In this study, we aimed to improve the Q-network to estimate the continuous excitation signal u for musculoskeletal systems. Then, we combined it with PTL to calculate appropriate control signals.

Let T be the number of finite timesteps and u_i be the excitation signal for muscle i. Each signal $u_i(t)$ at time t has two possible actions; either increase $[a_{i,1}(t)]$ or decrease $[a_{i,2}(t)]$. The adjustment of u_i affects the muscle and musculoskeletal model at time $(t + 1)$.

However, the two actions only determine the increment sign, and additional parameters are required to calculate the step sizes. Furthermore, the difference between adjacent states can hinder perceptron learning from input states during training. Moreover, incorrect adjustments can lead to signal oscillation in the redundant musculoskeletal model.

In human cortical circuits, sensory information is encoded by neurons via opposite tuning (Romo and de Lafuente, 2013). Based on this mechanism, we redefine action-value function $Q_{u_i,j}$

as a probability of signal u_i executing action $a_{i,j}$. Equation (18) defines u_i as

$$u_i = \frac{1}{\sum_{j=1}^{2} Q_{u_i,j}}(Q_{u_i,1}u_{\max} + Q_{u_i,2}u_{\min}),\ i = 1, 2, ..., n \quad (18)$$

and action-value function $Q_{u_i,j}$ is redefined as

$$Q_{u_i,j}(s_t, a_{i,j,t}) = \mathbb{E}\left[E_{u_i}(s_{t+1}, a_{i,j,t+1}) + \gamma Q_{u_i,j}(s_{t+1}, a_{i,j,t+1})\right], \quad i = 1, 2, ..., n;\ j = 1, 2, \quad (19)$$

where E_{u_i} is an evaluation function related to arm motion. According to Equations (18) and (19), a specific action value of a function is not enough to obtain the excitation signal in our method. Instead, relative values of different functions determine an excitation signal, and thus $Q_{u_i,1}$ and $Q_{u_i,2}$ should be maintained balanced. In addition, note that E_{u_i} is used in Equation (19) instead of conventional reward function R_{u_i}. This is because the R_{u_i} is a decreasing function of the action error, and during training, reducing action errors increases R_{u_i} and $Q_{u_i,j}$. In this case, the balance of action-value functions is affected by increasing $Q_{u_i,j}$. Therefore, we employ evaluation function E_{u_i}, which is an increasing function of the action error. Reducing errors therefore imply smaller E_{u_i} and a weaker effect than R_{u_i} on the balance of action-value functions. Furthermore, $(E_{u_i})_{\min} > 0$ promotes stability, as detailed in section 3.1.3.

We obtain the performance evaluation function as follows:

$$E_{u_i}(e_R) = p \cdot \exp\left[m \cdot g^2(e_R)\right] + k \quad (20)$$

$$g(e_R) = \min\left[|e_R|, e_0\right], \quad (21)$$

where $p, m, k > 0$ are parameters of E_{u_i} and function $g(e_R)$ prevents exploding gradients under large errors.

3.1.3. Learning by Gradient Descent

We define the loss function by summing the squared errors between expected action value $Q'_{u_i,j}$ and actual action value $E_{u_i,j} + \gamma\ Q'_{u_i,j}$:

$$L(\theta) = \frac{1}{2}\mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{2}\left(E_{u_i,j} + \gamma Q'_{u_i,j}(s', a'; \theta') - Q_{u_i,j}(s, a; \theta)\right)^2\right], \quad (22)$$

where γ is a factor to discount the future action value. The gradient of the loss function is given by

$$\nabla L(\theta) = \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{2}\gamma\left(E_{u_i,j} + \gamma Q'_{u_i,j}(s', a'; \theta') - Q_{u_i,j}(s, a; \theta)\right) \nabla Q'_{u_i,j}(s', a'; \theta')\right]. \quad (23)$$

During backpropagation, the outputs of multi- layer perceptron in our model can be easily obtained. We suppose that $Q'_{u_i,j}$ represents the result of the output layer and can be expressed as

$$Q'_{u_i,j} = f\left(\sum_{h=1}^{n_h}\omega_{hk}y_h\right), \quad (24)$$

where $f(x)$ is the sigmoid activation function, ω_{hk} is the weight of the edge from the h-th node in the hidden layer to the k-th node in the output layer. Consider ω_{hk} as an example, the weight increment is given by

$$\triangle\omega_{hk} = -\eta\frac{\partial L}{\partial\omega_{hk}} \quad (25)$$

$$= -\eta\sum_{i=1}^{n}\sum_{j=1}^{2}\gamma(E_{u_i,j} + \gamma Q'_{u_i,j} - Q_{u_i,j})\frac{\partial Q'_{u_i,j}}{\partial\omega_{hk}} \quad (26)$$

$$= -\eta\sum_{i=1}^{n}\sum_{j=1}^{2}\gamma(E_{u_i,j} + \gamma Q'_{u_i,j} - Q_{u_i,j})f'\left(\sum_{h=1}^{n_h}\omega_{hk}y_h\right)\sum_{h=1}^{n_h}y_h, \quad (27)$$

where y_h is the output of the h-th node in the hidden layer. When the excitations become stable, the expected increment is $\Delta\omega_{hk} \to 0$ such that $\Delta Q_{u_i,j} \to 0$, and hence $E_{u_i,j} + \gamma Q'_{u_i,j} = Q_{u_i,j}$ at this time. Factor γ is known as a decimal, and we can infer $\gamma Q'_{u_i,j} < Q_{u_i,j}$, which explains why the performance evaluation function should satisfy $(E_{u_i,j})_{\min} > 0$.

3.2. Noise in Nervous System

Noise is ubiquitous in real-world systems, especially during information transmission. As motion learning consists of information transmission, noise is present. Recent research roughly identified noise sources in the nervous system at the sensor and action levels (A Aldo et al., 2008). We considered these noise sources in the proposed PTL framework.

3.2.1. Noise at Sensor Level

During the collection of visual information, photoreceptors receive photons reflected by objects under the influence of Poisson noise, which reduces the accuracy of optical information (Bialek, 1987). Although sensory noise is inevitable (Bialek and Setayeshgar, 2005), it also mitigates sensitivity of the redundant musculoskeletal system.

When motion tracking is performed on the redundant musculoskeletal arm model, the Q-network method can exhibit unstable training, because joint angles are affected by the action of many muscles, likely falling into local optima. Then, any small fluctuation of excitation signals can be amplified and cause divergent signals. However, when target motion is considered as a region, fluctuations are tolerated. We use Poisson noise to conform tolerance regions and prevent rapid fluctuations:

$$s_{RN} = s_R + N_1, \quad (28)$$

$$N_1 \sim Pois(\lambda), \quad (29)$$

where s_R is the actual arm state, s_{RN} is the observed arm state observed by the vision sensor, and N_1 is Poisson noise in the visual information. In our algorithm, let $s_R = s_{RN}$ represents the inputs of the improved Q-network.

3.2.2. Noise at Execution Level

Noise at the sensor level is also called planning noise, as it affects decision-making. In addition, execution noise exists and is superimposed on the original decision signals. In fact, execution

noise describes an uncontrollable noise whose standard deviation is linearly related to the mean muscle force (Hamilton et al., 2004; Dhawale et al., 2017) and can be expressed as

$$u_{Ni} = \min[\max[u_i + N_2, 0], 1], \tag{30}$$

where $N_2 \sim N(0, (vF^T)^2)$ simulates noise in the motor system periphery, u_i and u_{Ni} are undisturbed and noisy signals from perceptron, respectively, and v is a scale coefficient of tendon force F^T. Note that the square of vF^T defines the variance of execution noise, and like noise in sensor level, let $u_i = u_{Ni}$ represent the final outputs of the proposed network.

4. SIMULATION EXPERIMENTS

We conducted simulation experiments on the musculoskeletal system model to verify the performance of different algorithms. Moreover, the equilibria of action values are analyzed to explain the learning process of the proposed PTL framework.

4.1. Experimental Setup

As mentioned above, we designed a simplified musculoskeletal arm model to verify and evaluate the proposed learning method. After analyzing its dynamics (Equation 12), a basic control framework is devised. To validate the formulation and analyze performance, optimization should be performed.

In this study, the proposed PTL is applied to a point-to-point motion task with constant angular velocity as temporal-difference learning approach. For a final state of target motion, we calculated midpoints and required constraints using inverse kinematics. Then, we used joint angles as motion states to design the simplified target states. Assuming a constant angular velocity, four types of control strategies were evaluated: (1) Q-network, (2) Q-network with noises, (3) PTL, and (4) PTL with noises. The implemented method including PTL is detailed in Algorithm 1.

We set maximum number of iterations $K = 500$ and number of timesteps $T = 10,000$ to simulate 10 s. All the errors and control signals were recorded at each timestep.

4.2. Results and Analysis

We considered average error $\bar{e} = \frac{1}{T}\sum_{t=1}^{T} |\theta(t) - \theta_R(t)|$ as a key performance indicator, where $\theta(t)$ is the precise expected joint angle at time t. As $\bar{e}$ reflects the average error, motion performance can be evaluated from this measure.

Figure 6 shows the average error $\bar{e}$ according to iteration k. Clearly, the Q-network method, Q-network with noises, and PTL are trapped at local optima and unstable during training. Still, phased targets improve learning by increasing the randomness of exploration, and noises during training enhance fault tolerance and the exploration ability during control.

Assume that the ratio of action-value functions is convergent to local optimum b_i, which is defined as

$$b_i = \frac{Q_{u_i,1}}{Q_{u_i,2}}. \tag{31}$$

Algorithm 1: PTL with Noises for Motion Learning in Musculoskeletal System.

Require: Given precise motion states $s(t) \in \mathcal{S}$. Initialize parameters: interval d, maximum number of iterations K, excitation signal u_i. Obtain simplified motion state $s_T(t) \in \mathcal{S}_T$ using Equation 13.

for k=1 **to** K **do**
  Compute average allowed error $\bar{e}_T = \frac{1}{T}\sum_{t=1}^{T} |s(t) - s_T(t)|$
  if $\bar{e}_R < \bar{e}_T$ and $k \neq 1$ **then**
    Reduce d gradually ($d \in N_+$, $d_{\max} < T$)
    Set new target states $\mathcal{S}_\mathcal{T}$ by simplifying $\mathcal{S}$
  end if
  for t=0 **to** T **do**
    Calculate activation signal $a_i(u_i(t))$ and tendon force F_i^T
    Perform motion corresponding to $s_R(t+1)$ caused by F_i^T
    Obtain actual motion error $e_R(t) = |s_T(t) - s_R(t)|$
    Introduce noise at sensor level into motion states via Equation 28. Let $s_R(t) = s_{RN}(t)$ be the inputs of improved Q-network
    Estimate $Q'_{u_i,j}$ by improved Q-network method
    Update weights ω to obtain new action values $Q_{u_i,j}$ via Equation 23
    Obtain signal $u_i(t)$ using Equation 18
    Introduce noise at execution level into excitation signals via Equation 30. Let $u_i(t) = u_{Ni}(t)$ be the outputs of improved Q-network
  end for
end for

Then, u_i can be rewritten as

$$u_i = \frac{1}{\sum_{j=1}^{2} Q_{u_i,j}}(Q_{u_i,1}u_{\max} + Q_{u_i,2}u_{\min}) \tag{32}$$

$$= \frac{1}{b_i + 1}(b_i u_{\max} + u_{\min}), \tag{33}$$

and hence the equilibrium point b_i is the only parameter that affects excitation signal u_i. We prescribe that the control method adjusts $Q_{u_i,1}$ and $Q_{u_i,2}$ in an opposite way. In addition, increment $\Delta Q_{u_i,j}$ satisfies $\Delta Q_{u_i,j} > 0$ and $\Delta Q_{u_i,j} \ll Q_{u_i,j}$ at simulation onset. The next equilibrium point at time $(t+1)$ is $b'_i = (Q_{u_i,1} \mp \Delta Q_{u_i,1})/(Q_{u_i,2} \pm \Delta Q_{u_i,2})$, whose increment is given by

$$b_i - b'_i = \frac{Q_{u_i,1}}{Q_{u_i,2}} - \frac{Q_{u_i,1} \mp \Delta Q_{u_i,1}}{Q_{u_i,2} \pm \Delta Q_{u_i,2}}, \tag{34}$$

$$= \frac{\pm(Q_{u_i,1}\Delta Q_{u_i,2} + \Delta Q_{u_i,1}Q_{u_i,2})}{Q_{u_i,2}(Q_{u_i,2} \pm \Delta Q_{u_i,2})}. \tag{35}$$

For $(-\Delta Q_{u_i,1}, +\Delta Q_{u_i,2})$, we obtain $b_i - b'_i > 0$, and excitation signal u_i becomes smaller. For $(+\Delta Q_{u_i,1}, -\Delta Q_{u_i,2})$, as $Q_{u_i,2} - \Delta Q_{u_i,2} > 0$, we obtain $b_i - b'_i < 0$, and excitation signal u_i becomes larger.

However, with reducing motion error, the increment of function $Q_{u_i,j}$ is smaller for $Q_{u_i,j} \approx \Delta Q_{u_i,j}$. From Equation

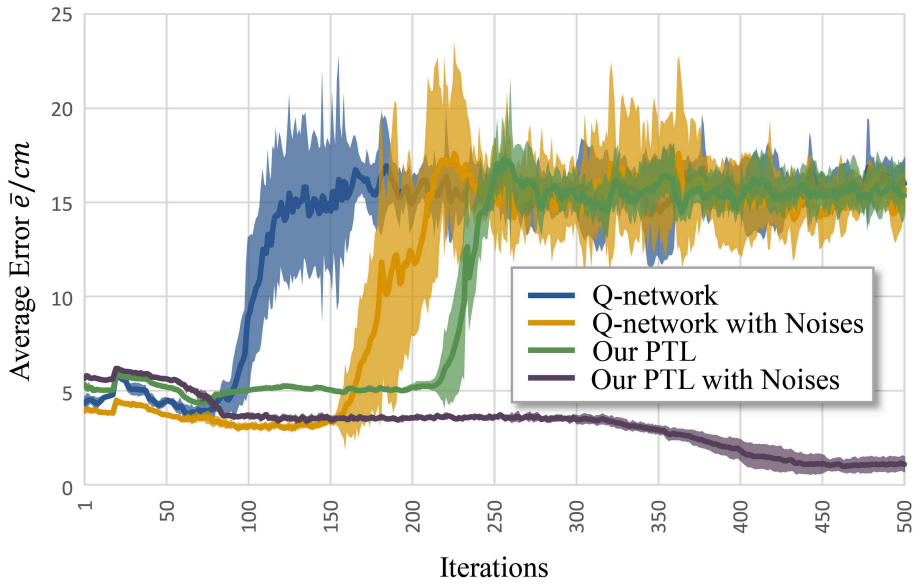


FIGURE 6 | Average error for different methods to control musculoskeletal arm model for motion tracking. Curves correspond to average errors over 10 trials.

(35), when $(+\Delta Q_{u_i,1}, -\Delta Q_{u_i,2})$, the sign of $(b_i - b_i')$ depends on the sign of $(Q_{u_i,2} - \Delta Q_{u_i,2})$. Nevertheless, it is difficult to guarantee either $(Q_{u_i,2} \leqslant \Delta Q_{u_i,2})$ or $(Q_{u_i,2} \geqslant \Delta Q_{u_i,2})$. The uncertain sign causes chattering on the excitation signal (Equation 33), which can cause signal divergence at the final state.

In addition, random factors like ϵ and noise can give rise to fluctuations of $\Delta Q_{u_i,j}$, which may increase the adjustment extent. For example, if $(+\Delta Q_{u_i,1}, +\Delta Q_{u_i,2})$ or $(-\Delta Q_{u_i,1}, -\Delta Q_{u_i,2})$, the increment of b_i is given by

$$b_i - b_i' = \frac{Q_{u_i,1}}{Q_{u_i,2}} - \frac{Q_{u_i,1} \pm \Delta Q_{u_i,1}}{Q_{u_i,2} \pm \Delta Q_{u_i,2}} \tag{36}$$

$$= \frac{\pm(Q_{u_i,1}\Delta Q_{u_i,2} - \Delta Q_{u_i,1}Q_{u_i,2})}{Q_{u_i,2}(Q_{u_i,2} \pm \Delta Q_{u_i,2})}, \tag{37}$$

where $(Q_{u_i,1}\Delta Q_{u_i,2} - \Delta Q_{u_i,1}Q_{u_i,2})$ with an uncertain sign can seriously undermine performance, as it is directly related to the sign of $(b_i - b_i')$. Furthermore, performance may decay even without condition $Q_{u_i,j} \approx \Delta Q_{u_i,j}$, and the method will be unreliable under its influence. Fortunately, with appropriate training, performance degradation by random effects can almost be eliminated.

Another problem is early convergence during learning. **Figure 7** shows the evolution of the average allowed error. The four evaluated methods terminate searching when reaching different local optima. Generally, premature convergence occurs through the insufficient exploration of solutions. Given its exploration ability, the proposed PTL with noises was guided by simplified targets to avoid premature convergence. This method achieved the lowest error (average $\bar{e} < 0.746cm$) and the most advanced learning level throughout repeated experiments.

$$b_i = \frac{Q_{u_i,1}}{Q_{u_i,2}} + \Delta b_i \tag{38}$$

We define Δb_i in Equation (38) as a small increment of the equilibrium point caused by the allowed error e_T. As $\frac{Q_{u_i,1}}{Q_{u_i,2}}$ is not at the expected equilibrium point b_i, $Q_{u_i,j}$ cannot easily generate large fluctuations. According to the analyses above, $\frac{Q_{u_i,1}}{Q_{u_i,2}}$ will converge to the final equilibrium point b_i when $t = T$.

Figure 8 shows signal u_i learned using PTL with noises and the corresponding tendon force, F_i^T. **Figure 9** shows the final position of the arm and joint angles. These results show that the most substantial errors occur at motion onset, and only slight fluctuations remain afterwards. At motion onset, it is reasonable to believe that unexpected muscle forces, especially passive forces of muscles 1 and 3, disturb the force balance. As the simulation proceeds, the arm model returns to a balance state by adjusting u_i. Therefore, PTL extends learning and guides toward the next expected solutions. In addition, the noises foster an extensive exploration of the solution space during training.

To further evaluate PTL framework, we consider point-to-point motion through two scenarios. First, motion begins from a stable position ($\theta_i = 0$) and finishes at another position (**Figure 10**).

When motion starts from a stable position, the next state s_{t+1} does not considerably change if $F^T = 0$. Therefore, the algorithm should not deal with large and rapid fluctuations, and the PTL performance is high. In contrast, in the second scenario, motion starts from an unstable position, and s_{t+1} exhibits a large difference when compared with s_t in the initial period

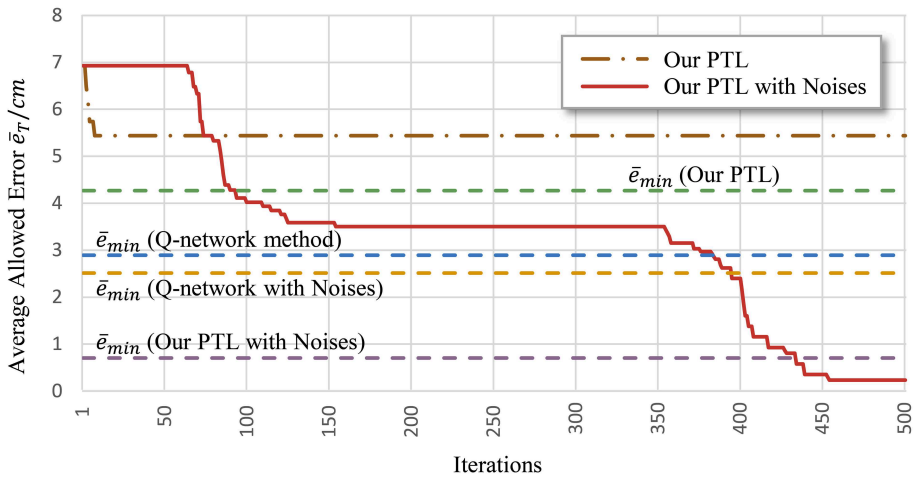


FIGURE 7 | Average allowed error during training. Most algorithms stop learning before processing all the simplified targets.

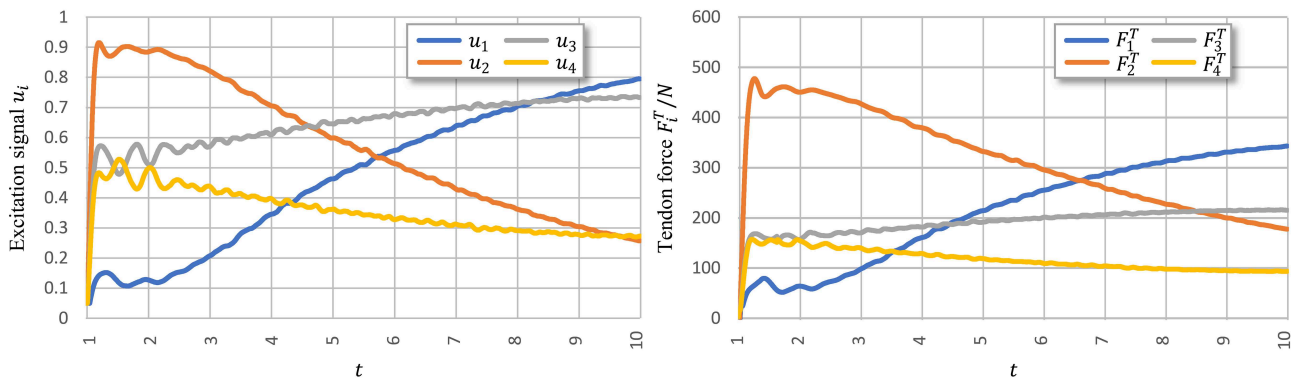


FIGURE 8 | Execution signals trained using PTL with noises after 500 iterations. All excitation signals are filtered with a Butterworth lowpass filter to separate signals from execution noise.

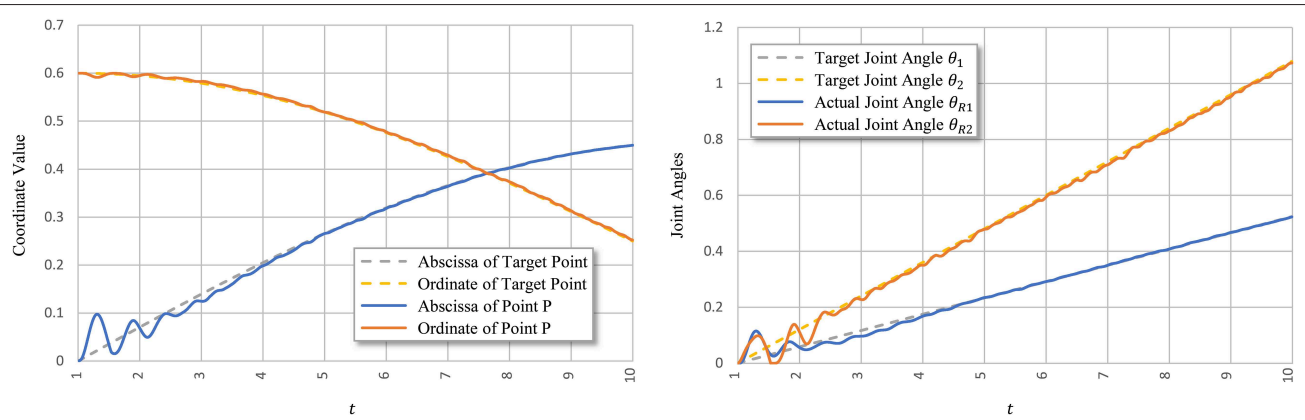


FIGURE 9 | Tracking performance of PTL with noises. Point *P* is the terminal point for arm motion.

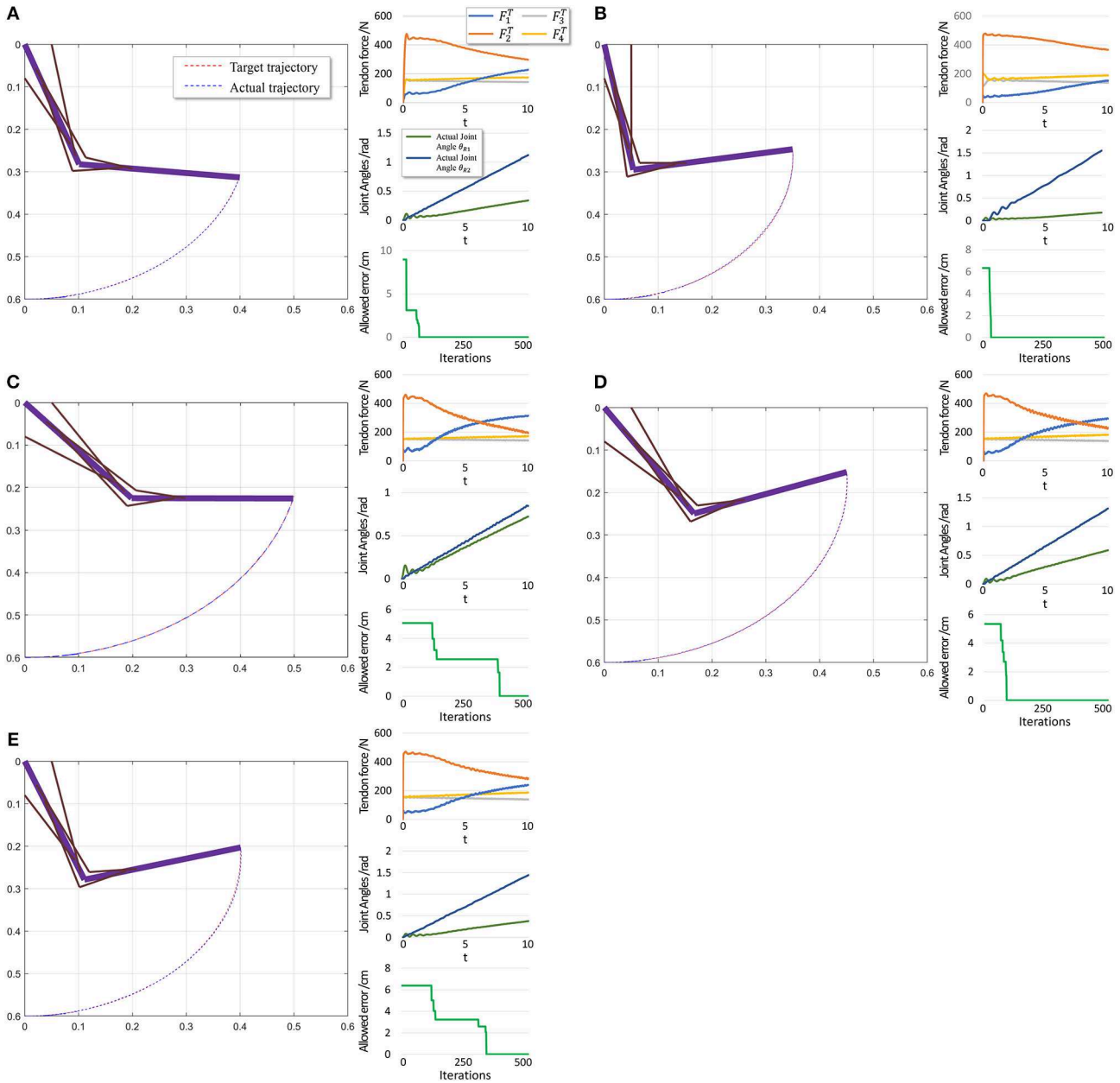


FIGURE 10 | Scenario 1: Motion with constant angular velocity begins from a stable position and reaches another position in the motion space. **(A–E)** are five different trajectories selected randomly from operation space. Especially, all the initial states are the same ($\theta_1 = 0, \theta_2 = 0$). In each situation, Left: actual motion trajectory of endpoint achieved by PTL. Right: (Top) corresponding tendon forces caused by signal μ. (Middle) Actual joint angles during the motion. Remember that each trajectory task is required a constant angular velocity. (Bottom) Allowed error during training, which can be considered as phased target of motion learning.

even if $F^T = 0$, as the gravitational torque contributes to a large angular acceleration. Consequently, learning is unstable.

The performance in the second scenario (**Figure 11**) confirms our prediction of large initial fluctuations. In fact, inappropriate initial parameters in musculotendon model will also degrade the performance. As inappropriate parameters lead to inappropriate muscle force, and some timesteps are necessary to adjust those parameters. In addition, the trajectory length is notably shorter than that in the first scenario, leading to a shorter trajectory for adjustment and learning. Consequently, errors increase in this scenario.

5. CONCLUSIONS

In this paper, we propose a human-inspired motion learning framework for a musculoskeletal system, called PTL. We analyze the learning process and equilibrium point of $Q_{u_i,j}$,

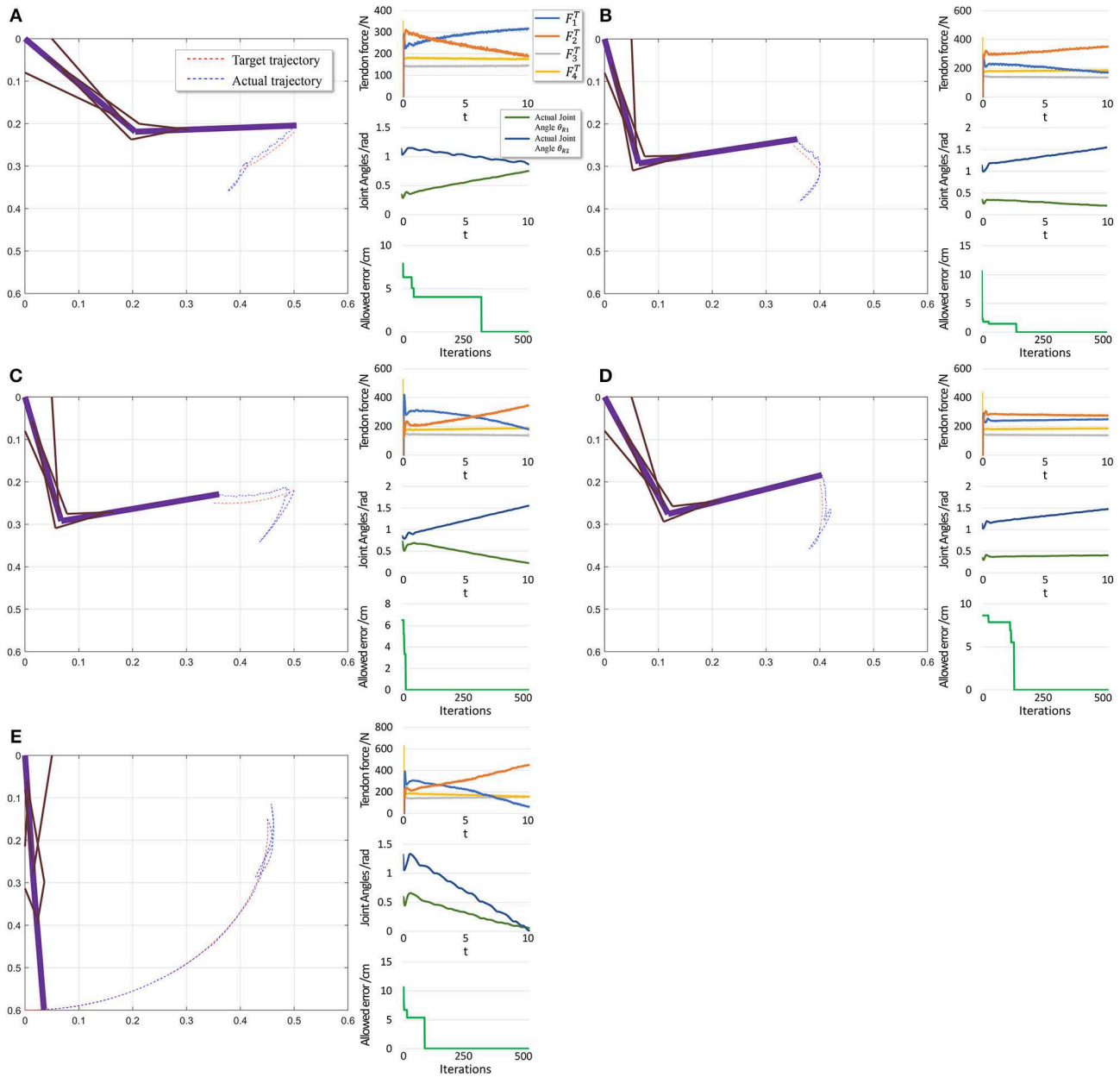


FIGURE 11 | Scenario 2: Motion with constant angular velocity begins from an unstable position and reaches another position in the motion space. **(A–D)** are five different trajectories selected randomly from operation space. In each situation, Left: actual motion trajectory of endpoint achieved by PTL. Right: (Top) corresponding tendon forces caused by signal μ. (Middle) Actual joint angles during the motion. Remember that each trajectory task is required a constant angular velocity. (Bottom) Allowed error during training, which can be considered as phased target of motion learning. To evaluate performance, situation **(E)** is designed particularly to move from a unstable state to the stable position ($\theta_1 = 0$ *and* $\theta_2 = 0$).

determining that phased targets guide excitation signals toward expected values during learning. Two types of biological noise sources are considered in the PTL framework to increase the exploration ability in an expanded solution space, making the algorithm suitably follow the guidance of phased targets. Theoretically, as PTL is based on a human learning process, it can be expanded as a general-purpose learning framework if we find appropriate ways to simplify different kinds of tasks, such as capture and pattern recognition tasks.

In future work, we will apply advanced methods in PTL to improve performance, especially when motion starts from an unstable position. Furthermore, better approaches for simplifying tasks and more biological mechanisms of motion control should be investigated to expand the application scope of the PTL framework.

DATA AVAILABILITY

All datasets generated for this study are included in the manuscript and/or the supplementary files.

AUTHOR CONTRIBUTIONS

JZ provided the main ideas of this research, wrote the manuscript and codes of experiments. JC and HD provided suggestions about PTL framework. HQ and other authors discussed and revised the manuscript.

FUNDING

This work was supported in part by the National Key Research and Development Program of China (2017YFB1300200, 2017YFB1300203), the National Natural Science Foundation of China under Grants 91648205 and 61627808, the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32000000, and the development of science and technology of Guangdong Province special fund project under Grant 2016B090910001.

REFERENCES

A Aldo, F., Selen, L. P. J., and Wolpert, D. M. (2008). Noise in the nervous system. *Nat. Rev. Neurosci.* 9, 292–303. doi: 10.1038/nrn2258

Agarwal, G. C., Berman, B. M., and Stark, L. (1970). Studies in postural control systems part I: Torque disturbance input. *IEEE Trans. Syst. Sci. Cybern.* 6, 116–121. doi: 10.1109/TSSC.1970.300285

Arnold, E. M., and Delp, S. L. (2011). Fibre operating lengths of human lower limb muscles during walking. *Philos. Trans. R. Soc. B Biol. Sci.* 366, 1530–1539. doi: 10.1098/rstb.2010.0345

Asano, Y., Kozuki, T., Ookubo, S., Kawasaki, K., Shirai, T., Kimura, K., et al. (2015). "A sensor-driver integrated muscle module with high-tension measurability and flexibility for tendon-driven robots," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg: IEEE), 5960–5965.

Asano, Y., Okada, K., and Inaba, M. (2017). Design principles of a human mimetic humanoid: humanoid platform to study human intelligence and internal body system. *Sci. Robot.* 2:eaaq0899. doi: 10.1126/scirobotics.aaq0899

Bialek, W. (1987). Physical limits to sensation and perception. *Annu. Rev. Biophys. Biophys. Chem.* 16, 455–478. doi: 10.1146/annurev.bb.16.060187.002323

Bialek, W., and Setayeshgar, S. (2005). Physical limits to biochemical signaling. *Proc. Natl. Acad. Sci. U.S.A.* 102, 10040–10045. doi: 10.1073/pnas.0504321102

Chen, J., Zhong, S., Kang, E., and Qiao, H. (2018). Realizing human-like manipulation with musculoskeletal system and biologically inspired control. *Neurocomputing* 339, 116–129. doi: 10.1016/j.neucom.2018.12.069

Cook, G., and Stark, L. (1968). The human eye-movement mechanism: experiments, modeling, and model testing. *Arch. Ophthalmol.* 79, 428–436. doi: 10.1001/archopht.1968.03850040430012

Crowninshield, R. D., and Brand, R. A. (1981). A physiologically based criterion of muscle force prediction in locomotion. *J. Biomech.* 14, 793–801. doi: 10.1016/0021-9290(81)90035-X

Dhawale, A. K., Smith, M. A., and Ölveczky, B. P. (2017). The role of variability in motor learning. *Annu. Rev. Neurosci.* 40, 479–498. doi: 10.1146/annurev-neuro-072116-031548

Diuk, C., Cohen, A., and Littman, M. L. (2008). "An object-oriented representation for efficient reinforcement learning," in *Proceedings of the 25th International Conference on Machine Learning* (ACM), 240–247.

Eisenberg, E., Hill, T. L., and Chen, Y.-D. (1980). Cross-bridge model of muscle contraction. quantitative analysis. *Biophys. J.* 29, 195–227. doi: 10.1016/S0006-3495(80)85126-5

Garner, B. A., and Pandy, M. G. (2003). Estimation of musculotendon properties in the human upper limb. *Ann. Biomed. Eng.* 31, 207–220. doi: 10.1114/1.1540105

Hahn, H. (2013). *Rigid Body Dynamics of Mechanisms: 1 Theoretical Basis*. Springer Science & Business Media.

Haines, C. S., Lima, M. D., Li, N., Spinks, G. M., Foroughi, J., Madden, J. D., et al. (2014). Artificial muscles from fishing line and sewing thread. *Science* 343, 868–872. doi: 10.1126/science.1246906

Hamilton, A. F. D. C., Jones, K. E., and Wolpert, D. M. (2004). The scaling of motor noise with muscle strength and motor unit number in humans. *Exp. Brain Res.* 157, 417–430. doi: 10.1007/s00221-004-1856-7

Hill, A. V. (1938). The heat of shortening and the dynamic constants of muscle. *Proc. R. Soc. Lond. Ser. B Biol. Sci.* 126, 136–195. doi: 10.1098/rspb.1938.0050

Hou, Y., Liu, L., Wei, Q., Xu, X., and Chen, C. (2017). "A novel ddpg method with prioritized experience replay," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (IEEE), 316–321.

Huxley, A. F. (1957). Muscle structure and theories of contraction. *Prog. Biophys. Biophys. Chem.* 7, 255–318. doi: 10.1016/S0096-4174(18)30128-8

Huxley, A. F., and Niedergerke, R. (1954). Structural changes in muscle during contraction: interference microscopy of living muscle fibres. *Nature* 173, 971–973. doi: 10.1038/173971a0

Jagodnik, K. M., and van den Bogert, A. J. (2010). Optimization and evaluation of a proportional derivative controller for planar arm movement. *J. Biomech.* 43, 1086–1091. doi: 10.1016/j.jbiomech.2009.12.017

Jäntsch, M., Wittmeier, S., Dalamagkidis, K., Panos, A., Volkart, F., and Knoll, A. (2013). "Anthrob-a printed anthropomimetic robot," In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (Atlanta, GA: IEEE), 342–347.

Kurumaya, S., Suzumori, K., Nabae, H., and Wakimoto, S. (2016). Musculoskeletal lower-limb robot driven by multifilament muscles. *Robomech J.* 3:18. doi: 10.1186/s40648-016-0061-3

Law, C.-T., and Gold, J. I. (2009). Reinforcement learning can account for associative and perceptual learning on a visual-decision task. *Nat. Neurosci.* 12, 655–663. doi: 10.1038/nn.2304

Matthew, M., Thomas, U., Ajay, S., and Delp, S. L. (2013). Flexing computational muscle: modeling and simulation of musculotendon dynamics. *J. Biomech. Eng.* 135:021005. doi: 10.1115/1.4023390

Miriyev, A., Stack, K., and Lipson, H. (2017). Soft material for soft actuators. *Nat. Commun.* 8:596. doi: 10.1038/s41467-017-00685-3

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature* 518, 529–533. doi: 10.1038/nature14236

Pennestrì, E., Stefanelli, R., Valentini, P., and Vita, L. (2007). Virtual musculo-skeletal model for the biomechanical analysis of the upper limb. *J. Biomech.* 40, 1350–1361. doi: 10.1016/j.jbiomech.2006.05.013

Proske, U., and Morgan, D. L. (1987). Tendon stiffness: methods of measurement and significance for the control of movement. A review. *J. Biomech.* 20, 75–82. doi: 10.1016/0021-9290(87)90269-7

Rasmussen, J., Damsgaard, M., and Voigt, M. (2001). Muscle recruitment by the min/max criterion–a comparative numerical study. *J. Biomech.* 34, 409–415. doi: 10.1016/S0021-9290(00)00191-3

Riedmiller, M., Gabel, T., Hafner, R., and Lange, S. (2009). Reinforcement learning for robot soccer. *Auton. Robots* 27, 55–73. doi: 10.1007/s10514-009-9120-4

Romo, R., and de Lafuente, V. (2013). Conversion of sensory signals into perceptual decisions. *Prog. Neurobiol.* 103, 41–75. doi: 10.1016/j.pneurobio.2012.03.007

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science* 275, 1593–1599. doi: 10.1126/science.275.5306.1593

Shi, P., and McPhee, J. (2000). Dynamics of flexible multibody systems using virtual work and linear graph theory. *Multibody Syst. Dyn.* 4, 355–381. doi: 10.1023/A:1009841017268

Stoianovici, D., and Hurmuzlu, Y. (1996). A critical study of the applicability of rigid-body collision theory. *J. Appl. Mech.* 63, 307–316. doi: 10.1115/1.2788865

Sutton, R. S., and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tahara, K., and Kino, H. (2010). "Iterative learning scheme for a redundant musculoskeletal arm: Task space learning with joint and muscle redundancies," in *2010 International Conference on Broadband, Wireless Computing, Communication and Applications* (Fukuoka: IEEE), 760–765.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Commun. ACM* 38, 58–69. doi: 10.1145/203330.203343

Thelen, D. G. (2003). Adjustment of muscle mechanics model parameters to simulate dynamic contractions in older adults. *J. Biomech. Eng.* 125, 70–77. doi: 10.1115/1.1531112

Thelen, D. G., Anderson, F. C., and Delp, S. L. (2003). Generating dynamic simulations of movement using computed muscle control. *J. Biomech.* 36, 321–328. doi: 10.1016/S0021-9290(02)00432-3

Van Hasselt, H., Guez, A., and Silver, D. (2016). "Deep reinforcement learning with double Q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence* (Phoenix, AR).

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., Freitas, N. (2016). "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning* (New York, NY: PMLR), 1995–2003. Available online at: http://proceedings.mlr.press/v48/wangf16.pdf

Winters, J. M. (1990). "Hill-based muscle models: a systems engineering perspective," in *Multiple Muscle Systems Biomech. & Movem.organiz*, eds J. M. Winters and S. LY. Woo (New York, NY: Springer), 69–93.

Winters, J. M. (1995). An improved muscle-reflex actuator for use in large-scale neuromusculoskeletal models. *Ann. Biomed. Eng.* 23, 359–374. doi: 10.1007/BF02584437

Winters, J. M., and Stark, L. (1987). Muscle models: what is gained and what is lost by varying model complexity. *Biol. Cybern.* 55, 403–420. doi: 10.1007/BF00318375

Zahalak, G. I., and Ma, S.-P. (1990). Muscle activation and contraction: constitutive relations based directly on cross-bridge kinetics. *J. Biomech. Eng.* 112, 52–62. doi: 10.1115/1.2891126

Zajac, F. E. (1989). Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. *Crit. Rev. Biomed. Eng.* 17, 359–411.

Zixing, C. (2000). *Robotics*. Beijing: Tsinghua University Press.

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 10 October 2019
doi: 10.3389/fnbot.2019.00082

Robust Event-Based Object Tracking Combining Correlation Filter and CNN Representation

Hongmin Li and Luping Shi**

Department of Precision Instrument, Center for Brain-Inspired Computing Research, Tsinghua University, Beijing, China

Object tracking based on the event-based camera or dynamic vision sensor (DVS) remains a challenging task due to the noise events, rapid change of event-stream shape, chaos of complex background textures, and occlusion. To address the challenges, this paper presents a robust event-stream object tracking method based on correlation filter mechanism and convolutional neural network (CNN) representation. In the proposed method, rate coding is used to encode the event-stream object. Feature representations from hierarchical convolutional layers of a pre-trained CNN are used to represent the appearance of the rate encoded event-stream object. Results prove that the proposed method not only achieves good tracking performance in many complicated scenes with noise events, complex background textures, occlusion, and intersected trajectories, but also is robust to variable scale, variable pose, and non-rigid deformations. In addition, the correlation filter-based method has the advantage of high speed. The proposed approach will promote the potential applications of these event-based vision sensors in autonomous driving, robots and many other high-speed scenes.

Keywords: event-based vision, object tracking, dynamic vision sensor, convolutional neural network, correlation filter

OPEN ACCESS

Edited by:
Guang Chen,
Tongji University, China

Reviewed by:
Liu Xiang Yong,
Tongji University, China
Rui Yan,
Sichuan University, China

****Correspondence:***
Hongmin Li
lihongmin0110@gmail.com
Luping Shi
lpshi@mail.tsinghua.edu.cn

Received: *25 March 2019*
Accepted: *20 September 2019*
Published: *10 October 2019*

Citation:
Li H and Shi L (2019) Robust Event-Based Object Tracking Combining Correlation Filter and CNN Representation. Front. Neurorobot. 13:82. doi: 10.3389/fnbot.2019.00082

INTRODUCTION

Different from the traditional frame-based imager, event-based camera, or dynamic vision sensor (DVS) converts the temporal contrast of the light intensity into spatiotemporal, sparse event streams (Lichtsteiner et al., 2008; Serrano-Gotarredona and Linares-Barranco, 2013; Brandli et al., 2014). DVS has the advantages of low information redundancy, high dynamic range, and high speed in visual sensing, and has the potential applications in the high-speed scenes. Recently, DVS has been used for estimating the high-speed optical flow and intensity field (Kim et al., 2008; Bardow et al., 2016). A visual processing system based on event camera demands low energy consumption. The outputted events are represented in the form of Address-Event Representation (AER) (Boahen, 2000). AER is often utilized to model the signal of neural systems, like the retina using discrete time events to convey information, and other spike coded neural systems in living organisms.

Visual tracking has a wide range of applications in the fields of autonomous driving, robot vision, trajectory analysis and so on. When an object is detected at a certain moment, it is often useful to track that object in subsequent recordings. Many works of object tracking based on the retina-inspired DVS sensors have been reported (Litzenberger et al., 2006; Conradt et al., 2009; Schraml and Belbachir, 2010; Drazen et al., 2011; Ni et al., 2012, 2015; Piatkowska et al., 2012; Zhenjiang et al., 2012; Saner et al., 2014; Zhu et al., 2017). However, event-based object tracking is challenging due to the significant appearance variations caused by the noise events,

complex background textures, occlusion and randomness of event generating in each pixel circuit. Firstly, events stimulated by the contour and textures of the object are easy to be confused by events from the background objects. If the target event-stream object has a similar spatiotemporal shape with that of a background object, a tracker based on simple feature representation would be easily confused. Besides, the event-stream shape of the non-rigid object changes and deforms all the time, which demands a more discriminative feature representation. **Figure 1** shows some successive reconstructed frames from several event-stream recordings. The ground-truth position of the target object is shown with a bounding box in each segment. From the pictures, we can see that the appearance of the target event-stream object changes obviously even between two adjoining segments, which demands a robust tracker for tracking the rapid changed appearance.

This paper presents a robust event-stream pattern tracking method based on correlation filter (CF) mechanism. Hierarchical convolutional layers of a convolutional neural network (CNN) are used to extract the feature representation from rate coding frame of event streams. The performance of the proposed method is evaluated on the DVS recordings of several complicated visual scenes. Among the recordings, three are captured by a DVS128 sensor (Lichtsteiner et al., 2008) by ourselves and the rest are from an event-stream tracking dataset (Hu et al., 2016) captured by a DAVIS sensor (Brandli et al., 2014). The results prove that the proposed method can successfully track the objects in some visual scenes with noise events, complicated background textures, occlusion, and intersected trajectories. This is because we use features from multiple CNN layers to represent the appearance of the target object which combines semantics that are robust to significant appearance variations and spatial details that are effective for precise localization.

Related Works

Object tracking methods based on event cameras can be classified into two categories. The first category is the event-driven mechanism in which each incoming event is processed and determined whether it belongs to the target object. In Litzenberger et al. (2006) implemented a continuous clustering of AER events and tracking of clusters. Each new event was assigned to a cluster based on a distance criterion and then the

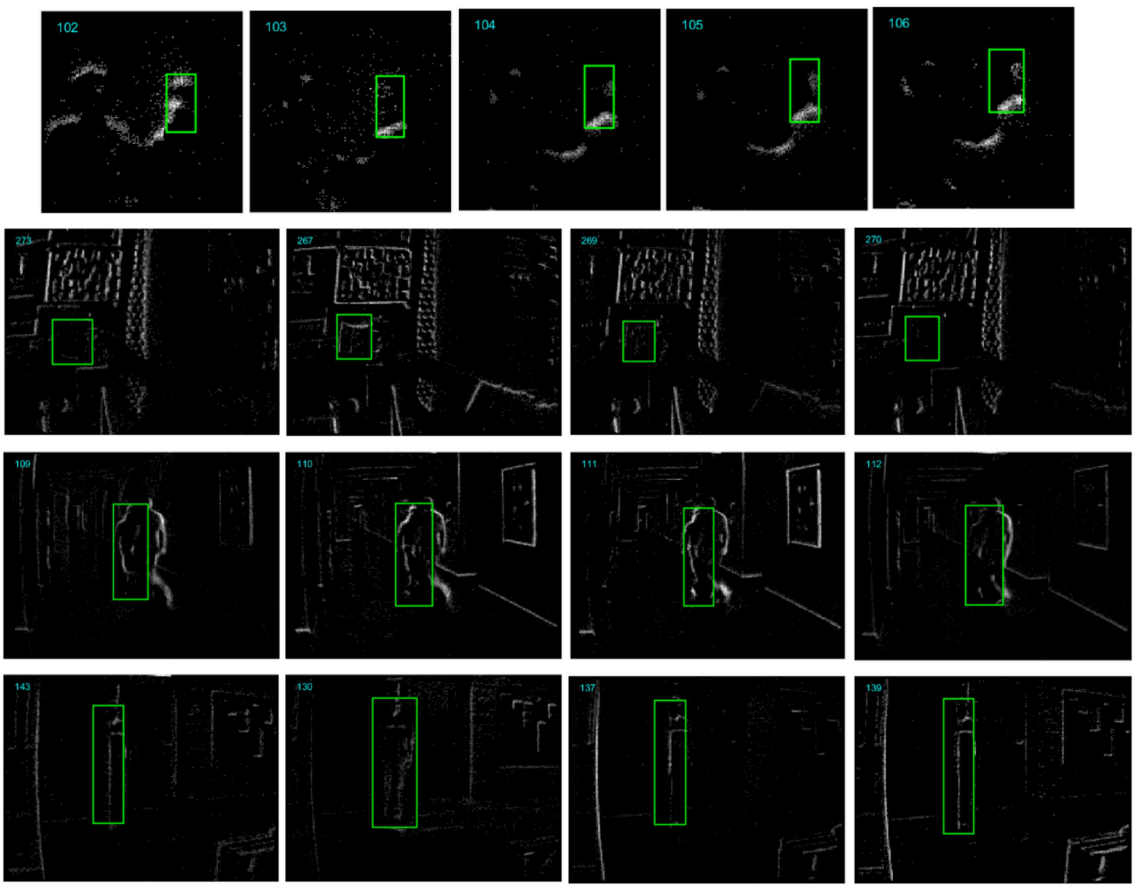


FIGURE 1 | Some example reconstructed frames from several event-stream recordings (From top to bottom are the "Horse toy" from DVS128 sensor, "Vid_B_cup," "Vid_J_person_floor," and "Vid_E_person_part_occluded" from DAVIS sensor). The appearance of the target object (in bounding box with green line) changes obviously for the noise events, background texture, and randomness in event-generating of the pixels.

clusters weight and center position was updated. In addition, point cloud method is also introduced to model the event-stream object. In Ni et al. (2015) proposed an iterative closest point based tracking method by providing a continuous and iterative estimation of the geometric transformation between the model and the events of the tracked object. In Ni et al. (2012), the authors applied the iterative closest point tracking algorithm to track a microgripper position in an event-based microrobotic system. One disadvantage of these kinds of methods is that noise events occur will cause the tracker to make a wrong inference. Adding noise event filtering modules to the tracking system will unavoidably filter many informative events while increase the computational complexity of the system. In addition, although these event-based sensors are based on the event-driven nature, it is still a difficult task to recognize an object from each single event. The second category is based on feature representation of the target object. In Zhu et al. (2017), the authors proposed a soft data association modeled with probabilities relying on grouping events into a model and computing optical flow after assigning events to the model. In Lagorce et al. (2014), proposed an event-based multi-kernel algorithm, and various kernels, such as Gaussian, Gabor, and arbitrary user-defined kernels were used to handle the variations in position, scale and orientation. In Li et al. (2015), the authors prosed a compressive sensing based method for the robust tracking based on the event camera. The representation or appearance model of event-stream object is based on features extracted from the multi-scale space with a data-independent basis and employs non-adaptive random projections that preserve the structure of the feature space of objects.

The core of most modern trackers is a discriminative classifier to distinguish the target from the surrounding environment. In computer vision, CF based methods has enjoyed great popularity due to the high computational efficiency with the use of fast Fourier transforms. In Bolme et al. (2010) learned a correlation filter over luminance channel the first time for real-time visual tracking, named MOSSE tracker. In Henriques et al. (2012, 2015), a kernelized correlation filter (KCF) is introduced to allow non-linear classification boundaries. Nowadays, features from convolutional neural network (CNN) are used to encode the object appearance and achieved good performance (Danelljan et al., 2015; Ma et al., 2015). In Danelljan et al. (2015), the authors proposed a method combining activations from the convolutional layer of a CNN in discriminative correlation filter based tracking frameworks, achieving a superior performance by using convolutional features compared to standard hand-crafted feature representations. They also show that activations from the first layer provides superior tracking performance compared to the deeper layers of the network. In Ma et al. (2015), they exploit the hierarchies of convolutional layers as a non-linear counterpart of an image pyramid representation and these multiple levels of abstraction to improve tracking accuracy and robustness. They demonstrate that representation by multiple layers of CNN is of great importance as semantics are robust to significant appearance variations and spatial details are effective for precise localization. Although feature-based methods show robustness and real-time capability, the most serious defect is that such algorithms should accumulate the events in a time window and then perform feature extraction. Then the length of the time window may be different under different scenes.

METHODOLOGY

Temporal Contrast Pixel

The pixel of the DVS sensor is a type of temporal contrast pixel which only responds to the temporal contrast of the light intensity in the scene and generates a temporal event whenever the brightness change exceeds a pre-defined threshold. Each event is a quadruple (x, y, t, p), where (x, y) denotes the positions of the pixel, t denotes the time when the event is generated, the polarity $p = 1$ denotes the increasing brightness and $p = -1$ denotes the decreasing brightness. This temporal contrast pixel has the advantage of high dynamic range because it needs not to respond to the absolute light intensity. The time stamp of each event has the temporal resolution of microsecond. Then the DVS sensor is suitable to capture the dynamic scenes with high-speed changes.

In this work, we use event-stream recordings from both DVS128 and DAVIS sensors to evaluate the performance of the proposed method. Both sensors are based on the same event-generating mechanism. As the name of the sensor shows, DVS128 has the spatial resolution of 128×128. DAVIS is a new retina-inspired, event-based vision sensor with the spatial resolution of 240×180.

Although these kinds of sensors are based on an event-driven nature, it remains a difficult task to recognize an object from a single event. Many works have accumulated the event stream into multiple segments on which to extract feature for information processing, such as the event-stream object display in jAER open-source tool. There exist two accumulating methods, i.e., hard events segmentation (HES) and soft events segmentation (SES). HES divides the event flow into segments using fixed time slices or fixed number of events. Different from HES, SES adaptively obtains the segments according to the statistical characteristics of the events based on an event responding neuron, such as the leaky-integration-firing neuron.

For comparison, we segment the event streams into the same number of segments with the items of the groundtruth. Rate coding is utilized to encode the visual information of the event-stream object. Intuitively, each pixel value is represented as the number of events generated by this pixel within the segment. Event rate of the temporal contrast pixel can be represented as follows,

$$Rate(t) \approx \frac{TCON(t)}{\theta} = \frac{1}{\theta}\frac{d\ln(I)}{dt} \tag{1}$$

where TCON represents the temporal contrast, and $I(t)$ is the photocurrent. Within a time window, the physical meaning of the number of events of a pixel represents the frequency of which the temporal intensity change exceeds the threshold. In rate coding, the serious temporal noise of the events is suppressed by integrating the events of each pixel in the segment.

Correlation Filter Framework

Generally, a CF tracker learns a discriminative classifier and finds the maximum value of the correlation response map as the estimation of the position of the target object. The resulting classifier is a 2-dimensional correlation filter which is applied to the feature representation. Circular correlation is utilized in CF framework for efficiently train. Multi-channel feature maps from multiple layers of a deep CNN are used as representation of rate-coding event-stream object. Feature maps of the l-th layer is denoted as x^l with the size of H × W × C, where H, W and C denote the width, height, and channel number, respectively. The correlation filter f_t has the same size with the feature maps in the current event frame t. In CF framework, the correlation filter is trained by solving a linear least-squares problem as follows,

$$w = \arg\min_{f_t} \sum_{h,w} \left\| f_t \cdot x^l{}_{h,w} - y_{h,w} \right\|^2 + \lambda \left\| f_t \right\|_2^2 \tag{2}$$

where $x^l_{h,w}$ demotes the shifted sample. hε{0,1,2,…,H-1}, wε{0, 1, 2, …, W-1}. $y_{h,w}$ is the Gaussian function label, and where λ is a regularization parameter (λ > 0). $y_{h,w} = \exp\left(-\frac{(h-H/2)^2+(w-W/2)^2}{2\sigma^2}\right)$, where σ is the kernel width.

The minimization problem in (2) can be solved in each individual feature channel using fast Fourier transformation (FFT). We use the capital letters as the corresponding Fourier transform of the signal. The learned correlation filter in frequency domain on the *c-th,* cε{1, 2, …, C} channel is as follows,

$$F^c = \frac{Y \odot \overline{X}^c}{\sum_{i=1}^{C} X^i \odot \overline{X}^i + \lambda} \tag{3}$$

where the operation ⊙ denotes the element-wise product, Y is the Fourier transformation form of y={$y_{h,w}$ | hε{0, 1, 2, …, H-1}, wε{0, 1, 2, …, W-1}}, and the bar means complex conjugation. z^l with size of H × W × C represents the feature maps of the l-th layer of the neural network. The l-th correlation response map of size H × W can be calculated as follows,

$$r^l = \xi^{-1}\left(\sum_{c=1}^{C} F^c \odot \overline{Z}^c\right) \tag{4}$$

where the operation ξ^{-1} denotes the inverse FFT transform. The position of maximum value of the correlation response map r^l is used as the estimation of the target location on the l-th convolutional layer.

Representation Based on Convolutional Neural Network

In this paper, hierarchical convolutional feature representation is used for encoding the appearance of the event-stream object. An imagenet-pretrained 16-layer classification network (VGG-Net-16)[1] implemented based on the MatConvNet library is used in our method for feature extraction. **Figure 2** shows the network architecture of the VGG-Net-16. Hierarchical feature representation are obtained with the CNN forward propagation. Event streams in a short duration are integrated into a rate coding map which is taken an the input of CNN. As the input of the original VGGNet are three-channel, we set each channel of the input layer equal to the rate coding map. As the pooling operation would reduce the spatial resolution with increasing depth of convolutional layers, we first remove the layers higher than the conv3_3 layer and the output of conv1_1, conv2_2, and conv3_3 are taken as the feature. Multiple convolutional layers are combined to encode the changed appearance of the event-stream object. **Table 2** shows the spatial size and channels of the feature maps of the input layer, and three different convolutional layers. Instead of resizing the size of the input rate coding maps to equal the size of the input layer of CNN (i.e., 224 × 224) in Ma et al. (2015), we use the resulted model parameters of each layer to perform convolutional operation on the original input. In this work, we test some hierarchical composition of different convolutional layers for feature representation and found that the representation of combination of multiple layers of conv1_1, conv2_2, and conv3_3 could achieved a satisfactory tracking performance.

[1]The model can be download from: https://pan.baidu.com/s/1QWl6zw4DpMSCDg2X-Mb7mAA

EXPERIMENTAL RESULTS AND DISCUSSION

In this section, a series of experiments on several event-stream recordings are presented. The metric used in this paper is the center location error. We labeled the ground-truth position of tracked object manually. Section 4.1 introduces the event-stream tracking dataset on which we perform the tracking experiments. In section 4.2, we evaluate the influence of two hyper-parameters. Section 4.3 presents the tracking speed over different convolutional layers of CNN. In section 4.4, we compare the performance of the proposed method with some other event-stream tracking methods.

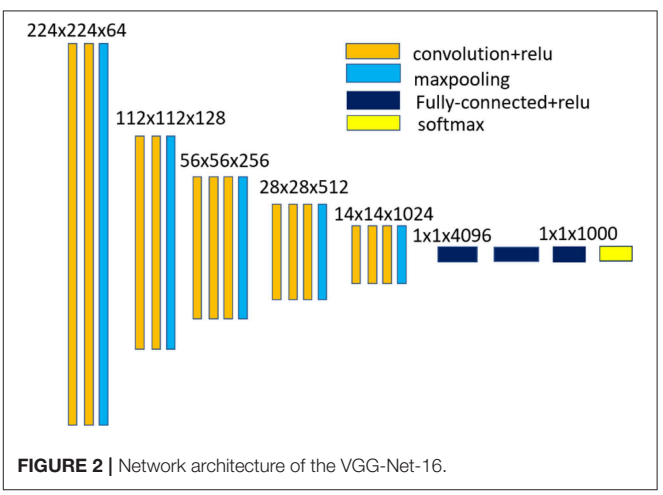


FIGURE 2 | Network architecture of the VGG-Net-16.

TABLE 1 | Challenges of each recording.

Challenge	Noise event	Complicated background	Occlusion	Intersected trajectories	Deformation	Scale variation	Pose variation
Digit3	1	0	0	0	0	1	0
Horse toy	1	0	0	0	1	1	1
Human face	1	0	0	0	1	1	1
Sylvestr	1	0	0	0	1	1	1
Vid_B_cup	1	1	0	0	0	0	1
Vid_C_juice	1	1	0	1	0	1	0
Vid_person_part_occluded	1	1	1	0	0	0	0
Vid_J_person_floor	1	1	1	1	1	1	1

If the recording (in the row) has this challenge (in the column), then the corresponding value is set to 1, otherwise 0.

Event-Stream Recordings

Eight event-stream recordings with labeled ground truth data are used to evaluate the proposed method. Among them, three recordings were captured by a DVS128 sensor by ourselves, and the rest are from an event-stream tracking dataset captured by a DAVIS sensor.

A. Three DVS128 recordings[2] Three event-stream recordings of three different scenes were captured with a DVS128 device. These recordings have the same spatial resolution of 128 × 128. We divided each recording into multiple segments and in each segment, we label the position and the size of the target object with bounding boxes. The first recording ("Digit3") is a scene containing several digits. The task is to track the digit3 in the event streams. This recording has a time range of about 38.8 s which is divided into 767 segments. The second recording ("Horse toy") is a moving human with a horse toy in his hand. The task is to track the horse in the event streams. This recording has a time duration of about 17.3 s and is divided into 347 segments. The appearance of the toy changed quickly with much rotation and deformation in the recording. The third recording ("Human face") has a duration of about 17.1 s and is divided in 343 segments. This task is to track the face of a human. The human face moved quickly with rotation and deformation and the appearance of the event stream changed all the time, which make the task difficult.

B. DAVIS recordings[3] Yuhuang Hu et al. (2016) proposed an event-stream tracking dataset with a DAViS240C camera which has a spatial resolution of 240 × 180. In some tracking sequences of this tracking dataset, the target objects are still, or cannot be distinguished from the background. We chose five recordings from this dataset and re-labeled the bounding boxes of the target objects because the provided bounding boxes in the dataset seem to have a little shift compared to the accurate positions of the target objects. These chosen five recordings are "Sylvestr," "Vid_B_cup," "Vid_C_juice," "Vid_E_person_part_occluded," and "Vid_J_person_floor," and are divided into 1344, 629, 404, 305, and 388 segments, respectively.

C. Challenges in each recording. We list the challenges in each recording as show in **Table 1**. Seven challenges are taken into consider, including the noise events, complicated background, occlusion, intersected trajectories, deformation, scale variation and pose variation. One or several challenges are contained in each recording.

Robustness to Hyperparameters

The proposed event-stream pattern tracking method requires the specification of two hyparameters, i.e., the event number in each segment and the convolutional layers for feature representation. To investigate the influence of these two hyperparameters, we performed a series of experiments on several event-stream recordings.

A. Event number in each segment. We investigate the influence of the number of events in each segment on several recordings, including the "Horse toy" from DVS128 sensor and the "Sylvestr," "Vid_J_person_floor" from DAVIS sensor. **Figure 3** shows the tracking trajectory of the proposed tracker under different event number in each segment. The change of the x position and y position of the center of the tracker along time are shown. As the number of segments would be different from that of the groundtruth with a different event number in each segment, it would be impossible to compare the location of the trackers when both have different number of segments. Then we use the index of the events to represent the time coordinate. We plot the curve of the x position and y position of the tracker over the index of events. Results show that the proposed method is robust to the number of events in each segment by comparing the degree of proximity of the trajectory of the tracker and the groundtruth. The effectiveness of the proposed method is owing to the discriminative feature representation transferred from the multiple layers of a pre-trained CNN on the computer vision task. The proposed method has many potential applications in many the high-speed scenes with less events in each time window.

B. Feature map from different layers in the CNN. We investigate the influence of different convolutional layers of VGG-Net-16 model on the tracking performance. Feature representations with four kinds of combination of convolutional layers, i.e., the Conv1_1 convolutional layer, the Conv2_2

[2]The DVS128 recordings and the groundtruth, named "DVS recordings and groundtruth.zip" can be download from: https://figshare.com/s/70565903453eef7c3965

[3]The DAVIS recordings and the groundtruth, named "DAVIS recordings and groundtruth.zip" can be download from: https://figshare.com/s/70565903453eef7c3965

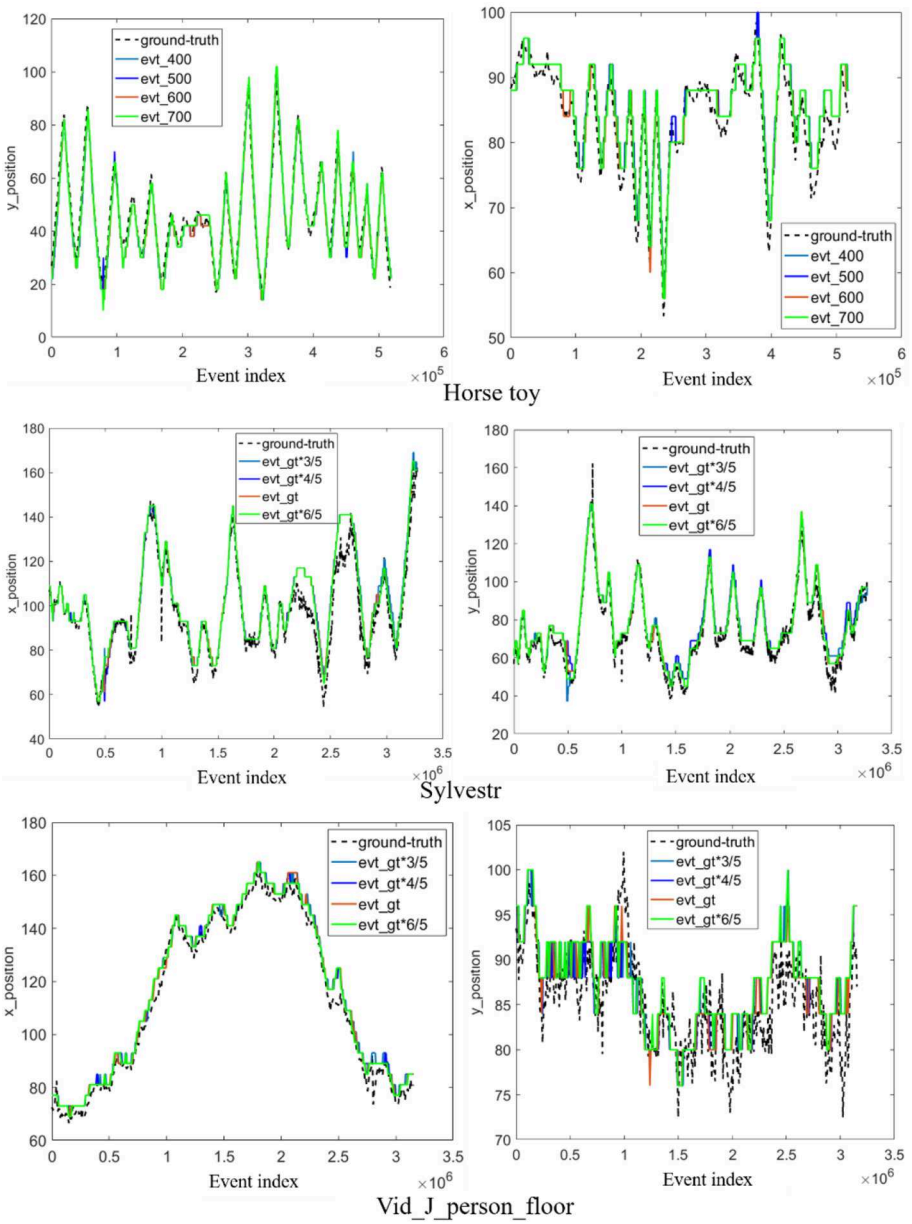


FIGURE 3 | Tracking performance under different event number per time bin on three event-stream recordings (from top to bottom are "Horse toy," "Sylvestr," and "Vid_J_person_floor" scenes, respectively). The x (left) and y (right) positions of the center of the tracker over the time are displayed. We use the index of event to represent the time coordinate, and the * mean multiplication.

layer, the Conv3_3 layer, and the composition of the three convolutional layers were evaluated. **Table 2** shows the spatial size and dimensionality of the feature maps of the input layer, and three different convolutional layers. All the DAVIS recordings ("Sylvestr," "Vid_B_cup," "Vid_C_juice," "Vid_E_person_part_occluded," and "Vid_J_person_floor") are used in this test. For the "Vid_B_cup" scene, the tracking would fail when a single convolutional layer or a composition of two layers (Conv1_1 or Conv2_2 or Conv3_3) is used. In "Vid_B_cup," the target cup is moved over a complicated background, then the events from the target object are very easy to be mixed up with the events from background. **Figure 4** shows the metric results on the rest four event-stream recordings with different convolutional layers. Feature representation from the higher convolutional layers results in better tracking performance. For the more complicated scenes with complex background, such as the "Vid_B_cup," combination of hierarchical feature representation from multiple convolutional layers is required for effective object tracking. This is because the feature representation from multiple convolutional layers combines the low-level texture features and high-level semantic features and can

handle the rapid change of the appearance of the target object. While for the relatively simple scenes with less noise events and simple background textures, less and lower convolutional layers result in effective tracking with a higher speed.

Tracking Speed Under Different Layers

In this section, we investigate the tracking speed of the proposed method on several recordings. The speed is measured over different convolutional layers of the employed CNN. **Table 3** shows the results of tracking speed on five DAVIS recordings. We measure the tracking speed using the unit of segments per second which is similar to the frame per second in the computer vision tracking. In the experiments, a PC machine with Intel(R) Core(TM)i5-7300HQ CPU @ 2.5 GHz is used. We did not present the measurement result on some convolutional layers in some event-stream recordings (such as the Conv1_1, and Conv2_2 in the "Vid_C_ juice" scene) which have failed in the tracking of the corresponding target object.

Intuitively, high-level representation requires more computational operations, which leads to the decrease of the tracking speed. In the proposed method, the precision and the speed are a tradeoff. In the simple scenes, such as monitoring an object with a fixed sensor, low-level convolutional layers are enough for effectively tracking with high speed. In other complicated scenes such as complicated background textures or moving DVS sensor, high-level convolutional layers are demanded for accurate tracking, which limits the tracking speed. In fact, due to the high computational efficiency in the frequency

TABLE 2 | Spatial size and channels of the feature maps from three different convolutional layers of the employed network.

	Input	Conv1_1	Conv2_2	Conv3_3
Spatial size	M × N	M × N	M/2 × N/2	M/4 × N/4
Channels	3	64	128	256

Input layer denotes the input rate coding map, after the necessary preprocessing steps.

TABLE 3 | Tracking speed under different convolutional layers of the employed network.

Event recordings	Conv1_1	Conv2_2	Conv3_3	All
Sylvestr	114.1	68.8	38.2	30.1
Vid_C_juice	–	–	37.8	29.8
Vid_E_person_part_occluded	–	68.5	37.9	29.2
Vid_J_person_floor	–	68.3	38.1	29.0
Vid_B_cup	–	–	–	29.3

The tracking speed is measured by the segments per second. We measure the tracking speed on the DAVIS recordings.

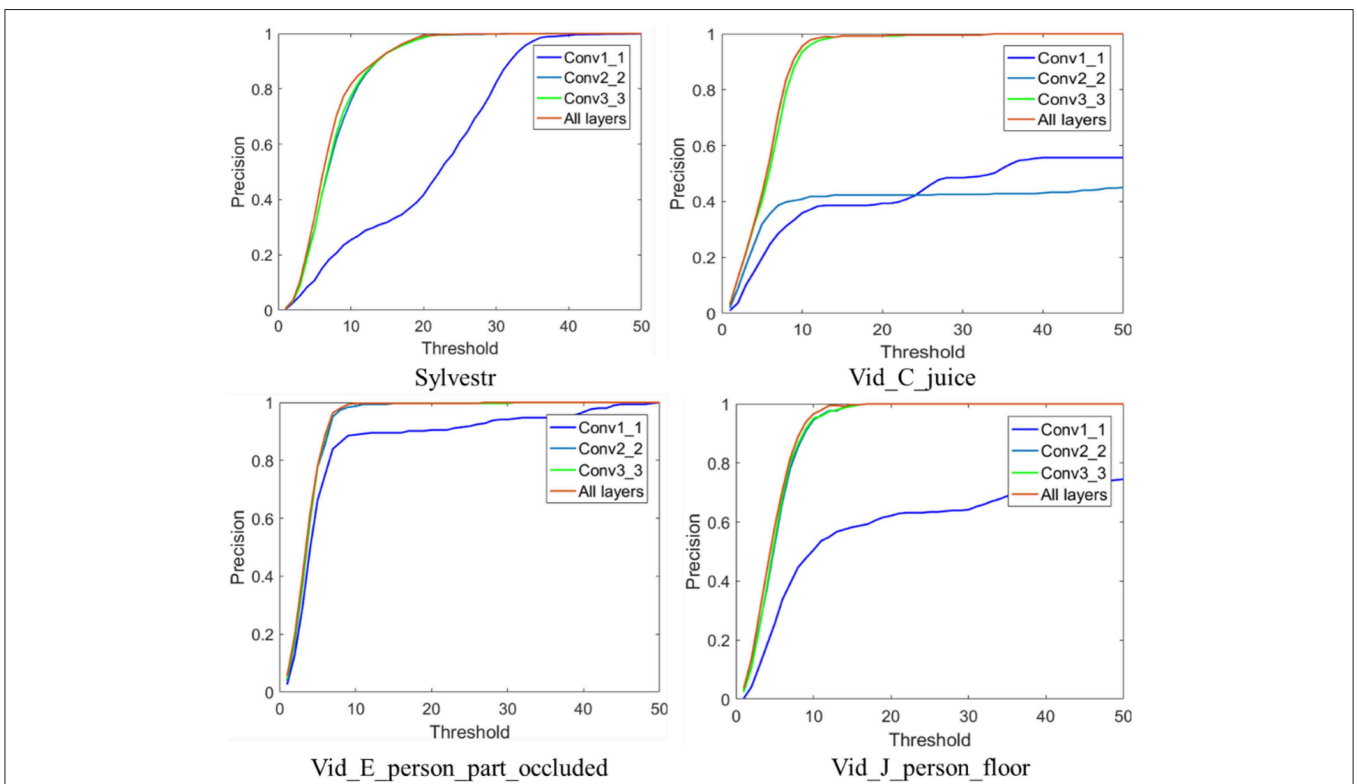


FIGURE 4 | Tracking performance with different feature representation from different convolutional layers of the employed CNN.

domain of the CF mechanism, the proposed method achieved relatively high tracking speed even using high-level feature map.

Comparison With Other Methods

In this section, we compare the performance of the proposed method to several other event-stream pattern tracking methods. We perform the experiments on all the eight event-stream recordings. In the first place, we introduced the four other tracking methods as follows:

A. Three Tracking Methods in jAER Software

These three methods are based on the three event filters in jAER source available within the jAER sourceforge repository, including "Rectangular Cluster Tracker," "Einstein Tracker," and "Median Tracker," respectively. Since the three methods have failed in the more complicated scenes, we only provide the tracking results of the three methods on the three simple scenes captured by DVS128 sensor.

B. Compressive Tracking-Based Rate Coding Feature

This is a feature-based event-stream tracking method based on compressive sensing and has achieved good performance on some simple scenes captured by DVS128. The compressive tracking learns a classifier on the compressive coding of multi-scale haar-like feature extracted on the rate coding map. We provide the results of this method on all the event-stream recordings for comparison with the proposed method.

Figure 5 shows the comparison results of the proposed method with all the four methods on the DVS128 recordings. Center location error is used to measure the tracking performance. Results show that the proposed tracking method achieves the best performance. Three tracking methods in jAER software show poor tracking performance. The event-driven jAER methods often failed to assign each event to the accurate position or cluster of the target object. Compared to the compressive tracking method, the proposed method achieved better performance owning to the more discriminative CNN feature presentation than the haar-like feature in the compressive tracking.

On the five DAVIS recordings, we only compared the tracking performance of the proposed method with the compressive tracking method because the three tracking methods in jAER software fail to track the target object in the more complicated scenes. **Figure 6** shows the tracking location precision of the two methods. The proposed method achieved better performance on the five DAVIS recordings. Especially in the "Vid_B_cup" scene where many objects in the background have the same shape with the target object "cup," the proposed tracker can track the target object correctly. Simple hand-crafted feature representation cannot handle many complicated scenes with noise events, complicated background textures, rapid changed appearance, and occlusion. By combining the low-level texture features and high-level semantic features, the feature representation from multiple convolutional layers can handle the complicated scenes very well. Results demonstrate that the proposed method is

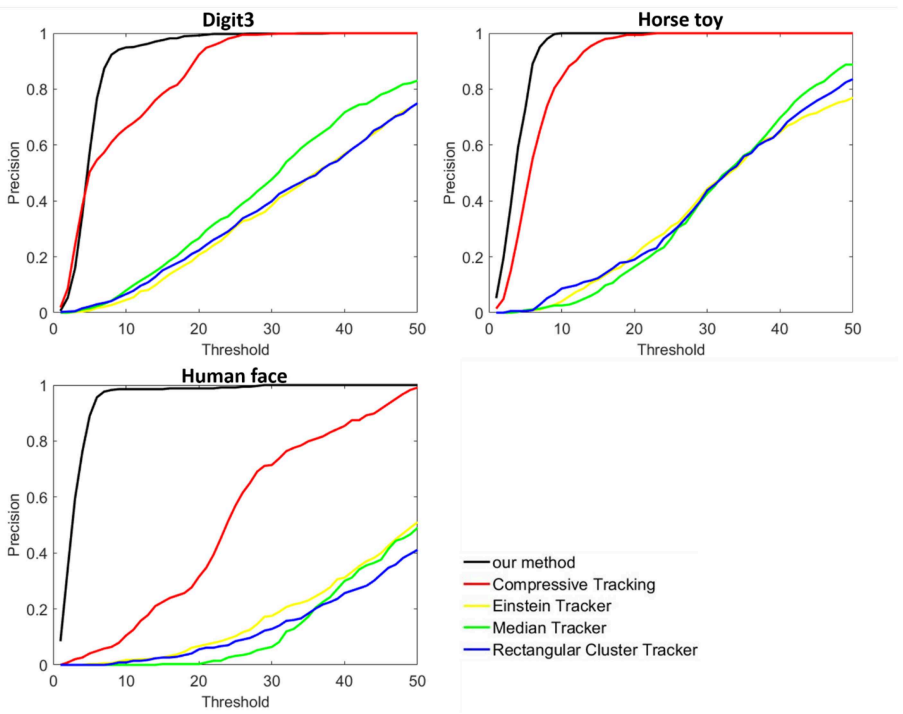


FIGURE 5 | Tracking results on three event-stream recordings captured by DVA128 sensor (from left to right are Digit3, Horse toy, and Human face, respectively). Comparison were done with four other tracking methods.

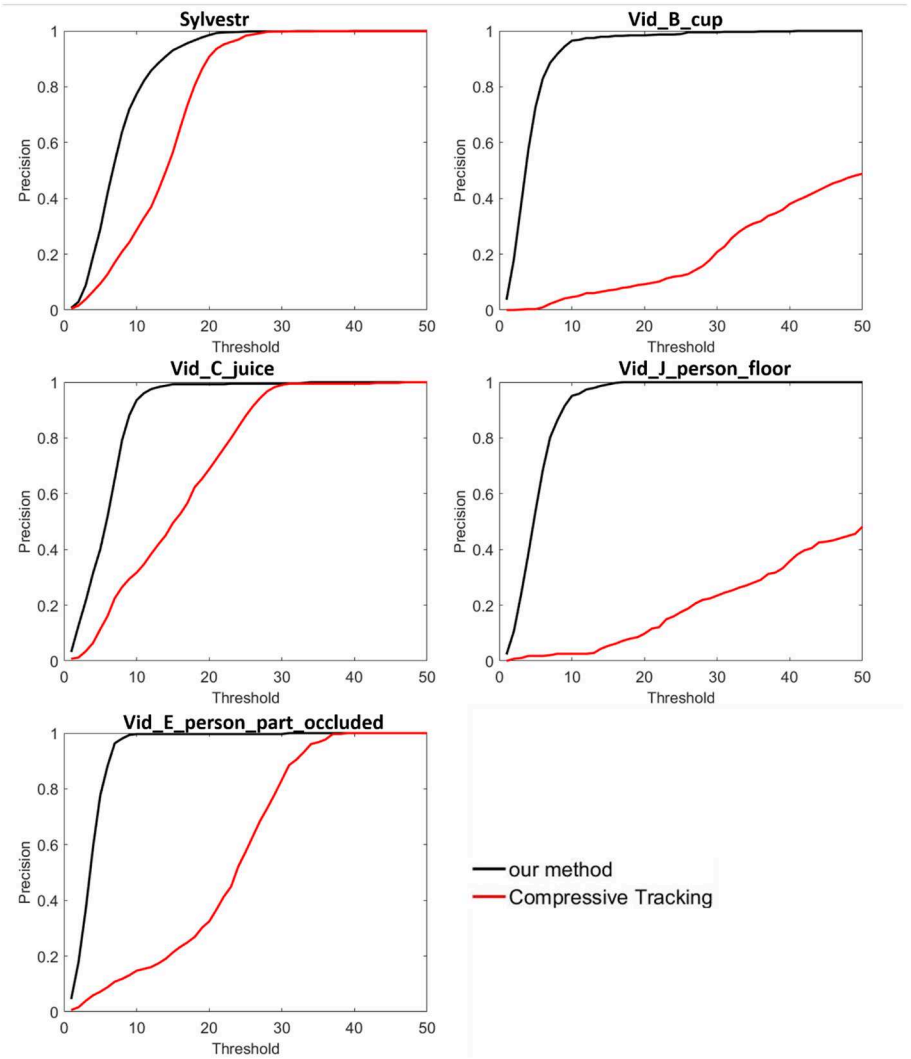


FIGURE 6 | Tracking results on five event-stream recordings captured by DAVIS sensor, compared with the compressive sensing-based tracking methods.

robust to many challenging visual scenes with better tracking performance than other methods.

Figure 7 shows some tracking examples by integrating the events into reconstructed frames. In the tracking process, we did not change the scale of the tracker. The proposed tracker tracked the target object with high location precision while the compressive sensing-based tracker drifts very easily. Even with complicated background and occlusion, our tracker achieved very high tracking precision.

CONCLUSION

In this work, we proposed a robust event-stream object tracking method based on the CF tracking mechanism. Our method overcomes some challenges in the event-stream tracking, such as the noise events, chaos of the complex background texture, occlusion, and randomness of event generating in the pixel circuit. Rate coding is used to encode the visual information of the target event-stream object. Correlation response map is computed on the feature representation from the hierarchical convolutional layers of a pre-trained deep CNN.

The proposed method shows good performance in many complicated visual scenes. Compared with other feature-based methods, our method is more robust in many visual scenes with noise and complicated background textures. Compared with other event-driven method, the proposed method has real-time advantage on event streams with large number of events. To utilize each single event for tracking and updating the appearance of the target object without segment reconstruction is a more interesting and challenging task which has lied in the heart of current event-based vision research and will be explored in the future. In event-driven tracking tasks, how to suppress the noise events is an import step for correct tracking as the noise events will lead the tracker to make wrong estimation.

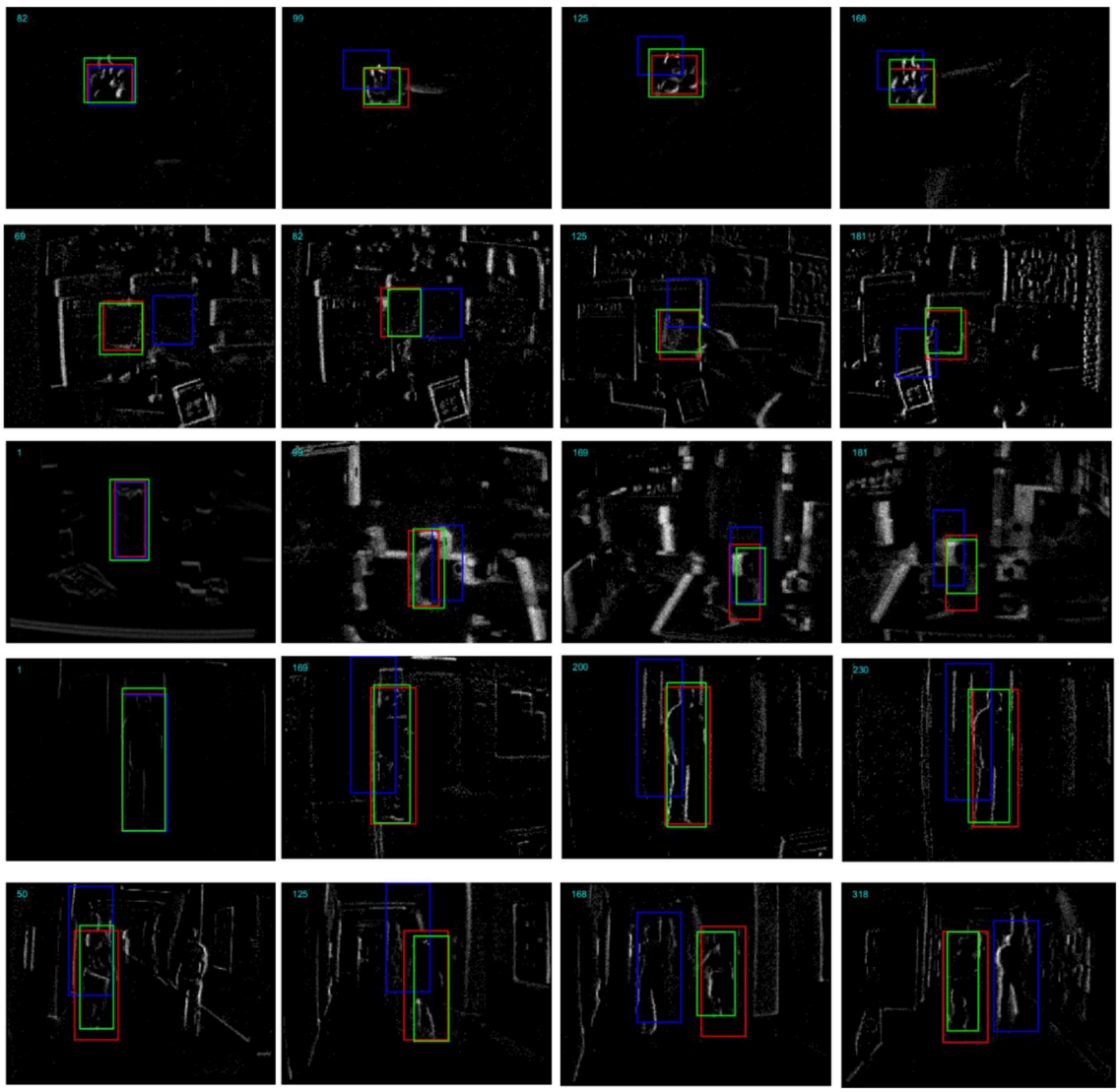


FIGURE 7 | Comparison of the proposed tracking method with the compressive sensing based method on example segments from the five DAVIS recordings. From top to bottom are the "Sylvestr," "Vid_B_cup," "Vid_C_juice," "Vid_E_person_part_occluded," and "Vid_J_person_floor," respectively. The bounding boxes with green, red, and blue color represent the groundtruth, our proposed tracker and the compressive sensing based tracker, respectively.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: https://figshare.com/s/70565903453eef7c3965; https://figshare.com/s/70565903453eef7c3965.

AUTHOR CONTRIBUTIONS

HL proposes the algorithm and design the experiment setup. Besides, data analysis and classification metric are also done by HL. LS supports the research of the neuromorphic vision and contributes the principle of event-based object tracking.

FUNDING

The work was partially supported by the Project of NSFC (No. 61327902), Beijing program on study of functional chip and related core technologies of ten million class of brain inspired computing system (Z151100000915071), and Study of Brain-Inspired Computing System of Tsinghua University program (20141080934, 20151080467).

REFERENCES

Bardow, P., Davison, A. J., and Leutenegger, S. (2016). "Simultaneous optical flow and intensity estimation from an event camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV), 884–892. doi: 10.1109/CVPR.2016.102

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

Bolme, D. S. Ross Beverid, J., Drap, B. A., and Lui, Y. M (2010). "Visual object tracking using adaptive correlation filters," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (San Francisco, CA).

Brandli, C., Berner, R., Yang, M., Liu, S. C., and Delbruck, T. (2014). A 240× 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid State Circ.* 49, 2333–2341. doi: 10.1109/JSSC.2014.2342715

Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R. J., and Delbruck, T. (2009). "A pencil balancing robot using a pair of AER dynamic vision sensors," in *22nd IEEE International Symposium on Circuits* and *Systems* (Taipei). doi: 10.1109/ISCAS.2009.5117867

Danelljan, M., Häger, G., Shahbaz Khan, F., and Felsberg, M. (2015). "Convolutional features for correlation filter based visual tracking," in *IEEE International Conference on Computer Vision Workshops* (Santiago). doi: 10.1109/ICCVW.2015.84

Drazen, D., Patrick, L., Philipp, H., Tobias, D., and Atle, J. (2011). Toward real-time particle tracking using an event-based dynamic vision sensor. *Exp. Fluids* 51, 1465–1469. doi: 10.1007/s00348-011-1207-y

Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2015). High-speed tracking with kernelized correlation filters. *IEEE Trans. Pattern Analy. Mach. Intell.* 37, 583–596. doi: 10.1109/TPAMI.2014.2345390

Henriques, J. F., Caseiro, R., Martins, R., and Batista, B. (2012). "Exploiting the circulant structure of tracking-by-detection with kernels," in *European Conference on Computer Vision* (Berlin, Heidelberg). doi: 10.1007/978-3-642-33765-9_50

Hu, Y., Liu, H., Pfeiffer, M., and Delbruck, T. (2016). DVS benchmark datasets for object tracking, action recognition, and object recognition. *Front. Neurosci.* 10:405. doi: 10.3389/fnins.2016.00405

Kim, H., Handa, A., Benosman, R., Ieng, S. H., and Davison, S. H. (2008). Simultaneous mosaicing and tracking with an event camera. *J. Solid State Circ.* 43, 566–576. doi: 10.5244/C.28.26

Lagorce, X., Meyer, C., Ieng, S. H., Filliat, D., and Benosman, R. (2014). Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Trans. Neural Netw. Learn. Syst.* 26, 1710–1720. doi: 10.1109/TNNLS.2014.2352401

Li, H., Jing, P., and Li, G. (2015). "Real-time tracking based on neuromrophic vision. in Non-Volatile Memory Technology Symposium (NVMTS)," in *2015 15th Non-Volatile Memory Technology Symposium (NVMTS)* (Beijing). doi: 10.1109/NVMTS.2015.7457498

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

Litzenberger, M., Posch, C., Bauer, D., Belbachir, A. N., Schon, P., Kohn, B., et al. (2006). "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Digital Signal Processing Workshop, 12th-Signal Processing Education Workshop, 4th* (Teton National Park, WY:IEEE), 173–178. doi: 10.1109/DSPWS.2006.265448

Ma, C., Huang, J. B., Yang, X., and Yang, M. H. (2015). "Hierarchical convolutional features for visual tracking," in *IEEE International Conference on Computer Vision* (Santiago). doi: 10.1109/ICCV.2015.352

Ni, Z., Ieng, S. H., Posch, C., Régnier, S., and Benosman, R. (2015). Visual tracking using neuromorphic asynchronous event-based cameras. *Neural Comput.* 27, 925–953. doi: 10.1162/NECO_a_00720

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

Piatkowska, E., Nabil Belbachir, A., Schraml, S., and Gelautz, M. (2012). "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (Providence, RI), 35–40. doi: 10.1109/CVPRW.2012.6238892

Saner, D., Wang, O., Heinzle, S., Pritch, Y., Smolic, A., Sorkine-Hornung, A., et al. (2014). "High-speed object tracking using an asynchronous temporal contrast sensor," in *19th International Workshop on Vision, Modeling and Visualization* (Darmstadt), 87–94. doi: 10.2312/vmv.20141280

Schraml, S., and Belbachir, A. N. (2010). "Dynamic stereo vision system for real-time tracking," *in IEEE International Symposium on Circuits and Systems (ISCAS)* (Paris). doi: 10.1109/ISCAS.2010.5537289

Serrano-Gotarredona, T., and Linares-Barranco, B. (2013). A 128x128 1.5% contrast sensitivity 0.9% FPN 3μs Latency4mWAsynchronous frame-free dynamic vision sensor using transimpedance preamplifier. *IEEE J. Solid State Circ.* 48, 827–838. doi: 10.1109/JSSC.2012.2230553

Zhenjiang, N., Bolopion, A., Agnus, J., Benosman, R., and Regnier, S. (2012). Asynchronsou event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Trans. Robot.* 28, 1081–1089. doi: 10.1109/TRO.2012.2198930

Zhu, A. Z., Atanasov, N., and Daniilidis, K. (2017). "Event-based feature tracking with probabilistic data association," in 2017 *IEEE International Conference on Robotics and Automation (ICRA)* (Singapore: IEEE), 4465–4470. doi: 10.1109/ICRA.2017.7989517

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.

frontiers in Neurorobotics

ORIGINAL RESEARCH
published: 16 October 2019
doi: 10.3389/fnbot.2019.00084

An Investigation of Vehicle Behavior Prediction Using a Vector Power Representation to Encode Spatial Positions of Multiple Objects and Neural Networks

Florian Mirus[1,2]*, *Peter Blouw*[3], *Terrence C. Stewart*[3] and *Jörg Conradt*[4]

[1] BMW Group, Research, New Technologies, Garching, Germany, [2] Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany, [3] Applied Brain Research Inc., Waterloo, ON, Canada, [4] Department of Computational Science and Technology, KTH Royal Institute of Technology, Stockholm, Sweden

Predicting future behavior and positions of other traffic participants from observations is a key problem that needs to be solved by human drivers and automated vehicles alike to safely navigate their environment and to reach their desired goal. In this paper, we expand on previous work on an automotive environment model based on vector symbolic architectures (VSAs). We investigate a vector-representation to encapsulate spatial information of multiple objects based on a convolutive power encoding. Assuming that future positions of vehicles are influenced not only by their own past positions and dynamics (e.g., velocity and acceleration) but also by the behavior of the other traffic participants in the vehicle's surroundings, our motivation is 3-fold: we hypothesize that our structured vector-representation will be able to capture these relations and mutual influence between multiple traffic participants. Furthermore, the dimension of the encoding vectors remains fixed while being independent of the number of other vehicles encoded in addition to the target vehicle. Finally, a VSA-based encoding allows us to combine symbol-like processing with the advantages of neural network learning. In this work, we use our vector representation as input for a long short-term memory (LSTM) network for sequence to sequence prediction of vehicle positions. In an extensive evaluation, we compare this approach to other LSTM-based benchmark systems using alternative data encoding schemes, simple feed-forward neural networks as well as a simple linear prediction model for reference. We analyze advantages and drawbacks of the presented methods and identify specific driving situations where our approach performs best. We use characteristics specifying such situations as a foundation for an online-learning mixture-of-experts prototype, which chooses at run time between several available predictors depending on the current driving situation to achieve the best possible forecast.

Keywords: vehicle prediction, long short-term memories, artificial neural networks, vector symbolic architectures, online learning, spiking neural networks

OPEN ACCESS

Edited by:
Pascual Campoy, Polytechnic University of Madrid, Spain

Reviewed by:
Subramanian Ramamoorthy, University of Edinburgh, United Kingdom
Michael Beyeler, University of Washington, United States

***Correspondence:**
Florian Mirus
florian.mirus@bmwgroup.com

Received: *15 January 2019*
Accepted: *26 September 2019*
Published: *16 October 2019*

Citation:
Mirus F, Blouw P, Stewart TC and Conradt J (2019) An Investigation of Vehicle Behavior Prediction Using a Vector Power Representation to Encode Spatial Positions of Multiple Objects and Neural Networks. Front. Neurorobot. 13:84. doi: 10.3389/fnbot.2019.00084

1. INTRODUCTION

The race to autonomous driving is currently one of the main forces for pushing research forward in the automotive domain. With highly automated vehicle prototypes gradually making their way to our public roads and fully-automated driving on the horizon, it seems to be a matter of time until we see robot taxis or cars navigating us through urban traffic or heavy stop-and-go on highways. One major reason for this development in recent years is the rapid progress of artificial intelligence (AI), especially the success of deep learning, which has shown remarkable results in tasks essential for automated driving like object detection, classification (Ciresan et al., 2012) and control (Bojarski et al., 2016).

Predicting future behavior and positions of other traffic participants from observations is essential for collision avoidance and thus safe motion planning, and needs to be solved by human drivers and automated vehicles alike to reach their desired goal. However, future positions of vehicles not only depend on each vehicle's own past positions and dynamics like velocity and acceleration, but also on the behavior of the other traffic participants in the vehicle's surroundings. Motion prediction for intelligent vehicles in general has seen extensive research in recent years (Polychronopoulos et al., 2007; Lawitzky et al., 2013; Lefèvre et al., 2014; Schmüdderich et al., 2015) as it is a cornerstone for collision-free automated driving. Lefèvre et al. (2014) classify such prediction approaches into three categories, namely *physics-based*, *maneuver-based*, and *interaction-aware*, depending on their level of abstraction. *Physics-based* and *maneuver-based* motion models consider the law of physics and the intended driving maneuver, respectively as the only influencing factors for future vehicle motion and ignore inter-dependencies between the motion of different vehicles. There exist a growing number of different *interaction-aware* approaches to account for those dependencies and mutual influences between traffic participants or, more generally, agents in the scene. Probabilistic models like costmaps (Bahram et al., 2016) account for physical constraints on the movements of the other vehicles. Classification approaches categorize and represent scenes in a hierarchy (Bonnin et al., 2012) based on the most generic ones to predict behavior for a variety of different situations.

Data-driven approaches to behavior prediction mainly rely on long short-term memory (LSTM) neural network architectures (Hochreiter and Schmidhuber, 1997), which have proven to be a powerful tool for sequential data analysis. Alahi et al. (2016) model interactions in the learning network architecture by introducing so-called social-pooling layers to connect several LSTM each predicting the distribution of the trajectory position of one agent at a time. Deo and Trivedi (2018a) adapted the combination of LSTM networks for encoding vehicle trajectories and (convolutional) social-pooling layers to account for interactions to vehicle prediction in highway situations. Altche and de La Fortelle (2017) use a LSTM network as well, but they account for interaction by including distances between the target vehicle and other agents directly in the training data rather than adapting the network architecture. A similar approach is proposed by Deo and Trivedi (2018b), but they combine LSTM networks with an additional maneuver classification network to predict future vehicle motion. One issue in data-driven approaches to behavior prediction accounting for relations between agents is that the number of other vehicles is variable. If information about vehicles other than the target are encapsulated in the input of the neural network, typically a specific number of other vehicles of interest are manually chosen a priori to avoid this issue (Altche and de La Fortelle, 2017; Deo and Trivedi, 2018b). If the information about other vehicles is encapsulated in the network architecture, it might be necessary to train several networks depending on the situation at hand.

In this paper, we expand our previous work (Mirus et al., 2018) on an automotive environment model based on VSAs (Gayler, 2003). In contrast to the representation shown in Mirus et al. (2018), this paper introduces a more sophisticated way of encapsulating spatial information of multiple objects in semantic vectors of fixed length. Therefore, we employ generalized exponentiation of high-dimensional vectors, referred to as the convolutive power, based on the VSA's binding operation, which in our case is circular convolution. We hypothesize that structured vector representations will be able to capture relations and mutual influence between traffic participants. For instance, in a situation as shown in **Figures 1A–C**, the behavior of the target vehicle, as depicted by a solid blue and dotted orange line for past and future positions, respectively, is influenced by the surrounding vehicles, as illustrated by solid and dotted gray lines for past and future positions, respectively. The target vehicle is approached from behind by a faster vehicle causing it to eventually change its lane to the right, which, for now, is still occupied by the ego-vehicle and another vehicle. All of these external influences have an impact on the target vehicle's motion (and vice versa). As we aim for a model-free data representation, we avoid introducing any physical constraints or restrictions regarding our data-representation or the predicting models.

In this work, we consider our main contributions to be the following: we present a representation of spatial information for multiple objects in semantic vectors of fixed length using the convolutive power. We use this representation as input for simple feed-forward neural networks and more sophisticated LSTM-based models and compare them against each other and a linear predictor as the simplest baseline. We conduct a thorough and detailed analysis for all of these models and show that by using our vector representation with a simple network architecture we can achieve results comparable to more complex models. This is particularly interesting for mobile applications, such as automated driving: combining our vector representation, which allows to encode spatial positions of several objects as well as efficient implementation in spiking neural networks (SNNs) (Eliasmith, 2013), with a simple feed-forward SNN would allow future deployment on dedicated, energy-efficient neuromorphic hardware. In case the performance of the simpler feed-forward networks is close enough to the more sophisticated ones, the possibility of efficient deployment could be an advantage over LSTM networks, which are by design harder to apply to dedicated computing hardware (Chang and Culurciello, 2017). Finally, we present a prototype of a mixture-of-experts online learning system, that chooses at run-time between several models, which

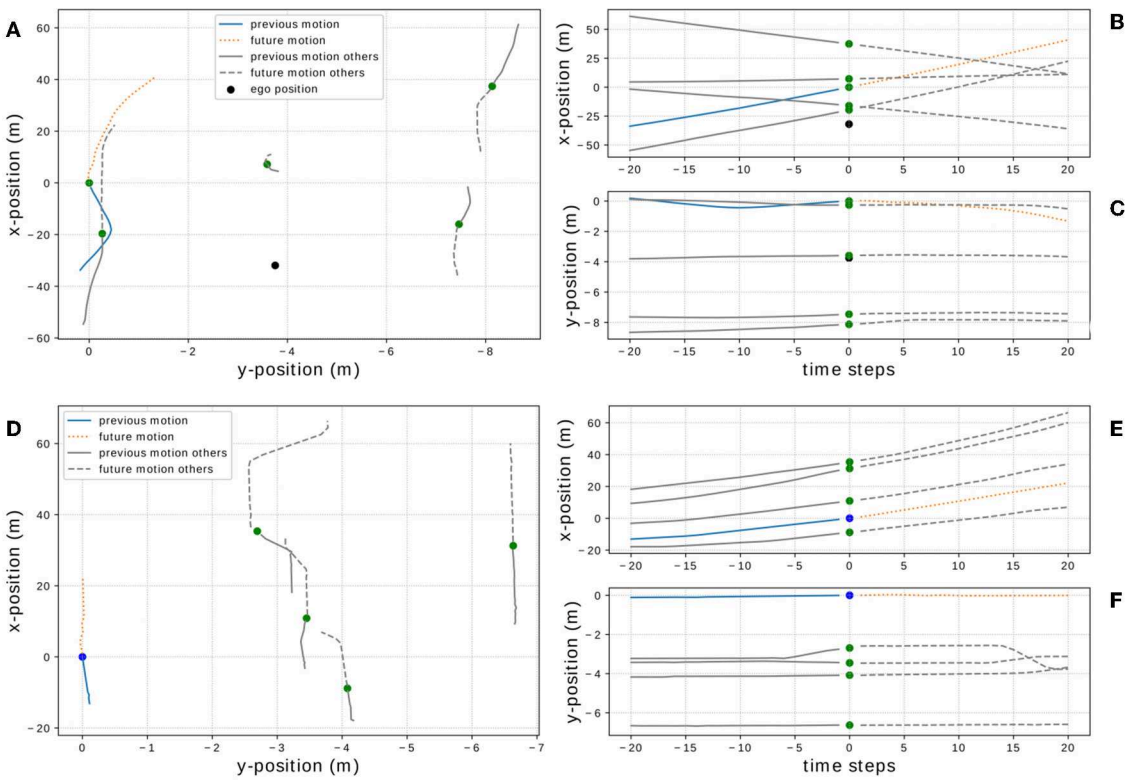


FIGURE 1 | Data visualization of one driving situation example from the *On-board* data set D_1 **(A–C)** and one example from the *NGSIM* data set D_2 **(D–F)**. The dots indicate the position of the vehicles and color-code the vehicle type (red = motorcycle, green = car, blue = truck, black = ego-vehicle), blue and orange lines show past and future motion of the target vehicle whereas gray lines depict the other vehicles' motion.

have been pre-trained offline, to achieve the best possible forecast. We show, that this online learning approach is able to improve its performance compared to the individual prediction models.

2. MATERIALS AND METHODS

2.1. Vector Symbolic Architectures

The term vector symbolic architectures (VSAs)—first coined by Gayler (2003)—refers to a family of approaches for cognitive modeling making use of distributed representations. The basic idea behind all of those approaches is to represent structure (e.g., cognitive concepts, symbols, or language) in a high-dimensional vector space by mapping each entity to be represented to a (possibly random) vector. One strength of VSAs is that they offer the possibility to manipulate their entities through algebraic operations, typically one *addition-like* and *multiplication-like* operation each. Vectors, which represent basic concepts not intended to be further decomposable and thus are not constructed from other vectors using the VSA's algebraic operations, are called *atomic vectors*.

2.1.1. Prerequisites

In this paper, we adopt the semantic pointer architecture (SPA) (Eliasmith, 2013), a variant of holographic reduced representations (HRRs) originally introduced by Plate (1994), to encode automotive scenes in high-dimensional vectors (note: the source-code for all models presented in this paper can be found at https://github.com/fmirus/spatrajectoryprediction). Thus, atomic vectors are picked from the real-valued unit sphere and the dot product serves as a measure of similarity. We call two vectors *similar*, if their dot-product is higher than a certain similarity threshold. The distribution of the dot-product of two randomly chosen unit vectors has a mean of 0 and a standard deviation of $\frac{1}{\sqrt{D}}$ (Widdows and Cohen, 2014). Thus, the similarity threshold is typically chosen as $\frac{c}{\sqrt{D}}$ for some constant c, which is a similarity value that we would expect from two randomly chosen vectors and only depends on the dimension D of the vector space. Furthermore, the algebraic operations are component-wise vector addition $\oplus$ and circular convolution $\circledast$, which is defined as

$$\mathbf{z} = \mathbf{v} \circledast \mathbf{w} \qquad \text{with } z_j := \sum_{k=0}^{D-1} v_k w_{(j-k) \bmod D} \tag{1}$$

for any two D-dimensional vectors $\mathbf{v}, \mathbf{w}$. One important property of this operation is the fact (Bracewell, 2000, Chapter 6), that circular convolution can efficiently be computed using the discrete Fourier transform:

$$\mathbf{v} \circledast \mathbf{w} = IDFT\left(DFT(\mathbf{v}) \odot DFT(\mathbf{w})\right), \tag{2}$$

where $\odot$ denotes element-wise multiplication, DFT and IDFT denote the discrete Fourier transform and inverse discrete Fourier transform, respectively. The neutral element regarding circular convolution is $\mathbf{1} = (1, 0, \cdots, 0)$. Furthermore, for any vector $\mathbf{v}$, the vector $\bar{\mathbf{v}} = (v_0, v_{D-1}, \ldots, v_1)$ is a *pseudo-inverse* element with respect to circular convolution, meaning that the vector derived from convolving them is similar to the neural element, i.e., $\mathbf{v} \circledast \bar{\mathbf{v}} \approx \mathbf{1}$. Although we can also find an exact inverse element $\mathbf{v}^{-1}$ for most vectors with $\mathbf{v} \circledast \mathbf{v}^{-1} = \mathbf{1}$, it is often more useful to work with pseudo-inverses instead of exact inverse elements, as they can become unstable in certain situations. However, we call the special class of vectors for which the pseudo- and exact inverse element coincide *unitary vectors*, i.e., $\mathbf{v}^{-1} = \bar{\mathbf{v}}$.

Using Equation (2), we define the *convolutive power* of a vector v by an exponent $p \in \mathbb{R}$ as

$$\mathbf{v}^p := \Re\left(IDFT\left(\left(DFT_0\left(\mathbf{v}\right)^p\right), \ldots, \left(DFT_{D-1}\left(\mathbf{v}\right)^p\right)\right)\right), \quad (3)$$

where $\Re$ denotes the real part of a complex number and $DFT_i(\mathbf{v})$ denotes the ith component of the vector $DFT(\mathbf{v})$. Denoting the set of unitary vectors by $\mathcal{U}$, we state three essential properties

- All elements of $\mathcal{U}$ have unit length, i.e., we have $\|\mathbf{u}\| = 1$ for any vector $\mathbf{u} \in \mathcal{U}$.
- $\mathcal{U}$ is closed under convolutive exponentiation, i.e., $\mathbf{u}^p \in \mathcal{U}$ for any $\mathbf{u} \in \mathcal{U}$ and $p \in \mathbb{R}$.
- Convolution with unitary vectors preserves the norm, i.e., $\|\mathbf{v}\| = \|\mathbf{v} \circledast \mathbf{u}\|$ for any $\mathbf{v}$ and any unitary vector $\mathbf{u} \in \mathcal{U}$.

2.1.2. Convolutive-Power Representation

In this paper, we adopt and improve the vector representation for automotive scenes introduced in earlier work (Mirus et al., 2018). Here, we introduce the convolutive vector-power shown in Equation (3) for encoding spatial positions of multiple vehicles and focus on investigating its expressive power. To create a vocabulary V of atomic vectors, we assign a random real-valued vector from the unit sphere to each category of dynamic objects (e.g., car, motorcycle, truck) as well as random unitary vectors $\mathbf{X}$ and $\mathbf{Y}$ to encode spatial positions. We use unitary vectors for $\mathbf{X}$ and $\mathbf{Y}$ as they have unit length and are closed under convolutive exponentiation. Therefore, by encoding spatial positions with powers of unitary vectors, we avoid exploding lengths of our final scene vectors, which would lead to additional noise and unwanted behavior when using them as input for neural networks. Furthermore, we use additional random ID-vectors **TARGET** and **EGO** representing the target object to be predicted and, if applicable, the ego-vehicle, respectively.

Given a situation as shown in **Figure 1A** with a sequence of prior positions (x_t, y_t) for the target vehicle at time step $t \in \{t_0, \ldots, t_N\}$ and equivalent sequences $(x_{obj,t}, y_{obj,t})$ for other traffic participants, we encapsulate this positional information in a scene vector

$$\mathbf{S}_t = \underbrace{\mathbf{TARGET} \circledast \mathbf{TYPE}_{target} \circledast \mathbf{X}^{x_t} \circledast \mathbf{Y}^{y_t}}_{\text{target-vehicle}} \oplus \underbrace{\sum_{obj} \mathbf{TYPE}_{obj} \circledast \mathbf{X}^{x_{obj,t}} \circledast \mathbf{Y}^{y_{obj,t}}}_{\text{other objects}} \quad (4)$$

for each time step t. This yields a sequence of semantic scene vectors $\mathbf{S}_t$ for $t \in \{t_0, \ldots, t_N\}$ encoding the past spatial development of objects in the current driving situation. **Figure 2** depicts the aforementioned scene vector representation: the left plots show similarities (depicted as heat map) between the vector $\mathbf{S}_t$ encoding the scene from **Figure 1A** and the vectors $v_i = \mathbf{TARGET} \circledast \mathbf{TYPE}_{target} \circledast \mathbf{X}^{\bar{x}_i} \circledast \mathbf{Y}^{\bar{y}_i}$ for a sequence of discrete position samples $\bar{x}_i, \bar{y}_i$. Similarly, the right plots show similarities between $\mathbf{S}_t$ and $\mathbf{CAR} \circledast \mathbf{X}^{\bar{x}_i} \circledast \mathbf{Y}^{\bar{y}_i}$ visualizing all other objects in the scene of type *car*. We observe clear peaks (bright yellow areas) of higher similarities at the true positions of the encoded objects depending on their type (e.g., car or truck) or if the object is the target object of interest. Hence, we can encode spatial information of several different objects in a sequence of semantic vectors and reliably decode it back out. This allows us to encode automotive scenes with varying number of dynamic objects in a vector representation of fixed dimensionality.

2.2. Models

2.2.1. LSTM Networks

In this work, we use a long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network-architecture for the prediction of vehicle positions. Our network consists of one LSTM encoder and decoder cell for sequence to sequence prediction, which means that the input and the final result of our model is sequential data. The encoder LSTM takes positional data for 20 past, equidistant time frames as input. That is, the input data is a sequence of 20 items of either positions of the target vehicle or a sequence of high-dimensional vectors encoding this positional data (see sections 1, 2.3.4 for further details). Thus, the resulting embedding vector encodes the history of the input data over those 20 time frames. This embedding vector is concatenated with additional auxiliary information to aid the model when predicting the future trajectory of the target vehicle. This auxiliary data is information, that is available to the system when the prediction is to happen, i.e., sensory data available at prediction time or future data about the ego-vehicle, such as its own planned trajectory (see section 3.1.1 for further details on this auxiliary data). Finally, the embedding vector is used as input for the decoder LSTM to predict future vehicle positions. The output of each model is a sequence of 20 positions of the target vehicle predicted over a certain temporal horizon into the future. We use the same network architecture for all encoding schemes of the input data and for both data sets. However, the dimensionality of the input varies over the different encoding schemes while the auxiliary information used to enrich the embedding vector is different depending on the data set (since only one data set is recorded from a driving ego-vehicle). We describe these implementation choices in more detail in section 3.1.1. **Figure 3** visualizes the architecture of our LSTM models indicating modules that change when varying the encoding scheme by a dashed red border whereas parts that change with the data set are highlighted through a dashed blue border.

2.2.2. NEF Networks

As an alternative to the LSTM-models, we also considered a much simpler single-hidden-layer network defined using the

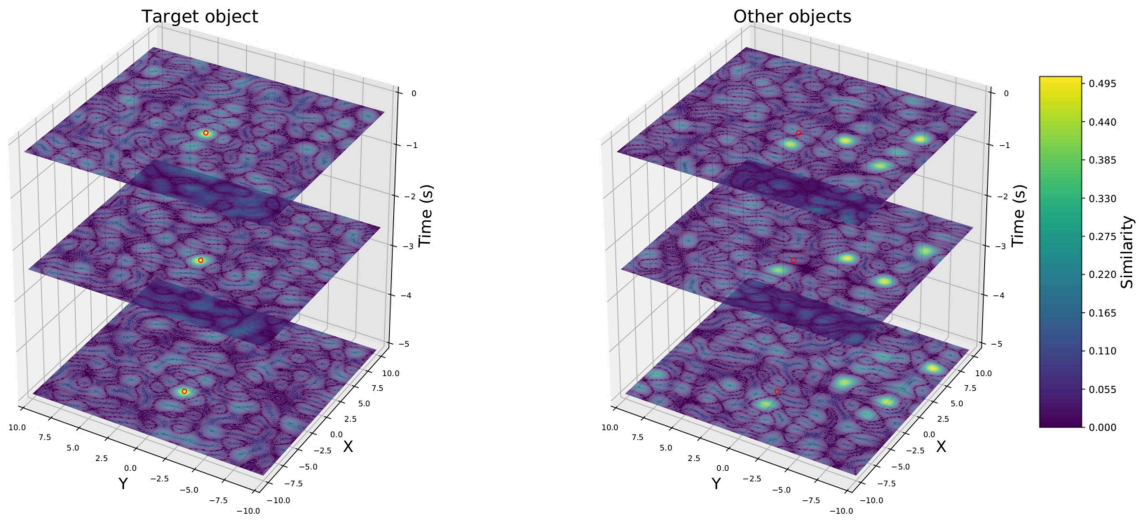


FIGURE 2 | Visualization of the convolutive vector-power representation of one particular driving situation over time at selected time-steps as a heat map of similarity values for 512-dimensional vectors. The red circles indicate the measured position of the target vehicle.

neural engineering framework (NEF) (Eliasmith and Anderson, 2003). While this is usually used for constructing large-scale biologically realistic neuron models (Eliasmith et al., 2012), the NEF software toolkit Nengo (Bekolay et al., 2014) also allows for traditional feed-forward artificial neural networks using either spiking or non-spiking neurons. Spiking neurons are of considerable interest for vehicle prediction algorithms due to the potential for reduced power consumption when run on hardware that is optimized for spiking neurons (i.e., neuromorphic hardware).

For these NEF networks, we use a single hidden layer containing N neurons, with randomly generated (and fixed) input weights, and use least-squares optimization to compute the output weights. That is, given the hidden layer spiking activity a_i for the ith neuron (i.e., a sequence of spikes)

$$a_i\left(\mathbf{x}(t)\right) = \sum_{j=1}^{m_i} h(t) * \delta(t - t_j) = \sum_{j=1}^{m_i} h(t - t_j), \tag{5}$$

where δ denotes the delta function, $h(t)$ is the post-synaptic current produced by a single spike and t_j are the m_i spike times of the ith neuron, we compute the network output $\mathbf{y}$ with output weights $\mathbf{d}_i$ as

$$\mathbf{y(t)} = \sum_{i=1}^{N} a_i\left(\mathbf{x}(t)\right)\mathbf{d}_i. \tag{6}$$

If we have a desired $\mathbf{y}(t)$ for every given input to the network, then we can provide that input, measure the resulting hidden layer activity for each input, and then find the optimal $\mathbf{d}_i$ values to make the network output match the desired output. This is a much faster alternative to using gradient descent rules (such as backpropagation). In particular, we find the $\mathbf{d}_i$ that minimize

$$E = \int \left(\mathbf{y}(t) - \sum_{i=1}^{N} a_i\left(\mathbf{x}(t)\right)\mathbf{d}_i\right)^2 \mathrm{d}\mathbf{x}(t). \tag{7}$$

As with any traditional network, we can have any number of input, output, and hidden neurons, all following this same process. The goal here is to provide a simple baseline for comparison to the LSTM networks, to see what (if any) performance gain is produced by the more complex network approach.

2.2.3. Mixture-of-Experts Online Learning

Given that we have multiple models p_i for $i = 1, \ldots, M$ for predicting vehicle positions, we also define mixture-of-experts models. These are models where the output is a weighted sum of the outputs from other models

$$\mathbf{v}_{mix,t} = \sum_{p} \mathbf{W}_{p,t}\mathbf{v}_{p,t}, \tag{8}$$

where $\mathbf{W}_{p,t}$ is the weight and $\mathbf{v}_{p,t}$ is the output value of the prediction model p for prediction time t. If each model produces a prediction of the x and y positions at N different time steps into the future and we have M models, w will be an $M \times N \times 2$ tensor. In other words, the particular weighting of models for predicting 0.5 s into the future may be very different from the weighting when predicting 5.0 s into the future.

The simplest way to generate these weights is to use standard delta-rule learning

$$\Delta\mathbf{W}_{p,t} = \kappa\mathbf{v}_{p,t}\underbrace{\left(\mathbf{v}_{observed,t} - \mathbf{v}_{mix,t}\right)}_{=\epsilon_t} = \kappa\mathbf{v}_{p,t}\epsilon_t. \tag{9}$$

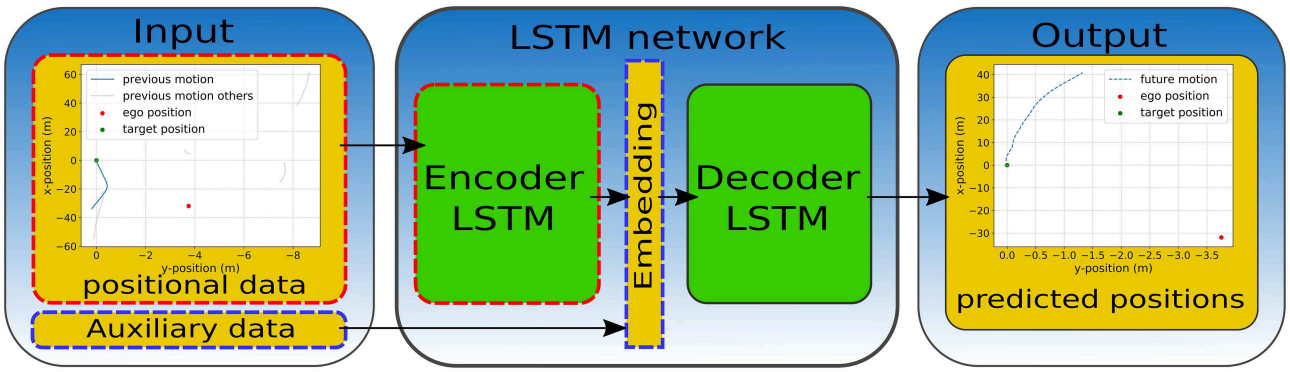


FIGURE 3 | Visualization of our LSTM-based learning architecture. Modules that change with varying encoding scheme of the input data are highlighted through dashed red borders whereas parts that change when varying the data set are highlighted through dashed blue borders.

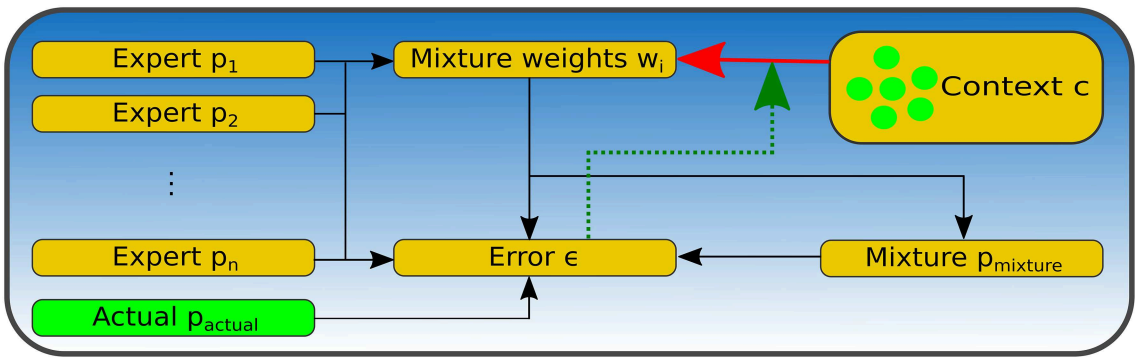


FIGURE 4 | Visualization of the network architecture of the context-sensitive mixture-of-experts online learning system. Yellow boxes indicate the individual components of the model, while the solid red line depicts the connection to decode out the weights $W_{p,t}$ for the individual expert predictors from the neural population encoding the context c as indicated by the green circles in the context component. The dotted green arrow indicates that the error signal is used to update the weights of this connection using delta-rule learning.

where κ is a learning rate and $\epsilon_t = \mathbf{v}_{observed,t} - \mathbf{v}_{mix,t}$ is the current prediction error, that is, the error between the mixture model's output $\mathbf{v}_{mix,t}$ and the target vehicle's actual position $\mathbf{v}_{observed,t}$. For this paper, we initialize the weights $\mathbf{W}_{p,t}$ to be $1/M$ (i.e., an equal weighting across all M models).

The above model attempts to find the best weighting of the offline models based only on the prediction error. However, it is also possible to learn a weighting that is based on the current *context*. That is, instead of learning $\mathbf{W}$, we can learn the function $f_{\mathbf{W}}(\mathbf{c}) = \mathbf{W}$, where $\mathbf{c}$ is some currently available sensor information.

Since neural networks are good function approximators, we implement this context-sensitive mixture-of-experts model as a single-hidden-layer artificial neural network (ANN) whose inputs are $\mathbf{c}$ and whose outputs are $\mathbf{W}$. As with the context-free mixture-of-experts model, we initialize the output to always produce $1/M$, and then train the network based on the prediction error.

Importantly, this context-sensitive mixture-of-experts model is meant to be trained *online*. That is, we do not pre-train it on a large corpus of data and then fix the final weights. Instead, we run the neural network training *while the system is running*, just like the context-free version. Indeed, the context-free version is equivalent to the context-sensitive model if the context is kept constant.

While any neural network learning algorithm could be used here, for simplicity we use delta rule again, and note that the delta rule is the first step of the classic backpropagation neural network learning algorithm. In other words, we only adjust the weights between the hidden layer and the output layer, and leave the other set of weights at their initial randomly generated values. This greatly reduces the computation cost of performing the online learning.

Figure 4 shows a schematic visualization of the mixture-of-expert model's architecture. Yellow boxes indicate the individual components of the model, while the solid red line depicts the connection to decode out the weights $\mathbf{W}_{p,t}$ for the individual expert predictors from the neural population encoding the context $\mathbf{c}$ as indicated by the green circles in the context component. Finally, the dotted green arrow indicates that the error signal is used to update the weights of this connection using delta-rule learning.

2.3. Data and Pre-processing

In this work, we use two different data sets for training and evaluation of our system, which we describe in more detail in the subsequent sections. We refer to those data sets as *On-board* or D_1 (see section 2.3.1), which is our main data set, and *NGSIM* or D_2 (see section 2.3.2), which is a publicly available data set used for reference and comparability. In this section, we describe both data sets regarding available features, available sources of information as well as their key differences and the preprocessing procedure.

2.3.1. On-Board-Sensors Data Set

This is our main data set used in this work. It contains real-world data gathered using the (ego-) vehicle's on-board sensors during test drives mainly on highways in southern Germany. The data contains object-lists with a variety of features obtained from different sensor sources. Apart from features about motion and behavior of the dynamic objects in the scene like position, velocity and acceleration, which are estimated from light detection and ranging (LIDAR) sensors, there is also visual information like object type probabilities or lane information, which is acquired from additional camera sensors. More detailed information on the test vehicle's sensor setup can be found in Aeberhard et al. (2015). The fused information about objects is available at a frequency of roughly 5 Hz. The main feature of this data set is that all information about other vehicles, such as position or velocity are measured with respect to the ego-vehicle and its coordinate system. The *On-board* data set contains 3,891 vehicles, which yield a total length of roughly 28.3 h when adding up the time each individual vehicle is visible.

2.3.2. NGSIM US-101 Data Set

The next generation simulation (NGSIM) US-101 data set (Colyar and Halkias, 2017) is a publicly available data set recorded on a segment of ∼640 m length with 6 lanes on the US-101 freeway in Los Angeles, California. Although the data set was originally intended for driver behavior and traffic flow models (He, 2017), it has also been used to train trajectory predictions models (Altche and de La Fortelle, 2017; Deo and Trivedi, 2018b). The data set was recorded using cameras observing freeway traffic from rooftops with trajectory-data being extracted later from the obtained video footage. It holds a total of 45 min of driving data split into three 15 min segments of mild, moderate and congesting traffic conditions. Apart from positional information in lateral and longitudinal direction (in a global and local coordinate system), additional features like instantaneous velocity, acceleration, vehicle size as well as the current lane are available for each vehicle. The trajectory data is sampled with a frequency of 10 Hz. The main difference to the *On-board* data set is the fact, that the *NGSIM* data set is recorded with an external stationary camera instead of on-board sensors of a driving vehicle. Thus, there is no ego-vehicle present in the data and all information are available in absolute coordinates instead of being measured relative to one particular ego-vehicle. The *NGSIM* data set contains 5,930 vehicles and therefore a total time of roughly 91.3 h when adding up the time each individual vehicle is visible.

2.3.3. Pre-processing

In this section, we describe the preprocessing steps performed a priori to prepare the information from our two data sets as neural network input. Although we aim to keep these preprocessing steps as consistent as possible across the data sets, there are some mild differences, which we will also point out. We aim to predict future positions of dynamic objects 5 s into the future based on their positions 5 s prior to their current location. As the two data sets are sampled at different frequencies, we interpolate the available data over 20 equidistant steps to achieve intervals of 0.25 s to improve consistency and comparability. Furthermore, we translate the current position of the target vehicle (the vehicle to be predicted) into the origin, i.e., position $(0, 0)$ (see **Figure 1**), to prevent our models from treating similar trajectories differently due to positional variations. Finally, to improve suitability of the data as input for neural networks, we divide all x-positions by a factor of 10 such that x-/y-values are scaled to a similar order of magnitude. Thus, one data sample consists of a sequence of length 20 of positional information over the past 5 s, which is used as input for our models with different encoding, and a sequence of 20 positions 5 s into the future used as labels or ground truth for the models to be trained with. For the *NGSIM* data set D_2, we use only every 10th data point, to avoid the creation of too many overlapping, and therefore too similar, data samples. Furthermore, we converted all values to the metric system and swapped the dimensions of the positions in D_2 such that for both data sets x- and y-direction correspond to longitudinal and lateral positions, respectively. For training and evaluating our models, we split both data sets into training $T_i \subset D_i$ and validation data $V_i \subset D_i$ containing 90% and 10% of the objects, respectively to avoid testing our models on vehicles they have been trained with.

2.3.4. Encoding Schemes

We use different encoding schemes of the positional input data in this work. The main encoding scheme is the convolutive vector-power representation as depicted in section 2.1.2. To avoid accumulation of noise while focusing on the vehicles most relevant for prediction, we only use objects closer than 40 m to the target vehicle in the *On-board* data set. For the *NGSIM* data set D_2, we additionally include only objects on the same lane as the target vehicle and on adjacent lanes. Thereby, we aim for consistency across both data sets and we keep the input data as comparable as possible to what a driving vehicle could be able to detect using its on-board sensors.

For the *On-board* data set D_1, we use two different variants of this representation, which differ in that the ego-vehicle's position is used or excluded in the *other objects* part of Equation (4), yielding two sequences $(\mathbf{S}_t^{ego})_{t_0}^{t_N}$ and $(\mathbf{S}_t)_{t_0}^{t_N}$. We used Nengo's SPA package for implementation and therefore refer to these encoding two schemes $(\mathbf{S}_t)_{t_0}^{t_N}$ and $(\mathbf{S}_t^{ego})_{t_0}^{t_N}$ as "SPA-power" and "SPA-power-with-ego," respectively. As the *NGSIM* data set D_2 does not contain an ego-vehicle, we only investigate the "SPA-power" encoding scheme there.

For a simple reference vector-representation, we add the positional vectors $\mathbf{X}$ and $\mathbf{Y}$ scaled with the target vehicle's prior positions (x_t, y_t) at each time step t, yielding the sequence $\tilde{\mathbf{S}}_t =$

$x_t \cdot \mathbf{X} + y_t \cdot \mathbf{Y}$. Finally, we also use plain numerical position values $p_t = (x_t, y_t)$ as input data. Note, that only the SPA-power representation variants $(\mathbf{S}_t)_{t_0}^{t_N}$ and $(\mathbf{S}_t^{ego})_{t_0}^{t_N}$ contain positional information about vehicles other than the target.

3. EXPERIMENTS AND RESULTS

In this section, we describe the training process and parameters of all our models and give a detailed analysis and evaluation of the results achieved. The LSTM models are implemented in Tensorflow (Abadi et al., 2016) whereas the NEF models and the mixture-of-experts online learning model are implemented using the Nengo software suite (Bekolay et al., 2014). We use the root-mean-square error (RMSE) as our main metric for evaluation purposes. In contrast to earlier work, we inspect the RMSE for lateral and longitudinal directions separately to give more detailed insights into the models' behavior. Calculating the RMSE of the Euclidean distance would absorb the influence of the lateral RMSE since it is an order of magnitude smaller than the longitudinal RMSE, while we consider both directions to be at least equally important. The lateral RMSE is even more informative regarding the models' performance on, for instance, lane change maneuvers. Note however, that this means that the *y*-axes in **Figures 5**, **6**, **8–11** show a different order of magnitude for lateral (RMSE X) and longitudinal (RMSE Y) direction. Finally, we investigate where the models shows their best performance looking for correlations between prediction accuracy and specific driving situations.

Table 1 summarizes the models evaluated in this section. The models LSTM SPA 1–3 as well as LSTM numerical employ the same network architecture as described in section 2.2.1 with sequential information as input data (using the different encoding schemes presented in section 2.3.4) and are analyzed in section 3.2.1. The models NEF SPA 1 and 2 employ the simpler, single-layer, feed-forward architecture as described in section 2.2.2 with a vector obtained as partial sum of vectors from the whole sequence used as input for the LSTM models (see section 3.1.2 for further details). Finally, mix online denotes the mixture-of-experts online learning model as described in section 2.2.3 using the predictions from some of the aforementioned offline models as input (see section 3.1.3 for further details). The models will be denoted in figure legends by their short name given in **Table 1**.

In section 2.3.4, we have described the different encoding schemes we will use to evaluate our models. We mentioned that the models employing the convolutive power to encode the input data are (i.e., LSTM SPA 1, 3 and NEF SPA 1 and 2) are the only ones having access to information about objects other than the target vehicle. Although these model therefore have access to more data than the other reference models, such as LSTM numerical, we are interested in evaluating the benefits of encoding the interconnections between vehicles implicitly in the input data using our semantic vector encoding instead of introducing a more complex network architecture. Therefore, we focus on the same network architecture for all encoding schemes in this paper and leave a comparison with more sophisticated network architectures, for instance, ones combining LSTM with social pooling layers as in Deo and Trivedi (2018a) or Alahi et al. (2016) for future work.

3.1. Model Training

3.1.1. LSTM Networks

We trained several instantiations of our LSTM-network architecture as described in section 2.2.1 on the *On-board* data set D_1 in advance to find an optimal set of parameters. We varied the number of layers, the number of hidden dimensions and the number of epochs for the models to be trained. We found, that increasing the number of layers does not improve the models' performance on the validation data, even when training longer using more epochs. On the contrary, models with more layers needed more training time to achieve a performance on the validation data comparable to the networks with less layers. Thus, a LSTM model with one encoder and decoder cell each is not only the simplest network architecture but also the best in terms of accuracy as well as time needed for training.

For this architecture, we found that the network performs best with 150 dimensions in the encoder and decoder cell each. Furthermore, we employed early stopping, that is, we trained our models for 10 epochs as we found that the models' performance stagnate on both, training and validation data sets, when training for up until a total 20 epochs. **Figure 5** visualizes this result by showing the development of the RMSE of the LSTM SPA 1 model during the training process for the training set T_1 (**Figures 5A,C**) and validation set V_1 (**Figures 5B,D**) of the *On-board* data set D_1. On the *y*-axis of each sub-figure, we have the RMSE while the *x*-axis from left to right depicts the result after each epoch during the training process. Each colored line illustrates the RMSE of the model for one particular prediction time step while all points with the same value on the *x*-axis depict the model's performance after the respective epoch during the training process.

Using the aforementioned network architecture and hyperparameter set, we train one model instantiation for each encoding scheme mentioned in section 2.3.4, whereas only the input dimensionality of the encoder cell changes when varying the representation of the input data. Importantly, we focus on positional information as the only input for our LSTM models in this work for reasons of consistency to make all models as comparable as possible. Hence, we neglect for example the history of the target (or ego-) vehicle's velocity or acceleration as input here. Between the two data sets, the only difference between models is the auxiliary data, that is used as additional input to the LSTM decoder cell at each time step. For both data sets, we use the instantaneous velocity of the target vehicle to aid the model predicting the future trajectory at every time step. As there is no ego-vehicle present, we use no further auxiliary data for the *NGSIM* data set D_2. For the *On-board* data set D_1, we use the ego-vehicle's predicted acceleration and the estimated curvature of the ego-vehicle's current lane. Although this is future information, we argue that it is solely about the ego-vehicle, which we expect to be available at the time the prediction is to happen. We assume, that an automated vehicle, in order to safely navigate, will have an estimation of the future lane curvature as well as the acceleration values of its own planned trajectory.

TABLE 1 | Summary of the evaluated models regarding architecture, input data, encoding, and training.

Short name	Input	Position encoding	Network architecture	Training	Number of units/Neurons	Data set
Linear	Current position and velocity	–	Linear regression	–	–	Both
LSTM numerical	Sequence of positions	–	LSTM with one encoder/decoder cell each	Offline, backpropagation	150 units per cell	Both
LSTM SPA 1	Semantic vector sequence	Convolutive power	LSTM with one encoder/decoder cell each	Offline, backpropagation	150 units per cell	Both
LSTM SPA 2	Semantic vector sequence	Scalar multiplication	LSTM with one encoder/decoder cell each	Offline, backpropagation	150 units per cell	Both
LSTM SPA 3	Semantic vector sequence	Convolutive power incl. ego-vehicle	LSTM with one encoder/decoder cell each	Offline, backpropagation	150 units per cell	*On-board*
NEF numerical	Sequence of positions	–	NEF single-layer	Offline, least-squares	3,000 neurons	Both
NEF SPA 1	Semantic vector sum	Convolutive power incl. ego-vehicle	NEF single-layer	Offline, least-squares	3,000 neurons	*On-board*
NEF SPA 2	Semantic vector sum	Convolutive power	NEF single-layer	Offline, least-squares	3,000 neurons	*NGSIM*
Mix online	Predictions offline models	–	NEF single-layer	Online, delta-rule	3,000 neurons	Both

3.1.2. NEF Networks

For our NEF networks, the main parameters influencing the models' performance are the number of neurons in the learning population (i.e., the hidden layer in terms of traditional neural networks), and the neuron model. For simplicity, we use Nengo's rate-variant of the leaky-integrate-and-fire (LIF) neuron model. From the NEF-theory (Eliasmith and Anderson, 2003) we know that increasing the number of neurons in a population yields a more accurate representation of the data encoded in the population's activity. Thus, we expect more accurate predictions when increasing the number of neurons. In our experiments, we found that a number of 3,000 spiking neurons is sufficient and further increasing the number of neurons does not improve the model's prediction accuracy. The neural weights are calculated using Nengo's default least-squares-optimization method with the exception, that we calculate the weights over smaller subsets of the input data for computational reasons.

We train two different variants of our simpler NEF-models using numerical input (NEF SPA numerical) as well as the SPA-power-with-ego (NEF SPA 1) and SPA-power encoding (NEF SPA 2) for the *On-board* and the *NGSIM* data set, respectively. Here, we adapt the input data such that for the model NEF numerical, we use the vector $(x_{t_0}, \ldots, x_{t_N}, y_{t_0}, \ldots, y_{t_N}, v)$ as input with v denoting the instantaneous velocity of the target vehicle. For the NEF SPA 1 and 2 models, instead of flattening the whole sequence of vectors into a giant single vector, we created a single semantic vector by summing the first, middle, and last element of the original vector sequences

$$\hat{\mathbf{S}} = \mathbf{S}_{t_0} \oplus \mathbf{S}_{t_{N/2}} \oplus \mathbf{S}_{t_N} = (\hat{s}_0, \ldots, \hat{s}_{D-1}). \tag{10}$$

We only sum up these vectors instead of the whole sequence $(\mathbf{S}_t)_{t_0}^{t_N}$ to avoid the accumulation of noise in the vector representation. Note that thereby the NEF model using the SPA-power representation does not use the full trajectory history but only selected time steps, namely those visualized in **Figure 2**. To make these simpler models as comparable as possible to the LSTM models in terms of information available to the network, we add the instantaneously velocity v of the target vehicle as an additional element to the input, which yields $(\hat{s}_0, \ldots, \hat{s}_{D-1}, v)$ as input of our model, since there is no intermediate embedding vector here where it could be included.

3.1.3. Mixture-of-Experts Online Learning

There are two different possible variants to our mixture-of-experts online learning model. One issue of such a learning system is that the actual position information of the target vehicle $\mathbf{v}_{observed,t}$ and thus the error ϵ_t in Equation (9) is not available at the time the model makes its predictions, since it is future data. In this paper, we show a first prototype that, for simplicity, ignores this delay issue and assumes that position information of the target vehicle $\mathbf{v}_{observed,t}$ actually is available at prediction time. In the future, we aim to investigate an online learning system that updates its weights $\mathbf{W}_{p,t}$ once the error signal ϵ_t gradually becomes available. However, the architecture of the model itself remains the same. The only difference to the prototype shown here is the time when Equation (9) is applied to update the neural weights. For the context-sensitive mixture-of-experts model, we use information about the current driving situation as identified in section 3.2.1 and **Figure 7** as context **c** for the learning system.

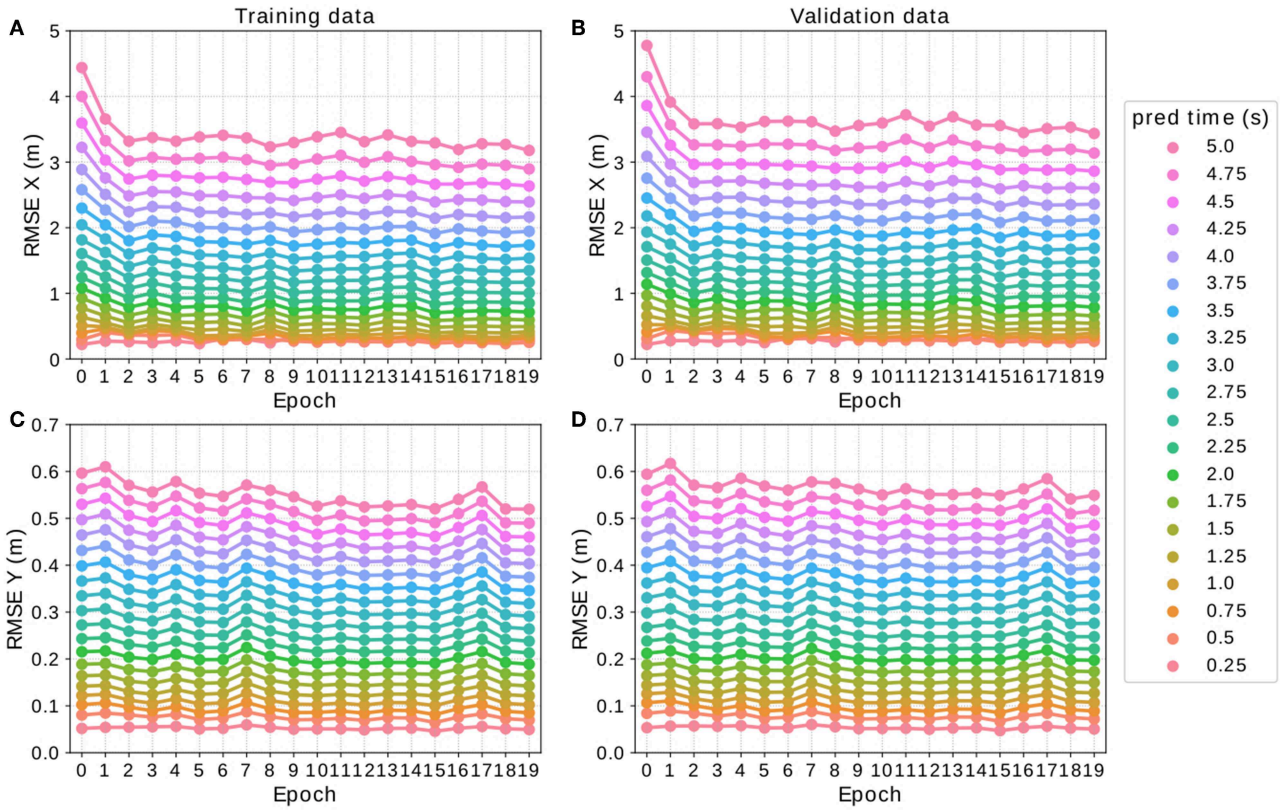


FIGURE 5 | Development of the RMSE at every prediction time step during the training process of the LSTM model using the SPA-power-with-ego vector representation (LSTM SPA 3) after each epoch on the training **(A,C)** and validation part **(B,D)** of the *On-board* data set. Each colored line illustrates the RMSE of the model for one particular prediction time step while all points with the same value on the *x*-axis depict the model's performance after the respective epoch during the training process. One observes comparable trends on both training and validation set and that the RMSE stagnates after 10 epochs.

For the *NGSIM* data set, we use the distance between the target-vehicle and the closest other vehicle as well as the number of surrounding relevant vehicles as context information. Relevant means that those vehicles that are included in the SPA-power representation are counted (see section 2.3.4). For the *On-board* data set, the distance between the target and the ego-vehicle is additionally included in the context.

In this work, we employ the pre-trained LSTM models using numerical inputs (i.e., LSTM numerical), the best-performing SPA-power encoding scheme for each data set (i.e., LSTM SPA 3 for the *On-board* data set and LSTM SPA 1 for the *NGSIM* data set), and a simple linear prediction as input experts for our online learning prototype. For training the model, we simulate online deployment by presenting the offline models' predictions on the validation subsets to the system. Thereby, the individual experts perform their prediction on previously unseen data samples. We conduct individual simulation runs for both data sets.

3.2. Evaluation

In this section, we evaluate the performance of our models and conduct a thorough analysis of the results achieved. For all evaluations in this section, we refer to the longitudinal and lateral direction as *x*- and *y*-direction, respectively.

3.2.1. LSTM Models

Figure 6 visualizes the RMSE of all LSTM-based models on the validation-set $V_1 \subset D_1$ of the *On-board* data set using 512-dimensional vectors. **Figures 6A,C** show the performance on the complete validation-set in *x*- and *y*-direction, respectively, whereas **Figures 6B,D** depict only situations with at least 3 other vehicles present, the distance between the target and the ego-vehicle being lower than 20 m and the distance between the target and the closest other vehicle being <10 m, again for *x*- and *y*-direction, respectively. We observe that all approaches yield comparable results with notable differences in certain situations. Although the SPA-power encoding schemes (LSTM SPA 1 and 3) tend to perform worst in *x*-direction, we observe that they perform better in *y*-direction in crowded situations with closely driving vehicles with LSTM SPA 3 ranking best along LSTM numerical.

To further investigate this result, we evaluated certain metrics, chosen to characterize crowded and potentially dangerous situations, for items in the validation set, where the LSTM

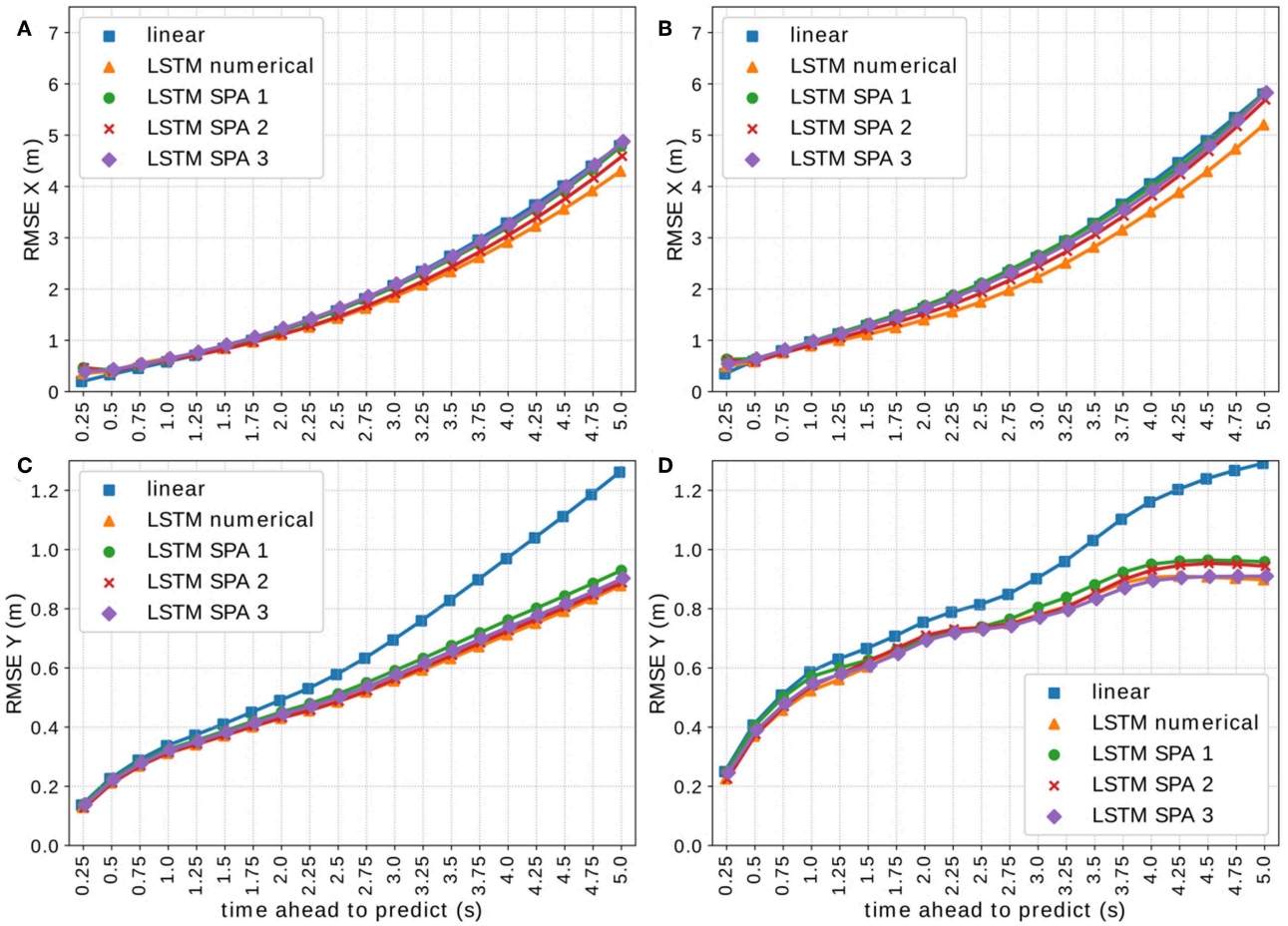


FIGURE 6 | Visualization of the RMSE of all LSTM models on the complete *On-board* validation set $V_1 \subset D_1$ in *x*- **(A)** and *y*-direction **(C)** and on a subset of situations with at least 3 other vehicles present and distance between the target and ego vehicle lower than 20 m and between target and closest other vehicle lower than 10 m, again in *x*- **(B)** and *y*-direction **(D)**.

SPA 3 model outperforms all other approaches with respect to the RMSE in *y*-direction (see **Figure 7**). We observe that the number of samples, where the distance between the target and the ego vehicle and/or the closest other object being small is significantly higher when the LSTM SPA 3 model outperforms all other approaches. For samples where the LSTM SPA 3 model performs best, the number of samples with a distance <20 m between the target- and ego-vehicle is 50.5 % higher compared to samples where any of the other models performs best. For distances <20 m between the target vehicle and the closest other vehicle, the number of samples is still 11.4 % higher when the LSTM SPA 3 model performs best. Finally, the number of situations with at least 3 other vehicles present is also 7.8 % higher compared to samples where any other model performs best. Thus, we consider these characteristics suitable candidates to serve as context variables on which our online-learning mixture-of-experts system could base its weighting decision on. However, we aim to investigate more sophisticated options, such as clustering methods in future work to uncover other, potentially more meaningful features compared to the ones shown in this paper, distinguishing between situations where LSTM SPA 3 performs best compared to another model showing the best performance.

Figure 8 visualizes the RMSE of all LSTM-based models on the validation-set $V_2 \subset D_2$ of the *NGSIM* data set for 512-dimensional vectors (**Figures 8A,C**) and for 1,024-dimensional vectors (**Figures 8B,D**). We observe, that all LSTM models achieve a very similar performance (almost identical in *y*-direction) with LSTM SPA 1 achieving the best performance in *x*-direction being on par with the numerical encoding for 512-dimensional vectors. For 1,024-dimensional vectors, LSTM SPA 1 even slightly outperforms all other approaches in *x*-direction, whereas we do not observe significant improvements in *y*-direction.

3.2.2. NEF Networks

Figure 9 visualizes the RMSE of our NEF-network models on both data sets. The NEF-network using the SPA-power encoding

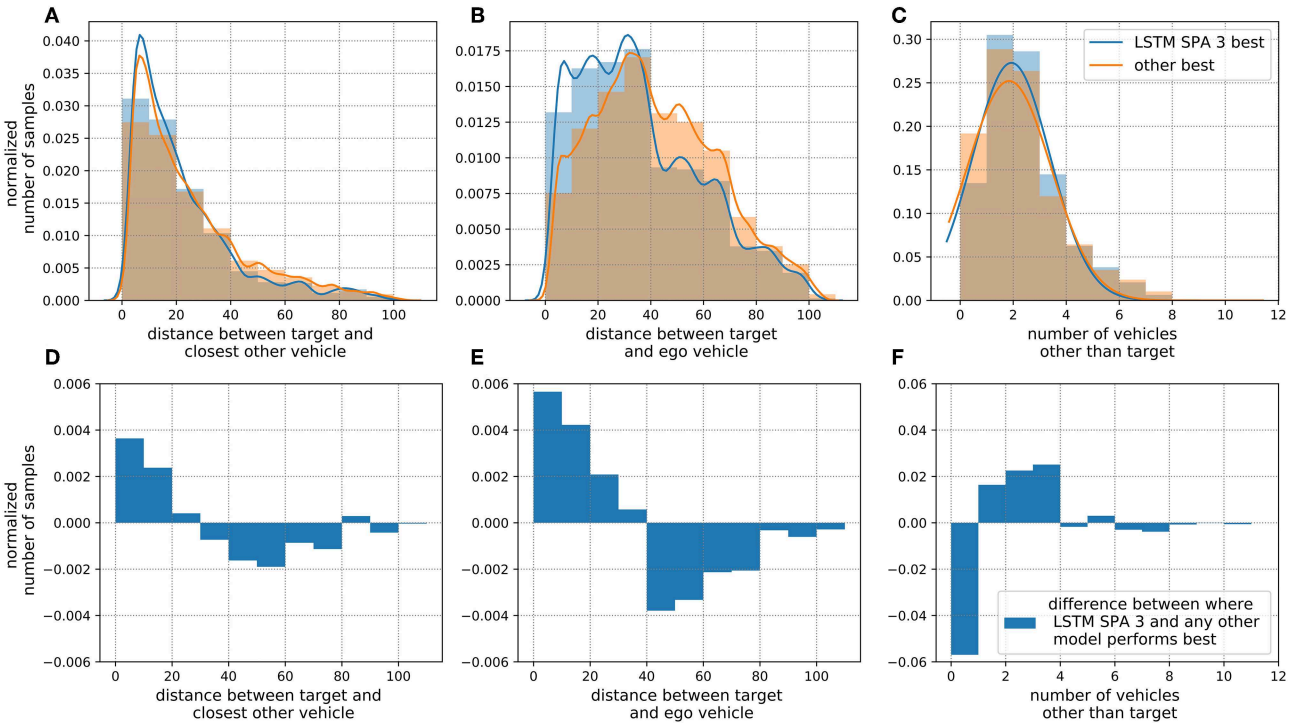


FIGURE 7 | Metric evaluation specifying situations where the LSTM SPA 3 model outperforms all other approaches regarding the RMSE in *y*-direction on the *On-board* data set D_1. In the upper row **(A–C)**, blue bars illustrate samples where LSTM SPA 3 performs better than all other models while the orange bars depict samples where any other model performs best. Panel **(A)** illustrates the distance between the target vehicle and the closest other vehicle, **(B)** illustrates the distance between the target and the ego-vehicle and **(C)** shows the number of vehicles other than the target. The lower row **(D–F)** illustrates the difference between the blue and orange bars in the corresponding upper panel.

schemes processes 512-dimensional for the *On-board* (NEF SPA 1) and 1,024-dimensional vectors for the *NGSIM* data set (NEF SPA 2). For reference, we included the performance of the most relevant LSTM models, namely LSTM SPA 1 and 3 for the *NGSIM* and *On-board* data set, respectively as well as LSTM numerical, in **Figure 9** as well. We observe that, despite a simpler network architecture and learning algorithm, the NEF-networks achieve a performance comparable to the more sophisticated LSTM models on both data sets. For the *NGSIM* data set, the NEF SPA 1 model performs on par with its LSTM model counterpart LSTM SPA 3. In this case, the NEF-model is not only simpler, but also has access to less information as its input data is a sum of a subset of the input sequence used for the corresponding LSTM-model.

3.2.3. Mixture-of-Experts Online Learning

Figure 10 shows the RMSE on selected slices of the validation-sets achieved by our context-sensitive mixture-of-experts online learning prototype, which assumes the error signal is available at the time the prediction needs to happen in comparison to the offline models. The four left plots (**Figures 10A,B,E,F**) show two data slices of the validation set D_1 of the *On-board* data set: **Figures 10A,E** show the RMSE at the start of training process while **Figures 10B,F** show the RMSE performance on the first 70 vehicles. Similarly, the four right plots (**Figures 10C,D,G,H**) show two data slices of the validation set D_2 of the *NGSIM* data set: **Figures 10C,G** show the RMSE at the start of the training process while **Figures 10D,H** show the RMSE on the first 92 vehicles. From **Figures 10A,E,C,G** we observe, that the model needs some time for adapting its weights yielding a RMSE performance worse than the individual experts for both data sets. However, the model's performance improves quickly and clearly outperforms all individual experts in *x*-direction while achieving RMSEs as low as the best individual experts in *y*-direction after a comparably low number of vehicles presented to the system. **Figures 10B,F** illustrate this result for the *On-board* data set, while **Figures 10D,H** show comparable results achieved by the mixture model on the *NGSIM* data set.

To get a better idea of how our model weights the individual predictors, we inspect one example driving situation. **Figure 11** visualizes the performance of our mixture-of-experts online learning prototype on one particular example of the *On-board* data set. We use a situation not directly after the start of the training process, i.e., the mixture model was already exposed to some vehicles and thus was able to consolidate its weights. **Figures 11A–C** show the driving situation with the vehicles' true trajectories as well as the trajectory predictions

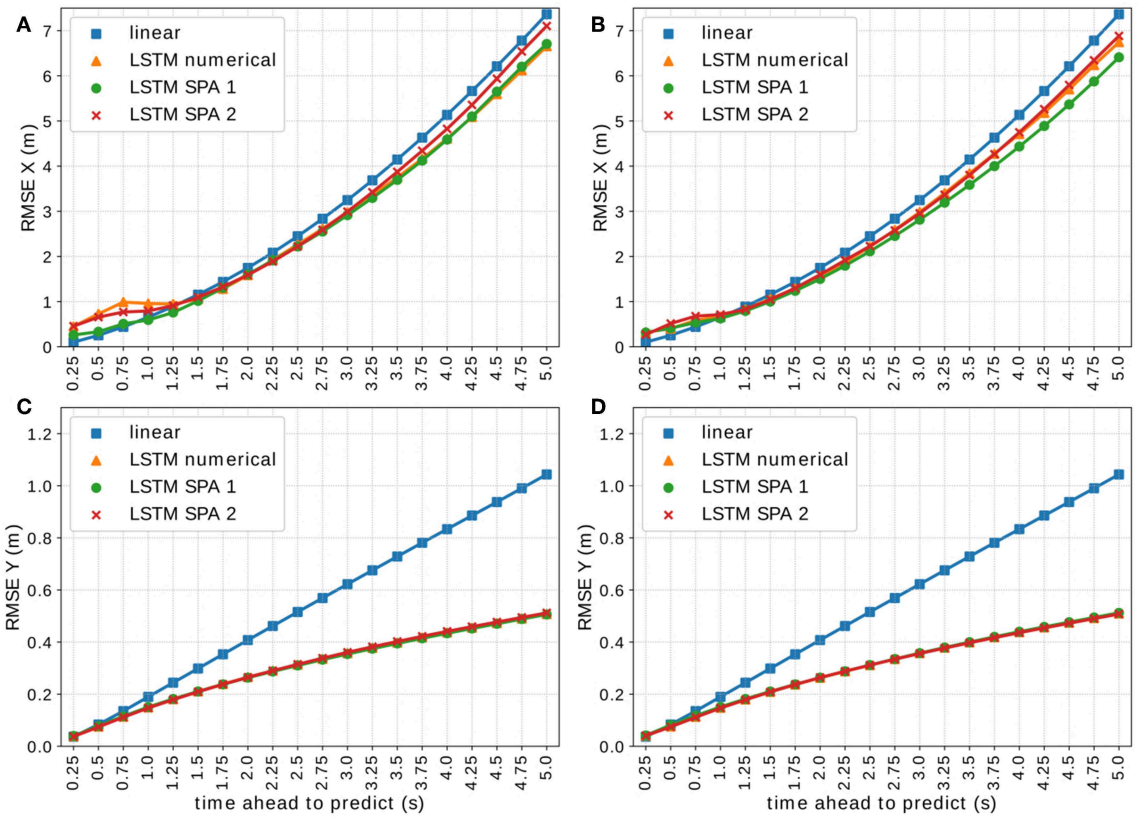


FIGURE 8 | Visualization of the RMSE of all LSTM models on the *NGSIM* validation set $V_2 \subset D_2$ using vectors of dimension 512 for the LSTM SPA 1 and 2 models in *x*- **(A)** and *y*-direction **(C)** and using vectors of dimension 1,024 for the LSTM SPA 1 and 2 models in *x*- **(B)** and *y*-direction **(D)**.

given by the offline models and the mixture-of-experts online learning prototype. **Figures 11D,E** show the absolute error of all approaches while **Figures 11F,G** visualize, how the model weights the individual experts for every prediction time step in this particular driving situation. We observe that the overall trend of our model shows in this example as well. The mixture-of-experts prototype achieves significant improvements in the *x*-direction while achieving RMSEs comparably low as the best individual expert in *y*-direction (**Figures 11D,E**). Furthermore, **Figures 11F,G** show that the model weights the expert predictors independently at individual time steps and hence is able to pick the best possible predictor at each time step. However, we also observe, that the error of the mixture-of-experts model in the *y*-direction is higher than the best individual predictor and that the weighting, especially for later prediction steps, could be improved.

4. DISCUSSION

For both data sets used in this paper, we observe that already the simple linear prediction models achieve solid accuracy, especially in longitudinal direction. This makes sense as both data sets almost exclusively contain highway driving situations, which in turn consist mainly of straight driving and rather rare lane-change maneuvers. For straight driving, linear prediction based on a constant velocity assumption is already a solid prediction approach, especially if all dynamic information (position, velocity etc.) are given relative to an already moving ego-vehicle like with the *On-board* data set D_1. **Table 2** summarizes the composition of both data sets.

For the *On-board* data set, in 86.1 % of all data samples the target vehicle does not perform a lane, i.e., only 13.8 % of all data samples contain a lane change performed by the target vehicle. We further distinguish between lane changes performed during the trajectory history, i.e., the past 5 s before the current time step (labeled as *past* in **Table 2**) and lane changes that are performed in the future, i.e., the future 5 s from the current time step (labeled as *future* in **Table 2**). For the *NGSIM*, the percentage of samples without a target vehicle lane change is 95.1 % while only 4.9 % of the samples contain a lane change performed by the target vehicle at all. The amount of samples containing a future lane change performed by the target vehicle is only 2.6 % of all samples in the *NGSIM* data set.

For the offline models, simple feed-forward NEF models and more sophisticated LSTM models alike, we observe that most improvements over the linear model are achieved in *y*-direction. That makes sense as linear prediction is unable to account for

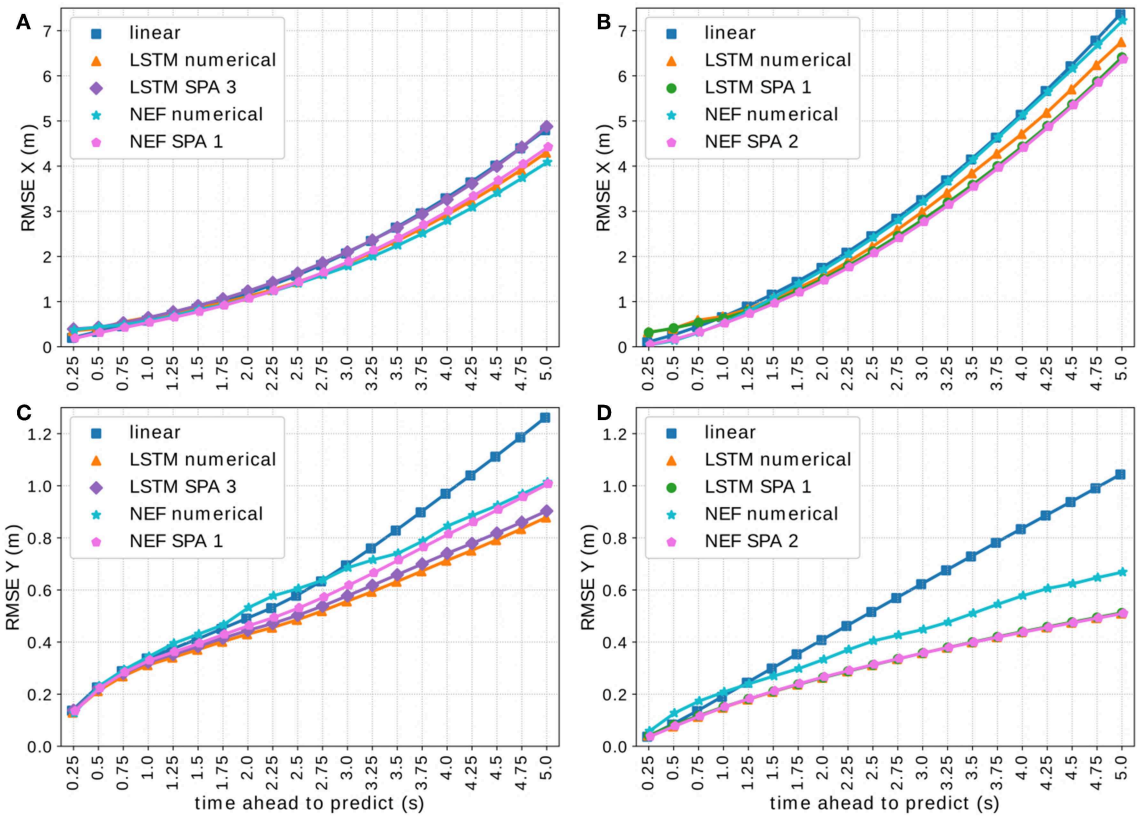


FIGURE 9 | Visualization of the RMSE in *x*- **(A,B)** and *y*-direction **(C,D)** of the NEF SPA 1 model on the *On-board* validation set $V_1 \subset D_1$ using 512-dimensional vectors for the SPA-power vectors **(A,C)** and the NEF SPA 2 model on the *NGSIM* data set D_2 using 1,024-dimensional vectors for the SPA-power vectors **(B,D)**.

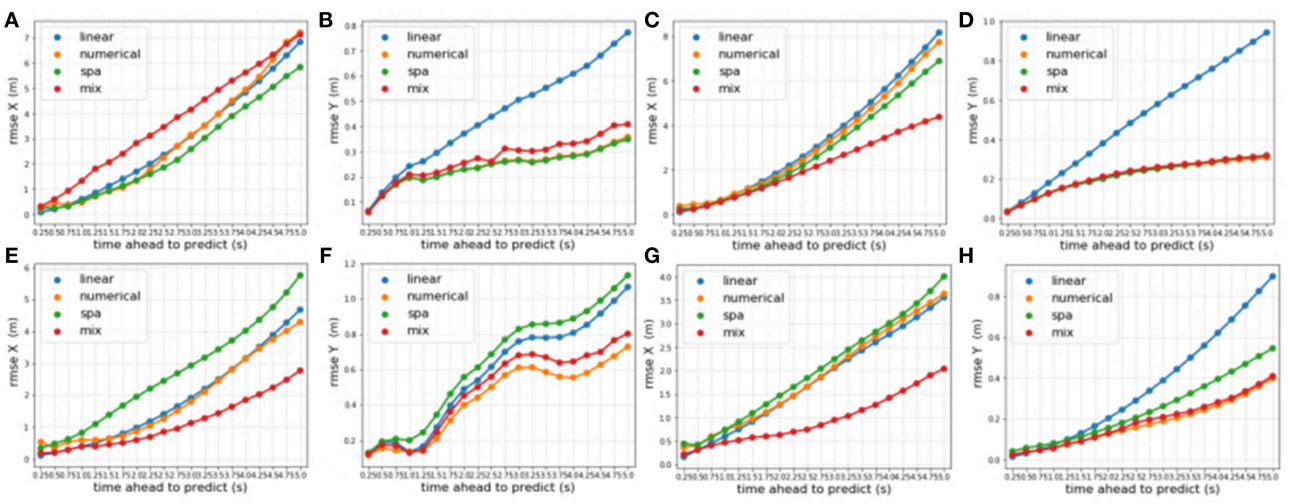


FIGURE 10 | Visualization of the RMSE of the context-sensitive mixture-of-experts online learning system on selected data-slices from the validation sets. The upper row shows the RMSE in *x*-direction **(A–D)**, while the lower row shows the RMSE in *y*-direction **(E–H)**. Panels **(A,E)** show the RMSE on the *On-board* data set at the start of training process while **(B,F)** show the RMSE performance on the first 70 vehicles. Similarly, panel **(C,G)** show the RMSE on the *NGSIM* data set at the start of the training process while **(D,H)** show the RMSE on the first 92 vehicles.

lane-changes or driving curves, which are mainly characterized by non-linear changes in lateral direction. We found that the LSTM models based on our SPA-power representation (LSTM SPA 1 and 3) achieve promising results on both data sets. However, for the *On-board* data set, this encoding scheme achieves its best result in crowded and potentially dangerous

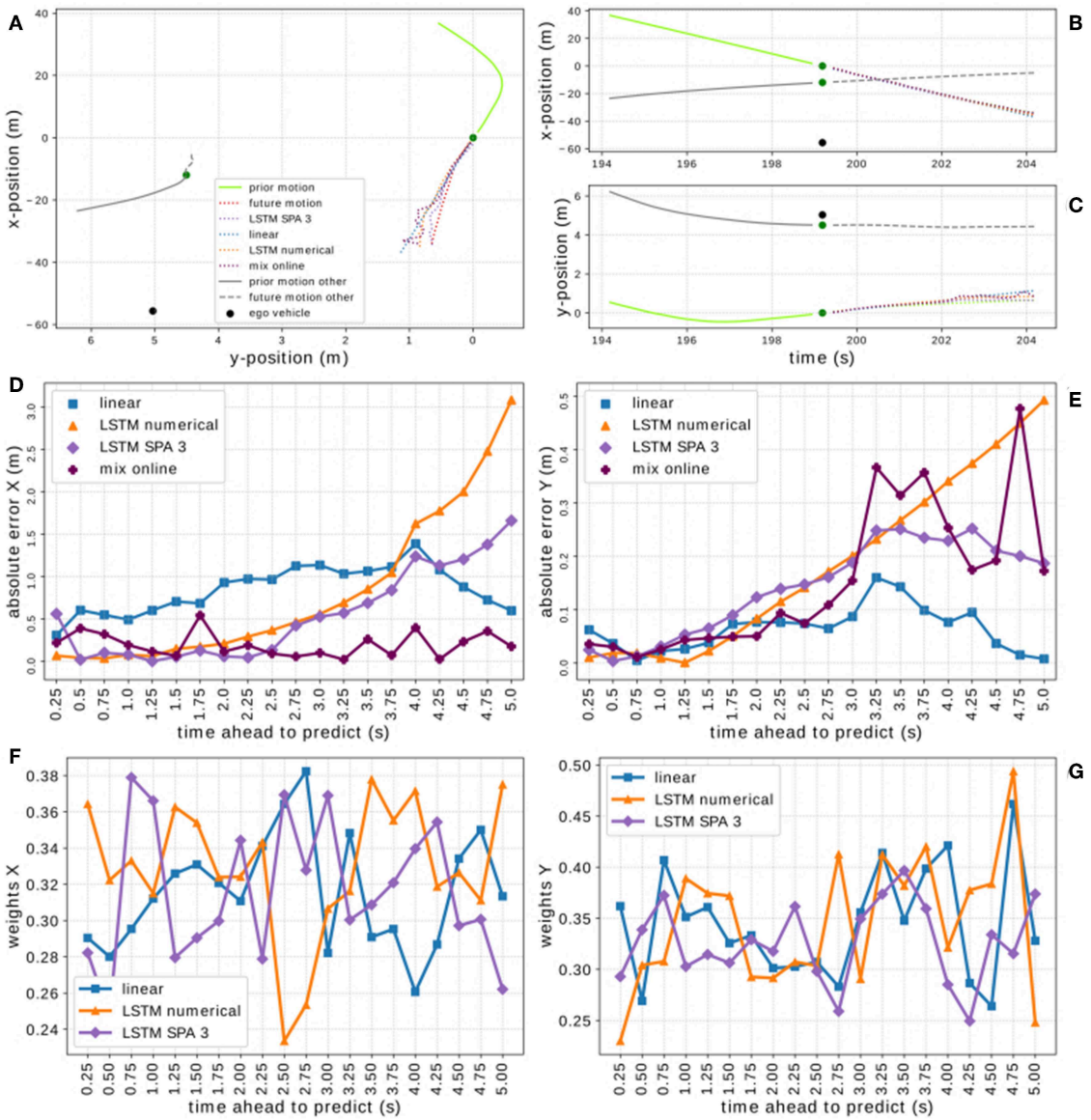


FIGURE 11 | Visualization of the context-sensitive mixture-of-experts online learning system on one particular driving situation from the *On-board* data set. Panels **(A–C)** depict the driving situation with the vehicles' true trajectories as well as the trajectory predictions given by the offline models and the mixture-of-experts online learning prototype. Panels **(D,E)** show the absolute error of all prediction models on that data-sample. Panels **(F,G)** visualize, how the mixture model weights the individual experts for every prediction time step in this particular driving situation.

TABLE 2 | Composition of both data sets regarding straight driving and samples containing a lane change performed by the target vehicle.

Data set	Straight driving	Total target vehicle lane changes	Past target vehicle lane changes	Future target vehicle lane changes
On-board	86.1%	13.9%	7%	8.2%
NGSIM	95.1%	4.9%	2.7%	2.6%

driving situations, without clearly outperforming the other approaches on the whole data set (see section 3.2.1 and **Figure 6**). Given these finding, we investigated situations, where the LSTM SPA 3 model does outperform all other approaches in y-direction and thereby came up with metrics characterizing such crowded situations (see **Figure 7**). This result did not hold that clearly on the *NGSIM* data set D_2, since the LSTM models achieve an almost identical performance in y-direction on this data set.

Nevertheless, we used the identified characteristics as context information for our first prototype of a mixture-of-experts online learning system based on simple delta-rule learning. For simplicity, the prototypical model shown here ignores the fact that measurements of the actual trajectory and thus the error signal for the learning system is future data, i.e., only available with a timing delay, and applies Equation (8) instantaneously. We tested and evaluated this prototype on both data sets achieving comparable results. We found that already shortly after initialization, the online learning system is able to adapt its weights to significantly improve its performance over the individual expert systems. Interestingly, the mixture-of-experts model achieves the most improvements over the individual experts in the *x*-direction although the characteristics used as context were derived from analyzing the LSTM models' performance in the *y*-direction. We assume that this is due to the fact, that the individual LSTM experts already show a closer-to-optimal performance in the *y*-direction with less room for improvements. Furthermore, the sample situation shown in **Figure 11** exemplifies another potential problem of the current model in the *y*-direction: with a distance of 12.8 m between the target and the closest other vehicle, a distance of 55.8 m between the target and the ego-vehicle and only one other vehicle present, this is not a typical situation for the LSTM SPA 3 model to perform best in the *y*-direction (cf. **Figure 7**) and thus this expert might not be weighted strongly enough by the model. However, these effects demand for further and more detailed investigation.

Another interesting result of our experiments is the fact, that the simple, feed-forward NEF networks show results comparable to the more sophisticated LSTM models. For those simple models, the SPA-power representation (NEF SPA 1 and 2) shows promising results comparable to the NEF numerical model on the *On-board* data set and clearly outperforming it on the *NGSIM* data set (**Figure 9**). Although the NEF models do not clearly outperform the LSTM models (which would be surprising), it is quite remarkable that they achieve results comparable to the more sophisticated models with a simpler network architecture, training procedure and, partly, less information. These results make those simple models using our proposed vector-representation as well as a numerical encoding scheme (possibly in combination with an online learning system like the one proposed in this paper) potential candidates to be deployed on dedicated neuromorphic hardware in mobile applications, as they can be efficiently implemented in a spiking neuron substrate. This could be an interesting, power-efficient approach in future automated vehicles.

4.1. Conclusion

In this paper, we showed a novel approach to encapsulate spatial information of multiple objects in a sequence of semantic pointers of fixed vector length. We used a LSTM sequence to sequence model as well as a simple feed-forward spiking neural network to predict future vehicle positions from this representation. For each of those models, we implemented at least one reference model using other encoding schemes to compare their performance to. Furthermore, we compared all our models to a simple linear prediction based on a constant velocity assumption. We evaluated our models on two different data sets, one recorded with on-board sensors from a driving vehicle and one publicly available trajectory data set recorded with an external camera observing a highway segment and conducted a thorough analysis. Finally, we used our pre-trained LSTM networks as basis for a mixture-of-experts online learning prototype and compared its performance to the individual expert systems. We consider our main contributions the proposed representation of spatial information for multiple objects in semantic vectors of fixed length using the convolutive power, the rigorous and detailed analysis of several simple and more advanced models, and the prototype of our online learning system.

4.2. Future Work

Although the results presented in this paper show promise, there are several directions for future work. Regarding our LSTM models, we aim to investigate if increasing the vector dimension further leads to improved model performance on the *On-board* data set, as the results on the *NGSIM* suggest that there is potential for improvements (see **Figures 8A,B**). Furthermore, our preliminary hyperparameter experiments suggest, that there is potential for improvements by incorporating the history of the target and/or ego-vehicle's velocity and/or acceleration. Therefore, we could investigate possibilities of how to encode such information in a semantic vector substrate. Another interesting option for the offline models is to investigate if a reduced, more balanced data set could improve the models accuracy or at least speed up the training process. As mentioned in section 4 and **Table 2**, both data sets are slightly unbalanced as they are dominated by straight driving, which is most common in highway situations. One possibility could be to use the current data sets and focus the training procedure on "interesting scenes," i.e., situations where for example a lane change is happening by for instance looking for data samples with significant differences in the lateral positions. Another option is to improve our current models to predict a probability distribution of the future positions instead of point predictions of raw position values to take uncertainties into account. Finally, we could also compare our current models to other state-of-the-art models, which combine LSTM and social pooling layers, which we did not include in the work at hand.

Regarding our mixture-of-experts online learning prototype, we have shown a simplified version ignoring the fact that the error signal is future data and thus can not be used instantaneously, but rather becomes gradually available over time. Although the network architecture and learning approach would remain unchanged, the timing when the weights' update happens needs to be implemented and investigated if and how this affects the models performance. However, the results achieved in this paper serve as an upper bound for the performance to be expected from models that have to deal with delayed error signals, that is, that the target vehicle's true motion is future data and thus not available at prediction time, The issue of delayed error signals was mentioned but, for simplicity, not addressed in this work. However, assuming that a model overpredicting the near future most likely will also overpredict

for later time steps, we could also experiment with model variants that update the weights for later prediction steps based on the error signal for earlier prediction steps before the error signal actually becomes available. Another direction could be to investigate if and how different context information affect the model's performance.

Since advanced driver assistance systems and, more generally, automated driving are mobile applications with tight energy restrictions, we finally aim to investigate if and how our current implementation could be deployed on dedicated, energy-efficient neuromorphic hardware for mobile, in-vehicle applications.

AUTHOR CONTRIBUTIONS

FM has designed and implemented all the model variants in Tensorflow and Nengo, designed and performed the experiments, pre-processed data, evaluated results, and wrote the manuscript. PB has designed the numerical LSTM models in Tensorflow and assisted in data pre-processing, experiments and evaluation, and revised the manuscript. TS has designed the models in Nengo, assisted in data pre-processing, experiments and evaluation, and contributed in writing the manuscript. JC coordinated and supervised the research work, and revised the manuscript.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., et al. (2016). "TensorFlow: a system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16 (Berkeley, CA: USENIX Association), 265–283.

Aeberhard, M., Rauch, S., Bahram, M., Tanzmeister, G., Thomas, J., Pilat, Y., et al. (2015). Experience, results and lessons learned from automated driving on Germany's highways. *IEEE Intell. Transport. Syst. Mag.* 7, 42–57. doi: 10.1109/MITS.2014.2360306

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). "Social LSTM: human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV), 961–971.

Altche, F., and de La Fortelle, A. (2017). "An LSTM network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (Yokohama: IEEE), 353–359.

Bahram, M., Hubmann, C., Lawitzky, A., Aeberhard, M., and Wollherr, D. (2016). A combined model- and learning-based framework for interaction-aware maneuver prediction. *IEEE Trans. Intell. Transport. Syst.* 17, 1538–1550. doi: 10.1109/TITS.2015.2506642

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., et al. (2014). Nengo: a Python tool for building large-scale functional brain models. *Front. Neuroinform.* 7:48. doi: 10.3389/fninf.2013.00048

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., et al. (2016). End to end learning for self-driving cars. *arXiv [Preprint] arXiv:1604.07316*.

Bonnin, S., Kummert, F., and Schmüdderich, J. (2012). "A generic concept of a system for predicting driving behaviors," in *2012 15th International IEEE Conference on Intelligent Transportation Systems* (Anchorage, AK), 1803–1808.

Bracewell, R. (2000). *The Fourier Transform and Its Applications*. Electrical Engineering Series. Tokyo: McGraw Hill.

Chang, A. X. M., and Culurciello, E. (2017). "Hardware accelerators for recurrent neural networks on FPGA," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (Baltimore, MD), 1–4.

Ciresan, D. C., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Netw.* 32, 333–338. doi: 10.1016/j.neunet.2012.02.023

Colyar, J., and Halkias, J. (2017). *US Highway 101 Dataset*. Available online at: https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm

Deo, N., and Trivedi, M. M. (2018a). Convolutional social pooling for vehicle trajectory prediction. *arXiv [Preprint] arXiv:1805.06771*.

Deo, N., and Trivedi, M. M. (2018b). "Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMS," in *2018 IEEE Intelligent Vehicles Symposium (IV)* (Changshu: IEEE), 1179–1184.

Eliasmith, C. (2013). *How to Build a Brain: A Neural Architecture for Biological Cognition*. New York, NY: Oxford University Press.

Eliasmith, C., and Anderson, C. H. (2003). *Neural Engineering : Computation, Representation, and Dynamics in Neurobiological Systems*. Computational Neuroscience. Cambridge, MA: MIT Press.

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

Gayler, R. W. (2003). "Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience," in *Proceedings of the ICCS/ASCS International Conference on Cognitive Science* (Sydney, NSW), 13–17 July 2003, 133–138.

He, Z. (2017). *Research Based on High-Fidelity NGSIM Vehicle Trajectory Datasets: A Review*. Technical Report. Beijing: Beijing University of Technology.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.* 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Lawitzky, A., Althoff, D., Passenberg, C. F., Tanzmeister, G., Wollherr, D., and Buss, M. (2013). "Interactive scene prediction for automotive applications," in *2013 IEEE Intelligent Vehicles Symposium (IV)* (Gold Coast, QLD), 1028–1033. doi: 10.1109/IVS.2013.6629601

Lefèvre, S., Vasquez, D., and Laugier, C. (2014). A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH J.* 1:1. doi: 10.1186/s40648-014-0001-z

Mirus, F., Stewart, T. C., and Conradt, J. (2018). "Towards cognitive automotive environment modelling: reasoning based on vector representations," in *26th European Symposium on Artificial Neural Networks, ESANN 2018* (Bruges), 55–60.

Plate, T. (1994). *Distributed representations and nested compositional structure* (PhD thesis). University of Toronto, Toronto, ON, Canada.

Polychronopoulos, A., Tsogas, M., Amditis, A., and Andreone, L. (2007). Sensor fusion for predicting vehicles' path for collision avoidance systems. *IEEE Trans. Intell. Transport. Syst.* 8, 549–562. doi: 10.1109/TITS.2007.903439

Schmüdderich, J., Rebhan, S., Weisswange, T., Kleinehagenbrock, M., Kastner, R., Nishigaki, M., et al. (2015). "A novel approach to driver behavior prediction using scene context and physical evidence for intelligent adaptive cruise control (I-ACC)," in *Future Active Safety Technology Towards Zero Traffic Accidents (FAST-Zero)* (Gothenburg: FISITA).

Widdows, D., and Cohen, T. (2014). Reasoning with vectors: a continuous model for fast robust inference. *Logic J. IGPL* 23, 141–173. doi: 10.1093/jigpal/jzu028

Conflict of Interest: FM was employed by BMW AG. PB and TS were employed by Applied Brain Research Inc.

The remaining 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.

Advantages of publishing in Frontiers

OPEN ACCESS
Articles are free to read for greatest visibility and readership

FAST PUBLICATION
Around 90 days from submission to decision

HIGH QUALITY PEER-REVIEW
Rigorous, collaborative, and constructive peer-review

TRANSPARENT PEER-REVIEW
Editors and reviewers acknowledged by name on published articles

Frontiers
Avenue du Tribunal-Fédéral 34
1005 Lausanne | Switzerland

Visit us: www.frontiersin.org
Contact us: info@frontiersin.org | +41 21 510 17 00

REPRODUCIBILITY OF RESEARCH
Support open data and methods to enhance research reproducibility

DIGITAL PUBLISHING
Articles designed for optimal readership across devices

FOLLOW US
@frontiersin

IMPACT METRICS
Advanced article metrics track visibility across digital media

EXTENSIVE PROMOTION
Marketing and promotion of impactful research

LOOP RESEARCH NETWORK
Our network increases your article's readership