Vision-guided Velocity Control of Micro Air Vehicles using Rate-decoded Spiking Neural Network

Yufei Peng, Ni Li, and Hann Woei Ho*
School of Aeronautics, Northwestern Polytechnical University,
and National Key Laboratory of Aircraft Configuration Design, 710072 xi'an, China

http://www.imavs.org/

ABSTRACT

Micro air vehicles (MAVs) operate under strict size, weight, power, and computational constraints, which make heavy deep neural controllers challenging for onboard deployment. Spiking neural networks (SNNs) and event-based vision have recently emerged as promising approaches for energy-efficient aerial autonomy. Existing studies mainly focus on low-level control or direct event-driven sensing, while the generation of high-level velocity commands from visual observations remains insufficiently explored. This paper proposes a rate-decoded SNN imitation-learning controller for vision-guided MAV velocity control. The proposed controller maps compact visual-state observations, including target offsets, apparent scale, and motion cues, to continuous three-axis translational velocity commands using leaky integrate-and-fire neurons and spike-rate decoding. The controller is trained through imitation learning from a privileged PID expert that has access to vehicle position and velocity feedback, whereas the SNN receives only visual observations. The training objective combines behavioral cloning, command-smoothness regularization, and spike-rate regularization. Experimental results in a visual target-approaching task show that the learned SNN reproduces the dominant target-following behavior of the expert controller, continuously reduces tracking error, and maintains stable spiking activity during closed-loop execution. The results demonstrate the feasibility of compact rate-decoded SNNs for high-level vision-guided MAV control.

1 INTRODUCTION

Micro Air Vehicles (MAVs) are increasingly expected to perform autonomous navigation and decision-making tasks under strict constraints in size, weight, power, and onboard computation. This makes efficient perception and control a central challenge for sustained autonomous flight. Recent studies have demonstrated that autonomous MAVs can perform highly dynamic tasks, such as deep-learning-based MAV racing and real-time visual navigation [2]. However, deploying computation-intensive neural controllers on resource-constrained MAV platforms remains challenging in practice. Therefore, perception and control algorithms for MAVs must satisfy stringent requirements in terms of computational overhead, memory footprint, inference latency, and energy consumption [3,4]. These constraints make the design of lightweight and energy-efficient neural control architectures a central challenge for enabling practical autonomous MAV flight.

To address the computational and energy constraints of MAV autonomy, spiking neural networks (SNNs) have emerged as a promising alternative to conventional Artificial Neural Networks (ANNs). SNNs are commonly regarded as the third generation of neural network models, where information is represented by temporally sparse spike events rather than continuous-valued activations [5]. By maintaining internal neuronal states, such as membrane potentials and synaptic currents, SNNs perform event-driven temporal computation and transmit information only when firing thresholds are reached. This sparse communication mechanism makes SNNs naturally compatible with event-based sensors, temporal sensory streams, and neuromorphic processors. Moreover, surrogate-gradient learning enables SNNs to be optimized with gradient-based methods while preserving discrete spike-based forward dynamics [6, 7]. Because SNNs rely on sparse event-driven computation, they can reduce computational overhead compared with dense activation-based neural models, which make them suitable for resource-constrained onboard MAV systems with limited latency, memory, and energy budgets.

Recent studies have demonstrated the potential of SNNs for MAV control. Fully neuromorphic vision-to-control systems have shown that raw event-camera data can be processed by SNNs to generate low-level control commands for autonomous MAV flight [8]. Neuromorphic attitude estimation and control has further shown that modular SNNs can map raw inertial measurements to motor-related control commands on an MAV platform [9]. In addition, fully spiking actor–critic control trained with proximal policy optimization has been applied to high-speed continuous MAV control [10]. These studies demonstrate the feasibility of SNN-based MAV autonomy, but existing methods mainly focus on low-level attitude control, event-camera-based control, or reinforcement-

*Email: hannwoei@nwpu.edu.cn

http://www.imavs.org/

learning-based agile flight. For MAV visual servoing, it remains necessary to investigate whether a compact SNN policy can learn high-level velocity commands from visual observations while maintaining stable closed-loop tracking behavior.

In this work, we propose an SNN-based imitation learning framework for high-level MAV velocity control. The proposed method focuses on designing a compact spiking neural controller that maps low-dimensional visual observations to translational velocity commands. The controller is implemented using leaky integrate-and-fire neurons, which allow temporal information to be represented through internal neuronal states and sparse spike activity. Given the visual observation vector, the SNN generates bounded velocity outputs along the x, y, and z axes for closed-loop target approaching. The SNN controller is trained through imitation learning from a privileged PID expert. The expert has access to vehicle position and velocity feedback and provides reference translational velocity commands, whereas the SNN learns to approximate the expert policy using only compact visual observations. A rate-decoding mechanism is adopted to convert spike activity into continuous control commands. To improve control accuracy and temporal stability, the training objective combines behavioral cloning loss, command-jerk regularization, and spike-rate regularization. This design encourages the SNN to produce accurate, smooth, and biologically plausible spiking control outputs. Experimental results demonstrate that the trained SNN can reproduce the main target-approaching behavior and achieve stable closed-loop visual tracking.

The main contributions of this work are summarized as follows:

- A rate-decoded leaky integrate-and-fire SNN controller is proposed for high-level MAV velocity control, which enables continuous three-axis translational velocity generation from compact visual observations.
- An imitation-learning framework is introduced that combines expert-command supervision, temporal command-smoothness regularization, and spike-rate regularization to improve tracking accuracy, control smoothness, and spiking stability.
- Closed-loop simulation experiments demonstrate that the proposed controller reproduces the dominant target-approaching behavior of a physical-feedback expert while maintaining stable spiking dynamics during visual servoing.

2 METHODOLOGY

2.1 Pipeline Overview

Fig. 1 illustrates the overview of the proposed closed-loop SNN imitation-learning pipeline for vision-guided MAV velocity control. The pipeline consists of four main components: visual observation, expert demonstration and imitation training, the rate-decoded SNN controller, and the control output applied to the MAV. The simulation environment provides vehicle odometry and visual target observations for closed-loop interaction. During training, a conventional expert controller computes reference translational velocity commands from the physical tracking error, while the SNN learns to predict corresponding velocity commands from compact visual observations. In the current training configuration, the expert command is used to drive the vehicle, and the SNN is optimized in parallel through supervised imitation. During evaluation, the trained SNN policy is deployed as the controller and directly sends its predicted velocity commands to the vehicle velocity-control interface.

The task is formulated as vision-guided target approaching for an MAV. The vehicle is required to approach a visual reference target and maintain stable tracking behavior through translational velocity control. Instead of using high-dimensional image inputs, the controller receives a compact observation vector obtained from the visual target feature stream:

$$\mathbf{o}_t = [q_t, d_{x,t}, d_{y,t}, s_t, g^e_{x,t}, g^e_{y,t}]^\top. \tag{1}$$

Here, q_t denotes the marker-identifier channel provided by the visual feature message, $d_{x,t}$ and $d_{y,t}$ denote the normalized horizontal and vertical image-plane offsets of the visual target, and s_t represents the normalized apparent target scale. The terms $g^e_{x,t}$ and $g^e_{y,t}$ denote the horizontal and vertical components of the time-surface motion vector. The resulting six-dimensional feature vector is directly used as the SNN observation. In the implementation, the horizontal image coordinate is inverted to match the vehicle lateral-control direction. The image-plane offsets, apparent target scale, and motion-vector components are normalized and clipped to bounded ranges to improve numerical stability and reduce the influence of outlier visual measurements during training.

The SNN policy maps the observation vector $\mathbf{o}_t$ to a bounded translational velocity command:

$$\mathbf{v}^{\mathrm{cmd}}_t = [v^{\mathrm{cmd}}_{x,t}, v^{\mathrm{cmd}}_{y,t}, v^{\mathrm{cmd}}_{z,t}]^\top. \tag{2}$$

Throughout this paper, $\mathbf{v}^{\mathrm{cmd}}_t$ denotes the velocity command predicted by the SNN, and $\mathbf{v}^{\mathrm{E}}_t$ denotes the reference velocity command generated by the expert controller. The yaw-rate command is fixed to zero in the current setting; therefore, the learning problem focuses on three-axis translational velocity control. The SNN controller consists of a continuous feature encoder, a leaky integrate-and-fire spiking layer, and a linear readout layer. Spike activity is accumulated over multiple internal simulation steps and converted into continuous control outputs through rate decoding. The decoded command is further bounded by a hyperbolic tangent function and axis-wise velocity limits of $[0.35, 0.35, 0.25]^\top$ m/s for the x, y, and z axes, respectively, to match the expert command range.

The privileged PID expert generates reference velocity commands using vehicle position and velocity feedback,

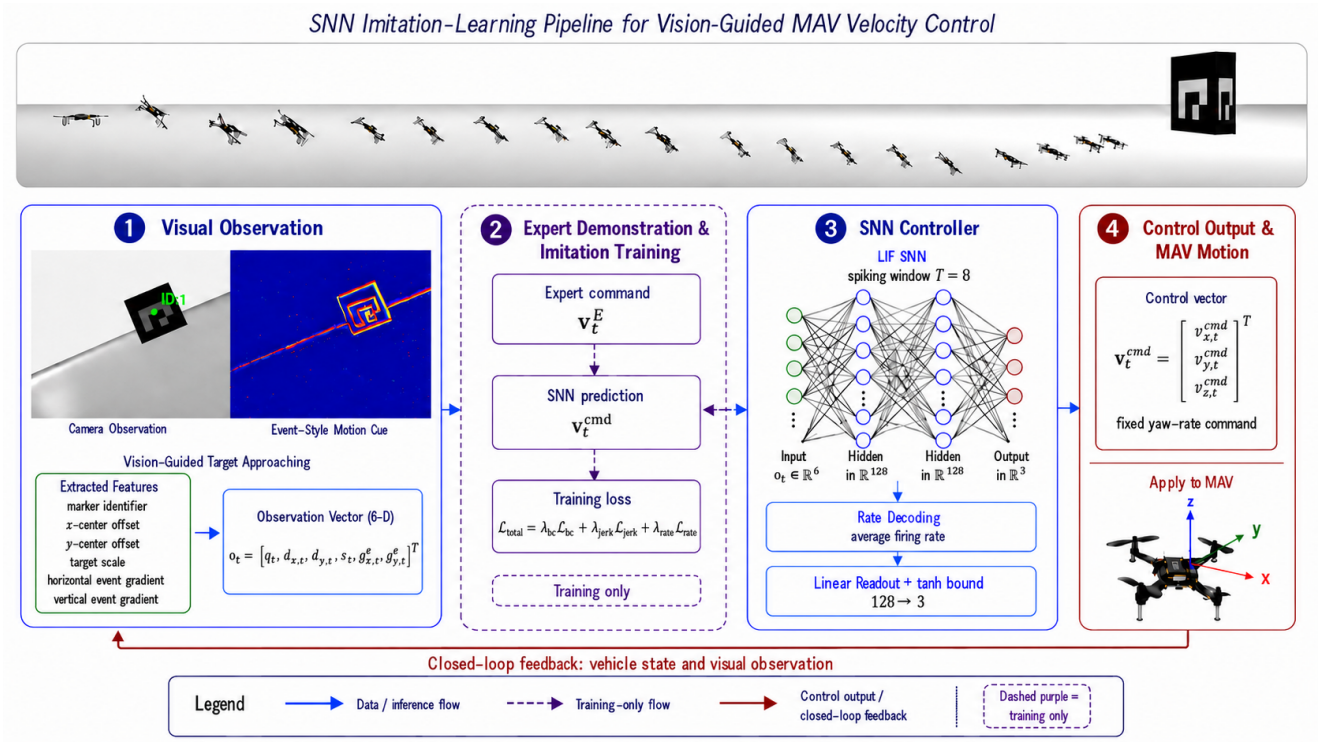

Figure 1: Overview of the proposed closed-loop imitation-learning pipeline. During training, the expert controller provides reference velocity commands for supervised imitation, while the rate-decoded SNN learns to map compact visual observations to three-axis velocity commands. During evaluation, the trained SNN is deployed as the closed-loop velocity controller.

which are not available to the SNN policy. These expert commands serve as supervised targets for the SNN. The training objective combines axis-weighted behavioral cloning loss, command-jerk regularization, and spike-rate range regularization. The behavioral cloning term encourages the SNN output to match the expert velocity command, the jerk term penalizes abrupt temporal variations in the predicted command sequence, and the spike-rate term constrains the firing activity within a desired range. This formulation enables the SNN controller to learn accurate, smooth, and stable spiking control outputs for closed-loop target approaching.

2.2 *Implementation of Spiking Neural Networks*

The proposed control policy is implemented as a rate-decoded spiking neural network (SNN) composed of leaky integrate-and-fire (LIF) neurons. Unlike conventional artificial neural networks that transmit continuous activations at each layer, the LIF-based SNN maintains an internal temporal state through membrane-potential dynamics and communicates information using binary spike events. Such temporal dynamics allow the controller to encode short-term variations in visual observations while preserving the event-driven computational characteristics of spiking neural networks.

The policy maps a compact visual observation vector $\mathbf{o}_t$ to a continuous translational velocity command. The observation vector contains the normalized visual and motion-related features extracted at control step t. The overall network consists of an input projection module, an LIF spiking computation layer, a rate-decoding module, and a continuous velocity readout. First, the observation is embedded into a hidden feature representation through a multilayer projection function:

$$\mathbf{h}_t = f_{\text{proj}}(\mathbf{o}_t), \tag{3}$$

where $f_{\text{proj}}(\cdot)$ is composed of fully connected layers with normalization and nonlinear activation. This projection transforms the low-dimensional observation into a latent representation suitable for subsequent spiking computation.

The projected feature $\mathbf{h}_t$ is then injected into the LIF spiking layer over a short internal time window of length T. In the reported experiments, the internal spiking window is set to $T = 8$. According to the implementation, the LIF layer contains 128 neurons, with $\tau_{\text{mem}}^{-1} = 10.0$, $\tau_{\text{syn}}^{-1} = 5.0$, and a firing threshold of $\vartheta_{\text{th}} = 0.30$. At each internal spiking step k, the LIF neurons update their membrane potential according to the incoming current and their previous hidden state. When the membrane potential exceeds the firing threshold, a binary spike is generated and the neuron state is reset. The spike-generation process can be written as

$$\boldsymbol{\sigma}_{t,k}, \boldsymbol{\xi}_{t,k} = f_{\text{LIF}}\left(\mathbf{h}_t, \boldsymbol{\xi}_{t,k-1}\right), \quad k = 1, \ldots, T, \tag{4}$$

where $\boldsymbol{\sigma}_{t,k}$ denotes the binary spike vector at the k-th internal

http://www.imavs.org/

step, and $\boldsymbol{\xi}_{t,k}$ represents the corresponding LIF neuron state, including the membrane potential and synaptic current. The generated spike sequence within one control step is denoted as

$$\mathcal{B}_t = \{\boldsymbol{\sigma}_{t,1}, \boldsymbol{\sigma}_{t,2}, \ldots, \boldsymbol{\sigma}_{t,T}\}. \tag{5}$$

This internal temporal unfolding enables the policy to convert a compact visual feature into a spike-based temporal representation before producing the final velocity command.

Since the quadrotor requires continuous velocity commands, a rate-decoding strategy is adopted to bridge the discrete spike representation and the continuous control space. Specifically, the spike-rate vector is computed by averaging the binary spike sequence over the internal time window:

$$\bar{\boldsymbol{\sigma}}_t = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{\sigma}_{t,k}. \tag{6}$$

The decoded spike-rate vector $\bar{\boldsymbol{\sigma}}_t$ provides a continuous representation of the spiking activity and is used as the input to the final velocity readout layer. The translational velocity command is obtained by

$$\mathbf{v}_t^{\mathrm{cmd}} = \mathbf{v}_{\max} \odot \tanh\left(\mathbf{W}_o \bar{\boldsymbol{\sigma}}_t + \mathbf{b}_o\right), \tag{7}$$

where $\mathbf{W}_o$ and $\mathbf{b}_o$ are the learnable parameters of the linear readout layer, $\tanh(\cdot)$ bounds the output range, $\mathbf{v}_{\max}$ denotes the maximum allowable velocity for each axis, and $\odot$ represents element-wise multiplication.

The final output of the policy is defined as

$$\mathbf{v}_t^{\mathrm{cmd}} = \left[v_{x,t}^{\mathrm{cmd}}, v_{y,t}^{\mathrm{cmd}}, v_{z,t}^{\mathrm{cmd}}\right]^\top, \tag{8}$$

which corresponds to the translational velocity commands along the x, y, and z axes. In the current setting, the yaw-rate command is not learned by the SNN and is fixed to zero during policy execution. Different from approaches that rely on an additional output low-pass filter, the proposed policy directly uses the rate-decoded SNN output as the control command. Therefore, the temporal smoothness of the generated velocity command is mainly determined by the intrinsic LIF dynamics, the spike-rate decoding mechanism, and the smoothness-aware training objective.

2.3 *Expert-Guided Imitation Learning*

The proposed SNN controller is trained using expert-guided imitation learning. During training, an expert policy is used to provide reference translational velocity commands for the visual servoing task. The expert policy is implemented as a privileged PID-based outer-loop velocity controller, which computes stable velocity demonstrations from vehicle position and velocity feedback according to the tracking error between the vehicle and the target. These expert commands serve as supervised labels for training the SNN policy.

At each control step, the perception module extracts the visual observation $\mathbf{o}_t$, and the SNN predicts a velocity command $\mathbf{v}_t^{\mathrm{cmd}}$ from this observation. In parallel, the expert policy generates the reference command $\mathbf{v}_t^{\mathrm{E}}$. The SNN is optimized to imitate the expert command while preserving smooth and functional spiking behavior. During the early training stage, the simulator is driven by the expert command rather than the SNN output. This prevents unstable commands from an untrained SNN from destabilizing the vehicle and allows the model to learn from stable closed-loop trajectories.

Algorithm 1 Closed-loop rate-decoded SNN imitation learning

```
Initialize environment, expert, and SNN policy
for episode = 1 to 1000 do
    Randomize the initial MAV pose
    Reset expert integrator and SNN hidden state
    Set rollout length by curriculum schedule
    for step = 1 to T_ep do
        Extract and normalize visual observation o_t
        Compute expert command v_t^E
        Predict SNN command v_t^cmd by rate decoding
        Accumulate losses after warm-up steps
        Apply the expert command to the simulator
        if step is divisible by 20 then
            Detach SNN hidden state from the graph
        end if
    end for
    Backpropagate the accumulated loss and update SNN
    parameters
end for
```

The training process follows a closed-loop rollout procedure. For each episode, the vehicle is initialized from a randomized pose, the expert and SNN hidden states are reset, and the system is rolled out for a fixed number of control steps. After an initial warm-up period, the SNN prediction and expert command are collected to compute the imitation loss. The SNN hidden state is periodically detached from the computational graph to avoid excessively long backpropagation histories while preserving its temporal state during execution. After each episode, the accumulated loss is backpropagated through the SNN using surrogate gradients, and the network parameters are updated with gradient-based optimization.

The rollout length follows a curriculum schedule from 100 to 600 control steps. For an episode of length T_{ep}, the warm-up duration is set to $N_{\mathrm{w}} = \min(50, \lfloor 0.3T_{\mathrm{ep}} \rfloor)$ control steps. These initial steps are excluded from loss accumulation to reduce the influence of reset transients in the LIF state and visual-motion features. Adam optimization is used with a learning rate of 10^{-4} and weight decay of 10^{-6}.

2.4 *Loss Function*

The SNN policy is trained by minimizing a composite loss that combines expert imitation, command smoothness,

http://www.imavs.org/

http://www.imavs.org/

and spike-activity regularization. The total loss is defined as

$$\mathcal{L}_{\text{total}} = \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{jerk}}\mathcal{L}_{\text{jerk}} + \lambda_{\text{rate}}\mathcal{L}_{\text{rate}}. \tag{9}$$

In the reported experiments, the loss weights are set to $\lambda_{\text{bc}} = 1.0$, $\lambda_{\text{jerk}} = 0.02$, and $\lambda_{\text{rate}} = 0.01$. An additional task-direction metric is computed during training for monitoring purposes, but its weight is set to zero and it is not used for parameter optimization.

The primary learning objective is axis-weighted behavioral cloning, which penalizes the discrepancy between the SNN command $\mathbf{v}_t^{\text{cmd}}$ and the expert command $\mathbf{v}_t^{\text{E}}$:

$$\mathcal{L}_{\text{bc}} = \frac{1}{3|\mathcal{T}|}\sum_{t\in\mathcal{T}}\sum_{j\in\{x,y,z\}} w_j\left(v_{j,t}^{\text{cmd}} - v_{j,t}^{\text{E}}\right)^2, \tag{10}$$

where $\mathcal{T}$ denotes the set of training steps after the warm-up period, and the axis-weight vector is set to $\mathbf{w} = [1.0, 6.0, 3.0]^\top$. Larger weights are assigned to the lateral and vertical axes because these channels are more sensitive to visual-servoing errors and were observed to be more difficult to fit during training.

To suppress abrupt command variations, a command-jerk regularization term is introduced. It penalizes the second-order temporal difference of the predicted velocity commands over a rollout:

$$\mathcal{L}_{\text{jerk}} = \frac{1}{T_{\text{ep}} - 2}\sum_{t=3}^{T_{\text{ep}}}\left\|\mathbf{v}_t^{\text{cmd}} - 2\mathbf{v}_{t-1}^{\text{cmd}} + \mathbf{v}_{t-2}^{\text{cmd}}\right\|_2^2. \tag{11}$$

This term encourages temporally smooth velocity outputs without applying an explicit low-pass filter to the final command.

The spike-rate regularization term is designed to maintain functional spiking activity in the hidden LIF layer. Instead of enforcing a fixed target firing rate, a range-based penalty is used:

$$\mathcal{L}_{\text{rate}} = \left[\max\left(0, r_{\text{low}} - \bar{r}\right)\right]^2 + \left[\max\left(0, \bar{r} - r_{\text{high}}\right)\right]^2, \tag{12}$$

where $\bar{r}$ denotes the mean spike rate of the hidden LIF layer. The lower and upper bounds are set to $r_{\text{low}} = 0.05$ and $r_{\text{high}} = 0.18$, respectively. This formulation prevents silent neurons and saturated firing activity while allowing the network to learn a flexible range of effective spike rates.

3 EXPERIMENTAL SETUP

The experiments are conducted in a ROS2-Gazebo simulation environment based on the Crazyflie nano-quadrotor platform. Following the software-in-the-loop simulation idea of CrazySim [12], the environment is implemented using ROS2, Gazebo, and the open-source Crazyflie ROS2-Gazebo package[1]. The package provides a Gazebo-based Crazyflie model, ROS2 communication interfaces, and velocity-command interaction, enabling repeatable closed-loop evaluation of vision-guided MAV control.

The perception and control pipeline is implemented as ROS2 nodes. The onboard camera stream is processed using OpenCV, and an ArUco marker is used as the visual reference. Compact visual features, including marker center, apparent scale, and short-term image-space motion cues, are extracted and used as the input to the SNN controller. The SNN is implemented with PyTorch and Norse [11], where leaky integrate-and-fire neurons and surrogate-gradient-compatible spiking computation are used. Velocity commands are published to the Crazyflie simulation interface, while odometry feedback is obtained through the ROS2-Gazebo bridge.

The task is defined as vision-guided target approaching in a three-dimensional workspace. A floating visual reference marker is placed at $[2.2, 0, 0.8]^\top$ m, while the desired target position is set to $\mathbf{p}^* = [2.0, 0, 0.8]^\top$ m. This offset reflects a practical visual-servoing setting, where the marker provides perception guidance and the UAV is expected to stabilize at a desired relative position rather than directly collide with the marker. At the beginning of each rollout, the UAV is initialized from a randomized position sampled from $x_0 \in [-2.0, 0.0]$ m, $y_0 \in [-1.0, 1.0]$ m, and $z_0 \in [0.75, 0.95]$ m. The range is chosen to ensure that the marker is visible in most initial states while still providing sufficient variation in distance and lateral offset.

For evaluation, the trained SNN controller and the expert reference controller are tested from the same initial poses over a fixed rollout length of 600 control steps. The comparison considers 3D trajectory, tracking error, per-axis velocity command, command jerk, and spike activity. Command jerk is computed from the discrete second-order difference of the velocity command, while spike activity is measured by the mean hidden-layer spike rate. These metrics jointly evaluate target-reaching performance, control smoothness, and the functional spiking behavior of the proposed controller.

4 EXPERIMENTAL RESULTS

This section evaluates the learned SNN controller in the vision-guided target-approaching task. The expert controller uses physical position and velocity feedback, whereas the SNN controller generates three-axis velocity commands from compact visual observations. Therefore, the evaluation focuses on whether the SNN can reproduce the dominant target-approaching behavior rather than exactly match the expert trajectory at every time step.

Fig. 2 presents a representative individual trajectory produced by the rate-decoded SNN controller. The stroboscopic sequence in panel (a) visualizes the complete target-approaching motion. Panels (b) and (c) show the trajectory projected onto the x–z and x–y planes, respectively, with color encoding the elapsed time. Starting from the light-colored square, the MAV progresses toward the target while

[1] https://github.com/bitcraze/crazyflie-simulation

http://www.imavs.org/

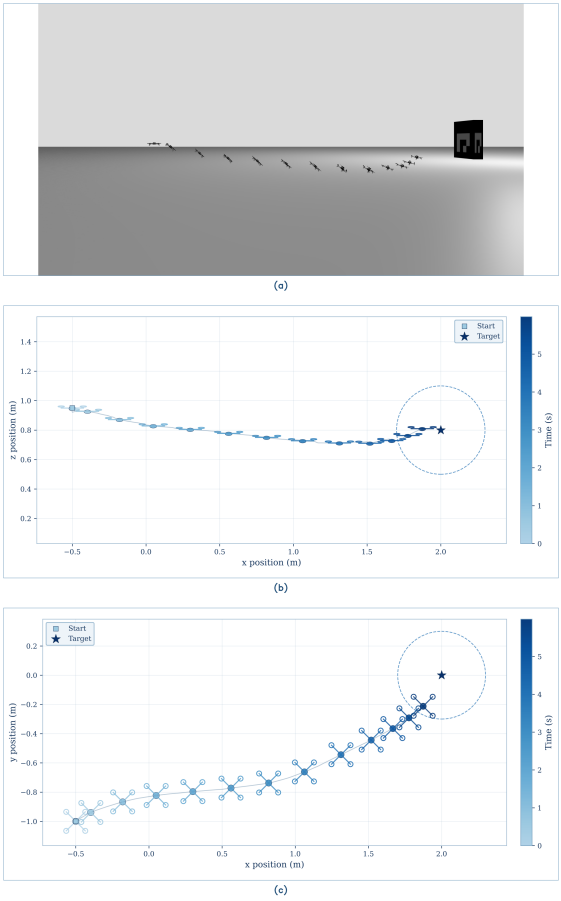

Figure 2: Visualization of the MAV target-approaching trajectory and convergence behavior.

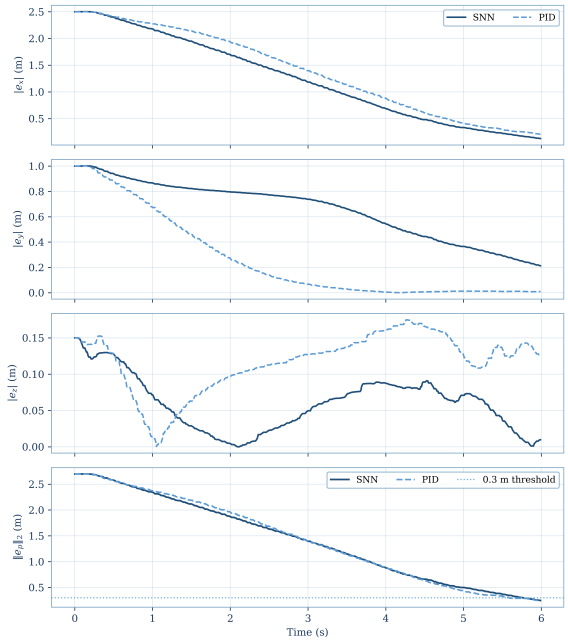

Figure 3: Position tracking error comparison over the evaluation rollout.

correcting its lateral position and altitude. The trajectory converges toward the dashed target region, demonstrating stable target-approaching behavior during closed-loop visual servoing.

The tracking-error curves in Fig. 3 show that both controllers continuously reduce the initial position error during the 600-step rollout. The SNN follows the same decreasing trend as the expert, indicating that the learned controller can generate effective closed-loop velocity commands from visual observations.

Fig. 4 reports the position responses along the x, y, and z axes. The x-axis response shows that both controllers move consistently toward the target, indicating that the SNN learns the dominant forward target-approaching behavior. The y-axis response indicates that the SNN can reduce the lateral deviation, although its convergence profile differs from that of the expert controller. The SNN z-axis response remains bounded around the target altitude but shows a larger residual offset than the expert response. The vertical dashed line indicates the time step at which the camera temporarily loses the visual target. This loss of visual reference may reduce the reliability of the compact visual observation and contribute to the subsequent lateral and vertical deviations. These results suggest that the SNN learns a usable visual servoing policy, while axis-level precision is still affected by visual-target visibility and closed-loop dynamics.

The velocity-command comparison in Fig. 5 further explains the observed motion. Along the x axis, the SNN produces a forward velocity close to the dominant expert command. Along the y and z axes, the SNN commands are less adaptive and show local fluctuations. Nevertheless, the generated commands remain sufficient to guide the MAV toward the target region.

Command smoothness is evaluated using discrete command jerk, as shown in Fig. 6. The SNN produces a short initial jerk peak after rollout begins, but the jerk remains low for most of the trajectory. Since no output low-pass filter is applied, this result suggests that the rate-decoding mechanism and smoothness-aware training provide usable command continuity.

Fig. 7 shows the hidden-layer spike activity during SNN evaluation. The spike rate increases at the beginning and then remains stable, indicating that the LIF layer stays active during closed-loop control and that the velocity command is produced through functional spike-rate representations.

To evaluate robustness under randomized initial conditions, the SNN controller was tested from different randomly sampled initial positions, as shown in Fig. 8. The resulting trajectories show that the SNN consistently drives the MAV toward the target region from different initial states. The

http://www.imavs.org/

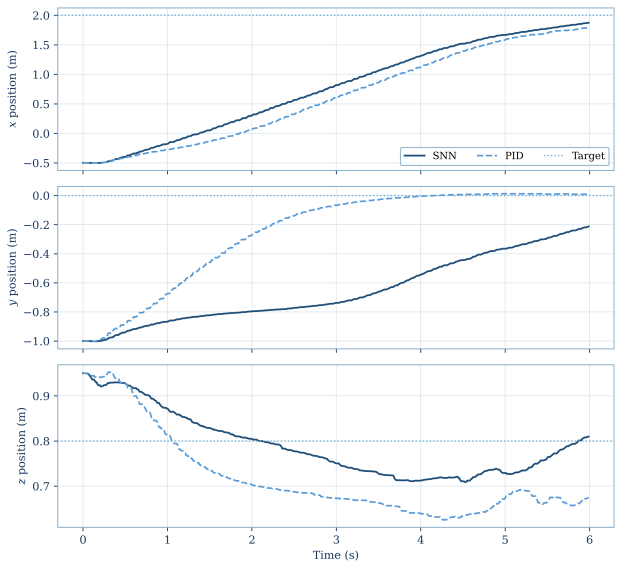

Figure 4: Position-axis comparison for the expert controller and the SNN controller.

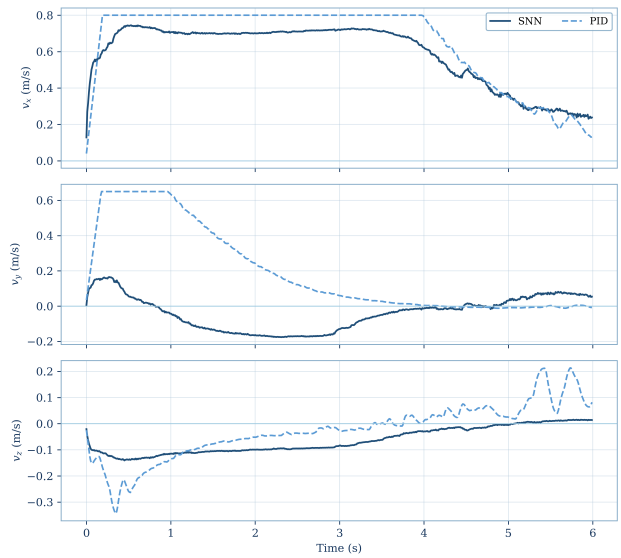

Figure 5: Velocity command comparison along the x, y, and z axes.

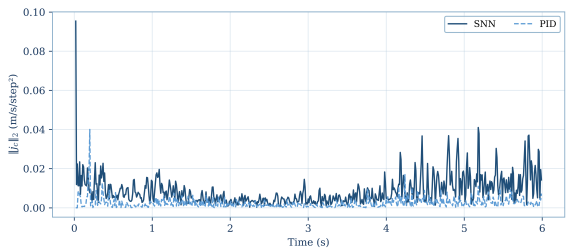

Figure 6: Command-jerk comparison between the expert controller and the SNN controller.

trajectories exhibit clear forward progress along the x direction, lateral correction from both sides of the workspace, and bounded vertical motion around the target altitude with small residual offsets. These results demonstrate that the learned controller preserves a consistent target-approaching tendency under randomized initial conditions.

Overall, the experimental results show that the rate-decoded SNN controller can reproduce the main visual target-approaching behavior of the expert controller. The learned policy reduces tracking error, generates bounded velocity commands, and maintains stable spike activity during closed-loop execution. Remaining lateral and vertical deviations indicate that axis-level precision can be further improved through better visual-state representation, smoother readout dynamics, and closed-loop fine-tuning.

5 CONCLUSION

This paper presented a rate-decoded SNN controller for vision-guided velocity control of an MAV. The proposed method uses compact visual-state observations as input and learns continuous three-axis translational velocity commands by imitating a privileged PID expert with access to vehicle position and velocity feedback. The SNN policy is built with leaky integrate-and-fire spiking dynamics, where hidden spike activity is converted into continuous control outputs through spike-rate decoding. To improve learning stability and control quality, the training objective combines axis-weighted behavioral cloning, command-jerk regularization, and spike-rate regularization. Experimental results demonstrate that the learned SNN controller can reproduce the dominant target-approaching behavior of the expert controller. The vehicle is driven toward the target region, the tracking error is gradually reduced, and the overall convergence trend is comparable to the expert reference. The spike-activity analysis further validates that the hidden spiking layer remains active during closed-loop execution, which indicates that the controller preserves functional spiking dynamics while producing continuous velocity commands. Overall, the results show that rate-decoded SNN imitation learning provides an effective and compact control framework for vision-guided MAV target approaching. Future work will extend the framework toward attitude-aware control and richer motion-cue representations to better handle coupled translational and rotational visual dynamics, together with onboard deployment on resource-constrained hardware.

REFERENCES

[1] O. K. Pal, "In-depth review of AI-enabled unmanned aerial vehicles," *Innovative Infrastructure Solutions*, vol. 9, Art. no. 415, 2024.

[2] L. Lamberti, E. Cereda, G. Abbate, L. Bellone, V. J. K. Morinigo, M. Barcis, A. Barcis, A. Giusti, F.

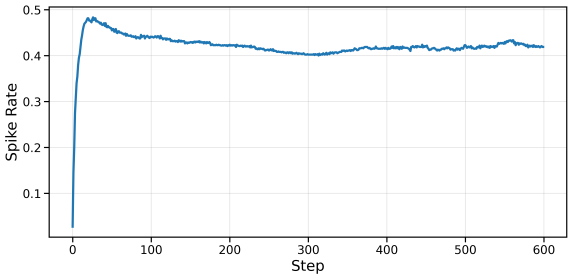

Figure 7: Average hidden-layer spike activity during SNN evaluation.

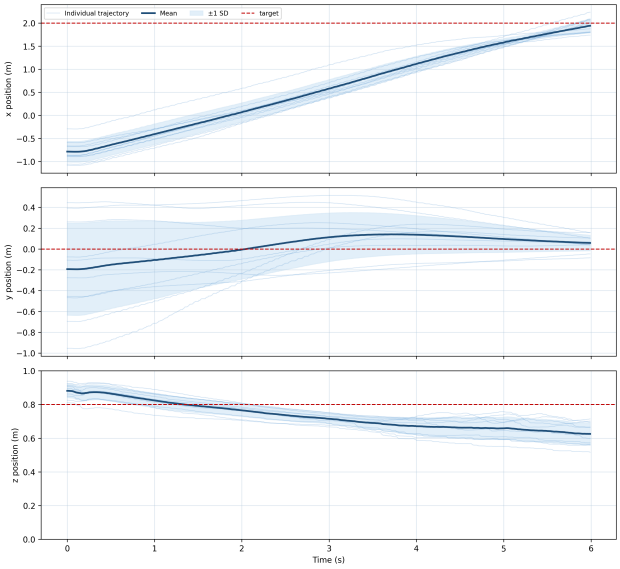

Figure 8: SNN trajectories under randomized initial conditions.

Conti, and D. Palossi, "A sim-to-real deep learning-based framework for autonomous nano-drone racing," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1899–1906, 2024.

[3] C. Vincent, L. Hirsch, N. Ul Hasan, C. C. E. Fernandes, A. Crainic, J. Zasada-James, and J. Baldwin, "Real-time edge intelligence in UAV for search-and-rescue: Onboard energy-efficient video summarisation with reduced data transmission," in *Proceedings of the 15th International Conference on the Internet of Things*, ACM, pp. 52–58, 2025.

[4] A. Abdalla, M. M. A. Mohammed, O. Adedeji, P. Dotray, and W. Guo, "Toward resource-efficient UAV systems: Deep learning model compression for onboard-ready weed detection in UAV imagery," *Smart Agricultural Technology*, vol. 12, Art. no. 101086, 2025.

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

[6] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019, doi: 10.1109/MSP.2019.2931595.

[7] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, vol. 111, no. 9, pp. 1016–1054, 2023, doi: 10.1109/JPROC.2023.3308088.

[8] F. Paredes-Valles, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. C. H. E. de Croon, "Fully neuromorphic vision and control for autonomous drone flight," *Science Robotics*, vol. 9, no. 90, Art. no. eadi0591, 2024, doi: 10.1126/scirobotics.adi0591.

[9] S. Stroobants, C. De Wagter, and G. C. H. E. de Croon, "Neuromorphic attitude estimation and control," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 4858–4865, 2025, doi: 10.1109/LRA.2025.3553418.

[10] M. B. van Breukelen Castillo, C. De Wagter, R. Ferede, R. Vos, and G. C. H. E. de Croon, "Spiking neural networks for high-speed continuous quadcopter control using proximal policy optimization," in *Proceedings of the 16th International Micro Air Vehicle Conference and Competition*, San Andres Cholula, Mexico, 2025, pp. 133–141.

[11] C. Pehle and J. E. Pedersen, "Norse – A deep learning library for spiking neural networks," Zenodo, version 0.0.7, 2021, doi: 10.5281/zenodo.4422025.

[12] C. Llanes, Z. Kakish, K. Williams, and S. Coogan, "CrazySim: A software-in-the-loop simulator for the Crazyflie nano quadrotor," in *Proceedings of the 2024 IEEE International Conference on Robotics and Automation*, Yokohama, Japan, 2024, pp. 12248–12254, doi: 10.1109/ICRA57147.2024.10610906.

http://www.imavs.org/