2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Spike-IMU: An Accurate and Low-power Spiking Neural Network for Pedestrian Velocity Estimation

Junye Zou[1], Xiaolei Li[2], Ziyang Meng[1*], and Guoqi Li[3]

***Abstract*— Accurate pedestrian navigation on edge devices is a critical problem. While artificial neural networks (ANNs) have been shown to effectively solve this problem with acceptable accuracy, their energy consumption limits applications on low-power computation platforms. Spiking neural networks (SNNs) are promising alternatives, while their applicability in using noisy, high-frequency IMU data is hindered by two key issues: information loss during spike encoding and simplistic neuron dynamics that fail to capture complex motion. This paper introduces Spike-IMU, an SNN-based velocity estimation network designed to overcome these issues for the pedestrian navigation problem. In particular, a dynamic spiking neuron (DSN) is introduced based on the integer firing mechanism. In addition, a temporal feature fusion spike encoder (TFFSE) and a dynamic spiking long short-term memory network (DS-LSTM) are proposed to encode and process IMU data into spike sequences. Our experiments on the RoNIN dataset show that Spike-IMU surpasses classical ANNs, reducing positioning error by 20% while consuming 70.3% less energy. This work demonstrates a novel pipeline to design SNNs that achieves both superior accuracy and energy efficiency, pushing applications of IMU-based pedestrian navigation to real-world low-power edge devices.**

I. INTRODUCTION

Accurate pedestrian navigation is essential for a wide range of applications, including indoor navigation, location-based services, and emergency response [1]. In recent years, with the development of micro-electro-mechanical systems (MEMS) technology, low-cost and miniaturized IMUs have become increasingly common, providing hardware support for pedestrian navigation using personal devices, including smartphones and wearables. However, inherent measurement errors in IMUs accumulate over time, leading to a rapid decrease in localization accuracy during long-term use. This limits the practical applications of the IMU-based pedestrian navigation problem.

To improve localization accuracy, early approaches relied on restrictive heuristics such as zero-velocity updates (ZUPT) [2] that are often invalid for unconstrained pedestrian motion [3], or multi-sensor fusion systems that increase hardware cost and deployment complexity [4], [5], [6]. More recently, artificial neural network (ANN)-based methods have demonstrated superior performance by learning to estimate velocity directly from raw IMU data, effectively capturing complex motion patterns [7], [8], [9]. These models generally outperform traditional pedestrian dead reckoning (PDR) methods [10]. However, the high computational and energy demands of these ANN models make them unsuitable to deploy on the resource-constrained edge devices.

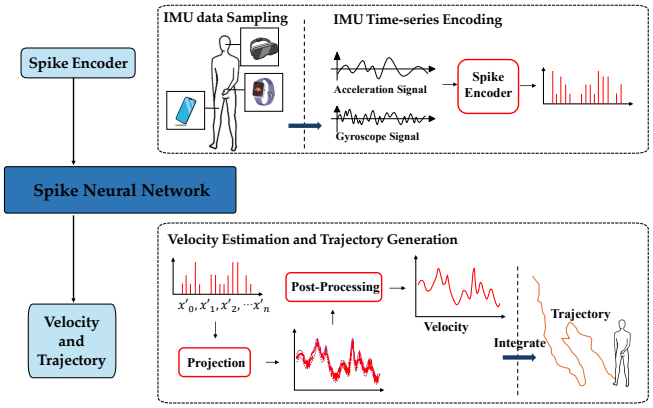

Fig. 1. Spike-IMU framework: from IMU data preprocessing to spiking neural network inference.

Spiking neural networks (SNNs) have emerged as a promising energy-efficient alternative, as they employ sparse, event-driven computation that mimics biological neural processing [11]. Despite their potential, SNNs face fundamental challenges when applied to high-precision regression tasks with noisy, high-frequency signals. Conventional SNNs, with their simple neuron dynamics and binary spike representations, often lack the information fidelity required to accurately model the subtle, high-frequency dynamics of human motion, limiting their applications in high-precision estimation tasks like PDR.

This paper fills this gap by proposing Spike-IMU, an SNN framework engineered to maximize information fidelity for IMU-based velocity estimation while retaining computational efficiency. We treat the task from a signal processing perspective, i.e., to transform the continuous IMU signal into a velocity estimation via a spike-based intermediate representation with minimal information loss. The contributions of this paper are as follows:

- A novel dynamic spiking neuron (DSN) model and the integer firing mechanism are introduced to enhance the representational bandwidth of the network. Such a

This work was supported in part by the Tsinghua-Toyota Joint Research Fund, in part by Beijing Natural Science Foundation (Grant No. L252095) and the Beijing Science and Technology Plan (Grant No. Z241100004224011), and in part by the National Natural Science Foundation of China (Grant Nos. 62273195, 62236009, and U22A20103).

[1]Junye Zou and Ziyang Meng are with the Department of Precision Instrument, Tsinghua University, Beijing 100084, China (e-mail: zoujy21@mails.tsinghua.edu.cn; ziyangmeng@tsinghua.edu.cn).

[2]Xiaolei Li is with the School of Instrument Science and Opto-electronic Engineering, Beijing Information Science and Technology University, Beijing 100192, China (email: 2023020213@bistu.edu.cn).

[3]Guoqi Li is with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: guoqi.li@ia.ac.cn).

*Corresponding author: Ziyang Meng.

design moves beyond coarse binary spikes to a finer-grained representation that better preserves the dynamic range of the IMU signal.
- In order to ensure the fidelity, we propose a temporal feature fusion spike encoder (TFFSE) to extract short-term and long-term robust temporal features, a dynamic spiking LSTM (DS-LSTM) to process the encoded spikes, and an adaptive post-processing filter to refine the final velocity estimations.
- To the best of our knowledge, this is the first study to demonstrate that a novel, fidelity-focused SNN architecture can surpass the accuracy of classical ANN models for IMU-based velocity estimation. Experiments show that the proposed approach reduces energy consumption by 70.3% compared with RoNIN-TCN [8], laying a robust foundation for future low-power inertial navigation applications.

II. RELATED WORKS

A. IMU-based Pedestrian Dead Reckoning

Traditional PDR methods rely on model-based algorithms that process IMU data through a sequence of discrete steps. These typically involve gait detection using peak detection [12] or zero-crossing detection [13], followed by step length estimation via biomechanical [14] or adaptive models [12]. The main drawback of traditional PDR approaches is that their performance degrades significantly during complex or irregular movements, as they depend on strong assumptions about pedestrian motion that are not always true in real-life scenarios [1].

With the rapid development of deep learning techniques, the solution of the PDR problem has gradually transferred to learning-based approaches. These approaches generally fall into two categories: estimating position directly [7], [15] or, more commonly, first estimating velocity and then integrating it over time. Seminal works like RoNIN [8] and CTIN [9] utilize deep networks like LSTM and transformer to map raw IMU data to velocity vectors, achieving superior accuracy and robustness. However, traditional deep learning algorithms require substantial computational resources, making them difficult to deploy on edge devices. Therefore, more efficient and energy-saving neural network models are expected to address this challenge.

B. Spiking Neural Networks for Time-Series Regression

Spiking neural networks (SNNs) have attracted widespread attention due to their ability to more accurately simulate the working mechanisms of biological nervous systems [11]. However, their applications to high-precision regression problems face the challenge of maintaining information fidelity. Converting continuous, high-frequency IMU signals into discrete spikes often results in significant information loss. This challenge causes two main bottlenecks in SNN design.

The first is the representational capacity of the spiking neuron model. The binary output of the standard leaky integrate-and-fire (LIF) neuron [16] is often too coarse to capture the fine-grained dynamics of motion data. To improve this, integer firing mechanisms have been proposed to reduce quantization error by allowing neurons to output integer-valued spikes, effectively increasing the communication bandwidth [17], [18]. Based on the integer firing mechanism, the dynamic parallel spiking neuron (DPSN) that adapts an input-dependent leakage rate instead of a fixed rate used in LIF has shown promising results on image classification [19]. Despite this progress, these advanced components have not yet been integrated into a cohesive framework designed for the specific demands of the inertial navigation problem.

The second bottleneck is temporal feature encoding, which dictates how the input signal is first converted to spikes. Early SNNs often used simple encoding schemes, such as repetition strategies [20], and therefore failed to preserve complex temporal dependencies. More advanced strategies have been developed, such as the delta encoder that captures local signal changes and the convolutional encoder that models sequence shapes [21]. However, IMU data consists of high-precision floating-point values and is highly sensitive to small variations. Existing methods that convert continuous time series into discrete spike sequences often suffer from information loss, which can significantly degrade model estimation performance. Therefore, developing a new encoding strategy that preserves temporal precision becomes particularly important.

III. METHODOLOGY

A. Overall Framework

In this paper, we aim to estimate pedestrian 2D linear velocity from a single IMU using SNNs. The proposed **Spike-IMU** framework, illustrated in Fig. 2, comprises three core modules:

- **Data preprocessing:** raw IMU data first goes through random angular augmentation (RAA) for robustness enhancement, followed by encoding the rotated IMU data into integer-valued spikes via our temporal feature fusion spike encoder (TFFSE).
- **Network inference:** a two-layer dynamic spiking LSTM (DS-LSTM) processes integer spike sequences. It combines long-term memory with the input-dependent leakage rate to model complex motion patterns.
- **Post-processing:** predictions are transformed back via inverse RAA. Then, to address the issue of prediction fluctuations commonly observed in SNNs, an adaptive multi-scale smoothing filter (AMSF) is applied to dynamically refine the output.

B. Integer Spike Firing Mechanism

To improve training stability and inference efficiency, we adopt an integer-valued spike activation scheme [17]. At layer l, the membrane potential U^l is mapped to an integer activation:

$$S_D^l = \mathrm{Fire}_D(U^l) = \left\lfloor \mathrm{clip}(U^l, 0, D) \right\rceil, \tag{1}$$

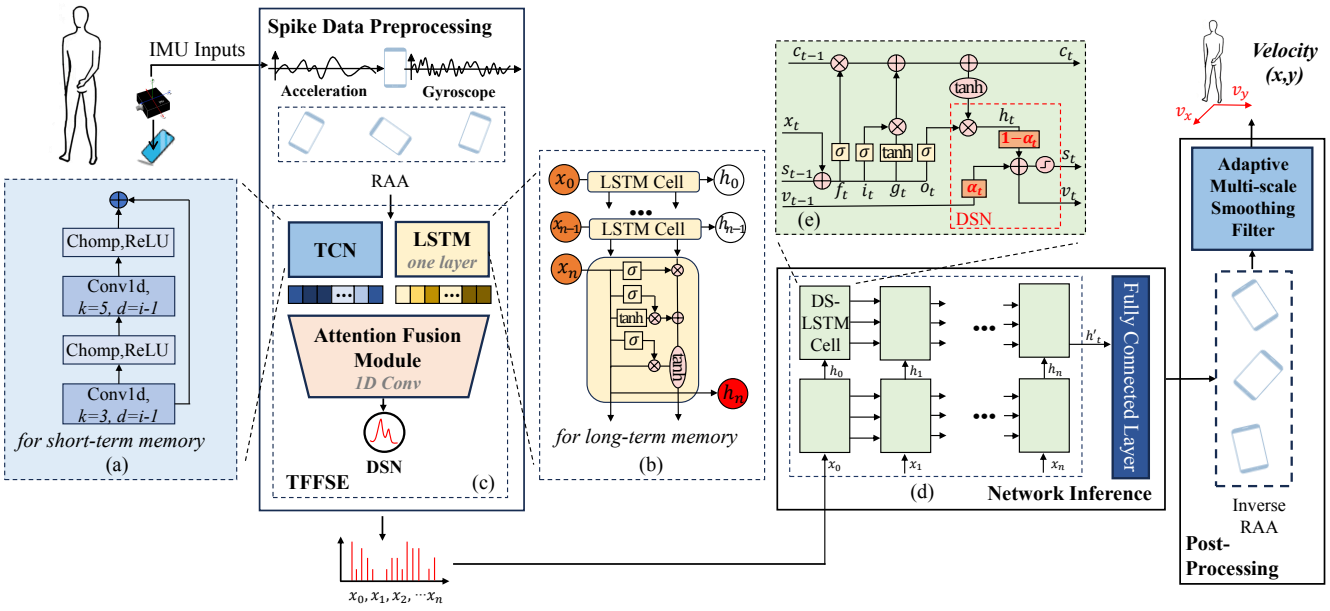

Fig. 2. Overall architecture of Spike-IMU, comprising data preprocessing, network inference, and post-processing modules: (a) TCN module, (b) LSTM network, (c) TFFSE module, (d) SNN-based DS-LSTM network, and (e) detailed structure of one cell of DS-LSTM network.

where D is a predefined maximum integer value, set to 4 in this paper. The clip$\{\cdot\}$ operation constrains the membrane potential within the range $[0, D]$, and $\lfloor\cdot\rceil$ denotes rounding to the nearest integer. This process can be interpreted as quantizing the continuous membrane potential into $D+1$ discrete levels, where each level represents varying degrees of importance. During inference, the integer spike is converted into binary spikes over D time steps. Specifically, the model generates a spike (1) for the first S_D^l steps, followed by non-spiking (0) for the remaining $D - S_D^l$ steps, which can be formulated as:

$$S_D^l = \sum_{d=1}^{D} \hat{S}^l[d], \tag{2}$$

where $\hat{S}^l[d]$ is 1 if $d \leq S_D^l$ and 0 otherwise.

Therefore, once a zero potential is detected, the computation for the remaining time steps can be skipped. This reduces computational cost while preserving the sparsity of SNNs.

C. Dynamic Spiking Neuron (DSN)

Standard LIF neurons use fixed leakage, limiting their adaptability to varying input dynamics. Inspired by DPSN model [19], we forgo its inherent parallelism for our sequential time-series data processing task, and propose the dynamic spiking neuron (DSN), which replaces fixed leakage with an input-dependent decay rate $\alpha_t \in [0, 1]$.

The membrane potential evolves as:

$$H_t = \alpha_t \odot H_{t-1} + (1 - \alpha_t) \odot X_t, \tag{3}$$

where H_t denotes the membrane potential at time t, X_t is the input, $\odot$ represents element-wise multiplication, and α_t is computed via:

$$\alpha_t = \sigma\left(W \cdot \text{CausalConv1D}(X_t)\right)^{1/\tau}, \tag{4}$$

where σ is sigmoid function, τ controls sharpness of its output distribution, and W fuses channel-wise features. Causal convolution offers a lightweight solution, enabling dynamic control of the leakage rate with minimal computational overhead. When α_t approaches 0, the system behaves like a hard reset, while values close to 1 correspond to a soft reset.

D. Random Angular Augmentation (RAA) Module

Leveraging the equivariance property of inertial navigation systems, rotating IMU data by an angle θ around the gravity-aligned axis obtains the same rotation of the motion state. This principle has been used to improve robustness in prior works [8], [15]. In this study, we exploit the property via a random angular augmentation (RAA) module applied during both training and testing.

Specifically, let an input sequence be denoted as $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{N\times 6}$, where N is the sequence length, $\mathbf{x}_i$ is the IMU data frame at i-th time step, which contains 3-axis accelerometer and 3-axis gyroscope measurements. For the entire sequence $\mathbf{X}$, three rotated sequences $\{\tilde{\mathbf{X}}_k\}_{k=1}^3$ are generated by applying three random rotations $\{\mathbf{R}(\theta_k) \in \mathbb{R}^6\}_{k=1}^3$, where each angle θ_k is drawn from a uniform distribution $\mathcal{U}(-\pi/2, \pi/2)$. The network processes each of these three sequences to produce three corresponding velocity prediction sequences, $\{\hat{\mathbf{V}}_k \in \mathbb{R}^{N\times 2}\}_{k=1}^3$, where $\hat{\mathbf{V}}_k = \{\hat{\mathbf{v}}_{k,1}, \hat{\mathbf{v}}_{k,2}, \ldots, \hat{\mathbf{v}}_{k,N}\}$. The final velocity prediction for each frame, $\hat{\mathbf{v}}_i$, is obtained by rotating back the corresponding velocity vectors using $\mathbf{R}^{-1}(\theta_k)$, and then averaging them as:

$$\hat{\mathbf{v}}_i = \frac{1}{3}\sum_{k=1}^{3} \mathbf{R}^{-1}(\theta_k)\hat{\mathbf{v}}_{k,i}. \tag{5}$$

E. Temporal Feature Fusion Spike Encoder (TFFSE)

The structure of the proposed temporal feature fusion spike encoder (TFFSE) is given in Fig. 2 (c). The raw continuous floating-point data are first fed into both the temporal convolutional network (TCN) and LSTM modules. TCN extracts local features within a short time window, and LSTM captures long-term temporal characteristics. The feature dimension is then expanded to 64. An attention fusion module is then introduced to integrate different features over short and long-time scales. This module consists of a lightweight single-layer 1D convolution, aiming to assign appropriate attention weights. The detailed structures of TCN module and LSTM network are depicted in Fig. 2 (a) and (b), respectively.

F. Dynamic Spiking Long Short-Term Memory (DS-LSTM) Network

LSTM networks are widely used for sequence modeling due to their ability to capture long-term dependencies. Inspired by the astrocyte-inspired ASLSTM proposed in [22], we propose a novel dynamic spiking long short-term memory (DS-LSTM) network. Our approach couples LSTM with DSN to estimate pedestrian linear velocity from IMU data.

As shown in Fig. 2 (e), the DS-LSTM cell maintains standard LSTM gates—input (i_t), forget (f_t), output (o_t), and cell gate (g_t)—with updates computed by:

$$\begin{cases} c_t = \sigma(f_t) \otimes c_{t-1} + \sigma(i_t) \otimes \tanh(g_t), \\ h_t = \sigma(o_t) \otimes \tanh(c_t), \\ v_t = \alpha_t v_{t-1} + (1-\alpha_t) h_t, \\ s_t = \lfloor \text{clip}(v_t, 0, D) \rceil, \end{cases} \tag{6}$$

where $i_t = W_i x_t + U_i s_{t-1}$ (similarly for f_t, g_t, o_t), W and U are learnable weight matrices, $\otimes$ denotes Hadamard product, c_t and h_t are the cell and hidden states, v_t is the membrane potential, and s_t is the quantized spiking output. Also note that the dynamic leakage rate α_t is computed via (4), enabling adaptive neuron dynamics.

Theoretically, the DS-LSTM guarantees Bounded-Input Bounded-Output (BIBO) stability. Since the LSTM output h_t is constrained by $\tanh(\cdot)$ and the dynamic decay α_t is within $(0, 1)$, the membrane potential update effectively forms a stable convex combination, ensuring the internal state dynamics remain strictly bounded.

G. Adaptive Multi-scale Smoothing Filter (AMSF)

Although the mean of the SNN's outputs over a time window is relatively close to the ground truth, the outputs exhibit significant instantaneous fluctuations (see blue line in Fig. 4 for example). While sliding window filtering can effectively suppress such high-frequency fluctuations, its performance is highly sensitive to the selection of window size. A larger filter window effectively smooths velocity estimations during steady-state walking. However, when the speed changes quickly, a smaller window is required to prevent over-smoothing. To balance this trade-off, we propose an adaptive multi-scale smoothing filter (AMSF) with dynamically adjusted window size:

$$w(t) = w_{\min} + \frac{w_{\max} - w_{\min}}{1 + \sum_{k=1}^{K} \left[\alpha_k \left(\frac{\sigma_k(t)}{\sigma_{0k}} \right)^p + \alpha_k \left(\frac{|\Delta\mu_k(t)|}{\mu_{0k}} \right)^q \right]}, \tag{7}$$

where $w_{\max}$ and $w_{\min}$ are selected to be 100, 50, respectively in this paper as the bounds of the smoothing windows, and $K = 3$ denotes the number of time scales: short ($s_1 = 50$), medium ($s_2 = 80$), and long ($s_3 = 100$), measured in time steps.

At each scale $k \in \{1, 2, 3\}$, for a given data sequence x, we compute three local statistics centered at time t: the local standard deviation $\sigma_k(t)$, the local mean $\mu_k(t)$, and the absolute change in local mean $\Delta\mu_k(t)$, defined as

$$\begin{cases} \sigma_k(t) = \text{Std}\left(x_{t-\lfloor s_k/2 \rceil : t+\lfloor s_k/2 \rceil} \right), \\ \mu_k(t) = \text{Mean}\left(x_{t-\lfloor s_k/2 \rceil : t+\lfloor s_k/2 \rceil} \right), \\ \Delta\mu_k(t) = |\mu_k(t) - \mu_k(t-1)|, \end{cases} \tag{8}$$

where $\text{Std}(\cdot)$ and $\text{Mean}(\cdot)$ denote the standard deviation and mean, respectively. Here, $\sigma_k(t)$ measures local fluctuation intensity, and $\Delta\mu_k(t)$ reflects the rate of trend change. In addition, the normalization baselines σ_{0k} and μ_{0k} are set to the median historical values. To enhance responsiveness, we set $p = 2$ and $q = 1$. The scale-dependent weights α_k follows an inverse-proportional rule:

$$\alpha_k = \frac{1/s_k}{\sum_{i=1}^{K} 1/s_i}. \tag{9}$$

The above design enables the proposed adaptive multi-scale smoothing filter to adaptively refine the smoothing window, suppressing noise during steady states while preserving dynamic information.

H. Loss Function and Training Strategy

We adopt a composite loss function that jointly optimizes velocity accuracy and cumulative drift:

$$\begin{aligned} \mathcal{L} &= \lambda \cdot \mathcal{L}_{\text{velocity}} + \mathcal{L}_{\text{cumulative}} \\ &= \lambda \cdot \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{v}}_i - \mathbf{v}_i\|_2^2 + \frac{1}{N} \sum_{i=1}^{N} \left\| \sum_{j=1}^{i} (\hat{\mathbf{v}}_j - \mathbf{v}_j) \right\|_2^2, \end{aligned} \tag{10}$$

where $\hat{\mathbf{v}}_i$ and $\mathbf{v}_i$ denote the predicted and ground-truth linear velocities at time step i, respectively, and λ is a weighting coefficient.

Importantly, the AMSF is excluded from the training pipeline in order to maximize the network's own prediction accuracy. Therefore, the loss is computed on raw SNN outputs before filtering. Despite this, as shown in the experiment section, applying AMSF during inference significantly improves velocity smoothness without degrading position accuracy.

We train and evaluate our model on the RoNIN dataset [8], which contains 42.7 hours of IMU data from 100 subjects performing diverse daily activities. The dataset is split into

training, validation, and test trajectories. Each trajectory is segmented into 400-step sequences for batch processing.

Model parameters are optimized using Adam with initial learning rates of 0.01 (encoder part) and 0.005 (inference network), dynamically adjusted based on loss and gradient. Training converges within 50 epochs.

IV. Experiments

In this section, a series of experiments are conducted to demonstrate the effectiveness of the proposed method. Firstly, we evaluate the fitting and prediction capacity of the proposed temporal feature fusion spike encoder. Secondly, to indicate the usefulness of the proposed adaptive multi-scale smoothing filter, we also evaluate its optimization performance of the velocity estimation. Thirdly, we present the test results of the proposed framework on the RoNIN dataset, and compare it with the classical ANN model. Finally, we analyze the computational efficiency of the proposed Spike-IMU.

A. Evaluation of TFFSE

Before evaluating the full Spike-IMU scheme, we first validate the hypothesis that the proposed TFFSE provides a superior temporal feature representation compared to simpler encoders. To do so, we design a challenging sequence fitting and prediction experiment using a composite periodic function, $y = sin(x) + sin(2x)$. Unlike the simple sin function used in prior works [23], [21], such a function blends both high and low frequency components, so it can be used to test an encoder's ability to extract and distinguish local features.

We generate 100 random instances of this function with phase shifts $d_i \sim \mathcal{U}(0, 2\pi)$: $y = \sin(x + d_i) + \sin(2(x + d_i))$. The dataset is split into 97 training and 3 validation sequences. Each sequence spans 1400 time steps, with the first 1,200 time steps used for fitting and the last 200 time steps for prediction. We use the root mean square error (RMSE) for quantitative evaluation.

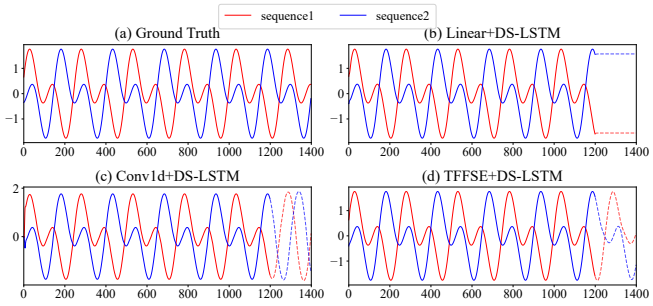

Fig. 3. Comparison of composite periodic function fitting (solid) and prediction (dashed) performance using three different encoders

TABLE I

Comparison of fitting and prediction performance on the composite periodic function using three different encoders

Sequences	Fitting			Prediction		
	TFFSE+ DS-LSTM	Linear+ DS-LSTM	Conv1d+ DS-LSTM	TFFSE+ DS-LSTM	Linear+ DS-SLSTM	Conv1d+ DS-LSTM
Sequence1	**0.0026**	0.0084	0.0447	**0.2124**	\	\
Sequence2	**0.0016**	0.0037	0.0218	**0.3899**	\	\
Sequence3	**0.0017**	0.0048	0.0127	**0.1593**	\	\
mean	**0.0020**	0.0056	0.0264	**0.2539**	\	\

Note: The best results are marked in **bold**. "\" indicates task failure.

We compare the proposed TFFSE+DS-LSTM against two baseline models: Linear+DS-LSTM and Conv1d+DS-LSTM. These baselines replace the proposed TFFSE with a standard linear fully connected layer and a 1D convolutional layer, respectively. The details of DS-LSTM architecture are described in Section III-F.

The results are presented in Fig. 3 and Table I. In the fitting task, TFFSE achieves the lowest RMSE across all sequences—reducing errors by 64.2% over the Linear encoder and 92.4% over the Conv1d encoder on average. More appealing, in the more challenging prediction task, TFFSE is the only method that successfully forecasts the correct results. Both baseline methods fail to generate meaningful predictions. For instance, the Conv1d encoder only learns major peaks/troughs but fails to capture the minor ones. These findings validate the effectiveness of the proposed encoder in complex sequence modeling tasks, particularly demonstrating its robustness in the prediction stage.

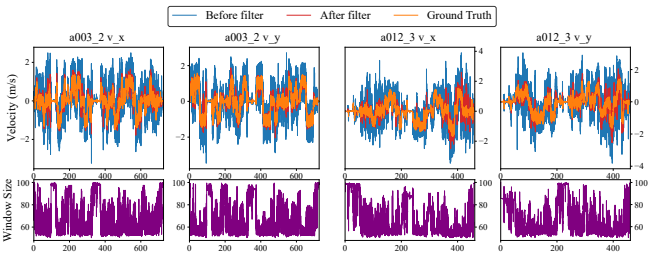

Fig. 4. Validation of AMSF filter performance and the analysis of adaptive window size, with the abscissa in seconds.

TABLE II

The Comparison of AVE Before and After Filter

Sequences	Before Filter			After Filter		
	v_x	v_y	v	v_x	v_y	v
a003_2	0.43	0.42	0.43	0.30	0.30	0.31
a012_3	0.52	0.52	0.051	0.40	0.39	0.38
mean	0.48	0.47	0.47	0.35	0.35	0.35

Note: Unit: m/s.

B. Evaluation of AMSF

This section validates the effectiveness of the proposed AMSF in refining the SNN's raw velocity predictions, which often exhibit large fluctuations.

As shown in Table II, applying the AMSF as a post-processing step leads to a substantial reduction in the absolute velocity error (AVE). For example, in sequence `a003_2`, the overall velocity error drops by approximately 30% (from 0.43 m/s to 0.31 m/s). Fig. 4 visually confirms this improvement, illustrating a smoother and more stable velocity trajectory after filtering.

More importantly, the results also demonstrate the filter's adaptive mechanism in action: the filter window dynamically narrows during rapid motion changes to preserve important details and widens during steady motion to maximize noise suppression. The filter's impact on the final localization

TABLE III

TRAJECTORY AND VELOCITY ERROR ON THE RONIN DATASET

sequence	length (m)	time (s)	RoNIN-TCN			Spike-IMU-Binary			Spike-IMU			Spike-IMU w/o AMSF			Spike-IMU w/o RAA		
			ATE	RTE-30s	AVE	ATE	RTE-30s	AVE	ATE	RTE-30s	AVE	ATE	RTE-30s	AVE	ATE	RTE-30s	AVE
a001_1	446.74	455.5	2.66	1.26	0.39	10.02	3.16	0.51	2.96	**1.16**	**0.3**	2.87	1.88	0.42	**2.33**	1.24	0.32
a003_1	350.64	387.2	1.17	1.46	0.4	4.57	3.31	0.34	**1.03**	**0.91**	**0.26**	1.05	1.37	0.4	6.14	1.64	0.29
a003_2	593.3	727.4	**1.26**	**0.89**	0.41	8.07	2.34	0.35	1.93	0.94	0.31	2.03	1.38	0.43	6.53	1.04	**0.30**
a007_2	199.17	612.3	3.94	**0.85**	0.28	3.85	2.03	0.25	**2.03**	0.96	**0.18**	**2.03**	1.4	0.25	2.77	1.1	0.21
a012_3	335.03	461.05	**2.19**	1.48	0.56	6.85	5.96	0.44	2.92	**1.36**	**0.38**	2.96	2.03	0.51	5.8	1.43	0.42
a013_3	402.74	570.6	**2.1**	**1.06**	0.51	6.01	2.75	0.38	2.25	1.41	**0.32**	2.3	2.18	0.46	8.93	1.6	0.33
a017_1	401.45	414.1	3.01	1.27	0.38	2.38	2.17	0.41	2.29	**1.11**	**0.29**	**2.05**	1.61	0.45	7.54	1.42	**0.29**
a023_3	386.51	590.4	**1.43**	0.8	0.29	7.46	2.89	0.29	1.91	0.81	0.18	1.91	1.34	0.33	2.52	**0.48**	**0.17**
a025_1	592.26	509.2	2.58	1.61	0.46	3.27	2.35	0.45	2.31	**1.51**	**0.33**	**2.29**	2.17	0.47	5.89	1.55	0.35
a026_3	987.96	831.3	8.05	1.59	0.37	9.28	2.82	0.35	3.37	**1.19**	**0.27**	**3.27**	1.89	0.5	11.39	1.29	**0.27**
mean	469.58	555.9	2.84	1.23	0.41	6.18	2.98	0.38	2.30	**1.14**	**0.28**	**2.28**	1.73	0.42	5.98	1.28	0.30

Note: The best results are marked in **bold**, and the second-best results are underlined. ATE/RTE values are in meters (m), and AVE is in meters per second (m/s).

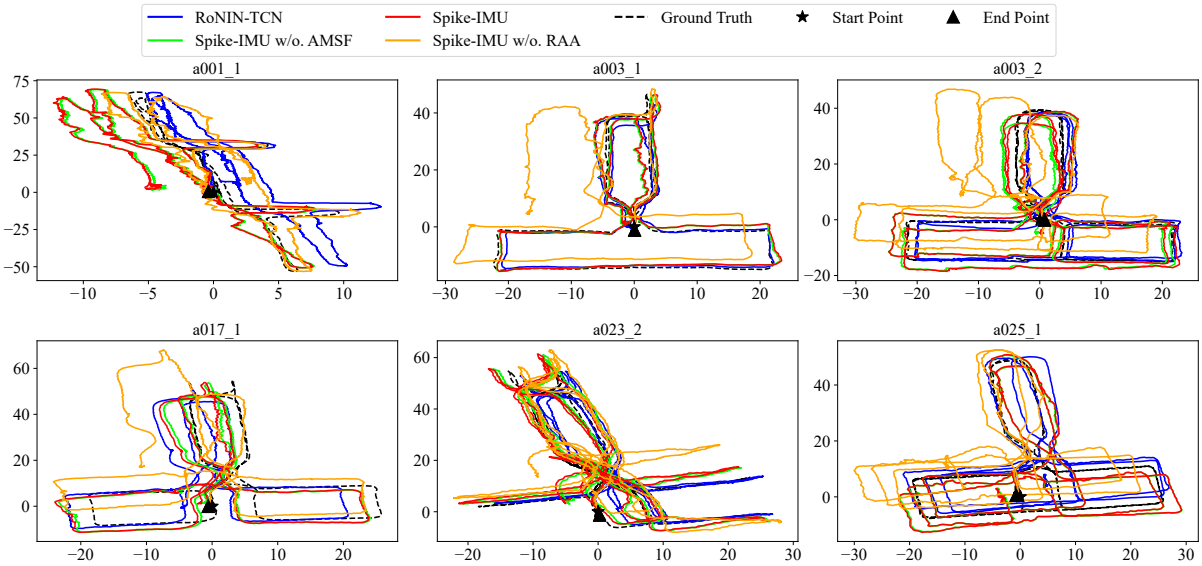

Fig. 5. Predicted trajectories on the RoNIN dataset. Comparison of localization accuracy between Spike-IMU and baseline models. (Unit: m)

accuracy will be comprehensively analyzed in the following section.

C. Evaluation of Spike-IMU

Having demonstrated the effectiveness of our key components of the proposed network, we now proceed to an overall evaluation of the entire Spike-IMU framework on the RoNIN dataset. To the best of our knowledge, this work is the first to apply SNNs to pedestrian linear velocity estimation problem using only IMU data. We compare the proposed Spike-IMU with the following models:

- **RoNIN-TCN** [8]: a classical ANN-based method that directly estimates velocity from IMU inputs, using a multi-scale TCN.
- **Spike-IMU w/o RAA**: ablation algorithm without the RAA module.
- **Spike-IMU w/o AMSF**: ablation algorithm without the AMSF post-processing.
- **Spike-IMU-Binary**: variant algorithm that utilizes the traditional LIF neuron. This neuron employs binary spiking instead of integer firing, with a fixed leakage rate of 0.8.

Performance is evaluated using three standard metrics: absolute trajectory error (ATE) for global accuracy, relative trajectory error (RTE-30s) for local drift over 30-second windows, and absolute velocity error (AVE).

The detailed quantitative results on 10 sequences from the RoNIN dataset are summarized in Table III, and representative trajectory visualizations are shown in Fig. 5. The full Spike-IMU model demonstrates competitive performance against the ANN-based model, RoNIN-TCN, achieving a lower mean ATE (2.30m vs. 2.84m) and RTE (1.14m vs. 1.23m), significantly outperforming it in AVE (0.28 m/s vs. 0.41 m/s).

The ablation study highlights the critical contributions of the proposed modules. Removing the RAA module (Spike-

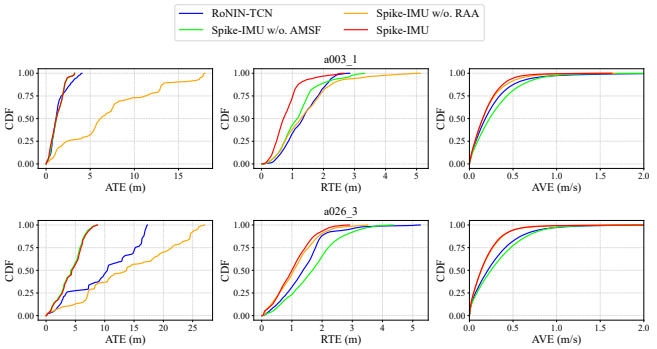

Fig. 6. Cumulative distribution functions of the prediction errors.

IMU w/o RAA) drastically degrades performance, with the mean ATE increasing from 2.30m to 5.98m. This verifies RAA's effectiveness in enhancing generalizability. Similarly, although the model without the filter (Spike-IMU w/o AMSF) achieves a competitive ATE, the full Spike-IMU model leverages AMSF to reduce the mean AVE by 33% and RTE by 34%, demonstrating the effectiveness of the filter in improving velocity smoothness and local trajectory accuracy. Furthermore, Spike-IMU-Binary shows much higher trajectory error than that of the proposed integer-firing model. This can be attributed to the binary firing mechanism, which encodes information solely using 0 and 1. Consequently, it fails to effectively quantify the differences between features, resulting in a limited capacity to represent complex temporal information.

To analyze error distributions, we plot the cumulative distribution functions (CDF) of ATE, RTE, and AVE in Fig. 6. The CDF can then be expressed as:

$$F_E(\varepsilon) = P(E \leq \varepsilon), \tag{11}$$

where $F_E(\varepsilon) \in [0, 1]$ represents the probability that the error variable E takes a value less than or equal to ε.

As shown in the Fig. 6, the CDF curve of Spike-IMU is the steepest and most concentrated across all evaluation metrics. From the CDF plot of the ATE, it is evident that RAA significantly improves the error distribution. Meanwhile, from the CDF plots of the RTE and AVE, the velocity optimization brought by AMSF also effectively improves the RTE.

D. Computational Efficiency Analysis

To validate the low-power property of the proposed Spike-IMU, we conduct a theoretical analysis on its computational efficiency. We follow the standard methodology in SNN literature [24], [25] by estimating energy consumption based on operation counts.

In theory, neural network operations can be categorized into two types: multiply-accumulate (MAC) and accumulation-only (AC). Their computational complexity is typically measured in floating point operations (FLOPs). The primary computational cost in ANNs comes from MAC operations. In contrast, SNNs leverage the sparsity of spike events to replace most MACs with more energy-efficient AC operations [26]. Following [24], we adopt the energy costs for a 45nm fabrication process, where a MAC operation

TABLE IV

COMPARISON OF OPERATION TYPES BETWEEN SNN AND ANN NETWORKS

Network Type	Layer Type	Mathematical Representation	Computation Formula	Operation Type
ANN	Convolution Layer	$FL_{conv}^{i\to o,k}$	$i \cdot o \cdot k$	MAC
	Downsampling Layer	$FL_{down}^{i\to o}$	$i \cdot o$	MAC
	Fully Connected Layer	$FL_{fc}^{i\to o}$	$i \cdot o$	MAC
	LSTM Layer	$FL_{LSTM}^{i\to h}$	$4 \times (i \cdot (i + h))$	MAC
SNN	Fully Connected Layer	$FL_{fc(SNN)}^{i\to o}$	$\phi \cdot i \cdot o \cdot k$	AC

TABLE V

COMPUTATIONAL COMPONENTS OF RONIN-TCN AND SPIKE-IMU

Model	Module	Components	Sum	
			MAC	AC
RoNIN -TCN	TCNblock (×6)	Conv1: $FL_{conv}^{i\to o,k}$, Conv2: $FL_{conv}^{o\to o,k}$, DS: $FL_{down}^{i\to o}$,	537,456	
	Decoder	$FL_conv^{i} \to o, k$	72	
Spike -IMU	Encoder	Conv1: $FL_{conv}^{i\to h_1,k_1}$, Conv2: $FL_{conv}^{h_1\to h_2,k_2}$, LSTM: $FL_{LSTM}^{i\to h_2}$ Attention: $FL_{conv}^{h_2\to h_2,1}$ CasualConv(CC): $FL_{conv}^{h_2\to h_2,1}$	36,928	
	DS-LSTM	input_lin: $4FL_{fc(SNN)}^{h_2\to h_3}$, hidden_lin: $4FL_{fc(SNN)}^{h_3\to o_1}$ CasualConv: $FL_{conv}^{o_1\to o_1,1}$	16,384	15,140
	Decoder	$FL_{conv(SNN)}^{o_1\to o,k}$		40
	AMSF	-	186	1077

Note: The total energy consumption corresponds to processing 10,000 data points, i.e., multiplying the total operations by 10,000. Due to the use of the RAA module, the energy consumption needs to be multiplied by 3 again in Spike-IMU.

TABLE VI

ENERGY CONSUMPTION COMPARISON BETWEEN RONIN-TCN AND SPIKE-IMU

Model	Energy Consumption (mJ)	FPS	CPU Core Usage (%)	Parameters Count	Weight File Size (MB)
RoNIN-TCN	26.4	174	396	3.11M	6.25
Spike-IMU	7.83	943	283	780K	3.00

consumes 4.6 pJ (E_{MAC}) and an AC operation consumes 0.9 pJ (E_{AC}). The energy consumption for ANNs and SNNs can be estimated as:

$$\text{E}_{\text{ANN}} = \text{FLOPs} \times E_{\text{MAC}}, \tag{12}$$

$$\text{E}_{\text{SNN}} = \text{FLOPs} \times E_{\text{AC}} \times \phi, \tag{13}$$

where ϕ represents the average spike firing rate.

Table IV systematically defines the typical operation types of ANNs and SNNs. In this table, FL denotes a forward layer, $i/o/h$ represents the input/output/hidden layer dimensions, k is the convolution kernel size, and ϕ is experimentally measured as 15.4%.

The operation breakdowns for the RoNIN-TCN baseline and the Spike-IMU are detailed in Table V, and the theoretical energy consumption is presented in Table VI. The analysis reveals that for processing 10,000 data points, RoNIN-TCN requires 26.4 mJ, whereas Spike-IMU consumes only 7.83 mJ—a 70.3% reduction in energy. This substantial saving is primarily attributed to the DS-LSTM module, whose sparse, AC-based computations drastically reduce the operational cost. More importantly, this efficiency is achieved without compromising the estimation accuracy.

Beyond theoretical analysis, we also evaluate the practical performance speed. The Spike-IMU and RoNIN-TCN models are deployed on a smartphone (Xiaomi 10S, MIUI 14.0.6, Snapdragon 870), using the `pytorch_android:1.13.1` framework. During the runtime, we close all other non-essential processes. The proposed Spike-IMU model achieves a processing speed of 943 FPS, 5.4 times faster than that of RoNIN-TCN baseline. Given that the input IMU data is sampled at 200 Hz, the performance of Spike-IMU comfortably exceeds the real-time processing requirement. This result verifies the superior computational speed of Spike-IMU and its capability to achieve real-time performance on resource-constrained edge devices.

V. Conclusions

This paper introduces Spike-IMU, an SNN framework designed for accurate and energy-efficient pedestrian velocity estimation problem using only IMU data. We address the critical challenges of information loss in signal-to-spike conversion and the simplistic dynamics of conventional spiking neurons. The proposed solution integrates a temporal feature fusion spike encoder (TFFSE) to ensure rich feature preservation, and a dynamic spiking LSTM (DS-LSTM) network built on the dynamic spiking neuron (DSN) with the integer firing mechanism to capture complex motion dynamics. Experimental results on the RoNIN dataset demonstrate that Spike-IMU not only realizes the energy-saving property of SNNs but also surpasses the accuracy of a classical ANN-based method, reducing positioning error by approximately 20% while consuming 70.3% less energy. Future work includes optimizing spike encoding mechanisms and testing the framework's robustness in more diverse and challenging scenarios.

References

[1] I. Klein, "Pedestrian inertial navigation: An overview of model and data-driven approaches," *Results in Engineering*, vol. 25, p. 104077, 2025.

[2] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara, "A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu," in *2009 IEEE International Symposium on Intelligent Signal Processing*. IEEE, 2009, pp. 37–42.

[3] A. Brajdic and R. Harle, "Walk detection and step counting on unconstrained smartphones," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013, pp. 225–234.

[4] D. Khan, Z. Cheng, H. Uchiyama, S. Ali, M. Asshad, and K. Kiyokawa, "Recent advances in vision-based indoor navigation: A systematic literature review," *Computers & Graphics*, vol. 104, pp. 24–45, 2022.

[5] C. Yang and H.-R. Shao, "Wifi-based indoor positioning," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 150–157, 2015.

[6] L. Bai, F. Ciravegna, R. Bond, and M. Mulvenna, "A low cost indoor positioning system using bluetooth low energy," *IEEE Access*, vol. 8, pp. 136 858–136 871, 2020.

[7] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 6468–6476.

[8] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, new methods," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3146–3152.

[9] B. Rao, E. Kazemi, Y. Ding, D. M. Shila, F. M. Tucker, and L. Wang, "Ctin: Robust contextual transformer network for inertial navigation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 5, 2022, pp. 5413–5421.

[10] N. M. R. M. Farhad Mortezapour Shiri, Thinagaran Perumal, "A comprehensive overview and comparative analysis on deep learning models," *Journal on Artificial Intelligence*, vol. 6, no. 1, pp. 301–360, 2024.

[11] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[12] L. Fang, P. J. Antsaklis, L. A. Montestruque, M. B. McMickell, M. Lemmon, Y. Sun, H. Fang, I. Koutroulis, M. Haenggi, M. Xie, *et al.*, "Design of a wireless assisted pedestrian dead reckoning system-the navmote experience," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 6, pp. 2342–2358, 2005.

[13] J. Park, Y. Kim, and J. Lee, "Waist mounted pedestrian dead-reckoning system," in *9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2012, pp. 335–336.

[14] H. Weinberg, "Using the adxl202 in pedometer and personal navigation applications," *Analog Devices AN-602 Application Note*, vol. 2, no. 2, pp. 1–6, 2002.

[15] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "Tlio: Tight learned inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5653–5660, 2020.

[16] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[17] M. Yao, X. Qiu, T. Hu, J. Hu, Y. Chou, K. Tian, J. Liao, L. Leng, B. Xu, and G. Li, "Scaling spike-driven transformer with efficient spike firing approximation training," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 4, pp. 2973–2990, 2025.

[18] X. Luo, M. Yao, Y. Chou, B. Xu, and G. Li, "Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection," in *European Conference on Computer Vision*. Springer, 2024, pp. 253–272.

[19] Y. Huang, M. Yao, Y. Pan, C. Lv, S. Xu, X. Zheng, B. Xu, and G. Li, "Parallel training in spiking neural networks," *arXiv preprint arXiv:2602.01133*, 2026.

[20] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 056–21 069, 2021.

[21] C. Lv, Y. Wang, D. Han, X. Zheng, X. Huang, and D. Li, "Efficient and effective time-series forecasting with spiking neural networks," *arXiv preprint arXiv:2402.01533*, 2024.

[22] X. Li, X. Chen, R. Guo, Y. Wu, Z. Zhou, F. Yu, and H. Lu, "Neurove: Brain-inspired linear-angular velocity estimation with spiking neural networks," *IEEE Robotics and Automation Letters*, vol. 10, no. 3, pp. 2375–2382, 2025.

[23] G. Li, L. Deng, H. Tang, G. Pan, Y. Tian, K. Roy, and W. Maass, "Brain-inspired computing: A systematic survey and future trends," *Proceedings of the IEEE*, vol. 112, no. 6, pp. 544–584, 2024.

[24] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-state Circuits Conference Digest (ISSCC)*. IEEE, 2014, pp. 10–14.

[25] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[26] X. Wu, W. He, M. Yao, Z. Zhang, Y. Wang, B. Xu, and G. Li, "Event-based depth prediction with deep spiking neural network," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 6, pp. 2008–2018, 2024.