A Survey on Spiking Neural Network for Drone Flight Control

スパイキングニューラルネットワークを用いたドローン飛行制御に関する研究調査

平田 涼馬(Ryoma Hirata)
Fukuda lab. M1

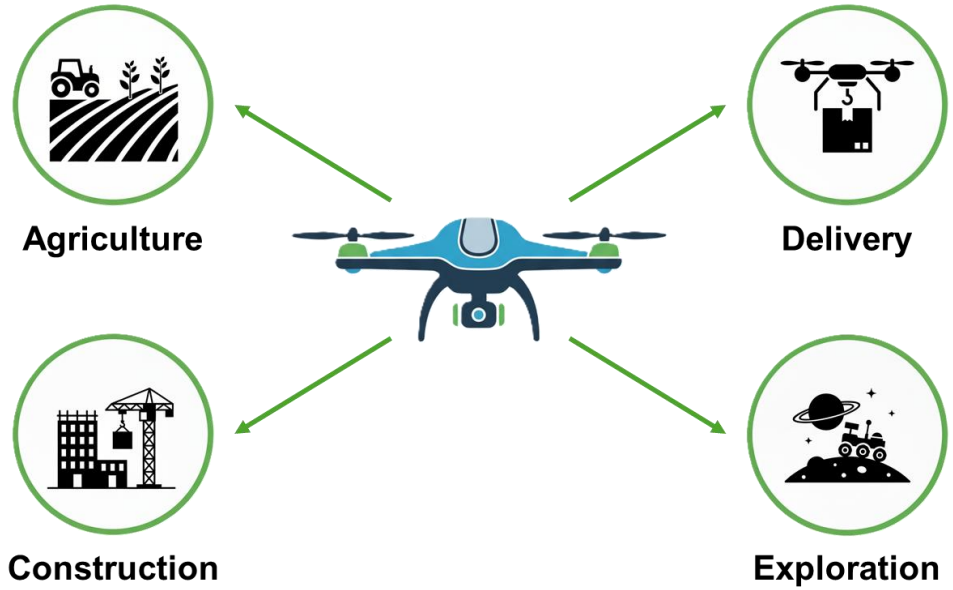
Outline

1. Introduction
2. Neuromorphic Attitude Estimation and Control using IMU
3. Neuromorphic Control using Event-based Vision
4. Agile Navigation using Reinforcement Learning-Trained SNN
5. Conclusion

1. Introduction

- Benefits of Drones
  1. High mobility
  2. Low cost of introduction / operation

- Benefits of AI Control
  - **autonomous**
  - **performing complex tasks**



- Comparison: Traditional Control vs AI Control

features	Conventional Control (PID / Model base…)	AI Control
Adaptability	Low	**High**
Target Environment	Static	**Dynamic, unpredictable, complex**
Interpretability	**High**	Low
Reliability	**High**	Low

1. Introduction

Two methods of AI control for drones

On-board

- ○ Real-time processing
- × Power consumption
- × Inference accuracy

Ground-base

- ○ Inference accuracy
- × Real-time processing

Trade-off in On-board AI

Factor	Focusing on accuracy	Focusing on Flight Time
① Inference Accuracy	High	Low
② Model Size	Large	Small
③ Calculation amount	High ⟷	Low
④ Power Consumption	High	Low
⑤ Flight Time	Short	Long

- In the **Martian environment**, processing must be done **on-board**
  - There is too much latency in communication with Earth.

- NASA researchers propose using Spiking Neural Networks to control future Mars drones

Fig. Ingenuity[1]

[1] Harbour, David AR, et al, "Martian Flight: Enabling Motion Estimation of NASA's Next-Generation Mars Flying Drone by Implementing a Neuromorphic Event-Camera and Explainable Fuzzy Spiking Neural Network Model." *2024 AIAA DATC/IEEE 43rd Digital Avionics Systems Conference (DASC)*. IEEE, 2024.

1. Introduction

Spiking Neural Network (SNN)

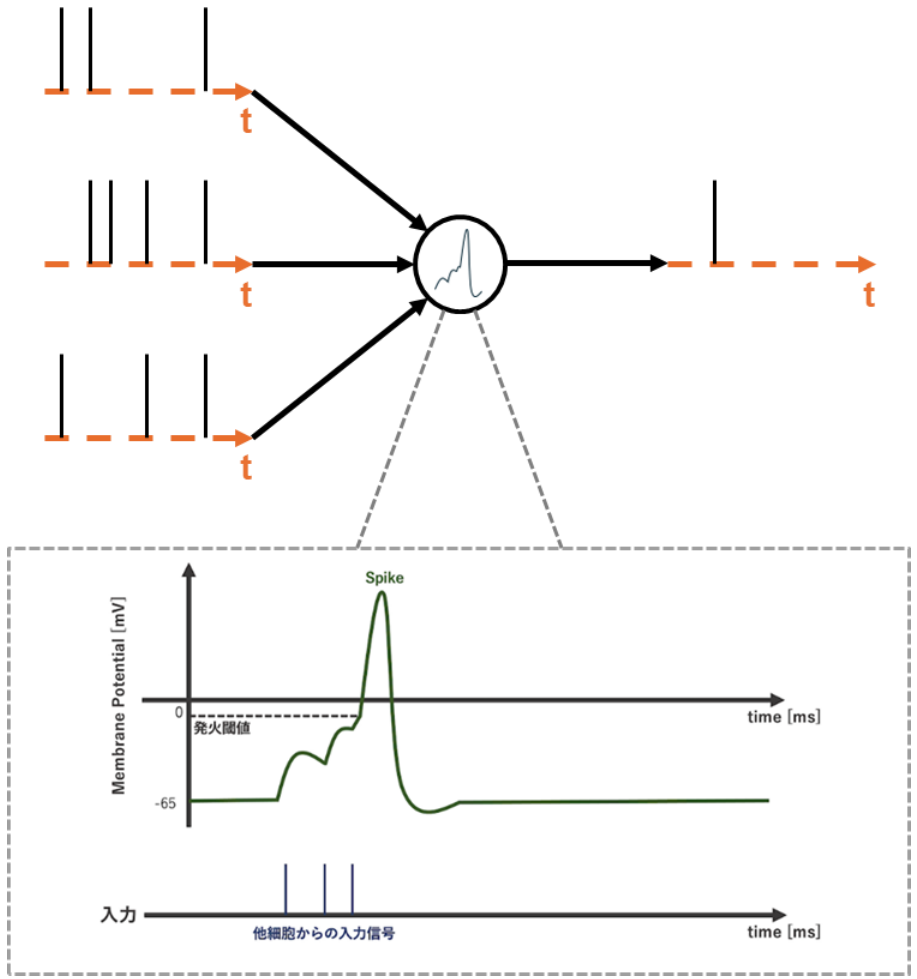


Fig. SNN Operation (Modified from [2])

LIFモデル

$$\tau_m \frac{dV(t)}{dt} = \underbrace{(-V(t) + E_{rest})}_{\text{Leak Term}} + \underbrace{I(t)}_{\text{Input Current}}$$

τ_m : Membrane Time Constant
$I(t)$: Input Current
$V(t)$: Membrane Potential
E_{rest} : Resting Membrane Potential

- ○ Low Power Consumption
- ○ Low Latency Response
- ✕ Training Difficulty
- ✕ Hardware Dependency

[2] ゼロから学ぶスパイキングニューラルネットワーク https://snn.hirlab.net/?s=2/

1. Introduction

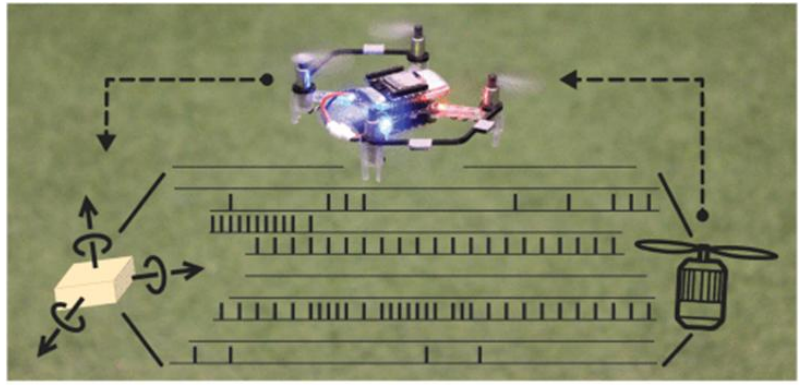
- This investigation focused on two applications:
  low-level control (attitude control) and **path planning**

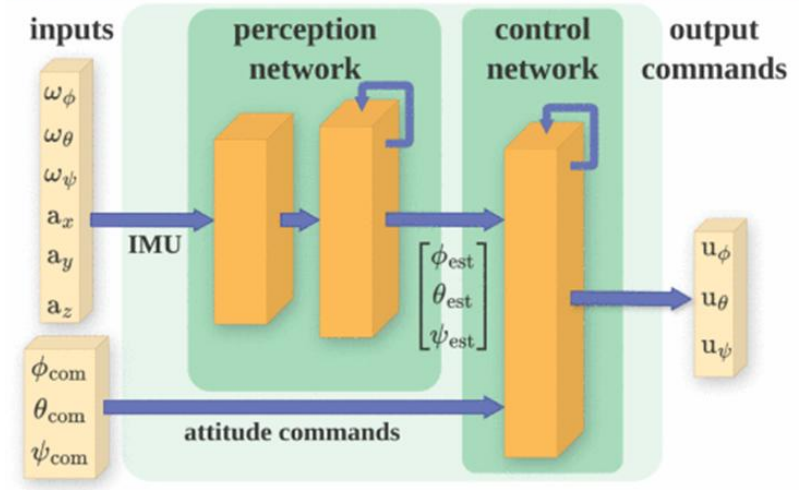
Application	Sensor	SNN output	Platform	Introduction paper
Low-level control (Attitude control)	**IMU**	**Attitude / control**	Microprocessor	**Paper 1**
Low-level control (Attitude control)	**Event camera**	**Optical flow**	Neuromorphic processor	**Paper 2**
Path planning	-	**Attitude**	Simulator	**Paper 3**

2. Neuromorphic Attitude Estimation and Control using IMU

Paper1
S. Stroobants, C. De. Wagter, and G. CHE. De. Croon,
”Neuromorphic attitude estimation and control”,
IEEE Robotics and Automation Letters, (2025).

- **Key point**
  - This paper propose an **SNN model** that outputs **low-level control** commands from **IMU** inputs.
- **Network Architecture**
  - Divided into two subnetworks
    1. Perception network (SNN)
       - Input : [Angular velocity, Acceleration]
       - Output : [Attitude (Euler kernel)
    2. Control network (SNN)
       - Input : [Attitude (estimation + target)]
       - Output : [**Torque**]

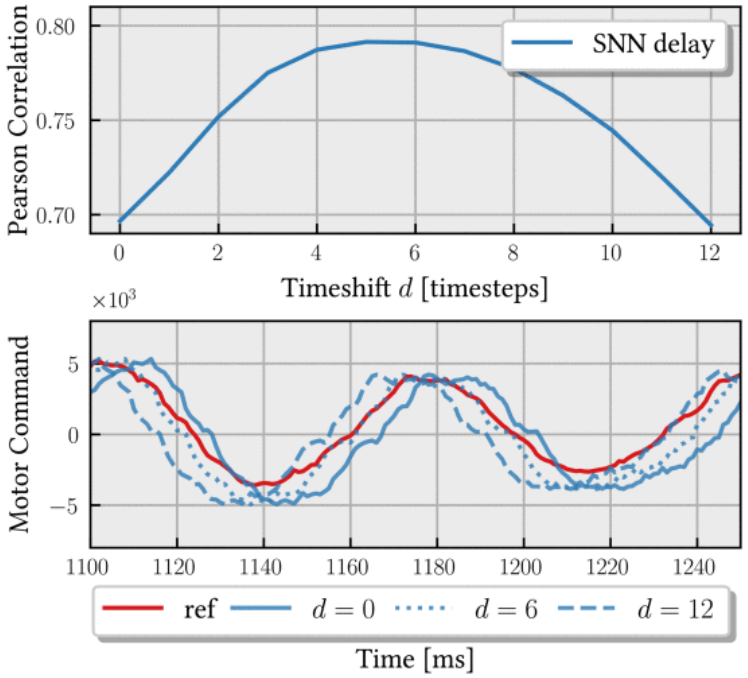




2. Neuromorphic Attitude Estimation and Control using IMU

Training techniques

1. Handling SNN-specific delays



– Shift the ground truth labels by 6 steps

2. Data augmentation

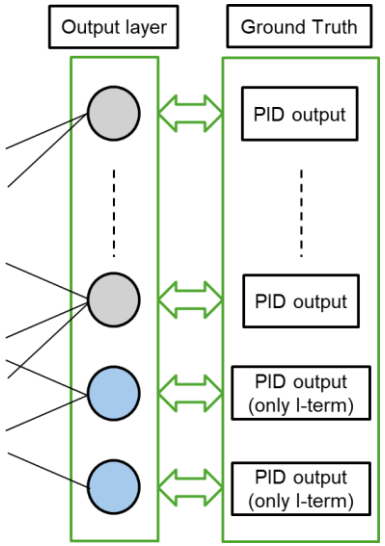
    1. Flying using a SNN trained with the initial training data

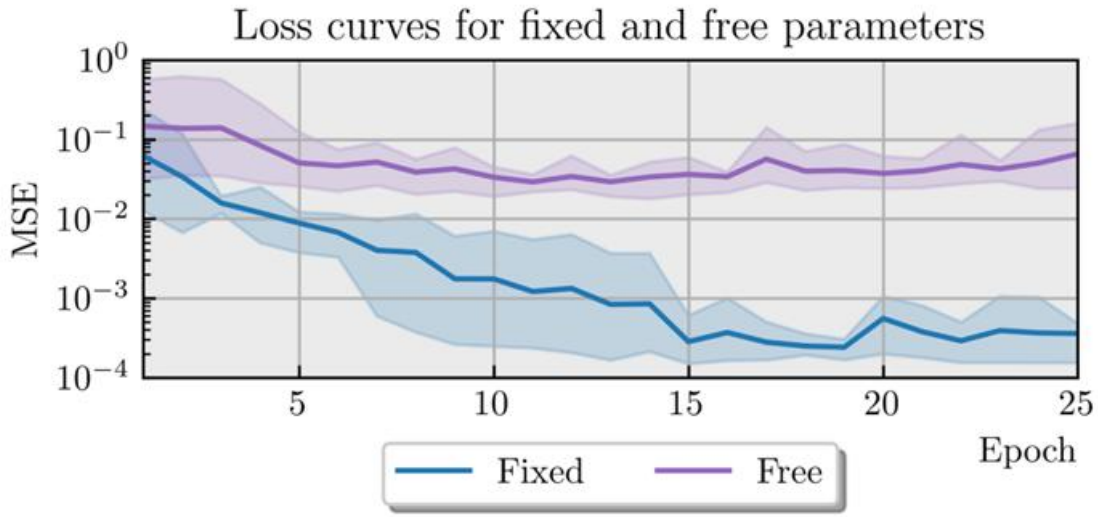
    2. Gathering data by adding random disturbances to the PID controller

2. Neuromorphic Attitude Estimation and Control using IMU

Training : Integral control compensation

- Learning integral control of PID is difficult.
  - Learning PD control takes priority
  - membrane potential of neurons can’t retain information for a long time due to **leakage**

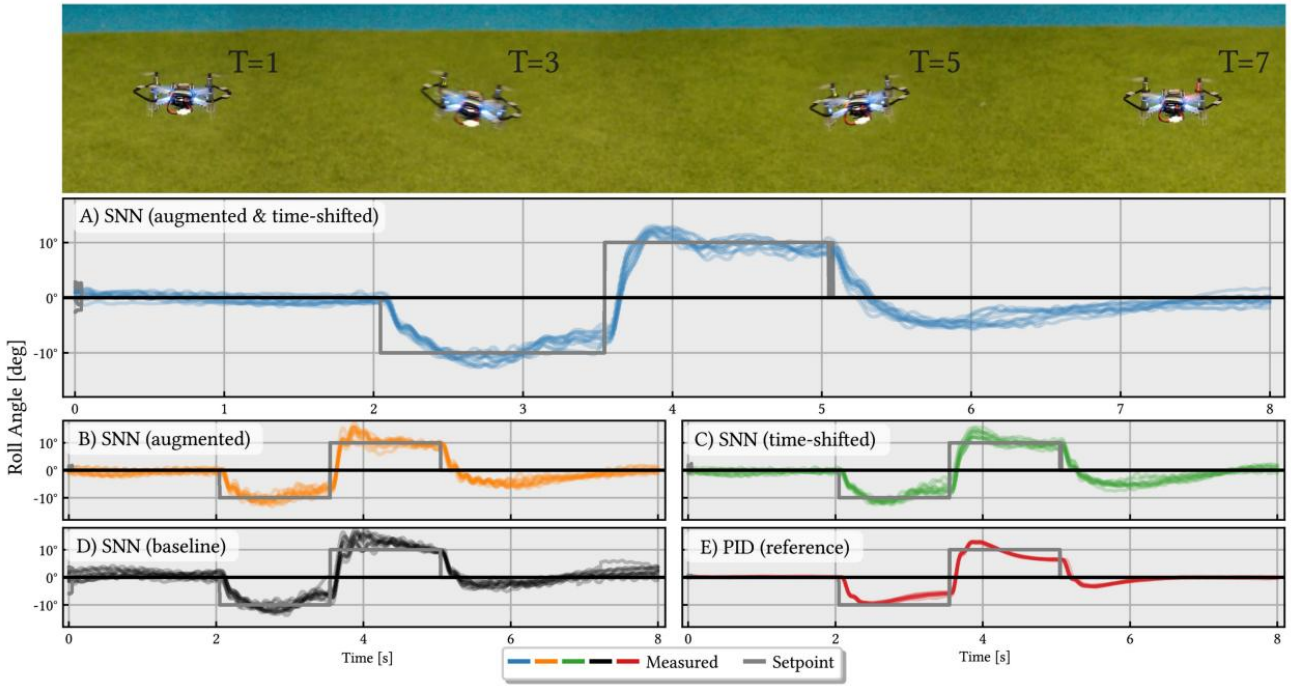




  - Fixing the leak parameters of 10 neurons to integrate input in the membrane potential
  - Set the GT labels for those neurons to I-term of PID control

2. Neuromorphic Attitude Estimation and Control using IMU

Result : Roll angle control



Definition of RMSE
(Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}$$

y_i : Ground truth,
$\hat{y}_i$: Predicted value,
n : total data count

Controller	RMSE	avg. SD	avg. RT
SNN (time-shifted & augm.)	3.03°	0.77	145ms
SNN (augmented)	3.10°	0.95	130ms
SNN (time-shifted)	3.24°	0.92	145ms
SNN (baseline)	3.14°	1.16	135ms
PID	2.67°	0.23	125ms

Note that the PID receives the estimated attitude as input, while the SNN needs to calculate this internally.

PID has higher tracking accuracy

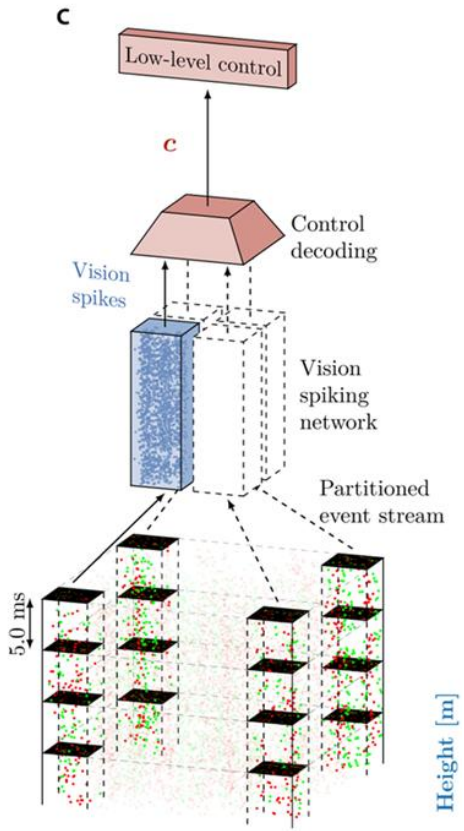
- SNN's performance is **comparable** when considering that it is **also estimating attitude**

3. Neuromorphic Control using Event-based Vision

Paper2
F Paredes-Vall´es, J J Hagenaars, J Dupeyroux, S Stroobants, Y Xu, and G CHE de Croon, ”fully neuromorphic vision and control for autonomous drone flight”, Science Robotics, Vol. 9, No. 90, eadi0591, (2024)

- **Key point**
  - This paper propose an **SNN model** that outputs **low-level control** commands from **event camera** inputs.

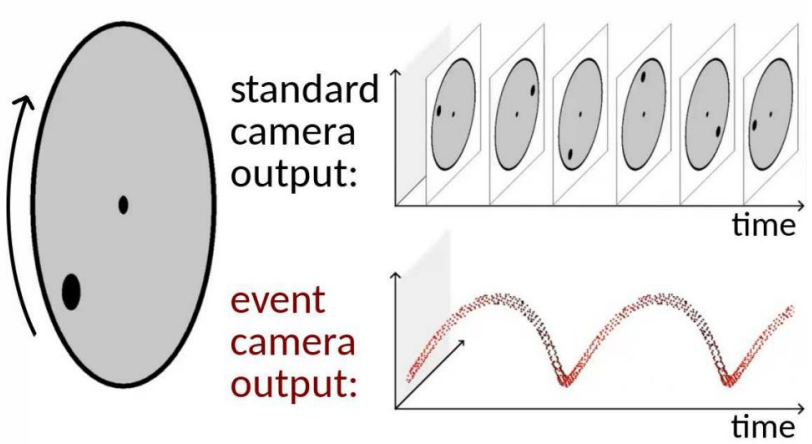
- **Network Architecture**
  - Divided into two blocks
    1. Vision spiking network (SNN)
       - Input : [**Event stream (Event camera)**]
       - Output : [Ego-motion (translational velocity, angular velocity)]
    2. Control decoder (Linear map)
       - Input : [Ego-motion, Target Ego-motion]
       - Output : [Thrust, Torque]



3. Neuromorphic Control using Event-based Vision

- **What's Event Camera**
  - **Asynchronously** detects changes in brightness for each pixel and outputs them as events
    ⇔ Records brightness values for all pixels **simultaneously** at each time step (RGB camera)

- **Features of Event Camera**
  - Temporal resolution
  - Low power consumption
  - High Dynamic Range (HDR)



Event camera operation [1]

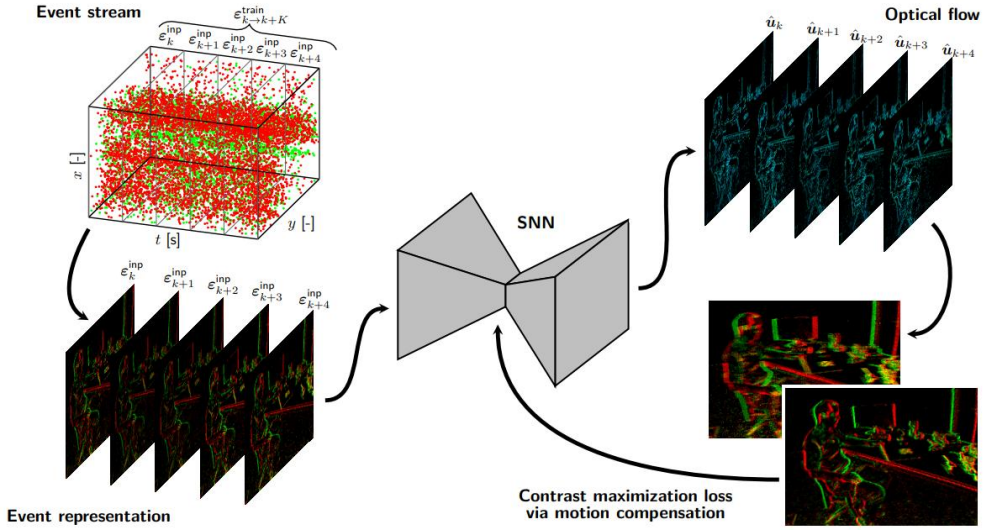
- **Advantages of combining with SNN**
  - **Low latency response** through asynchronous processing
  - Efficient processing of sparse event data, enabling **low power consumption**
  - Natural handling of temporal information

[1] Gehrig+ "Asynchronous, Photometric Feature Tracking using Events and Frames" @ECCV 2018

3. Neuromorphic Control using Event-based Vision

- **Optical-flow**
  - Vectors representing the movement (or motion) of objects and textures between consecutive images

- **Event-based Optical flow**
  - Implemented as a self-supervised learning
  - Using an encoder-decoder model



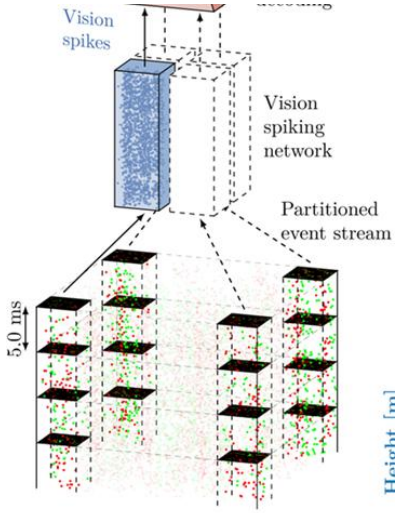
  - Accuracy comparable to ANNs
    The model size is too large to be directly implemented on a neuromorphic chip

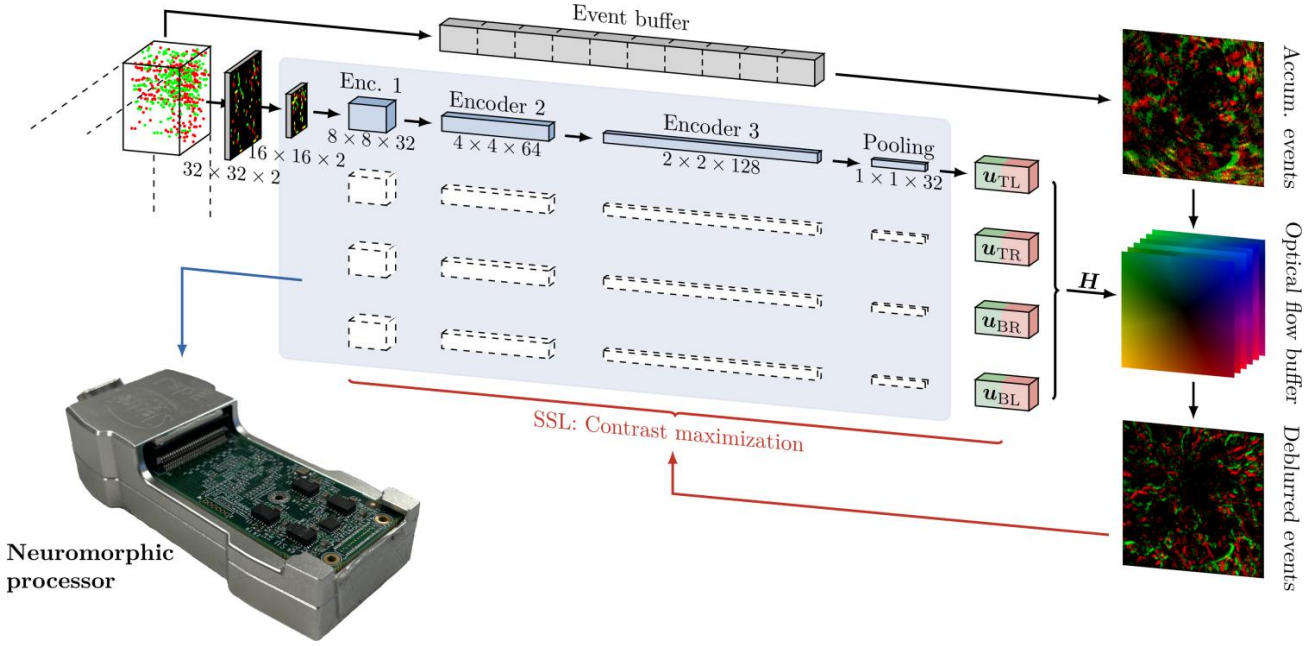
  - Techniques to reduce the network size are required

J Hagenaars, F Paredes-Vall´es, and G De Croon, ”self-supervised learning of event-based optical flow with spiking neural networks”, Advances in Neural Information Processing Systems, Vol. 34, pp. 7167–7179, 2021.

3. Neuromorphic Control using Event-based Vision

- **Vision Network Architecture**
  - **Input (Event camera) / Event buffer**
    - Uses corner regions instead of the entire camera image
    - Temporal accumulation of events to create an event grid
  - **Encoder Network** (SNN)
    - Output the feature vector (Encoder×3+Pooling)
  - **Optical flow estimation**
    - Estimating local optical flow from feature vectors
    - Combine four local optical flows to create optical flow buffer.
  - **Output**
    - Calculating egomotion from optical flow buffer





3. Neuromorphic Control using Event-based Vision

Controller (Linear map, 9×4)

[3DF velocity, 3DoF acceleration, Pitch, Roll, Target velocity]

↓

[Thrust, Roll, Pitch, Yaw]

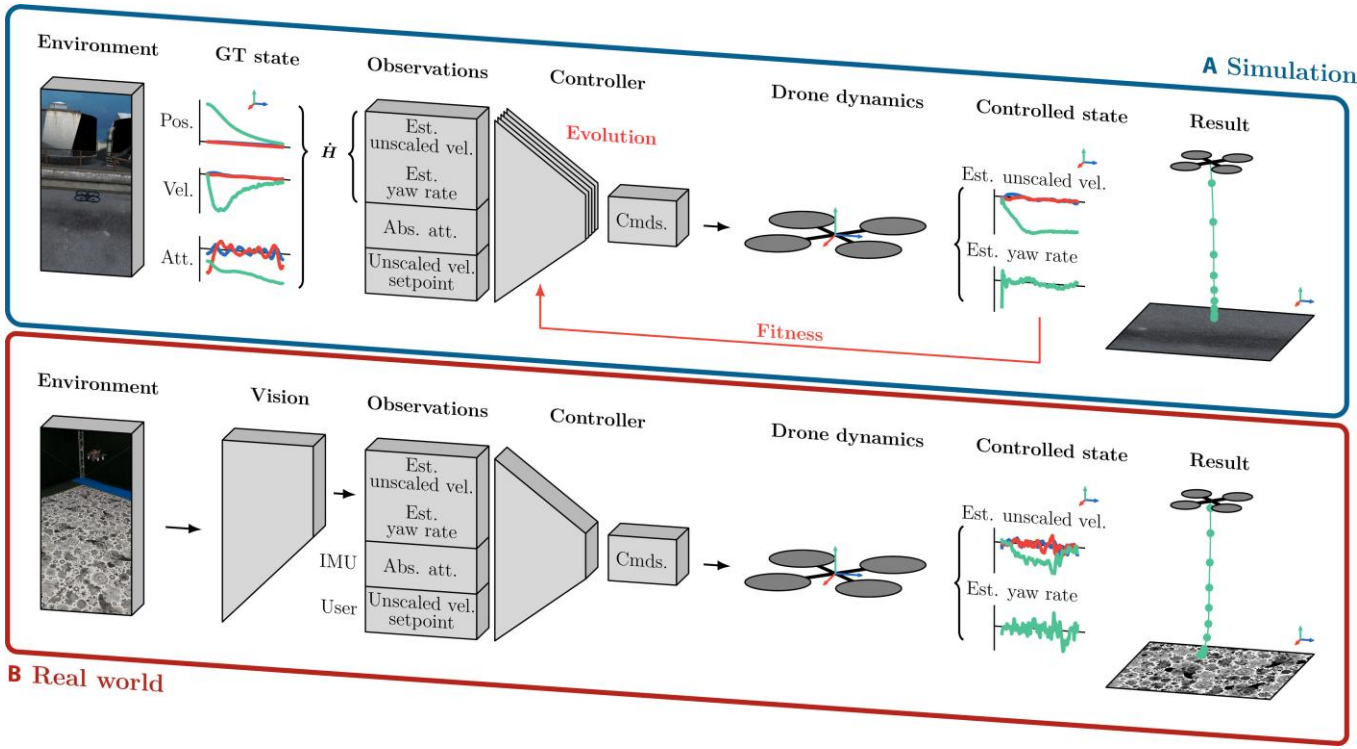
Evolutionary algorithm

×25000

100 random 9×4 matrix

↓ ← Add $N(0, 0.001)$ noise to increase the variety

Select the 100 items with the highest fitness



Fitness function

$$F = \frac{1}{N_{\text{eval}}} \sum_{i \in N_{\text{eval}}} \sum_{j \in N_{\text{steps}}} \mathbf{w} \cdot \left(v^{\mathcal{B}}_{\text{sp},i} - \begin{bmatrix} \hat{V}^{\mathcal{B}}_x \\ \hat{V}^{\mathcal{B}}_y \\ \hat{V}^{\mathcal{B}}_z \end{bmatrix}_j \right)^2 + \left(\hat{\omega}^{\mathcal{B}}_z \right)^2$$

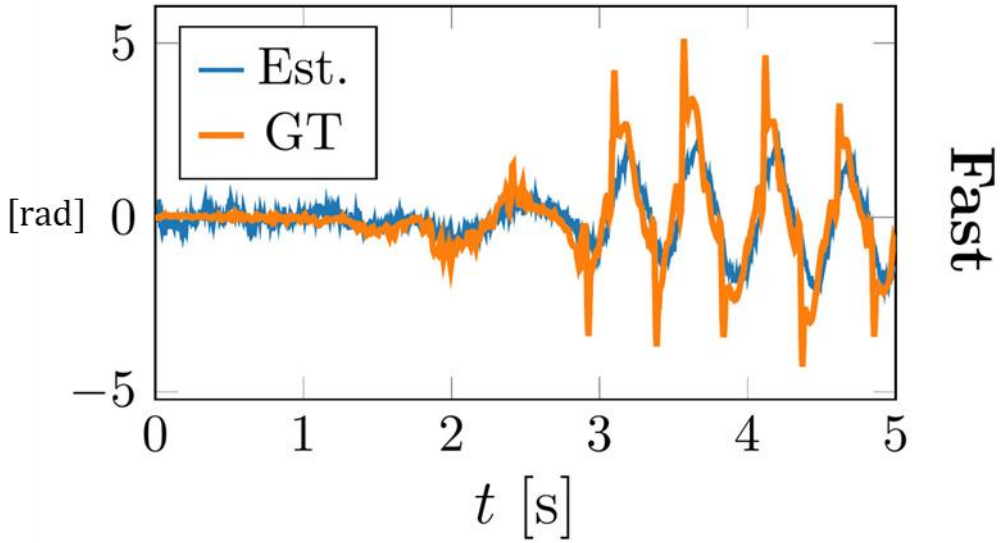
Translational velocity error　Yaw penalty

3. Neuromorphic Control using Event-based Vision

Result : Vision network

- **Condition**
  - Training data : Event sequence in a real experimental environment
  - GT : Records of the drone's position and attitude during event data collection

- **Yaw rotation estimation**
  - It was able to accurately follow
    even at a high rotation speed of 4 rad/s.

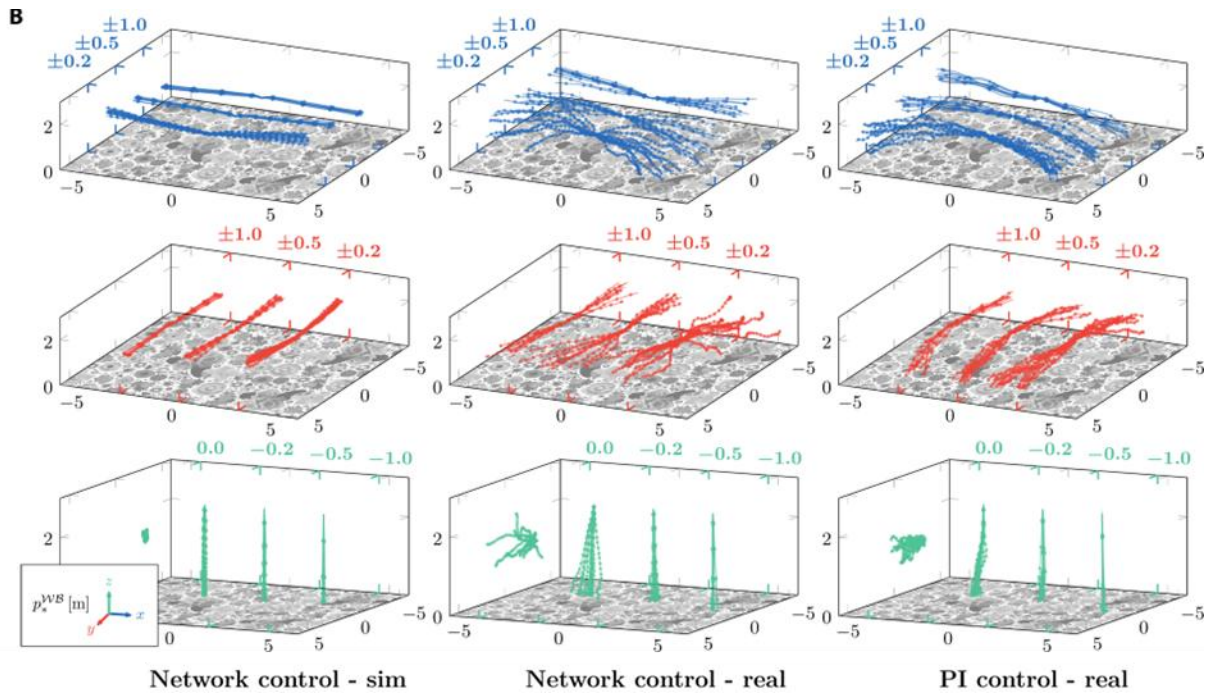


3. Neuromorphic Control using Event-based Vision

- **Result : Evaluation of control**

- **Simulation**
  - Success in all flight patterns

- **Real**
  - Failed in many patterns
    - Reality gap
    - Insufficient integral control



trajectory

- **Trajectory stability at high speed**
  - Reduction in noise ratio by increasing the number of events

3. Neuromorphic Control using Event-based Vision

- **Power Consumption**
  - Comparison of SNN execution devices
    - Loihi : 0.95 W / Jetson : 2.98 W

  → Power saving achieved by neuromorphic chips.

  → Achieving greater power savings by reducing **idle power consumption**.

- **Inference speed**
  - SNN : 1637 [inf/s] / ANN : 83 [inf/s]

  → SNNs enable significantly **faster inference**.

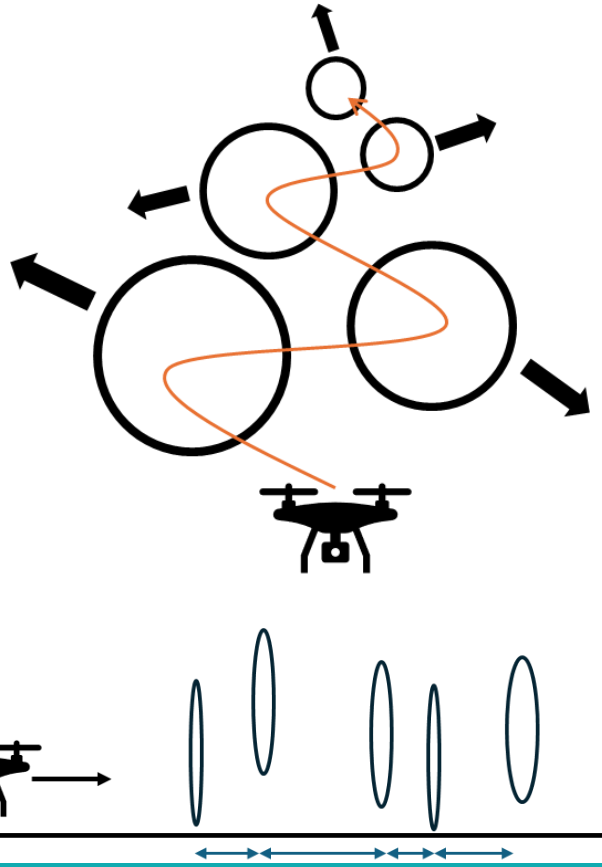
Device	Seq.	Static (W)	Dynamic (W)	Idle (W)	Running (W)	Delta (W)	inf/s	mJ/inf
Nahuku 32 (empty)	Any	0.86	0.04	0.90	0.90	4×10^{-3}	60,496	71×10^{-6}
3*Nahuku 32: SNN	Slow	0.90	0.05	0.94	0.95	12×10^{-3}	1,637	7×10^{-3}
	Medium	0.90	0.04	0.94	0.95	8×10^{-3}	411	21×10^{-3}
	Fast	0.90	0.04	0.94	0.95	7×10^{-3}	274	27×10^{-3}
3*Jetson Nano: SNN (10 W)	Slow	–	–	1.04	2.98	1.93	26	75.25
	Medium	–	–	1.06	2.98	1.92	26	75.35
	Fast	–	–	1.04	2.99	1.95	25	76.52
3*Jetson Nano: ANN (10 W)	Slow	–	–	1.04	3.30	2.27	83	27.36
	Medium	–	–	1.07	3.33	2.26	80	28.09
	Fast	–	–	1.07	3.30	2.23	80	27.80

4. Agile Navigation using Reinforcement Learning-Trained SNN

Paper3
Y-C. Lee, S. Mengozzi, L. Zanatta, A. Bartolini, A. Acquaviva, and F. Barchi,
”Bio-inspired drone control: A reinforcement learning trained spiking neural networks for agile navigation in dynamic environment”, In 2025 IEEE International Conference on Omni layer Intelligent Systems (COINS), pp. 1–8. IEEE, 2025.

- **Key Point**
  - Control in **dynamic environments** using SNN flight policies trained with **deep reinforcement learning** algorithms

- **Simulation**
  - Task : Pass through 5 consecutive rapid-transit gates
  - Situation :
    - Drone initial speed : 0 ~ 10 m/s
    - Distance between gates : 0.5 ~ 9 m
    - Gates angle : $-\pi \sim \pi$
    - Gates oscillates in the YZ plane

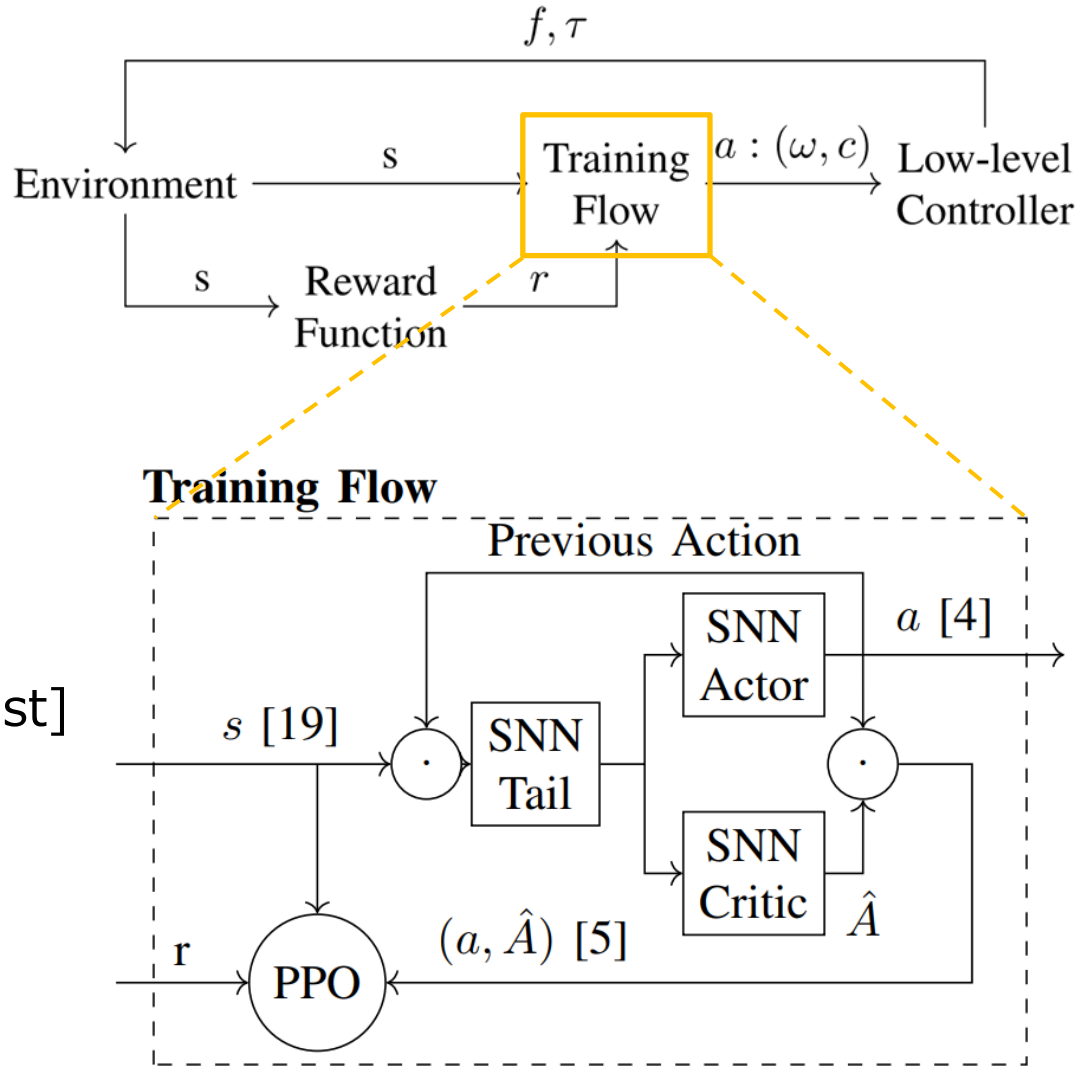


4. Agile Navigation using Reinforcement Learning-Trained SNN

- **Network Architecture (SNN agent)**
  - Input ($\boldsymbol{s}$):

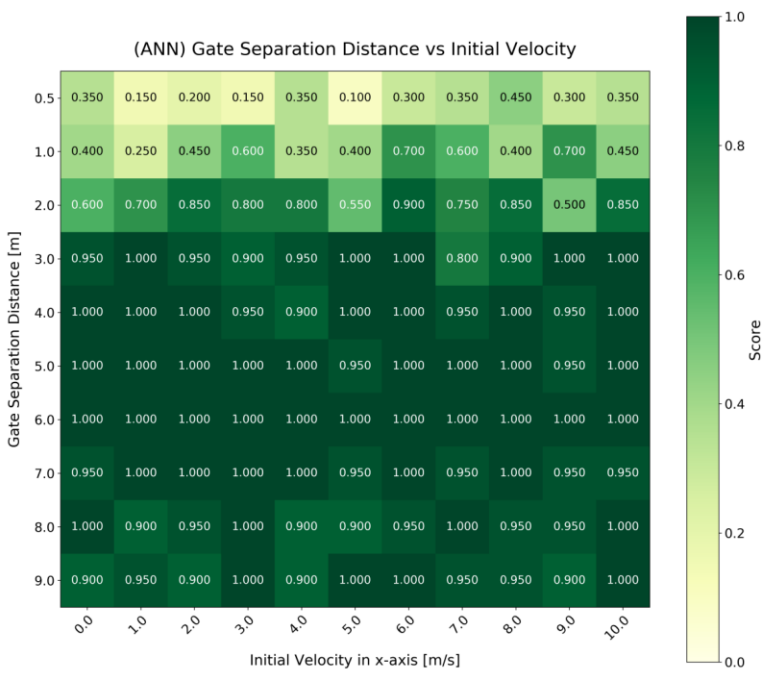
d_{tgt}	Distance between the drone and the goal
q	Attitude
v	Velocity
ω	Angular velocity
d_{gate}	Distance between the drone and the center of the next gate
v_{gate}	Gates velocity

  - Output : [Body Rate (Angular velocity), Thrust]
  - Model size :
    - Input layer : 19 neurons
    - Hidden layer : 1024 neurons × 2
    - Output layer : 4 neurons

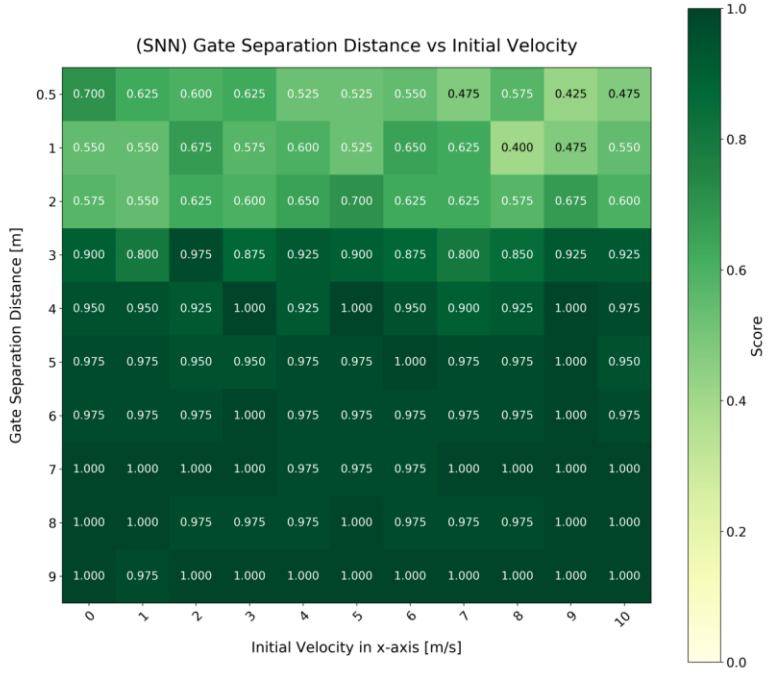


4. Agile Navigation using Reinforcement Learning-Trained SNN

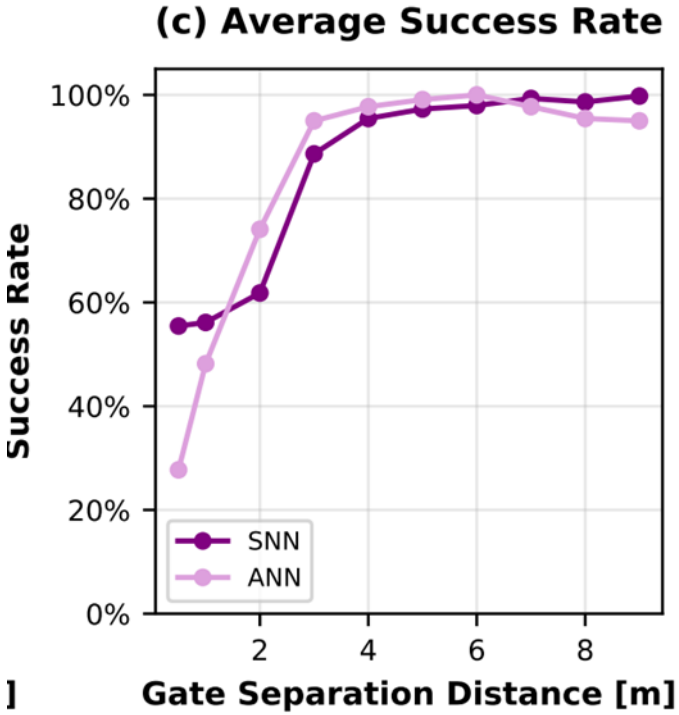
- **Result : Gate passing success rate (ANN vs SNN)**



(a) State-of-the-Art's Performance



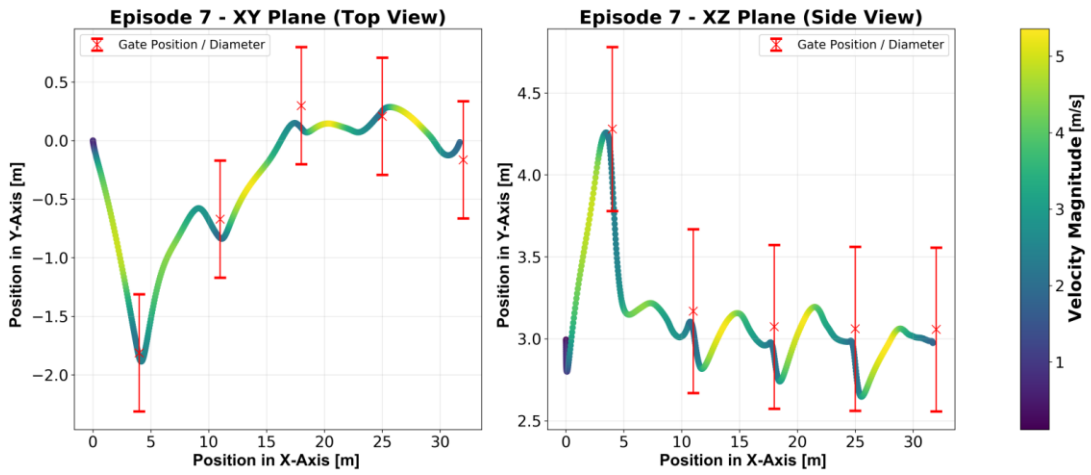
(b) SNN's Performance



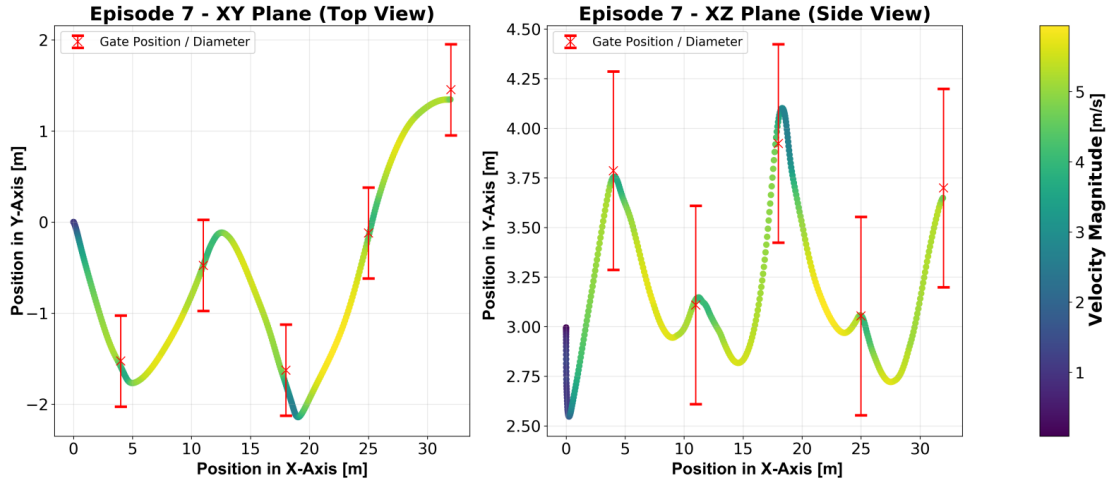
- When the gate separation distance is short, SNN is superior
  - This high success rate is due to the SNN's **temporal processing characteristics**

4. Agile Navigation using Reinforcement Learning-Trained SNN

Result : Comparison of trajectories (ANN vs SNN)



(a) State-of-the-Art ANN Trajectory



(b) SNN trajectory

- SNN creates **smoother trajectories**
- SNN consistently maintains **higher speeds** throughout the trajectory

- SNN can complete tasks efficiently in a short amount of time
  - This leads to reduced power consumption

5. Conclusion

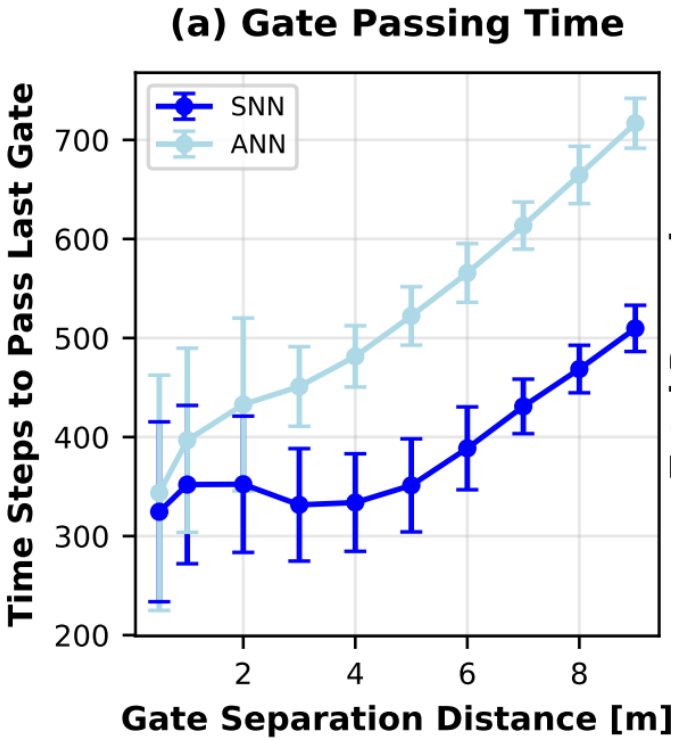
This investigation focused on two applications
low-level control (attitude control) and **path planning**

- **low-level control**
  - Paper 1 (IMU)
    - Attitude control using SNN with only an IMU is currently less useful than PID control, and integration with image recognition and other technologies is necessary
  - Paper 2 (Event camera)
    - There is a problem that the control accuracy deteriorates in response to the demand for reducing the model size.
- **Path-planning**
  - The challenge remains in designing an optimal reward function for real environments.
- **Towards Mars exploration**
  - While drone control using SNNs has the advantage of reducing power consumption, there are many issues that need to be resolved when applying it to real environments such as Mars.

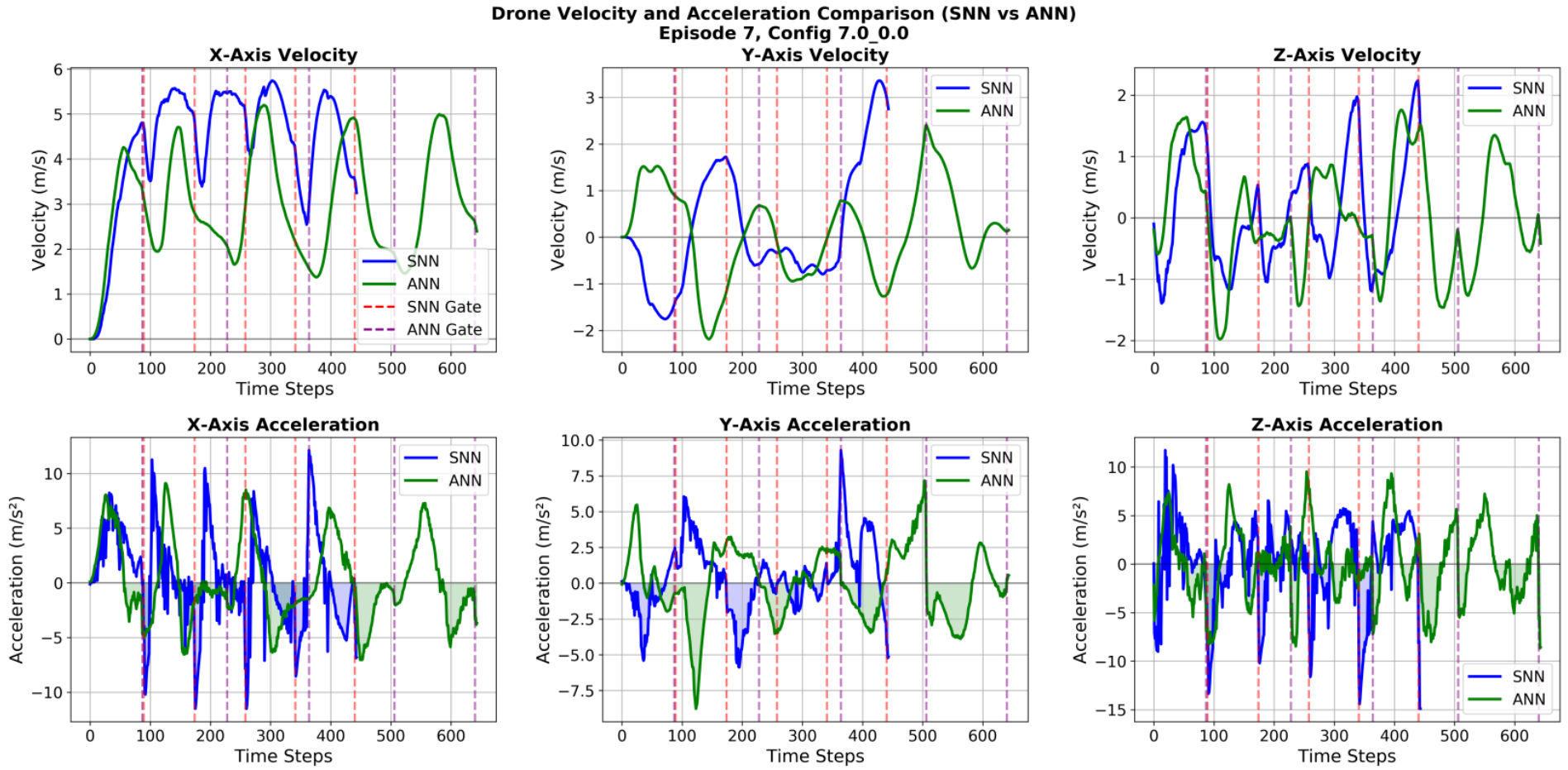
Appendix

4. Agile Navigation using Reinforcement Learning-Trained SNN

- **Result : Completion time**



4. Agile Navigation using Reinforcement Learning-Trained SNN



2. Neuromorphic Attitude Estimation and Control using IMU

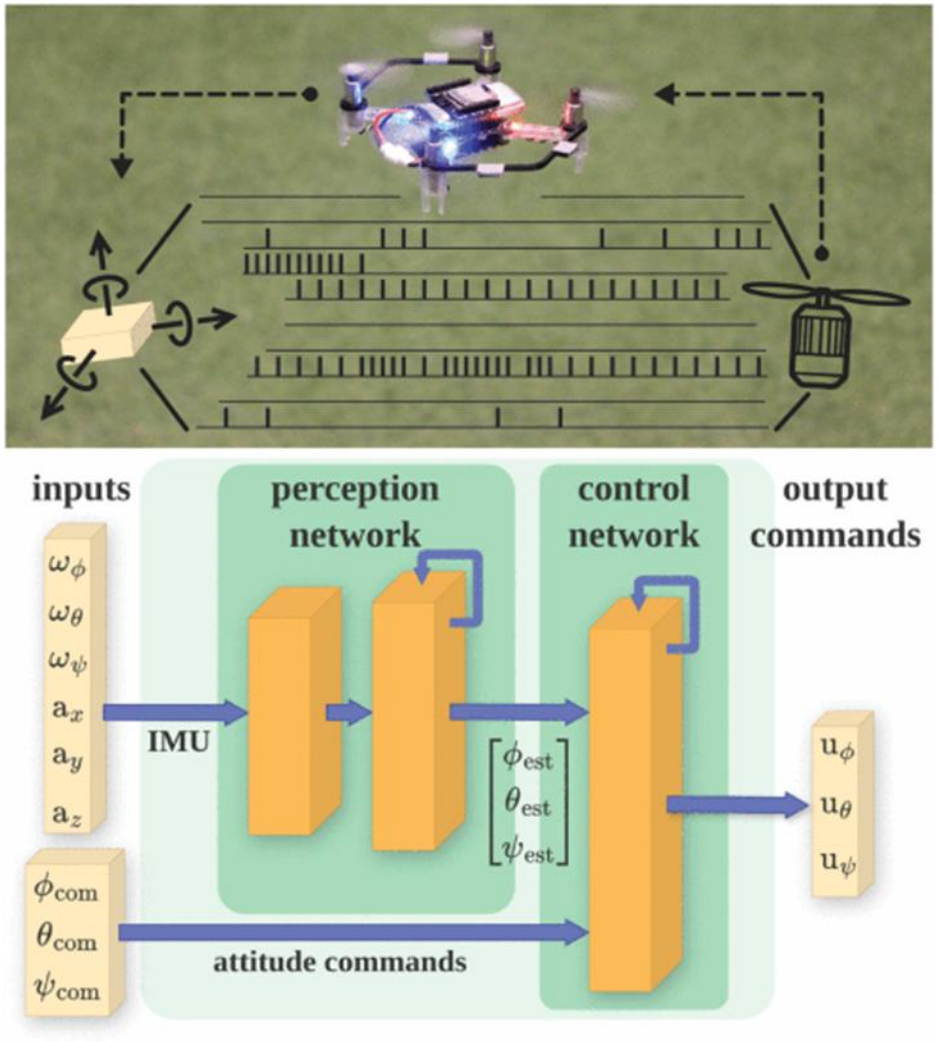
Paper1
S. Stroobants, C. De. Wagter, and G. CHE. De. Croon,
”Neuromorphic attitude estimation and control”,
IEEE Robotics and Automation Letters, (2025).

- **Key point**
  - End-to-end SNN for attitude estimation and control for real-world micro drone
- **Solution**
  - Division into two subnetworks
  - Introduced practical techniques
    1. Time-shifted Targets
    2. Integrator Neurons
    3. Data Augmentation

2. Neuromorphic Attitude Estimation and Control using IMU

Proposed Network Specifications

Item	Details
Neuron model	CUBA-LIF
Architecture	Perception : 2-layer (recurrent) Control : 1-layer (recurrent)
Number of Nerons	Training : 150-150-130 Inference : 150-100-80
Learning Algorithm	BPTT (Backpropagation Through Time)

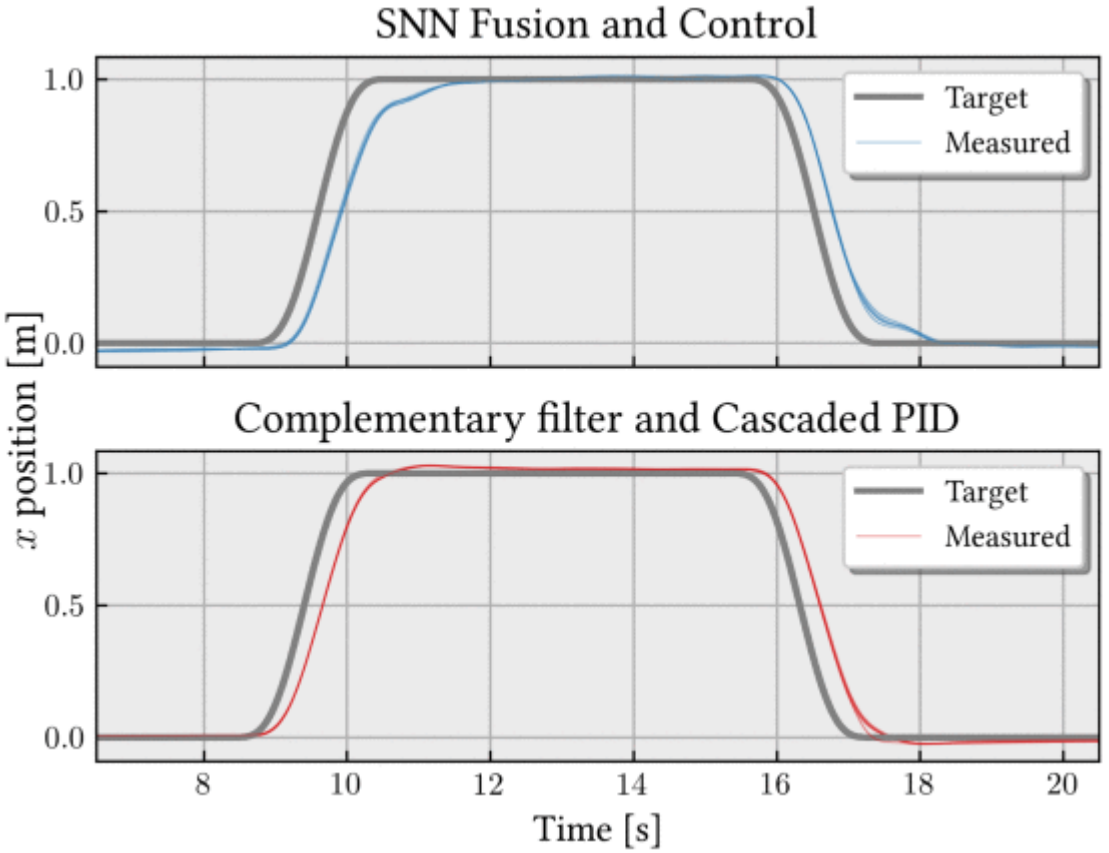


- Training data collection
  - Collected from 20 minutes of manual flight
    - Attitude : complementary filter
    - Control : cascaded PID controller

- Spike ⇔ Real Values
  - Transformation using a Weight Matrix

2. Neuromorphic Attitude Estimation and Control using IMU

1. Single-axis position control



SNN can achieve stable target tracking with performance comparable to PID control

2. Neuromorphic Attitude Estimation and Control using IMU

Power Consumption estimation

- Number of addition operation (1 step)
  - PID : **3000** times / step
  - SNN : **7500** times / step

  - Energy efficient is expected to be in the **same order**

- **Potential for further power consumption reduction**
  1. Integration with vision tasks
     - Highly compatible with the reduction in multiplications enabled by spike sparsity
  2. Reduction in states like hovering
     - Network sparsity increases when control is not applied

3. Neuromorphic Control using Event-based Vision

Paper2
F Paredes-Vall´es, J J Hagenaars, J Dupeyroux, S Stroobants, Y Xu, and G CHE de Croon,
”fully neuromorphic vision and control for autonomous drone flight”,
Science Robotics, Vol. 9, No. 90, eadi0591, (2024)

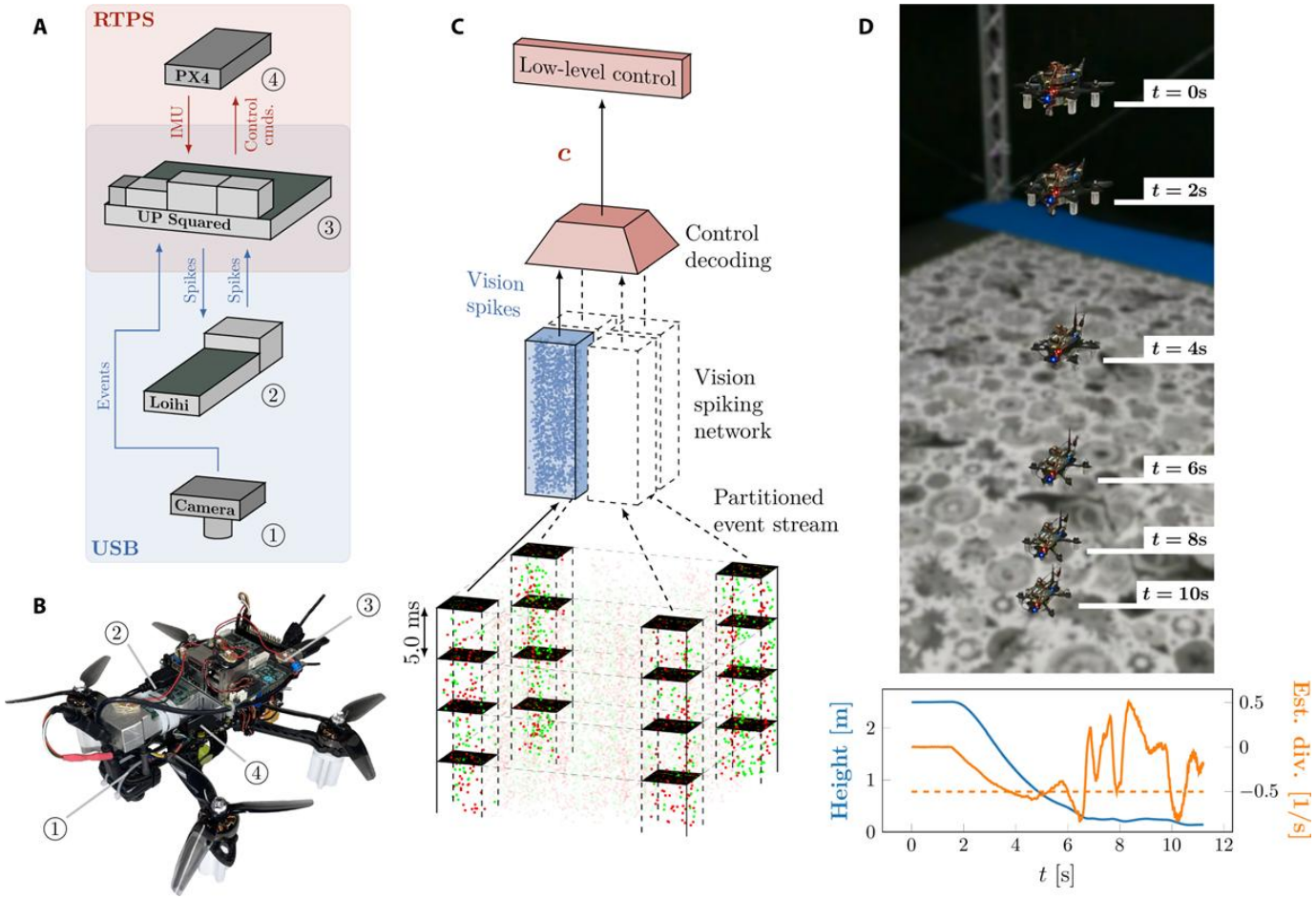
- **Key point**
  - Vision-based autonomous flight entirely on a neuromorphic chip

- **Solution**
  - Implemented a full pipeline on the Loihi chip
    1. **Ego-motion estimation** (optical flow) from event camera data
    2. Velocity control (trained by a genetic algorithm)

3. Neuromorphic Control using Event-based Vision

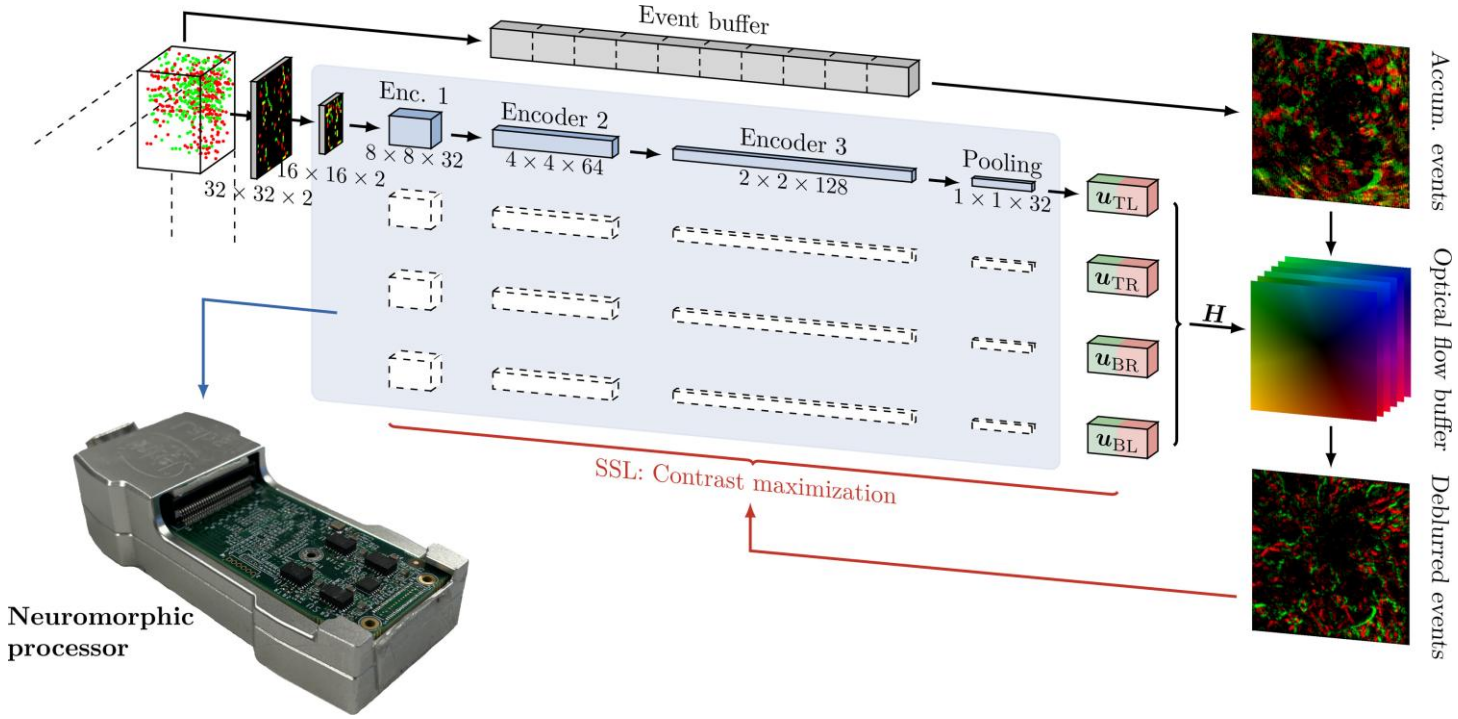
- Proposed system architecture
  - Vision Network (Optical flow)
  - Control decoder (Linear map)

- Network size reduction
  - The floor is a
    flat surface with a rich texture
  - Limit the number of input event
    (90 per corner ROI)



3. Neuromorphic Control using Event-based Vision

Vision Network Architecture



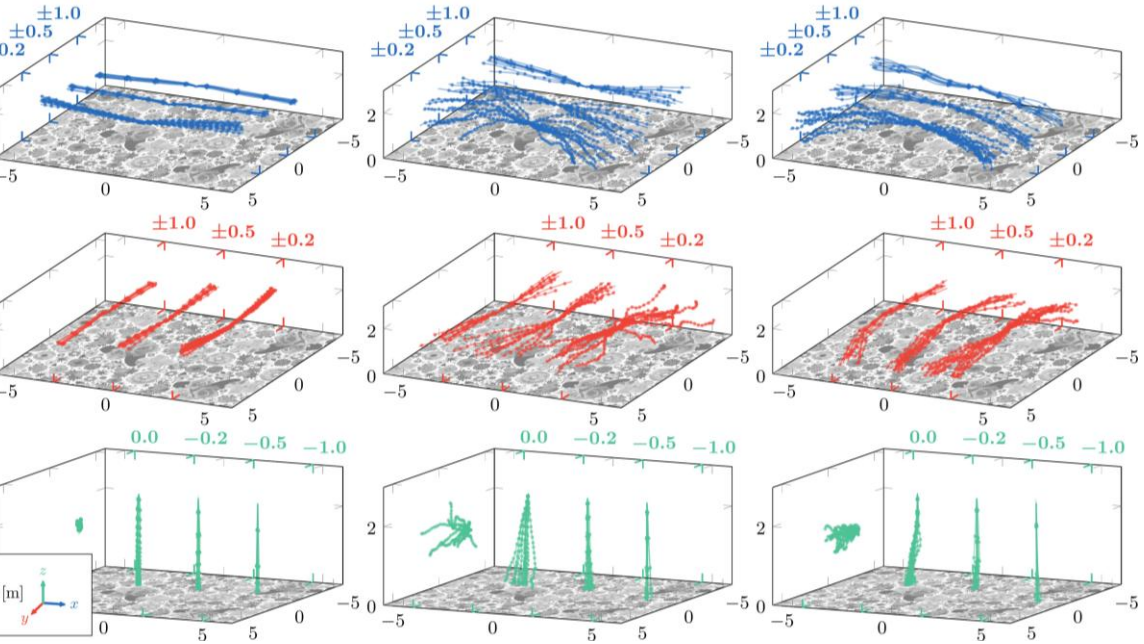
Optical flow estimates

$$\mathbf{u}(\mathbf{x},\mathbf{H}) = \begin{bmatrix} u(\mathbf{x},\mathbf{H}) \\ v(\mathbf{x},\mathbf{H}) \end{bmatrix} = \left(\mathbf{H}, \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \right)_{\text{Eucl}} - \begin{bmatrix} x \\ y \end{bmatrix}$$

H : holography matrix

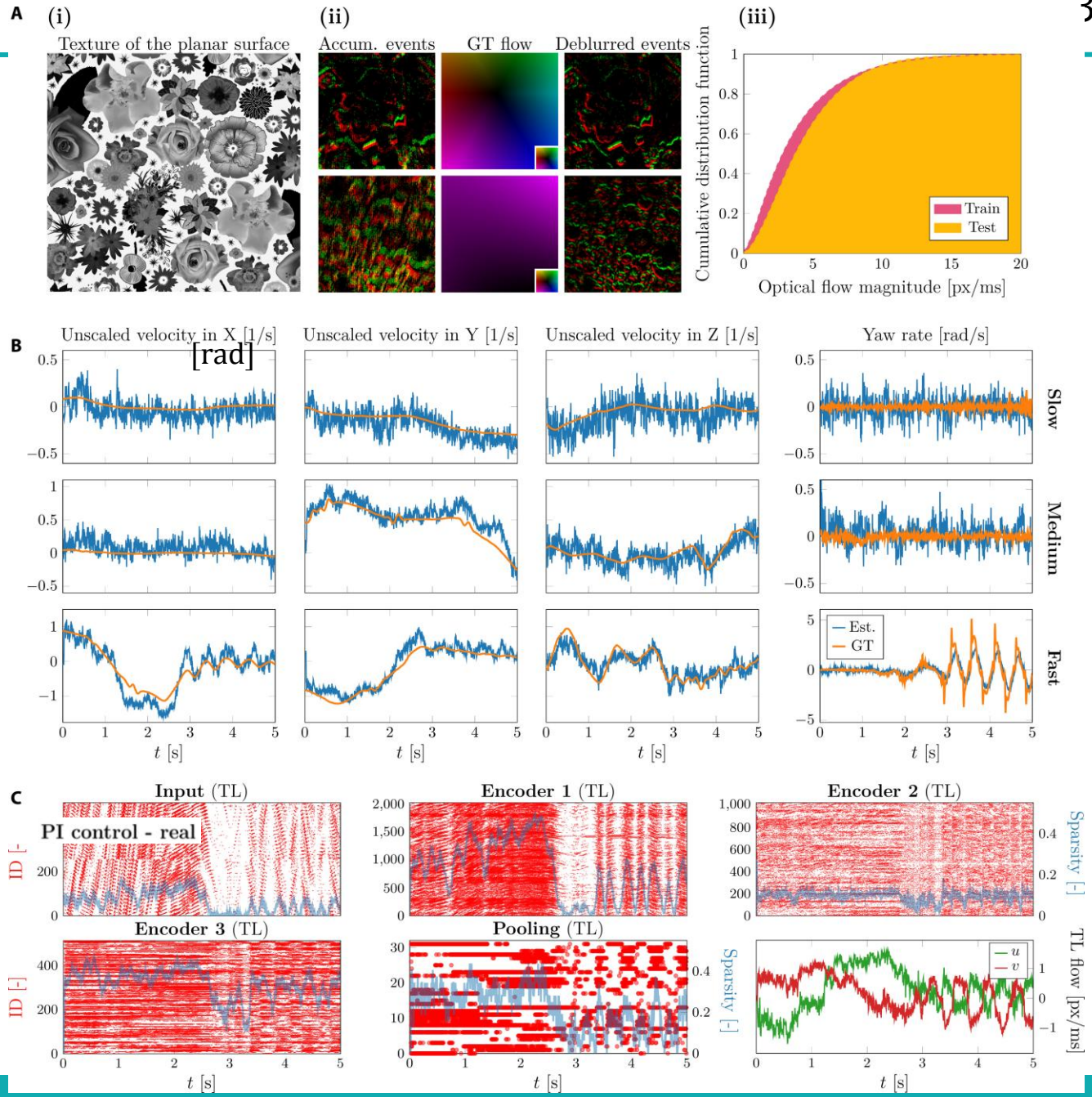
$$L_{\text{flow}} = L_{\text{contrast}} + \lambda L_{\text{smooth}}$$

$$\text{EPE} = \frac{1}{N} \sum_{i \in I}^{N} ||\boldsymbol{u}_i - \boldsymbol{u}_i^{\text{GT}}||$$



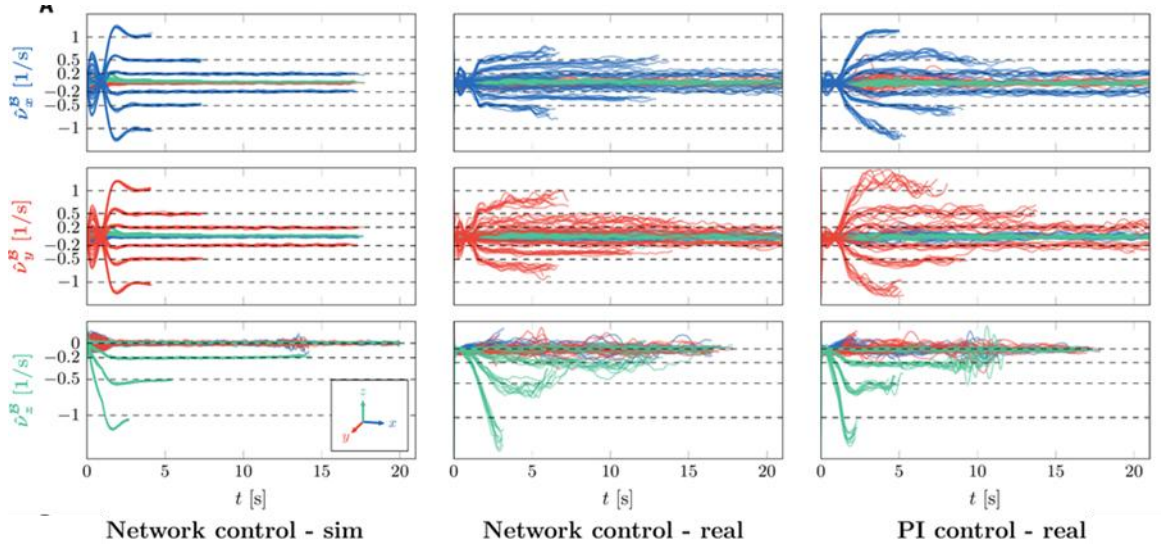
Network control - sim

Network control - real

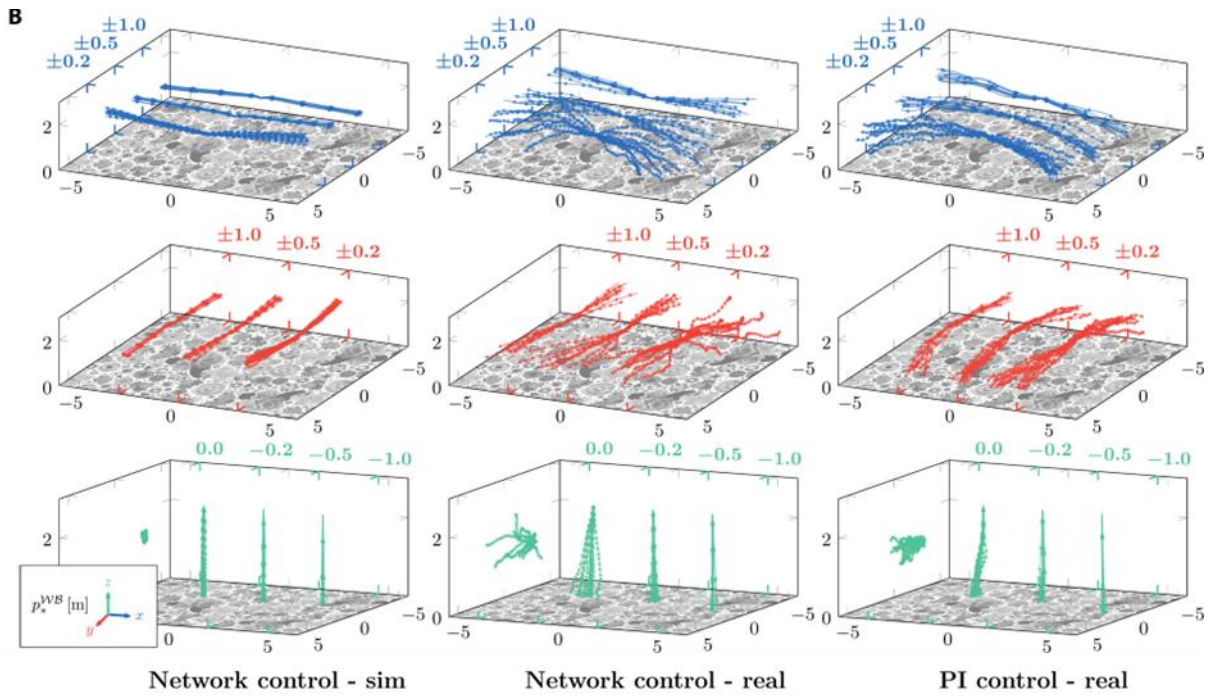


3. Neuromorphic Control using Event-based Vision

Result : Evaluation of control



velocity



trajectory

- Simulation
  - Success in all flight patterns
- Real
  - Failed in many patterns
    Cause → Reality gap / Insufficient integral control
- Trajectory stability at high speed
  - Reduction in **noise ratio** by increasing the number of events

Device	Seq.	Static (W)	Dynamic (W)	Idle (W)	Running (W)	Delta (W)	inf/s	mJ/inf
Nahuku 32 (empty)	Any	0.86	0.04	0.90	0.90	4×10^{-3}	60,496	71×10^{-6}
3*Nahuku 32: SNN	Slow	0.90	0.05	0.94	0.95	12×10^{-3}	1,637	7×10^{-3}
	Medium	0.90	0.04	0.94	0.95	8×10^{-3}	411	21×10^{-3}
	Fast	0.90	0.04	0.94	0.95	7×10^{-3}	274	27×10^{-3}
3*Jetson Nano: SNN (5 W)	Slow	–	–	1.05	2.23	1.18	14	86.11
	Medium	–	–	1.03	2.25	1.22	14	85.58
	Fast	–	–	1.03	2.24	1.21	14	86.19

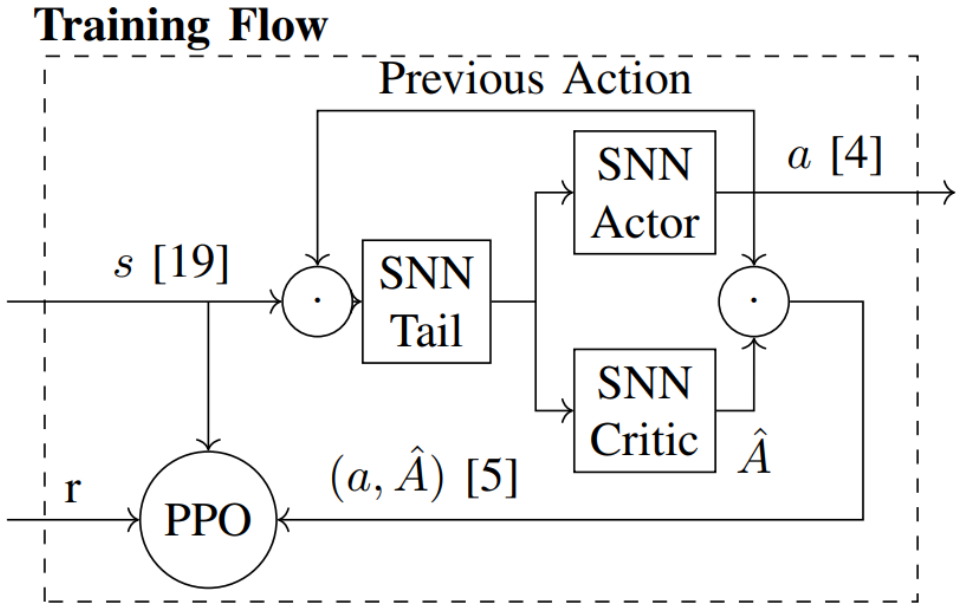
Device	Seq.	Static (W)	Dynamic (W)	Idle (W)	Running (W)	Delta (W)	inf/s	mJ/inf
3*Jetson Nano: SNN (10 W)	Slow	–	–	1.04	2.98	1.93	26	75.25
	Medium	–	–	1.06	2.98	1.92	26	75.35
	Fast	–	–	1.04	2.99	1.95	25	76.52
3*Jetson Nano: ANN (5 W)	Slow	–	–	1.05	2.66	1.61	59	27.46
	Medium	–	–	1.07	2.64	1.57	56	27.91
	Fast	–	–	1.06	2.64	1.58	57	27.52
3*Jetson Nano: ANN (10 W)	Slow	–	–	1.04	3.30	2.27	83	27.36
	Medium	–	–	1.07	3.33	2.26	80	28.09
	Fast	–	–	1.07	3.30	2.23	80	27.80

4. Agile Navigation using Reinforcement Learning-Trained SNN

- **Training Flow**
  - Input : state of system

$$s \in \mathbb{R}^{19} := (d_{tgt}, q, v, \omega, d_{gate}, v_{gate})$$

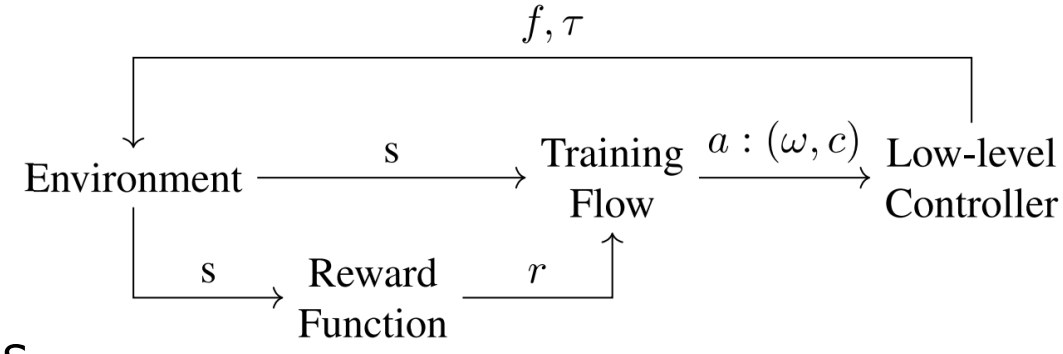
  - Output : Thrust, Body rates
  - SNN model architecture
    - Hidden layer (1024 neurons) ×2



4. Agile Navigation using Reinforcement Learning-Trained SNN

- **Framework of Reinforcement Learning**
  - In the proposed framework...
    - Physical simulation environment
    - SNN agent
    - Low-level controller

  → A task of passing through multiple **moving** rings



- **Proximal Policy Optimization(PPO)**

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]$$

Action probability update　　Clip function

- Clipping the policy to **prevent large changes**

  → **Stable learning** is achievable

4. Agile Navigation using Reinforcement Learning-Trained SNN

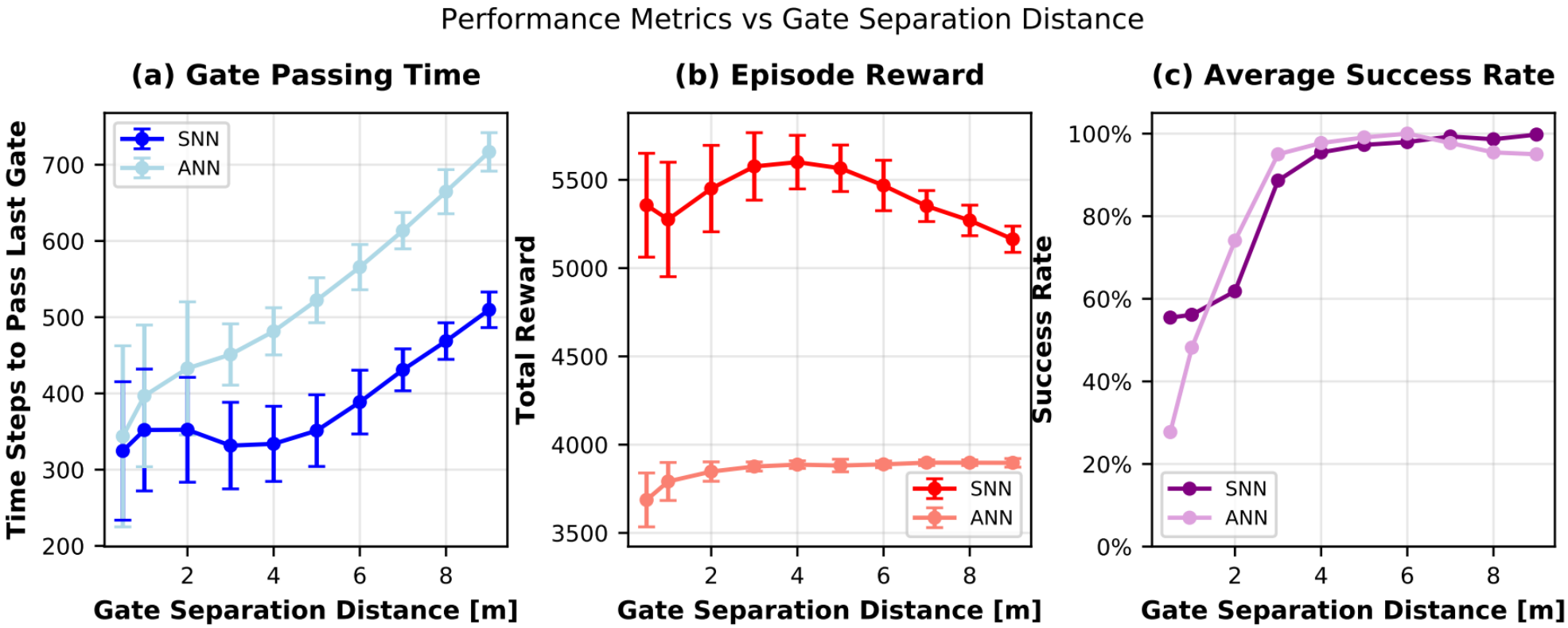
- Reward function

$$r_t = r_{gate,t} + r_{pos,t} + r_{pos,t} \cdot (r_{up,t} + r_{spin,t}) - \beta \cdot r_{acc,t}$$

Gate passing reward　Position reward　Horizontal reward　Rapid acceleration penalty

$$r_{pos,t} = \frac{0.05}{1 + d_{tgt}^2} + \frac{2.5}{1 + d_{gate}^2} - \frac{2.0}{1 + d_{ring}^2}$$

  - Reward
    1. Close to final goal point
    2. Close to the center of the gate
  - Penalty
    1. Close to the ring (boundary)

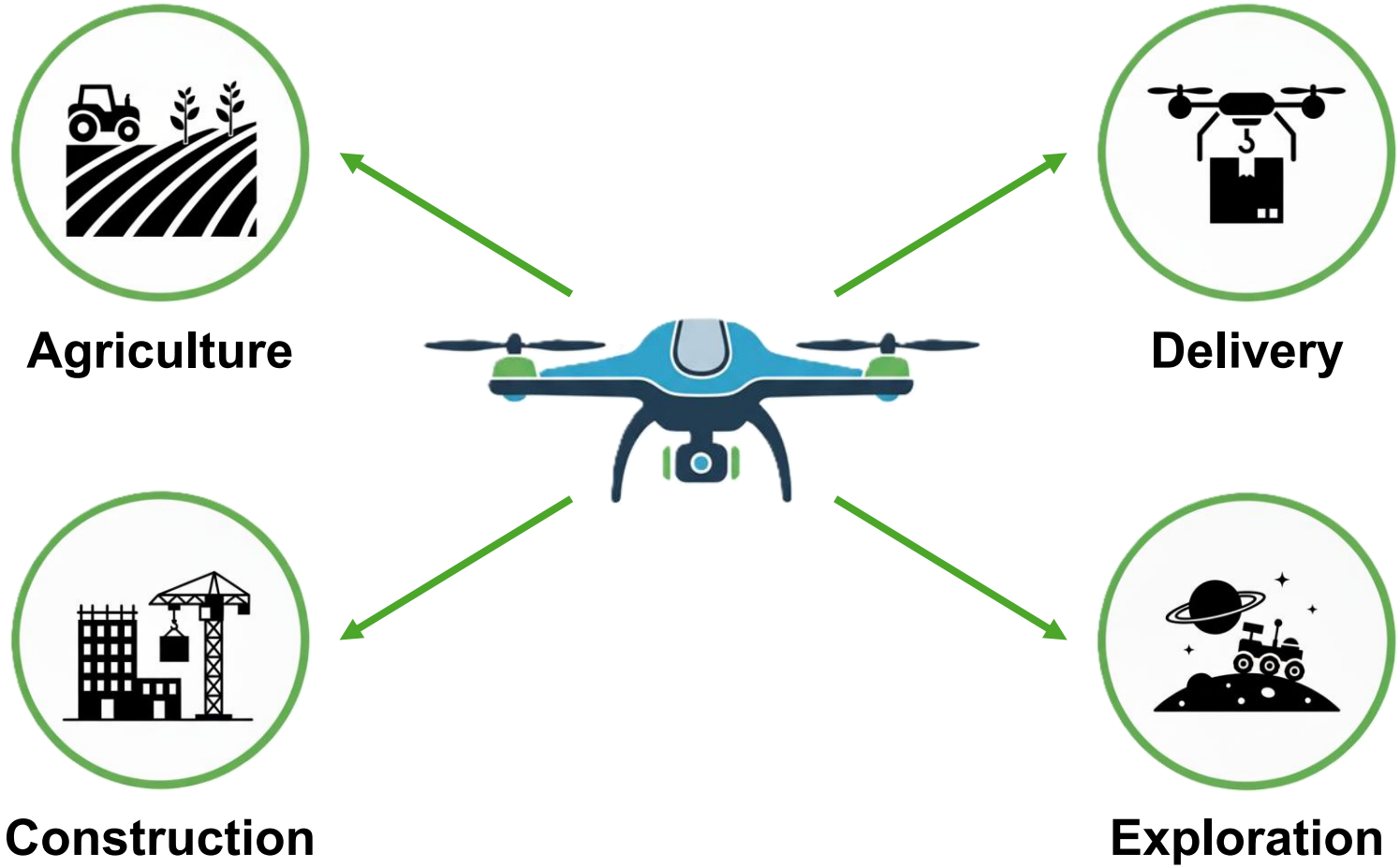

Performance Metrics vs Gate Separation Distance
(a) Gate Passing Time
Time Steps to Pass Last Gate
Gate Separation Distance [m]
SNN
ANN
(b) Episode Reward
Total Reward
Gate Separation Distance [m]
SNN
ANN
(c) Average Success Rate
Success Rate
Gate Separation Distance [m]
SNN
ANN

4. Agile Navigation using Reinforcement Learning-Trained SNN

- **Power Consumption**
  - Number of neurons
    - Input layer : 23
    - Hidden layer : 2048 (1024×2)
    - Output layer : 4
  - Firing rate : 15~20 %

  → Number of operation : 215,000

- **Memory usage**
  - float32 : 4.3 MB
  - Int8 : 1.08 MB

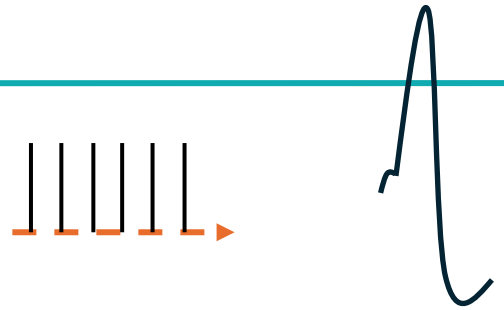

Agriculture
Delivery
Construction
Exploration

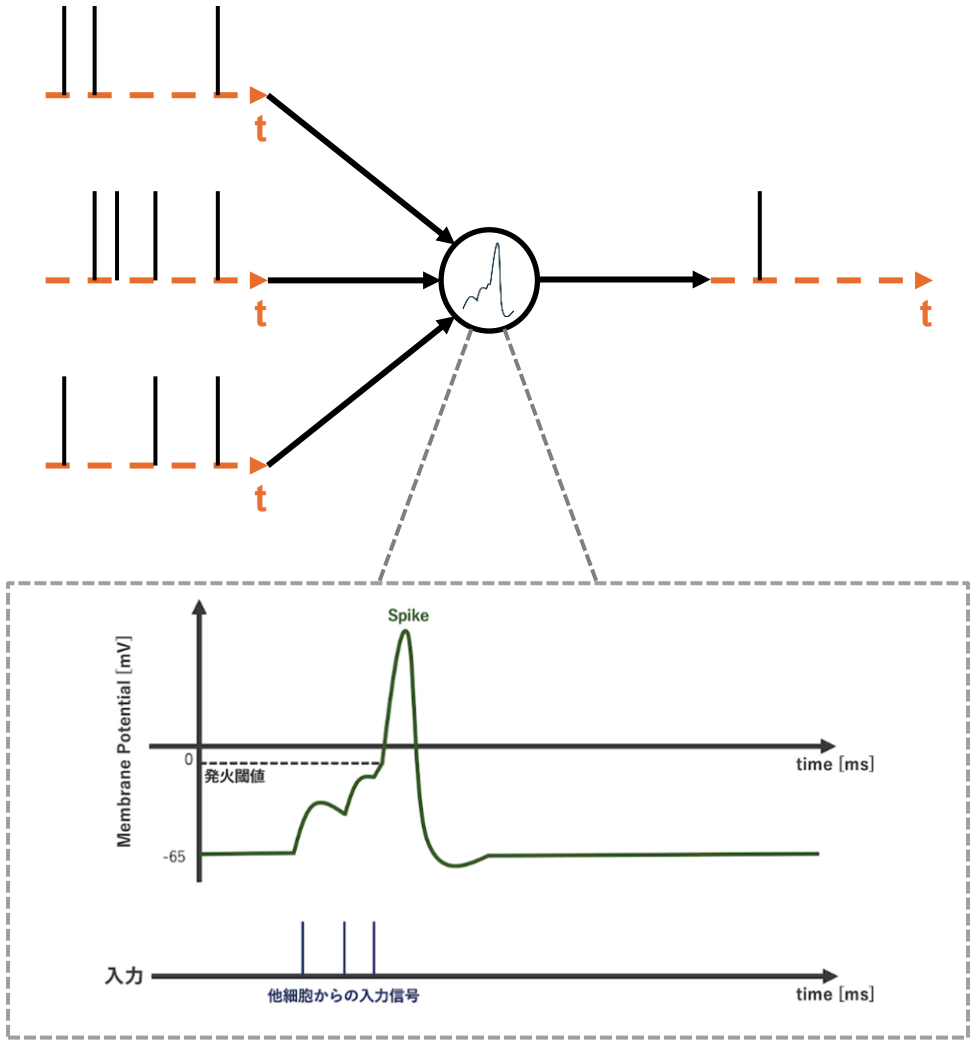
On-board

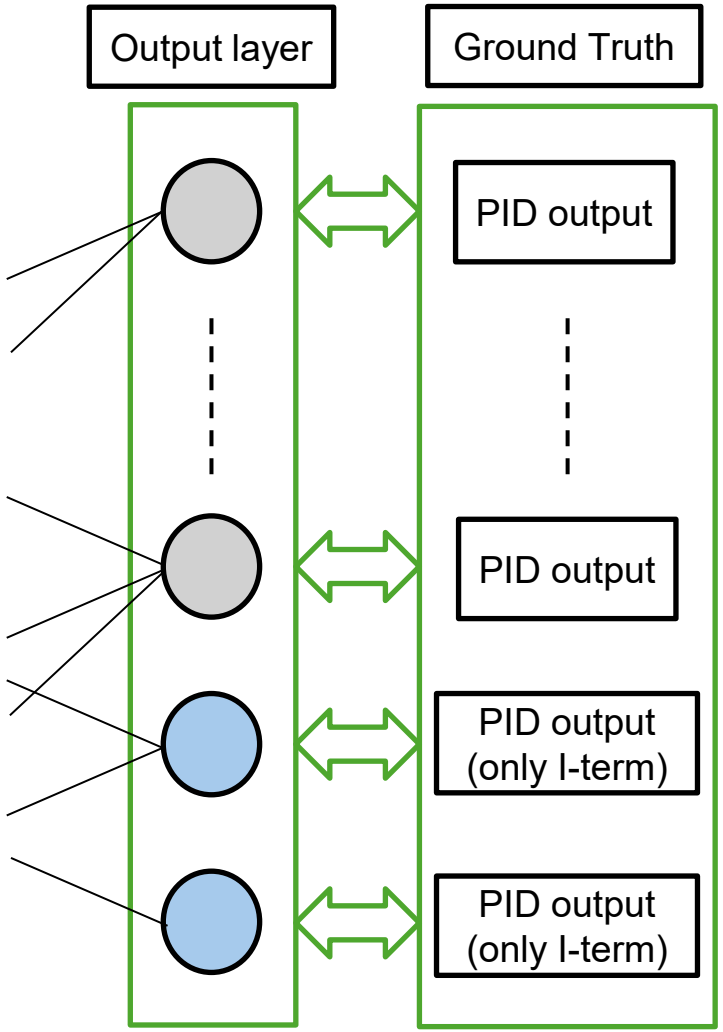
- ○ Real-time processing
- × Power consumption
- × Inference accuracy

Ground-base

- ○ Inference accuracy
- × Real-time processing

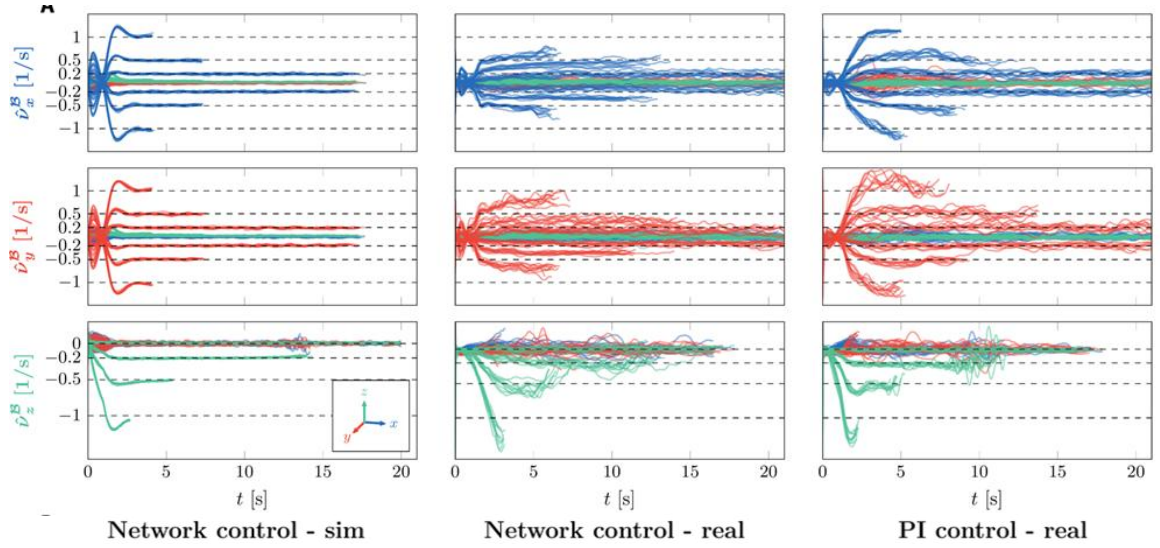



t
t
t
t
Spike
Membrane Potential [mV]
0
発火閾値
-65
time [ms]
入力
他細胞からの入力信号
time [ms]



$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}$$

y_i : Ground truth,
$\hat{y}_i$: Predicted value,
n : total data count


A
$\hat{v}_x^B$ [1/s]
$\hat{v}_y^B$ [1/s]
$\hat{v}_z^B$ [1/s]
t [s]
Network control - sim
Network control - real
PI control - real