








NeuroVE: Brain-Inspired Linear-Angular Velocity Estimation With Spiking Neural Networks

Xiao Li , Xieyuanli Chen , *Member, IEEE*, Ruibin Guo , Yujie Wu , *Member, IEEE*, Zongtan Zhou , Fangwen Yu , *Member, IEEE*, and Huimin Lu , *Member, IEEE*

Abstract—Vision-based ego-velocity estimation is a fundamental problem in robot state estimation. However, the constraints of frame-based cameras, including motion blur and insufficient frame rates in dynamic settings, readily lead to the failure of conventional velocity estimation techniques. Mammals exhibit a remarkable ability to accurately estimate their ego-velocity during aggressive movement. Hence, integrating this capability into robots shows great promise for addressing these challenges. In this letter, we propose a brain-inspired framework for linear-angular velocity estimation, dubbed NeuroVE. The NeuroVE framework employs an event camera to capture the motion information and implements spiking neural networks (SNNs) to simulate the brain's spatial cells' function for velocity estimation. We formulate the velocity estimation as a time-series forecasting problem. To this end, we design an Astrocyte Leaky Integrate-and-Fire (ALIF) neuron model to encode continuous values. Additionally, we have developed an Astrocyte Spiking Long Short-term Memory (ASLSTM) structure, which significantly improves the time-series forecasting capabilities, enabling an accurate estimate of ego-velocity. Results from both simulation and real-world experiments indicate that NeuroVE has achieved an approximate 60% increase in accuracy compared to other SNN-based approaches.

Index Terms—Neurorobotics, bioinspired robot learning, SLAM.

I. INTRODUCTION

VISION-BASED ego-velocity estimation is essential for robot state estimation, especially in dynamic environments. In intricate, highly dynamic scenes, conventional

Received 28 August 2024; accepted 3 January 2025. Date of publication 13 January 2025; date of current version 28 January 2025. This article was recommended for publication by Associate Editor L. Jamone and Editor T. Ogata upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation of China under Grant 42201456 and Grant 62403478, in part by the Young Elite Scientists Sponsorship Program by CAST under Grant 2023QNRC001, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20243064, and in part by the Independence Science Foundation of National University of Defense Technology under Grant 24-ZZCX-GZZ-13. (*Corresponding authors: Huimin Lu; Fangwen Yu.*)

Xiao Li is with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410000, China, and also with the Center for Brain-Inspired Computing Research, and Department of Precision Instrument, Tsinghua University, Beijing 100084, China.

Xieyuanli Chen, Ruibin Guo, Zongtan Zhou, and Huimin Lu are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410000, China (e-mail: lhmnew@nudt.edu.cn).

Yujie Wu is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, SAR, China.

Fangwen Yu is with the Center for Brain-Inspired Computing Research, and Department of Precision Instrument, Tsinghua University, Beijing 100084, China (e-mail: yufangwen@tsinghua.edu.cn).

Digital Object Identifier 10.1109/LRA.2025.3529319

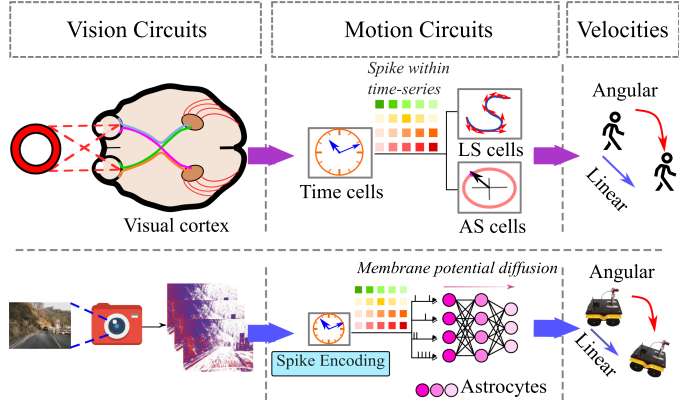


Fig. 1. The linear and angular speed estimation pipeline mirrors the processes of the human brain and robotic systems. In the vision circuits, the sensor transforms external light into spike signals. In the motion circuits, these signals are first encoded with temporal information. Subsequently, linear speed (LS) and angular speed (AS) cells translate these spikes into velocity signals.

frame-based cameras encounter limitations such as motion blur and varying illumination. These limitations often hinder the detection of visual features by traditional vision methods, potentially resulting in failures.

Humans' visual perception and motion estimation demonstrate impressive abilities, particularly in accurately estimating speed and responding swiftly. In the human brain, there exists a special class of cells known as spatial cells, which include linear speed (LS) cells, angular speed (AS) cells [1], and time cells [2]. As shown in Fig 1, the vision motion circuits receive external light signals, which are converted into spikes and then conveyed to the cerebral cortex, where they activate spatial cells. These essential discoveries have inspired us to solve the problem of velocity estimation by mimicking the vision motion circuits in the brain.

In terms of vision circuit modeling, the neuromorphic sensor [3], e.g., event cameras [4], has been developed by leveraging the principles of brain-based vision. It exhibits exceptional temporal resolution and dynamic range, enabling excellent performance in dynamic scenarios where frame-based cameras may fail. However, the asynchronous triggering property of neuromorphic sensors poses challenges for direct applications of these traditional computer vision approaches.

In terms of motion circuit modeling, SNNs are considered the most representative brain-inspired models for modeling human brain functions due to their biological plausibility [5], [6]. SNNs have shown excellent performance in the field of visual tasks [7],

including image classification. However, modeling the sequential velocity estimation problem for time-series learning poses two significant challenges for existing SNN models. Firstly, the inherent sparsity, as well as low-precision spike representation, leads to a deficiency in numerical accuracy, potentially affecting their expressive ability in complex tasks. Secondly, conventional SNN models, like Leaky Integrate-and-Fire (LIF) models, suffer from poor predictive capabilities. When confronted with the intricacies of time-series data. Therefore, it is necessary to devise a novel neuron model and network structure to tackle these challenges effectively.

In this letter, we propose a brain-inspired framework, NeuroVE, specifically tailored to model vision motion circuits, enabling accurate estimations of the linear and angular velocities from event camera data. To address the limitation of numerical precision, we design a novel ALIF neurons model inspired by astrocytes [8]. Astrocytes can synchronize neuronal firing by regulating the ion concentration and neurotransmitter levels among nerve cells, playing an indispensable role in maintaining brain functions such as the circadian rhythm.

To the best of our knowledge, this is the first brain-inspired approach to solving linear and angular velocity estimation problems by mimicking the brain's vision motion circuits. We summarize our main contributions as follows:

- We propose a brain-inspired framework that simulates the vision motion circuits of the brain for velocity estimation. The framework integrates an event camera for capturing motion information and employs brain-inspired SNNs to simulate spatial cells such as LS cells, AS cells, and time cells.
- Inspired by astrocytes, we design a brain-inspired neuron model, ALIF, that incorporates the neurotransmitter diffusion mechanism into the neuron dynamics.
- We introduce a brain-inspired SNN structure, ASLSTM, which integrates the dynamic property of ALIF neurons with the time-series forecasting capability of LSTM.

II. RELATED WORK

A. Numerical Regression With SNNs

Spiking neural networks, a category of computational models inspired by biological neural systems, exhibit distinctive superiority in processing dynamic visual data. The existing studies on numerical regression with SNNs usually employ population coding or membrane potential to encode numerical information.

Population coding represents values through the activation patterns of neuronal populations. N. Iannella et al. use the inter-spike interval (ISI) coding approach to approximate a nonlinear function [9]. Lv et al. propose SNNs designed for time-series forecasting tasks, which utilize fully connected (FC) layer coding to generate its outputs [10]. Employing the FC layer for output coding in SNNs represents an instantiation of population coding, which fundamentally uses the collective activity of neurons to encode information. Gehrig et al. used SNN to estimate the angular velocity of the event camera [11]. Their network architecture comprises six convolutional layers followed by a single fully connected layer.

The spike within SNNs are binary coding in nature. The binary code typically implies a more extensive neuronal population to encode continuous values effectively. To mitigate the redundancy in neuronal populations solely for numerical

encoding, the approach of membrane potential encoding has been introduced. The membrane potential encoding approach maps discrete spiking to continuous values by the non-spiking neurons. Consequently, the absence of a reset mechanism in the neuron's membrane potential implies that the membrane potential can represent continuous values.

Recently, several applications have attempted to use membrane potentials to encode continuous values, including optical flow estimation and image generation, etc [12], [13]. Furthermore, certain classification problems incorporate regression techniques to determine the probability distribution across various classes [14], [15]. Henkes et al. have provided an in-depth discussion on the regression challenges within SNNs and proposed a spiking long short-term memory (SLSTM) model to address numerical problems [16].

Nonetheless, despite these great progresses, the binary spike representation potentially limits the representational precision of existing models. As information theory dictates, a higher precision representation of continuous values necessitates more information. Our approach efficiently encodes continuous values in SNNs through the introduction of a mechanism for the diffusion of membrane potentials.

B. Event-Based Velocity Estimation

While significant progress has been achieved in navigation and localization, exploring first-order kinematics remains under-explored. In the current event-based odometry approach, some studies have optimized the pose and velocity of the event camera concurrently as state variables, which is especially prevalent in visual-inertial odometry (VIO) processes. Mueggler et al. have introduced a continuous-time VIO framework. Unlike discrete-time VIO, which estimates pose and velocity separately and may be susceptible to inconsistencies, the continuous-time approach offers a coherent and unified representation of pose and velocity [17]. In [18], a method is proposed for simultaneously estimating an event camera's velocity and pose, leveraging a known photometric 3D map. They derived an intensity-change residual loss and determined the pose and velocity of the event camera by nonlinear optimization.

Another vein of research focuses on the unique asynchronous spatio-temporal properties of event cameras for the direct estimation of velocity. A pioneering work goes back to the contrast maximization (CM) framework introduced by Guillermo et al. [19]. However, the computational cost associated with CM often impedes running in real-time. Peng et al. have introduced a geometry-based velocity estimation method. They have formulated a closed-form solution for linear velocity estimation by employing the trifocal tensor [20]. Li et al. have introduced a calibration method established on linear velocity correlation [21]. They estimate the camera's linear velocity under the constraint of linear motion for calibration. This geometry-based methodology assumes robust feature extraction to capture motion information effectively. More recently, the contributions of Lu et al. are particularly noteworthy, which proposed an advanced velometer [22]. They employed stereos event cameras to determine the depth and normal flow and integrated IMU data within a continuous-time framework.

The majority of existing approaches rely on traditional computer vision pipelines, which fail to fully harness the potential of event cameras. Our brain-inspired computational paradigm

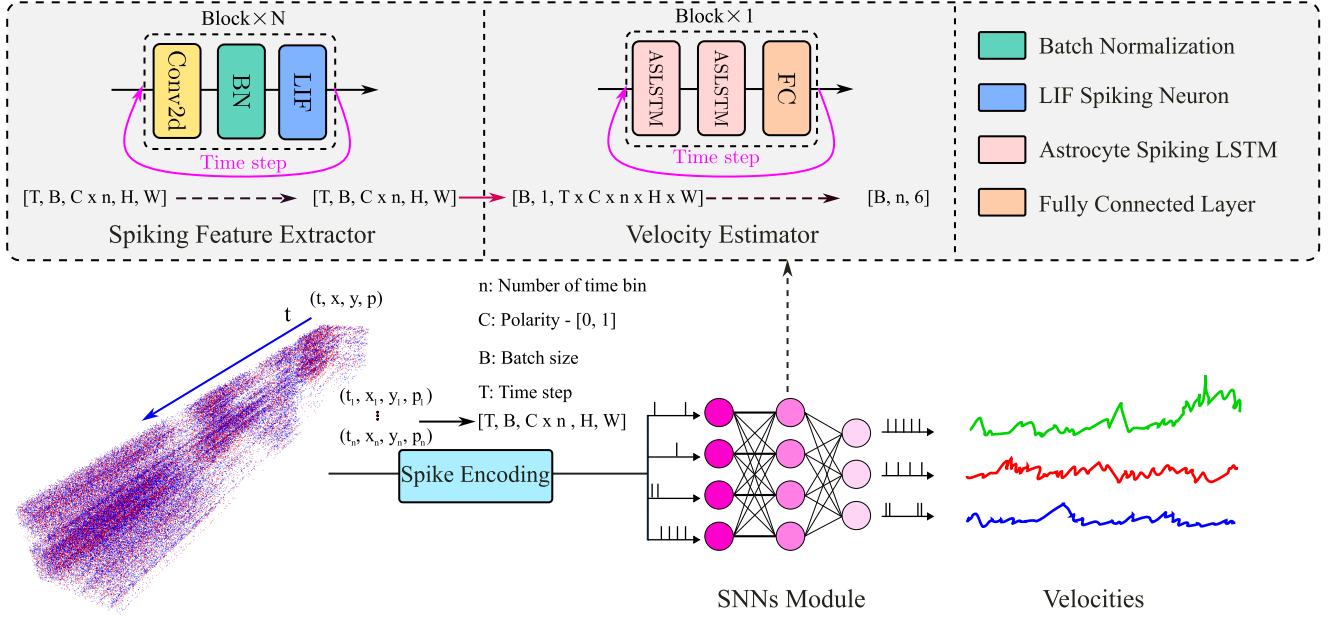


Fig. 2. NeuroVE framework. The event data is denoted as (t, x, y, p) and partitioned into n chronological bins. These events are processed by spike coding and transformed into time-series spikes $[T, B, C \times n, H, W]$, where T represents the time step, B represents the batch size, C represents the number of channels, n represents the number of pieces into which the events are chronologically divided, H and W is the height and width, respectively. Finally, these time-series spikes are processed by the spiking feature extractor and the velocity estimator to infer the linear and angular velocities directly.

effectively integrates the asynchronous spatio-temporal characteristics of event camera data.

III. METHODOLOGY

A. NeuroVE Framework

The NeuroVE framework is depicted in Fig. 2, which consists of three main components: spike encoding, spiking feature extractor, and velocity estimator.

- **Spike Encoding:** The data pre-processing phase, which categorizes event data over a specified time interval based on its polarity, assigns events with negative polarity to channel 0 and positive polarity to channel 1 in each chronological bin.
- **Spiking Feature Extractor:** The spiking feature extractor comprises N sequential blocks, each equipped with a convolutional layer, a batch normalization layer [23], and a LIF neuron.
- **Velocity Estimator:** The velocity estimator receives the high-level features generated by the Spiking Feature Extractor. Furthermore, the velocity estimator integrates ALIF neurons with ALSTM, enabling accurate estimation.

Activation of LS and AS cells in the brain has been demonstrated to exhibit a nonlinear correlation with the linear and angular velocities [1]. Moreover, time cells do not encode absolute temporal information, such as circadian rhythms. Instead, they are specialized in encoding sequential information [2].

Accordingly, the operational mechanism of these cells can be effectively modeled using SNNs. In the spike encoding phase, we simulate the function of time cells by mapping sequential event camera data into n chronological bins. Besides, we regard the SNNs as a nonlinear mapping, like LS and AS cells, that transforms high-dimensional event data into a six-dimensional

velocity vector. Once trained, the network predicts velocity at every moment, even with novel scenarios, by the inherent temporal structure of event data. This time-series forecasting-based velocity estimation approach enables our model to adapt to known data distributions and generalize to new environments.

In the following subsections, we dive into the foundational design principles and implementation details of the ALIF (Section III-B) and ASLSTM (Section III-C) models.

B. Astrocytes Leaky Integrate-and-Fire Neurons

In this section, we introduce the ALIF neuron model, which can effectively solve the instability of SNNs in numerical regression problems.

The basic LIF model can be written as

$$\begin{aligned} \tau \frac{dV(t)}{dt} &= -(V(t) - V_{rest}) + RI(t), \\ V(t) &= V_{rest}, \text{ if } V(t) > V_{th}, \end{aligned} \quad (1)$$

where $V(t)$ is the membrane potentials at time t , V_{rest} is the neuronal resting potential, $I(t)$ is the pre-synaptic input current, R is the neuronal resistance and τ is the leakage constant. When the membrane potential exceeds the threshold V_{th} , the neuron fires an action potential, followed by a membrane potential returns to the resting potential V_{rest} . If the membrane potential fails to reach the threshold V_{th} , it undergoes leakage at $1/\tau$ rate.

However, the sparsity firing pattern induced by the LIF neuron model poses challenges for numerical regression applications. Sparse signals cannot inherently maintain high precision values with integrity. The neuron's membrane potential is initialized at the resting level, requiring sufficient time to accumulate charge to reach the threshold for firing. While increasing the time step relieves this issue to a degree. It concurrently leads to exponentially increasing network dimensions and memory usage.

Moreover, the numerical challenges introduced by sparsity are not effectively addressed by simply increasing the time step, particularly in the time series forecasting problem.

Astrocytes orchestrate the synchronization of neural signals through the modulatory control of neurotransmitter transmission among neurons [8]. Inspired by astrocytes, we introduce a diffusion model grounded in Fick's law [24], designed to simulate the neurotransmitter transfer from astrocytes to neurons.

$$J(i, j) = D \frac{\partial C}{\partial t} = D(V_i - V_j), \quad (2)$$

where $J(i, j)$ represents the diffusion equation between the i th and j th neurons, D is the diffusion coefficient, and C represents the concentration of neurotransmitters on the postsynaptic terminal. However, to avoid the introduction of superfluous hyperparameters, the C is defined as the differential in membrane potentials across neurons. Consequently, we obtain the ALIF model.

$$\begin{aligned} \tau \frac{dV(t)}{dt} &= -(V(t) - V_{rest}) + R(I(t) + J), \\ V(t) &= V_{rest}, \text{ if } V(t) > V_{th}. \end{aligned} \quad (3)$$

Equation (3) implies that the membrane potential is transmitted to adjacent neurons. To derive a gradient-friendly solution for (3), it is effective to adapt the formula for iterative computation and assume that V_{rest} is equal to 0. We employ the Euler method for the first-order differential equation [7] as

$$\begin{aligned} V_j^{t+1} &= (1 - \frac{dt}{\tau})V_j^t + \frac{dt}{\tau}R(I_j^t + J(i, j)), \\ V_j^t &= V_{rest}, \text{ if } V_j^t > V_{th}, \end{aligned} \quad (4)$$

where the upper index t and lower index j denote time moment and neuron index, respectively. The pre-synaptic current $I(t)$ is the neuron's input signal X , and R is the operator $f(\cdot)$, encompassing various forms such as convolutional or linear operators. Additionally, we denote $(1 - \frac{dt}{\tau})$ as α and $\frac{dt}{\tau}$ as β . Thus, (4) can be simplified to

$$\begin{aligned} V_j^{t+1} &= \alpha V_j^t + \beta(f(X_j + D(V_i^{t_f} - V_j^{t_0}))), \\ V_j^t &= 0, \text{ if } V_j^t > V_{th}, \end{aligned} \quad (5)$$

where $V_j^{t_0}$ is the initial membrane potential of the j th neuron, $V_i^{t_f}$ is the membrane potential of the i th neuron at t_f moment.

Finally, by simplifying the constant term in (5) and assuming $V_j^{t_0} = 0$, we derive an iterative formulation for the ALIF neuron model.

$$V_j^{t+1} = (1 - S_j^t)V_j^t + f(X_j + DV_i^{t_f}), \quad (6a)$$

$$S_j^t = \mathcal{H}(V_j^t), \quad (6b)$$

$$\mathcal{H} = \begin{cases} 0, & V_j^t < V_{th} \\ 1, & V_j^t \geq V_{th} \end{cases}. \quad (6c)$$

The iterative process of the ALIF neuron model involves two sequential phases: updating the neuron's state (6a) and spike firing (6b). When the membrane potential exceeds the threshold value, the neuron generates a spike. Subsequently, at the next time step, the neuron will reset its membrane potential depending on the occurrence of spikes. Furthermore, due to the non-differentiable property of (6c), alternative surrogate function [7] is employed during the training phase.

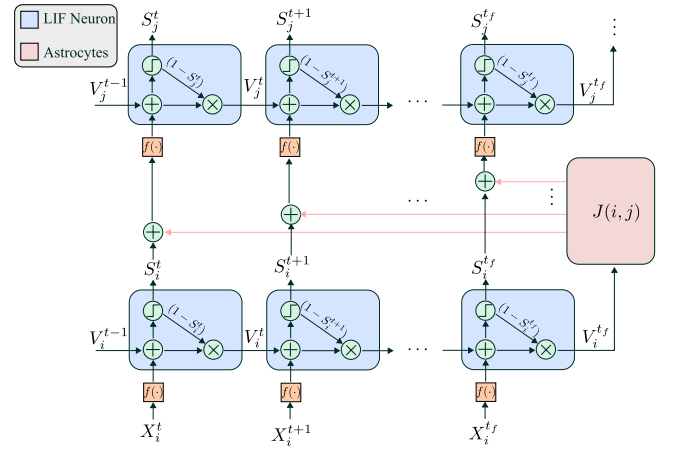


Fig. 3. The illustration describes the diffusion mechanism of membrane potentials in the two-dimensional computational graph of SNNs. Here, the blue block represents the LIF neurons, and the pink block represents the astrocyte-inspired diffusion mechanism.

Fig. 3 illustrates the phenomenon where the membrane potential of the i -th neuron diffuses into the j -th neuron at a predefined rate. In SNNs, inputs propagate concurrently across both temporal and spatial dimensions, which thereby can be unfolded as a two-dimensional computational graph.

We employ a temporally prioritized sequential computational method for our neural network. Computing the activation state of each neuron at a given time step before moving on to the next. This sequential computation ensures that astrocytes are informed of the membrane potential at the final step of the preceding neuron and enables the effective diffusion of these potentials across the network.

C. Astrocytes Spiking Long Short-Term Memory Network

The task of linear-angular velocity regression for an event camera is formulated as a time-series model, capturing the sequential progression of velocities over time. Despite the spatio-temporal dynamics inherent to SNNs, our empirical findings indicate that directly employing plain SNNs into a time-series model is prone to encountering gradient explosion issues during the backpropagation phase.

Accordingly, we first introduce the ASLSTM for the time-series forecasting task, built upon the ALIF model introduced in Section III-B. The ASLSTM introduces supplementary gating mechanisms, similar to standard LSTM [25] and SLSTM [16], enhancing the network's spatial-temporal dynamics.

The ASLSTM can be formulated as

$$c^t = \sigma(f^t) \otimes c^{t-1} + \sigma(i^t) \otimes \tanh(g^t),$$

$$h^t = \sigma(o_t) \otimes \tanh(c_t),$$

$$v^t = \mathcal{A}(h^t, x^t, v^{t-1}),$$

$$\zeta = \mathcal{O}(v^{t_f}, x^{t_f}),$$

$$s^t = \mathcal{H}(v^t),$$

with

$$i^t = W_i x^t + U_i v^{t-1},$$

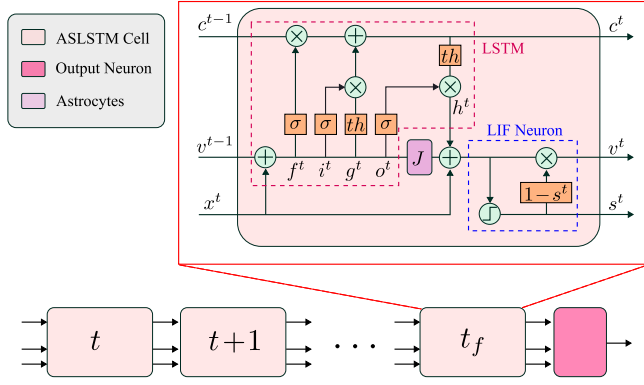


Fig. 4. Elaborate on the mechanism by which astrocytes diffuse membrane potentials and illustrate the integration process with the LIF neurons within the ASLSTM.

$$\begin{aligned} f^t &= W_f x^t + U_f v^{t-1}, \\ g^t &= W_g x^t + U_g v^{t-1}, \\ o^t &= W_o x^t + U_o v^{t-1}, \end{aligned} \quad (7)$$

where i^t , o^t , f^t , and g^t are inputs of the input gate, forget gate, cell state, respectively. The signals are consistent with the standard LSTM, where σ denotes the sigmoid function, \tanh denotes the tangent hyperbolic function, and \otimes is the Hadamard product. \mathcal{H} is the Heaviside step function formulated in Eq. 6c. $\mathcal{O}(\cdot)$ is the output neurons, which can be formulated by

$$\mathcal{O}(v^{t_f}, x^{t_f}) = \kappa v^{t_f} + f(x^{t_f} + J(t-1, t)), \quad 0 < \kappa < 1. \quad (8)$$

$\mathcal{A}(\cdot)$ is the state update function of the ALIF neuron, which can be written as

$$\mathcal{A}(h^t, x^t, v^{t-1}) = \alpha(1 - s^{t-1})h^t + f(x^t + J(t-1, t)), \quad (9)$$

$J(t-1, t)$ represents the diffusion across various time steps within the same neuron.

Fig. 4 represents the internal architecture and synaptic connections within the ASLSTM. The inputs of ASLSTM propagate exclusively along the temporal dimension, rather than spatially, at each time step. Upon reaching the final time step, the output neuron is engaged to emit membrane potentials that represent the continuous values. The region enclosed by the red dashed line in Fig. 4 is the standard LSTM cell. Due to the presence of LIF neurons within the ASLSTM cell, the inputs to ASLSTM comprise three parts: long-term memory c^{t-1} , membrane potential v^{t-1} , and spikes s^{t-1} .

D. Training Details

The loss L is constituted by two distinct components: the angular velocity loss L_a and the linear velocity loss L_l :

$$\begin{aligned} L &= L_a + L_l \\ L_i &= \frac{1}{2} \sqrt{e_i^T e_i}, \quad i \in \{a, l\} \end{aligned} \quad (10)$$

where $e_a = \omega_{gt} - \omega$, $e_l = l_{gt} - l$, ω is the angular velocity and l linear velocity, respectively. Both angular and linear velocities are defined within the three-dimensional Euclidean space \mathbb{R}^3 .

Since there is a substantial numerical value disparity between angular and linear velocities. The gradients associated with them vary by several orders of magnitude throughout the training process. Consequently, we implemented a dynamic scaling of the loss functions for angular and linear velocities to mitigate this discrepancy. The dynamic scaling balanced the influence of angular and linear velocities within the loss function.

As shown in Fig. 2, our training process is multi-step. Firstly, the event data is partitioned into n chronological bins. These events are assigned to different channels according to their polarity. Secondly, the polarity-divided events are subsequently converted into a time series spike tensor $[T, B, C \times n, H, W]$. Finally, these time-series spikes are fed into the SNN module to estimate the velocities $[B, n, 6]$, where the angular velocities are represented by Euler angles.

In the computational process, we sequentially allow each layer to complete its computations at the time step. The resultant information is passed on to the subsequent layer only after each layer has fully processed the data at a given time step. The SNNs are trained by the Adam optimizer within the public PyTorch framework [26].

IV. EXPERIMENTS

In the experimental section, we designed three experiments to evaluate our NeuroVE framework. Initially, we evaluated the performance of the ALIF neurons and the ASLSTM in nonlinear regression and time-series forecasting experiments. Subsequently, we evaluated the performance of NeuroVE in the linear-angular velocity estimation experiments. Finally, real-world experiments confirmed the effectiveness of NeuroVE.

A. Experimental Configuration

In this section, we detail the experimental configurations for the three experiments and the metrics employed to evaluate the results.

1) *Nonlinear Regression and Time-Series Forecasting Experiments*: The *nonlinear regression and time-series forecasting experiments* aim to assess the ASLSTM's ability to perform numerical precision and time-series forecasting. The existing studies on SNNs for nonlinear numerical regression experiments frequently employed simple configurations. In addition, the proficiency of SNNs in time-series prediction is inferior to ANNs. To this end, we have designed an experiment to fit the $y = \sin(x)$ curve, encompassing numerical regression and predictive components.

We randomly generated 100 instances of $\sin(x + d_i)$, each incorporating phase offsets d_i , allocating 97 sequences for training and reserving 3 sequences for validation. Furthermore, the whole training dataset is comprised of 1000 time steps. The experiment fits the data across the initial 0 to 1000 time steps and forecasts the outcomes for the subsequent 1000 to 2000 time steps.

Furthermore, we undertook a comparative analysis of the firing frequencies across various neuron types when subjected to identical random signals to demonstrate the dense firing frequency feature of ALIF neurons.

2) *Linear-Angular Velocity Estimation Experiments*: The *linear-angular velocity estimation experiments* is specifically designed to evaluate the framework's performance in accurately determining velocities from event camera data. This evaluation

was performed on two distinct datasets: the MVSEC dataset [27] and a synthetic dataset generated by the CARLA simulator [28], which provides a controlled environment for testing. For the training phase, we used the *out_door_day2* dataset and a subset of the *out_door_day1* dataset. Following the training phase, we evaluated the model's performance in an unused portion of the *out_door_day1* dataset.

3) *Real-World Experiments*: In real-world experiments, our objective is to confirm the efficacy of our proposed NeuroVE framework on the robot. To this end, we affixed an event camera to a four-wheeled robot and tracked its movement along a corridor within a room (see Fig. 7). We recorded a total of four distinct trajectories during the experiments to demonstrate the performance of NeuroVE. The outputs from the multi-sensor fusion algorithm were used as the ground truth for velocity estimation. To guarantee the precision of our evaluation, we systematically sampled a subset of each trajectory for a validation set, deliberately excluded from the training phase.

4) *Metrics*: In order to comprehensively evaluate the performance of the proposed NeuroVE framework, we employ different metrics to quantify the performance in the experiments. We have selected the Root Mean Square Error (RMSE) and Relative Error (RE) as the primary metrics for evaluating model performance. Additionally, we introduced the Continuous Ranked Probability Score (CRPS) [29] as a supplementary metric to provide a more comprehensive evaluation of model performance in time-series forecasting tasks. The formulas are shown as follows,

$$RMSE(x) = \sqrt{\frac{1}{n} \sum_i^n \|x_{gt}^i - x^i\|^2},$$

$$RE(x) = \frac{1}{n} \sum_i^n \frac{\|x_{gt}^i - x^i\|}{x_{gt}^i}, \quad (11)$$

where x_{gt}^i is the i th ground truth, x^i is the i th prediction value.

B. Results of Nonlinear Regression and Time-Series Forecasting Experiments

In the experiments, we evaluate our method and compare its performance with existing pipelines listed as follows:

- **ASLSTM (Ours)**: The method proposed in Section III-C, and the network streamlined architectural design, comprising two layers of ASLSTM cells.
- **ASLSTM-w/o**: The proposed networks utilize LIF neurons [7] in place of ALIF neurons, thereby excluding the astrocyte-inspired diffusion mechanism.
- **LTC [30]**: The SNNs incorporate liquid time-constant spiking neurons designed to address the challenges associated with long-term time series prediction.
- **SLSTM [16]**: The SLSTM proposed for regression using spiking neural networks.

We begin by discussing the numerical experiment part, depicting the 0-1000 time steps in Fig. 5. The numerical experiment part shows the smoothest curve of our results. In contrast to other methods, they exhibit fluctuations due to SNN numerical instability. Table I further illustrates the significant improvement of our method, with an $RMSE^\dagger$ of approximately 0.3, markedly better than the 0.8 of SLSTM and 3 of LTC. Since the $\sin(x)$ oscillates between an amplitude range of -1 to 1, the resulting

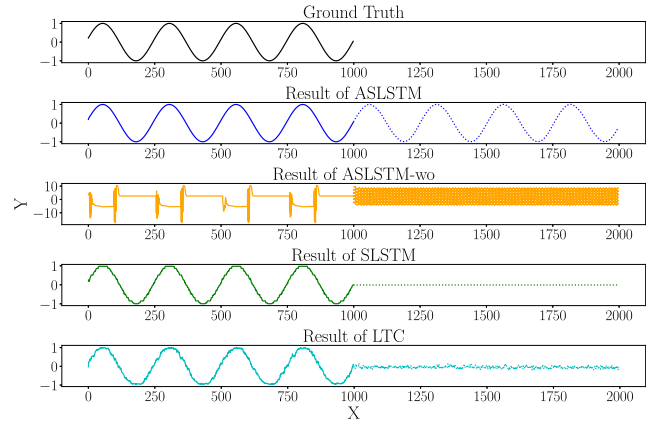


Fig. 5. The graph of the results of numeric regression and time-series forecasting for $y = \sin(x)$.

TABLE I
RMSE[†] FOR NUMERICAL REGRESSION TASKS

Sequence	SLSTM	LTC	ASLSTM-w/o	ASLSTM(Ours)
1	0.86	3.16	failed	0.31
2	0.84	3.67	failed	0.30
3	0.87	3.92	failed	0.31

[†] $RMSE^\dagger = RMSE \times 10^3$.

TABLE II
RMSE FOR TIME-SERIES FORECASTING TASKS

Sequence	SLSTM	LTC	ASLSTM-w/o	ASLSTM(Ours)
1	0.50	0.53	failed	0.02
2	0.88	0.86	failed	0.07
3	0.95	1.04	failed	0.11

TABLE III
CRPS FOR TIME-SERIES FORECASTING TASKS

Sequence	SLSTM	LTC	ASLSTM-w/o	ASLSTM(Ours)
1	0.64	0.64	failed	0.30
2	0.80	0.78	failed	0.41
3	0.74	0.77	failed	0.37

RMSE is inherently small. To facilitate a clear comparison, we introduced $RMSE^\dagger$, which magnified 1000 times from RMSE.

In the time-series forecasting part, depicted in the 1000-2000 time steps of Fig. 5. The experimental results indicate that most alternative methods struggle with time-series forecasting during 1000-2000 time steps. Table II quantitatively confirms our method's superior performance in time-series forecasting. To further evaluate the performance of our model in time series forecasting tasks, we introduced the CRPS as a performance metric. These evaluation results are presented in Table III. It is clear that our proposed method significantly outperforms other comparative methods in time-series forecasting tasks. Fig. 5 exposes other methods' deficiencies in time-series forecasting capabilities. Fig. 6 indicates that our method improves SNN's numerical stability through increased firing frequency.

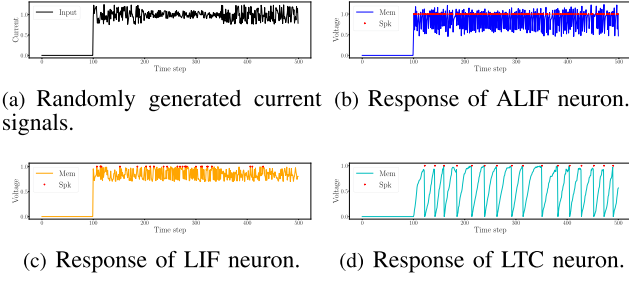


Fig. 6. Results of neuronal firing frequency.

TABLE IV
EVALUATION ON MVSEC DATASET

Sequence	Error	SNN-ANG		NVE-w/o		NVE(Ours)	
-	-	v_a	v_l	v_a	v_l	v_a	v_l
out_door_day1	RMSE*	0.91	0.7	0.64	0.3	0.11	0.12
	RE	1.63	0.45	0.89	0.22	0.15	0.10

¹ RMSE*: RMSE of angular velocity (v_a) magnified by a factor of 100.

² The unit of linear velocity (v_l) is m/s and angular velocity (v_a) is deg/s .

TABLE V
EVALUATION ON SYNTHETIC DATASET

Sequence	Error	SNN-ANG		NVE-w/o		NVE(Ours)	
-	-	v_a	v_l	v_a	v_l	v_a	v_l
Seq 1	RMSE*	5.77	0.75	5.37	0.30	3.56	0.28
	RE	1.84	0.91	1.02	0.34	0.73	0.32
Seq 2	RMSE*	4.41	0.99	3.82	0.38	3.37	0.35
	RE	1.92	0.92	1.14	0.38	1.01	0.31

C. Results of Linear-Angular Velocity Estimating Experiments

In this section, our primary objective is to validate the efficacy of brain-inspired methodologies in the estimation of velocities and angular velocities from event cameras. We have selected the MVSEC dataset, which offers accurate ground truth of trajectory. By differentiating these poses, we obtain the ground truth of linear and angular velocities, which provide a benchmark for model evaluation. Our methods and baseline methods are listed as follows:

- NVE (Ours): NeuroVE is proposed in Section III.
- NVE-w/o: The method employs LIF neurons [7] instead of ALIF neurons.
- SNN-ANG [11]: The approach uses SNN to implement angular velocity regression.

During the training phase, we do not utilize all the trajectories for model training. Instead, we allocate a portion for training and reserving another for the validation and testing phases. Ultimately, we employ the RMSE and RE as the primary evaluation metric.

The experimental results are presented in Tables IV and V. Our method shows better performance in estimating linear and angular velocities, outperforming other advanced approaches. In Table IV, our method demonstrates superior performance over other approaches, achieving an RMSE* of 0.11 for angular

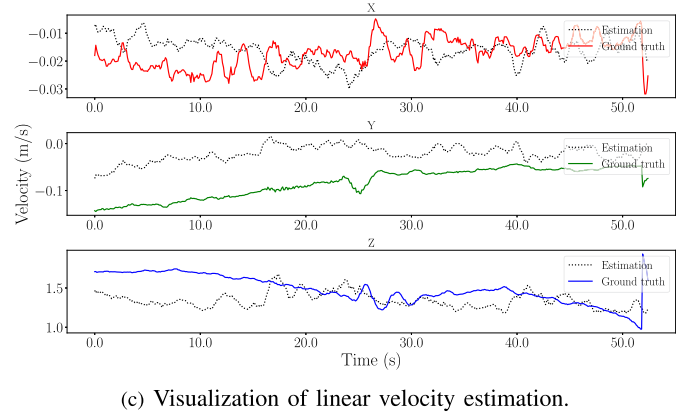
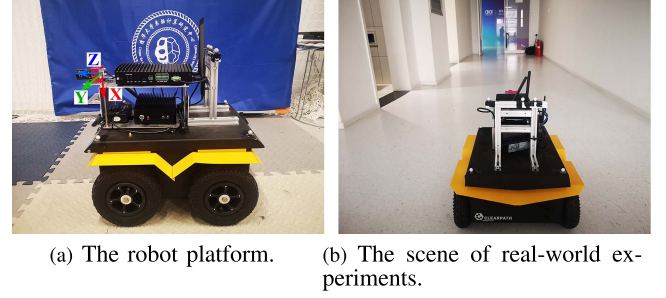


Fig. 7. Illustration of the robot platform, scene, and results of linear velocity estimation in the real-world experiments.

velocity and 0.12 for linear velocity. The RMSEs of angular and linear velocities are not balanced. Most of the samples in the dataset exhibit linear motion, resulting in very small values of mean angular velocity. The RMSE* represents the RMSE of angular velocity by a factor of 100.

In addition, we conducted numerous randomized experiments within the MVSEC dataset, which yielded results indicating that NeuroVE realizes a performance improvement of approximately 60% in velocity estimation.

D. Results of Real-World Experiments

Fig. 7(c) illustrates the estimation of linear velocity. The coordinate system is built on the camera frame, with the Z-axis aligned directly in front of the camera, the X-axis pointing to the ground, and the Y-axis established according to the right-hand rule. The experiment's evaluation phase yielded an RMSE of 0.08 and RE of 0.99 for angular velocities, whereas the linear velocities demonstrated an RMSE of 0.21 and RE of 0.23. This result exhibits the effectiveness of NeuroVE on real robots. Furthermore, the vehicle primarily moves in a straight line, rarely requiring steering, which results in angular velocities that are typically much closer to zero. Consequently, while the RMSE for angular velocity is smaller when compared to linear velocity, the RE is notably higher for angular velocity. Furthermore, the average power consumption for the NeuroVE model running on neuromorphic chips (Lynxi-HS110) was 1.3 W, while on the Xavier NX it was 3.6 W. NeuroVE's power consumption on the Xavier NX is nearly 2.77 times higher than on neuromorphic chips.

V. CONCLUSION

This letter introduces NeuroVE, a brain-inspired framework designed to estimate linear and angular velocities by mimicking the vision motion circuit. It captures the motion data from an event camera and employs mechanisms that mimic the function of LS and AS cells to process the information effectively. Firstly, we encode temporal information into spikes to simulate time cells in the initial phase. Subsequently, we introduce ALIF neurons to improve the representation precision of SNNs. Finally, we introduce the ASLSTM structure incorporated with ALIF neurons to enhance the accuracy of time-series forecasting. Our method has demonstrated its superior velocity estimation accuracy through synthetic datasets and real-world robot experiments. Moreover, numerical experiments have substantiated the advantages of NeuroVE in addressing numeric and time-series forecasting issues in SNNs. The NeuroVE framework provides a novel solution to the essential challenge of ego-velocity estimation, unleashing the potential of neuromorphic computing to address self-motion estimation problems.

REFERENCES

- [1] D. Spalla, A. Treves, and C. N. Boccara, "Angular and linear speed cells in the parahippocampal circuits," *Nature Commun.*, vol. 13, no. 1, Apr. 2022, Art. no. 1907.
- [2] H. Eichenbaum, "Time cells in the hippocampus: A new dimension for mapping memories," *Nature Rev. Neurosci.*, vol. 15, no. 11, pp. 732–744, Nov. 2014.
- [3] Z. Yang et al., "A vision chip with complementary pathways for open-world sensing," *Nature*, vol. 629, no. 8014, pp. 1027–1033, May 2024.
- [4] G. Gallego et al., "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2022.
- [5] Y. Guo, X. Huang, and Z. Ma, "Direct learning-based deep spiking neural networks: A review," *Front. Neurosci.*, vol. 17, 2023, Art. no. 1209795.
- [6] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019.
- [7] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Front. Neurosci.*, vol. 12, 2018, Art. no. 331.
- [8] L. Giantomasi et al., "Astrocytes actively support long-range molecular clock synchronization of segregated neuronal populations," *Sci. Rep.*, vol. 13, no. 1, Mar. 2023, Art. no. 4815.
- [9] N. Iannella and A. D. Back, "A spiking neural network architecture for nonlinear function approximation," *Neural Netw.*, vol. 14, no. 6–7, pp. 933–939, 2001.
- [10] C. Lv, Y. Wang, D. Han, X. Zheng, X. Huang, and D. Li, "Efficient and effective time-series forecasting with spiking neural networks," in *Proc. 41st Int. Conf. Mach. Learn.*, 2024, pp. 33624–33637.
- [11] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza, "Event-based angular velocity regression with spiking networks," in *Proc. 2020 IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4195–4202.
- [12] J. Cuadrado, U. Rançon, B. R. Cottureau, F. Barranco, and T. Masquelier, "Optical flow estimation from event-based cameras and spiking neural networks," *Front. Neurosci.*, vol. 17, 2023, Art. no. 1160034.
- [13] H. Kamata, Y. Mukuta, and T. Harada, "Fully spiking variational autoencoder," in *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 6, pp. 7059–7067.
- [14] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for energy-efficient object detection," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 7, pp. 11270–11277.
- [15] J. K. Eshraghian, X. Wang, and W. D. Lu, "Memristor-based binarized spiking neural networks: Challenges and applications," *IEEE Nanotechnol. Mag.*, vol. 16, no. 2, pp. 14–23, Apr. 2022.
- [16] A. Henkes, J. K. Eshraghian, and H. Wessels, "Spiking neural networks for nonlinear regression," *Roy. Soc. Open Sci.*, vol. 11, no. 5, 2024, Art. no. 231606.
- [17] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1425–1440, Dec. 2018.
- [18] S. Bryner, G. Gallego, H. Rebecq, and D. Scaramuzza, "Event-based, direct camera tracking from a photometric 3D map using nonlinear optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 325–331.
- [19] G. Gallego, H. Rebecq, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 3867–3876.
- [20] X.-Z. Peng, W. Xu, J. Yang, and L. Kneip, "Continuous event-line constraint for closed-form velocity initialization," in *Proc. Brit. Mach. Vis. Conf.*, 2021.
- [21] X. Li, Y. Zhou, R. Guo, X. Peng, Z. Zhou, and H. Lu, "Spatio-temporal calibration for omni-directional vehicle-mounted event cameras," *IEEE Robot. Automat. Lett.*, vol. 9, no. 3, pp. 2311–2318, Mar. 2024.
- [22] X. Lu, Y. Zhou, J. Niu, S. Zhong, and S. Shen, "Event-based visual inertial velometer," in *Proc. Robotics: Sci. Syst.*, 2024.
- [23] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 12, pp. 11062–11070.
- [24] D. Wheatley, "Diffusion theory, the cell and the synapse," *Biosystems*, vol. 45, no. 2, pp. 151–163, 1998.
- [25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [26] A. Paszke et al., "Automatic differentiation in PyTorch," in *Proc. Neural Inf. Process. Syst. Workshops*, 2017.
- [27] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2032–2039, Jul. 2018.
- [28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot. Learn.*, 2017, pp. 1–16.
- [29] R. Pic, C. Dombry, P. Naveau, and M. Taillardat, "Distributional regression and its evaluation with the CRPS: Bounds and convergence of the minimax risk," *Int. J. Forecasting*, vol. 39, no. 4, pp. 1564–1572, 2023.
- [30] B. Yin, F. Corradi, and S. M. Bohte, "Accurate online training of dynamical spiking neural networks through forward propagation through time," *Nature Mach. Intell.*, vol. 5, no. 5, pp. 518–527, May 2023.