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

# Agile in the Face of Delay: Asynchronous End-to-End Learning for Real-World Aerial Navigation

Yude Li†, Zhexuan Zhou†, Huizhe Li, Youmin Gong* and Jie Mei*

***Abstract*— Robust autonomous navigation for Autonomous Aerial Vehicles (AAVs) in complex environments is a critical capability. However, modern end-to-end navigation faces a key challenge: the high-frequency control loop needed for agile flight conflicts with low-frequency perception streams, which are limited by sensor update rates and significant computational cost. This mismatch forces conventional synchronous models into undesirably low control rates. To resolve this, we propose an asynchronous reinforcement learning framework that decouples perception and control, enabling a high-frequency policy to act on the latest IMU state for immediate reactivity, while incorporating perception features asynchronously. To manage the resulting data staleness, we introduce a theoretically-grounded Temporal Encoding Module (TEM) that explicitly conditions the policy on perception delays, a strategy complemented by a two-stage curriculum to ensure stable and efficient training. Validated in extensive simulations, our method was successfully deployed in zero-shot sim-to-real transfer on an onboard NUC, where it sustains a 100 Hz control rate and demonstrates robust, agile navigation in cluttered real-world environments. Our project details are available at `https://hitsz-mas.github.io/Agile-Asynch-Nav/`.**

## I. INTRODUCTION

Robust and agile autonomous navigation is crucial for the effective deployment of Autonomous Aerial Vehicles (AAVs) in complex and unstructured environments. Conventional approaches typically rely on a modular pipeline that separates the navigation task into distinct stages of perception [1], planning [2], and control [3]. While well-established, this architecture is prone to inter-module error propagation and significant decision-making latency, which can compromise flight safety and efficiency. In response to these challenges, end-to-end learning frameworks, particularly those leveraging reinforcement learning (RL) [4], [5], have emerged as a promising alternative. By directly mapping sensory inputs to control actions, these methods can overcome the limitations of modular systems, thereby reducing latency and improving robustness.

However, the practical deployment of these end-to-end models on computationally constrained AAVs introduces a significant challenge. Unlike platforms with substantial onboard processing capabilities, such as legged robots [6] or autonomous vehicles [7], AAVs face a critical resource contention. They must arbitrate between processing high-bandwidth sensor data and maintaining the high-frequency control loops essential for agile flight. This contention manifests as a fundamental temporal mismatch: while high-frequency state information from an Inertial Measurement Unit (IMU) is readily available, the perception stream is simultaneously constrained by both the low native update rates of its sensors (e.g., LiDAR and cameras) and the heavy computational overhead of processing their data on resource-limited hardware. This asynchrony between perception and control has become a key barrier to achieving highly agile and robust end-to-end navigation.

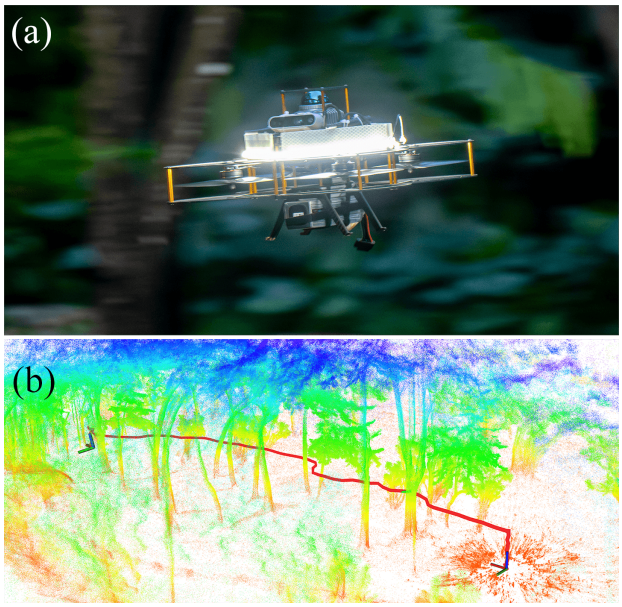



Fig. 1: The proposed asynchronous framework validated through zero-shot sim-to-real flight in a dense forest. (a) The custom-built AAV navigating through cluttered trees. (b) The complete, collision-free trajectory of a 30-meter flight test through the forest.

This work presents a novel asynchronous framework designed to resolve the temporal mismatch between perception and control. The proposed method operates by decoupling these two loops, enabling the control policy to execute at a high frequency using the latest available state. This decoupling, in turn, introduces the consequential problem of data staleness, formally known as the Age of Information (AoI) [8], where the policy must rely on delayed perception features. This challenge is addressed through a principled Temporal Encoding Module (TEM), a mechanism that allows

†These authors contributed equally to this work.

*Corresponding authors: Jie Mei jmei@hit.edu.cn and Youmin Gong gongyoumin@hit.edu.cn.

The authors are with the School of Intelligence and Engineering, and also with the Guangdong Provincial Key Laboratory of Intelligent Morphing Mechanisms and Adaptive Robotics, Harbin Institute of Technology, Shenzhen, Guangdong, China. This work was supported in part by the Shenzhen Science and Technology Program under Grant JCYJ20241202124010014, JCYJ20240813104923032, and GXWD20231129140908002.

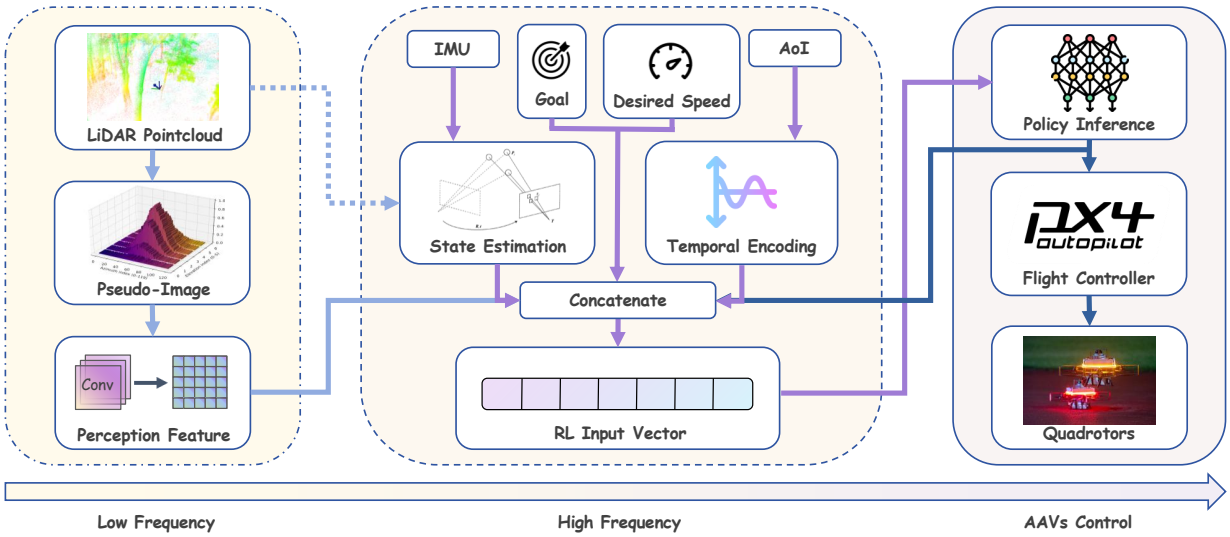



Fig. 2: Overview of the asynchronous framework. The low-frequency perception module converts LiDAR point clouds to a pseudo-image, which a CNN processes into a feature vector. The high-frequency control module concatenates this feature with the latest IMU state, previous action, desired speed, and a temporal encoding vector representing data staleness. An MLP policy then processes this combined state to generate high-frequency control commands.

the network to explicitly reason about information delay and compensate for the resulting partial observability. The framework further incorporates an efficient pillar-based perception module to minimize latency at the sensor processing stage. Through extensive validation in the NVIDIA Isaac Sim environment and a successful zero-shot transfer to physical AAVs, this framework is shown to enable robust, high-frequency navigation (Fig. 1). The main contributions of this work are summarized as follows:

- A novel end-to-end network architecture is presented, featuring a computationally efficient LiDAR data processing module that leverages spatial features to enable agile flight in complex environments.
- A theoretically-grounded Temporal Encoding Module is proposed to enable asynchronous policy inference. By modeling data staleness, this module resolves the partial observability induced by the low-frequency perception stream, facilitating robust and high-frequency control from low-frequency sensors on computationally limited platforms.
- A two-stage curriculum learning strategy is applied to optimize the asynchronous policy, enabling successful zero-shot, sim-to-real transfer. This result, validated on physical AAVs, confirms the practical applicability and robustness of the proposed asynchronous framework.

## II. RELATED WORK

Architectures for autonomous navigation have evolved from modular pipelines to learning-based paradigms to address error propagation and latency. In modular perception–planning–control pipelines, representative planners attain robust and agile flight yet remain exposed to inter-module error propagation and accumulated delays [9], [10], [11], [12].

Hybrid approaches integrate machine learning into specific system components, with examples in perception-aware imitation [13], one-stage planning [14], learned time allocation [15], and adaptive speed constraints [16]. Similarly, frameworks like NavRL employ a specialized perception module [17]. However, many of these hybrid architectures continue to use a synchronous update schedule. This model makes the decision-making latency directly dependent on the perception system's processing rate and the available onboard computing power.

End-to-end control via reinforcement learning demonstrates high-speed navigation [5], [18] and champion-level racing from visual inputs [4], [19]. In vision-based end-to-end methods, studies explore obstacle avoidance using learned motion cues or adaptive-speed control [20], [21], [22]. An alternative line directly processes point clouds—spanning model-based [23] and model-free methods [24], [25]—and typically relies on point, voxel, or pillar encoders (PointNet++ [26], VoxelNet [27], PointPillars [28]) proven effective in autonomous driving and quadrupedal locomotion [6], [7], suggesting transfer to AAVs. However, many end-to-end stacks typically operate under a single-inference-per-sensor-frame regime (policy updates gated by new perception frames, even if the low-level controller ticks faster), constraining control reactivity by sensor update rates and perception compute. Traditionally, sensor latency is addressed by decoupling perception and planning [29] or employing high-frequency reactive policies [30]. Our framework adopts a similar decoupled structure within end-to-end learning, but introduces TEM to explicitly compensate for the resulting data staleness.

To address the challenges of asynchrony and delay in systems, multiple lines of research offer distinct approaches. For instance, while theories like Reinforcement Learning (RL)

explicitly model perception-to-action delays [31], multi-rate and event-triggered control focuses on optimizing update schedules under resource constraints [32], and the concept of Age of Information (AoI) provides a metric to quantify data freshness [8], [33]. Collectively, these perspectives reveal that the temporal mismatch between perception and control is not merely an architectural issue of system design, but a fundamental structural problem rooted in physical constraints. This necessitates explicit temporal reasoning at the moment of decision-making. Yet, most work in this area has focused on low-level components like controllers and communication protocols. The integration of such temporal modeling into high-level, end-to-end perception-to-action policies remains a significant and underexplored challenge, which this work confronts.

Despite steady progress, most state-of-the-art end-to-end frameworks typically operate under a single-inference-per-sensor-frame schedule (e.g., [20], [21], [22], [24], [25]), effectively tying decision updates to perception throughput and overlooking data staleness, formally the Age of Information (AoI) [8], [33], and the partial observability inherent in real asynchronous systems. We address this gap by decoupling perception and control and by explicitly encoding AoI at decision time via a Temporal Encoding Module, which enables robust high-frequency control under realistic low-rate sensing without relying solely on implicit temporal memory.

## III. METHODOLOGY

Our proposed framework (Fig. 2) comprises a low-frequency perception pipeline and a high-frequency control loop that operate asynchronously. It transforms raw LiDAR data into a 2D Pseudo-Image (III-A) and formulates the task as a Markov Decision Process (MDP) (III-B). The core of our method is an Asynchronous End-to-End Network, whose temporal encoding is theoretically justified (III-C). The network's Architecture (III-D) and Policy Training details (III-E) are also presented.

### A. Pseudo-Image Generation

To process raw and unstructured point cloud data, a preliminary transformation into a structured pseudo-image representation is required. The proposed methodology employs a spherical coordinate projection for this conversion. This approach is specifically chosen for its efficacy in preserving range information, which is essential for subsequent collision avoidance tasks.

Each point in the raw point cloud, captured in the robot's body frame, can be represented in spherical coordinates as $\mathbf{p} = (r, \theta, \phi)$, where $r$ denotes the radial distance, $\theta$ the azimuthal angle in the XY-plane, and $\phi$ the polar angle measured from the positive Z-axis. The raw point cloud is the set of all points $\xi = \{\mathbf{p}_m \mid m = 1, \ldots, N\}$, where each $\mathbf{p}_m = (r_m, \theta_m, \phi_m)$.

To convert the unstructured point cloud into a structured pseudo-image, the point cloud is first discretized into a 2D grid by partitioning the sensor's field of view into angular bins. Each pillar $P_{i,j}$ is then defined as the set of all points within a specific angular bin

$$P_{i,j} = \{\mathbf{p} \mid \mathbf{p} \in \xi, \theta_i \le \theta < \theta_{i+1}, \phi_j \le \phi < \phi_{j+1}\}. \quad (1)$$

This grid spans the sensor's field of view, $[\theta_{\min}, \theta_{\max})$ and $[\phi_{\min}, \phi_{\max})$, with angular resolutions $\Delta\theta_{\text{pillar}}$ and $\Delta\phi_{\text{pillar}}$. This results in a grid of dimensions $(N_\phi, N_\theta)$, where $N_\theta = \lfloor(\theta_{\max} - \theta_{\min})/\Delta\theta_{\text{pillar}}\rfloor$ and $N_\phi = \lfloor(\phi_{\max} - \phi_{\min})/\Delta\phi_{\text{pillar}}\rfloor$.

The value of each pixel $I(i, j)$ in the pseudo-image is determined by the contents of its corresponding pillar $P_{i,j}$ according to the following function:

$$I(i,j) = \begin{cases} \min\{r \mid \mathbf{p} = (r, \theta, \phi) \in P_{i,j}\} & \text{if } P_{i,j} \neq \varnothing \\ r_{\max} & \text{if } P_{i,j} = \varnothing \end{cases}, \quad (2)$$

where $r_{\max}$ is the sensor's maximum range. The resulting collection of range values forms a single-channel pseudo-image of dimensions $(N_\phi, N_\theta, 1)$. This structured tensor serves as the input to a convolutional backbone for feature extraction.

### B. Reinforcement Learning Framework

*1) Problem Formulation:* The navigation task is formulated as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P$ is the state transition function, $R$ is the reward function, and $\gamma$ is the discount factor. The agent's objective is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward, given by

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \quad (3)$$

The state observation $s_t \in \mathcal{S}$ is a composite vector defined as the concatenation of perception features and the quadrotor's internal state:

$$s_t = [\mathbf{z}_t, \mathbf{p}_{\text{rel}}, \mathbf{q}, \mathbf{v}, \boldsymbol{\omega}, \mathbf{a}_{\text{prev}}, v_{\text{des}}, \Phi(\Delta t_{\text{lidar}})]^T, \quad (4)$$

where $\mathbf{z}_t \in \mathbb{R}^{D_{\text{perception}}}$ is the perception feature, $\mathbf{p}_{\text{rel}} \in \mathbb{R}^3$ is the relative position to the target expressed in the body frame, $\mathbf{q} \in \mathbb{R}^4$ is the orientation as a unit quaternion, $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity, $\mathbf{a}_{\text{prev}} \in \mathbb{R}^{D_{\text{action}}}$ is the previous action taken, $v_{\text{des}} \in \mathbb{R}$ is the desired speed and $\Phi(\Delta t_{\text{lidar}}) \in \mathbb{R}^{D_{\text{temp}}}$ is the output of the Temporal Encoding Module.

The action space $\mathcal{A} \subset \mathbb{R}^{D_{\text{action}}}$ defines the high-level control commands for AAVs. The policy network outputs these commands, which are then tracked by a low-level controller.

*2) Reward Function:* The reward function $R_t = R(s_t, a_t)$ is composed of dense and sparse components, formulated as $R_t = \sum_i w_i r_i + r_T$, where the first term is a weighted sum of dense reward components $r_i$ with corresponding weights $w_i$, and $r_T$ is a sparse terminal reward that is non-zero only upon episode termination. The dense components are defined as follows:

a) Static Safety Reward ($r_{\text{static}}$): This reward penalizes proximity to static obstacles based on LiDAR measurements. The calculation is defined by:

$$p_{\text{beam}} = \tanh\left(k \cdot \left(\frac{\min(d_{\text{obs}}, L_s)}{L_s} - c\right)\right) + 1, \tag{5}$$

$$r_{\text{static}} = Q_q\left(\log(p_{\text{beam}})\right), \tag{6}$$

where $d_{obs}$ denotes the distance to obstacles, and $L_s$ is the threshold distance at which the penalty becomes active. The penalty curve, parameterized by $k$ and $c$, employs $\tanh$ and $\log$ functions to induce a sharp transition that strongly discourages collisions. The quantile function $Q_q$ ensures robustness to measurement noise.

b) Velocity Reward ($r_{\text{velocity}}$): This term encourages the robot to move towards the goal at a desired speed. It penalizes any misalignment between the velocity vector $\mathbf{v}$ and the unit direction vector $\hat{\mathbf{g}}$ towards the goal. It also manages the speed $v = \|\mathbf{v}\|$, encouraging the agent to maintain a speed near $v_{\text{des}}$ within a desired range $[k_{v1}v_{\text{des}}, k_{v2}v_{\text{des}}]$, while penalizing speeds outside this range, where $0 < k_{v1} < 1 < k_{v2}$. The reward is formulated as:

$$\begin{aligned} r_{\text{velocity}} =& \mathbf{v} \cdot \hat{\mathbf{g}} - (\max(0, v - k_{v2}v_{\text{des}}))^2 \\ & - (\max(0, k_{v1}v_{\text{des}} - v))^2 \\ & + \mathbb{I}(k_{v1}v_{\text{des}} \le v \le k_{v2}v_{\text{des}}) \\ & \cdot \exp\left(-\frac{(v - v_{\text{des}})^2}{2\sigma^2}\right), \end{aligned} \tag{7}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $\sigma$ controls the reward shape.

c) Height Penalty ($r_{\text{height}}$): A penalty is applied for altitude $z$ straying from a vertical corridor $[z_{\min}, z_{\max}]$, given by

$$r_{\text{height}} = -(\max(0, z - z_{\max}))^2 - (\max(0, z_{\min} - z))^2. \tag{8}$$

d) Attitude Penalty ($r_{\text{attitude}}$): Excessive pitch $\alpha_y$ or roll $\alpha_x$ angles beyond a threshold $\alpha_{\max}$ are penalized through the term

$$\begin{aligned} r_{\text{attitude}} =& -(\max(0, |\alpha_y| - \alpha_{\max}))^2 \\ & - (\max(0, |\alpha_x| - \alpha_{\max}))^2. \end{aligned} \tag{9}$$

In addition to these dense rewards, large sparse rewards or penalties are issued at episode termination: a positive reward $r_{\text{goal}}$ for reaching the target, a negative penalty $r_{\text{collision}}$ for collision, and a penalty $r_{\text{limit}}$ for violating an operational boundary limit, such as exceeding maximum altitude or leaving the designated flight area.

### C. Asynchronous End-to-end Network with Temporal Encoding Module

Conventional end-to-end control suffers from the mismatch between high-rate IMU-based control and low-rate perception from sensors like LiDAR, whereas the proposed asynchronous architecture decouples them, enabling high-frequency control. This decoupling, however, introduces data staleness. This staleness is formally modeled via the Age of Information (AoI) [8]—specifically, the age upon decision (AuD)—defined as

$$\Delta t_{\text{lidar}} = t^i_{\text{ctrl}} - t_{\text{meas}}, \tag{10}$$

where $t^i_{\text{ctrl}}$ is the $i^{th}$ decision time and $t_{\text{meas}}$ is the measurement timestamp. The resulting high Age of Information (AoI) induces partial observability, breaking the Markov assumption in the original state space. To address this, a Temporal Encoding Module (TEM) that encodes the AoI is proposed as a policy input $\Phi(\Delta t_{\text{lidar}})$.

Let the standard observation

$$O'_t = [\mathbf{z}_t, \mathbf{p}_{\text{rel}}, \mathbf{q}, \mathbf{v}, \boldsymbol{\omega}, \mathbf{a}_{\text{prev}}, v_{\text{des}}]^T, \tag{11}$$

which is the full state $s_t$ from Eq.4 without temporal information. The augmented observation $O_t$ is defined as the concatenation of $O'_t$ and the encoded delay feature:

$$O_t := (O'_t, \Phi(\Delta t_{\text{lidar}})) = s_t. \tag{12}$$

By conditioning on its own velocity and the explicit time delay, the policy can learn to implicitly predict how the perceived environment has changed, compensating for the information staleness. Theoretically, augmenting the observation with the AoI reduces the conditional entropy of the state estimation ($H(S_t|O_t) \le H(S_t|O'_t)$) and, by the Law of Total Variance, eliminates the excess variance caused by delay uncertainty, thereby facilitating more stable policy learning.

### D. Network Architecture

Since the LiDAR perception data is structured as a 2D pseudo-image, a convolutional neural network (CNN) is utilized as a backbone to extract spatial features and transform them into a compact feature embedding. This perception embedding is then concatenated with the robot's internal state vector. This combined state is then processed by a MLP decoder to generate a unified feature representation for the subsequent actor and critic networks.

From this unified representation, the actor network outputs $(\alpha, \beta)$ for a Beta distribution to generate normalized continuous actions, which improves convergence in constrained spaces [34]. A constant offset $\epsilon$ is added after `Softplus` to ensure $\alpha, \beta > \epsilon$, maintaining a unimodal distribution and avoiding unstable U-shaped behaviors.

### E. Policy Training

A primary challenge in this asynchronous framework is the temporal discrepancy between low-frequency perception updates and high-frequency control actions. This mismatch provides a weak supervisory signal for the policy, making direct training inefficient. To overcome this challenge, a two-stage curriculum learning strategy is adopted.

**I.** Synchronous Training Stage: The network is trained synchronously within the simulator using ideal, high-frequency perception data. Within this idealized setting, the policy acquires a fundamental baseline navigation capability. Notably, throughout this learning process, the simulated AoI is maintained to be zero.

**II.** Asynchronous Training Stage: After convergence in the synchronous phase, training shifts to an asynchronous paradigm where the simulator provides low-frequency perception data to mimic real-world constraints. Unlike the synchronous stage (AoI = 0), the simulated AoI is now a non-zero, time-varying value that quantifies data staleness. The curriculum-based training strategy promotes robust policy generalization by employing the Temporal Encoding Module (TEM) to adapt decision-making under time-varying perception delays. This mechanism effectively compensates for the performance degradation induced by transmission latency.

Crucially, the synchronous training in the first stage provides the policy with a critical warm start, making the task of learning to leverage the TEM to handle time-varying delays in the second stage more tractable and thereby significantly improving training stability.

## IV. RESULTS AND DISCUSSIONS

### A. Implementation Details

*1) Network Architecture and Training:* In this paper, the actor-critic policy is optimized using the Proximal Policy Optimization (PPO) algorithm [35]. Training is conducted in a forest-like simulation environment developed with the Omnidrones framework [36] in NVIDIA Isaac Sim. The policy network is trained on a single NVIDIA RTX 5880 Ada GPU using 4000 parallel simulation environments as shown in Fig. 3, with a desired speed randomly sampled from $[1, 4]\ m/s$. The network's perception input is the LiDAR point cloud, which covers an azimuthal range of $[-\frac{1}{2}\pi, \frac{1}{2}\pi)$ with resolutions of $\Delta\theta_{\text{pillar}} = \frac{1}{60}\pi$ and $\Delta\phi_{\text{pillar}} = \frac{1}{36}\pi$. The convolutional backbone processes this pseudo-image and outputs a compact feature embedding $z_t \in \mathbb{R}^{32}$ . The policy network's action space is defined as target linear velocities in the robot's body frame. The non-perceptual state input is composed of the action vector from the previous timestep, other proprioceptive states, temporal feature from the Temporal Encoding Module (TEM), which processes the AoI and is implemented as a sinusoidal encoder inspired by [37].

For a continuous AoI $\Delta t_{\text{lidar}}$, the feature vector $\Phi(\Delta t_{\text{lidar}})$ is computed as:

$$\Phi(\Delta t_{\text{lidar}}) = [\sin(t_j), \cos(t_j), \sin(t_j/100), \cos(t_j/100)]^T, \tag{13}$$

where the discrete step $j$ and re-quantized time $t_j$ are derived using $j = \text{round}(\Delta t_{\text{lidar}}/\Delta t)$ with a resolution of $\Delta t = 0.01$s, and $t_j = j \cdot \Delta t$. The policy is trained using the AdamW optimizer and a PPO clip ratio of 0.1.

*2) Real-World Experimental Platform:* For real-world validation, the trained policy is deployed on a custom-built quadrotor, as shown in Fig. 1(a). Onboard computation is handled by an Intel NUC 13. This unit interfaces with an NxtPx4 flight controller. Environmental perception is provided by a Livox Mid-360 LiDAR [38]. An onboard Intel RealSense D435i camera is used solely for visualization and does not contribute to the navigation algorithm. To evaluate the framework's performance and portability on different embedded systems, the same policy is also successfully deployed and tested on an NVIDIA Jetson Orin NX module.

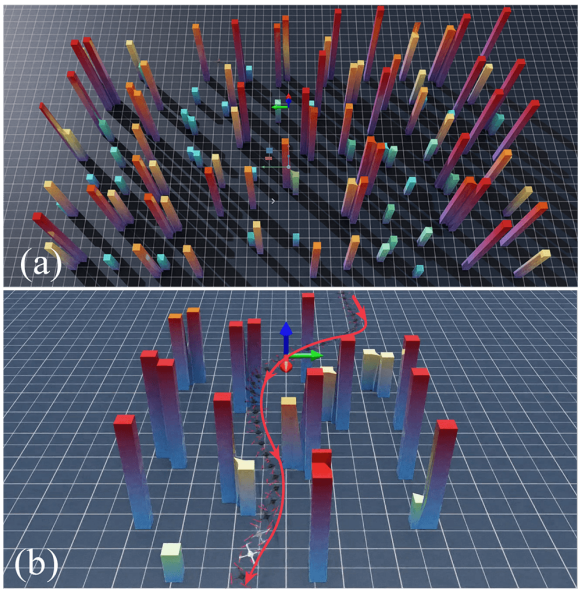



Fig. 3: Visualization of (a) a sample training environment and (b) the collision-free trajectory generated by the proposed end-to-end navigation model.

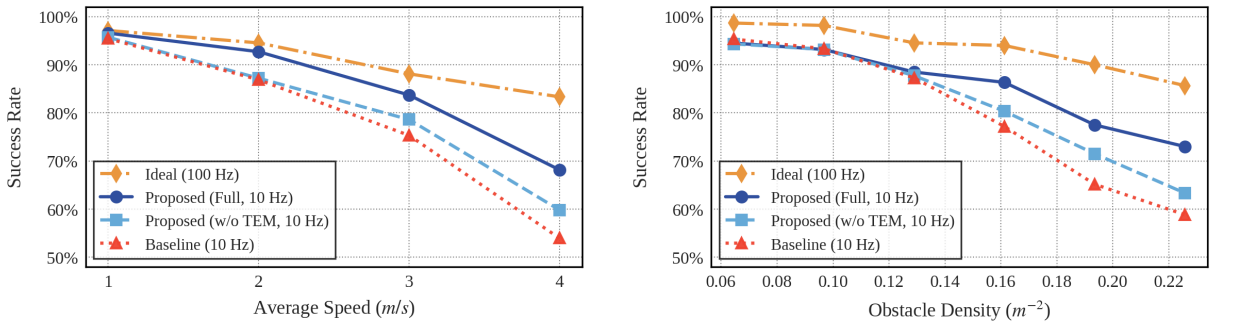
### B. Simulation Experiments

*1) Benchmark with Previous Systems:* A benchmark analysis is conducted to evaluate the proposed framework (a representative successful trajectory is shown in Fig. 3(b)) against state-of-the-art methods: two Learning-based (NavRL [17], YOPO [14]) and one optimization-based (EGO-Planner-v2 [2]) with each method evaluated 2000 trials. For each trial, the environment is randomly initialized, featuring a high-density obstacle field ($0.2\ m^{-2}$) composed of rectangular prisms with horizontal side lengths sampled from $[0.4, 0.6]\ m$. The agent's task is to navigate a path at a target average velocity of $2.5\ m/s$. To assess robustness to sensor constraints, we evaluated performance under both an ideal $100\ Hz$ and a constrained $10\ Hz$ perception rate, with the latter chosen to mirror the native update rate of the LiDAR sensor used in our physical tests [38]. To ensure a fair comparison, the underlying low-level controller for all tested methods is maintained at a constant $100\ Hz$, and, within each perception-rate regime, all methods share the same compute budget.

The results in Table I highlight our framework's exceptional robustness to varying sensor frequencies. At a $100\ Hz$ perception rate, our method achieves a success rate of 93.67%, which remained remarkably high at 91.08% (a minimal 2.6% drop) when the rate is reduced to $10\ Hz$. This stability stands in stark contrast to competing synchronous systems, such as NavRL [17], which suffers a much larger 11.6% performance degradation. This performance gap arises because the control frequency of conventional learning-based approaches is fundamentally coupled to the perception rate. Similarly, optimization-based planners like EGO-Planner-

TABLE I: PERFORMANCE BENCHMARK UNDER VARYING SENSOR FREQUENCIES. All trials are conducted in an environment with an obstacle density of 0.2 $m^{-2}$ and a target average velocity of 2.5 $m/s$.

| **Model** | **Reach Goal (↑)** | | **Performance Loss (↓)** | **Avg. Velocity ($m/s$)** | | **Onboard Processing Demand** | |
|---|---|---|---|---|---|---|---|
| | High Freq | Low Freq | | High Freq | Low Freq | High Freq | Low Freq |
| NavRL [17] | 86.96% | 75.37% | 11.6% | 2.49 | 2.46 | High | Low |
| YOPO [14] | 67.71% | 58.51% | 9.2% | 2.51 | 2.52 | Medium | Low |
| EGO-Planner-v2 [2] | 79.87% | 76.71% | 3.2% | 2.53 | 2.53 | Medium | Medium |
| Ours (Sync. Baseline / Async) | **93.67%** | **91.08%** | **2.6%** | 2.54 | 2.54 | Medium | Low |




(a) Performance as a function of average speed, with obstacle density held constant at 0.25 $m^{-2}$.

(b) Performance as a function of obstacle density, with average speed held constant at 4.0 $m/s$.

Fig. 4: Evaluation of model success rate under challenging conditions.

v2 [2], despite featuring a decoupled core computation, are ultimately bottlenecked by real-time map generation from the same slow data stream. In contrast, our framework's robustness is a direct consequence of its asynchronous architecture. By decoupling the perception and control loops, our design enables high-frequency policy execution while using the Temporal Encoding Module (TEM) to actively compensate for information delays, thereby overcoming the bottlenecks inherent to synchronous models.

*2) Ablation Studies and Performance Analysis:* To validate the proposed asynchronous framework, two sets of simulation experiments are conducted to analyze performance under varying speeds and obstacle densities, with 1000 trials for each tested condition. The environment consists of a 20-meter path populated with obstacles shaped as rectangular prisms, whose horizontal side lengths are sampled from 0.4 m to 0.6 m. The evaluation compares the following models:

- **Proposed Model**: The complete asynchronous framework with the sinusoidal TEM.
- **Ideal Case**: A synchronous version of the proposed model with an ideal high-frequency (100 $Hz$) perception input, serving as a performance upper bound.
- **Ablation Model**: The proposed model with the TEM removed, to specifically isolate its contribution.
- **Baseline Model**: A conventional synchronous end-to-end model, where the control loop frequency is limited by the LiDAR sensor (10 $Hz$).

In the first experiment (Fig. 4(a)), performance is evaluated as a function of average speed, with obstacle density held constant at 0.25 $m^{-2}$. In the second (Fig. 4(b)), robustness is analyzed by varying the obstacle density while maintaining a constant high speed of 4 $m/s$.

The findings indicate that conventional synchronous models, though effective in simple settings, degrade in high-speed, high-density scenarios, whereas our asynchronous method surpasses the baseline by over 14 percentage points. This performance is further enhanced by the Temporal Encoding Module (TEM); its removal caused a significant ($p < 0.05$) performance drop of 8.4 and 9.7 percentage points under the maximum speed and maximum density conditions, respectively. Notably, the TEM's contribution becomes more pronounced as the task grows more challenging, with the performance gap widening significantly at higher speeds and densities. Crucially, the complete framework also closes a significant portion of the performance gap to an ideal, fully synchronous system, validating it as a robust and effective solution for high-frequency, reactive navigation.

### C. Physical Flight Tests and Performance Analysis

The robustness of our framework is validated by its successful zero-shot sim-to-real transfer, deploying a policy trained entirely in simulation directly onto the physical platform without any fine-tuning. For these tests, the AAV relies solely on a low-frequency LiDAR operating at 10 $Hz$ (the typical native update rate of the onboard Livox Mid-360 sensor [38]) while navigating two distinct and challenging environments at an average speed of 1.3 $m/s$, with a maximum velocity of 2.0 $m/s$. Multiple successful trials are conducted in each environment to confirm the policy's consistency. During these flights, the policy consistently operates with a high Perception AoI, frequently exceeding 100 $ms$, which validates the effectiveness of the TEM in a real-world setting.

In a cluttered indoor space (0.25 $m^{-2}$ density), the AAV successfully navigates the planned course and demonstrates

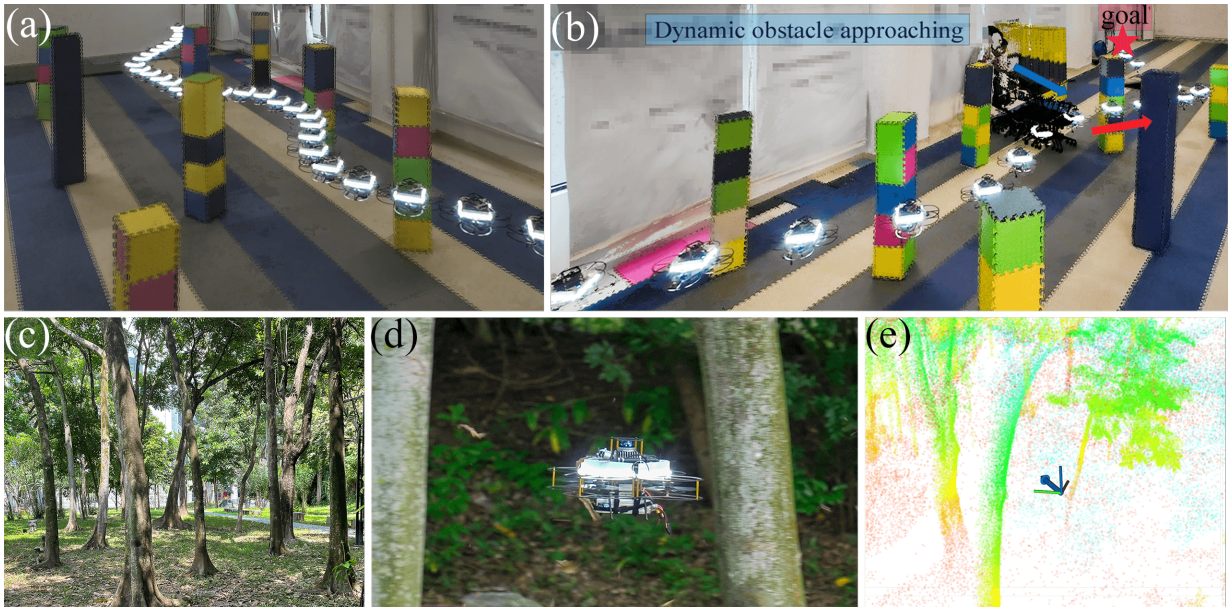



Fig. 5: Real-world flight validation of our asynchronous framework, demonstrating successful zero-shot sim-to-real transfer in cluttered environments. (a) The AAV navigating a dense indoor obstacle field ($0.25\ m^{-2}$). (b) The AAV's flight trajectory in a scenario involving a dynamic obstacle. (c) First-person and (d) third-person views of autonomous navigation through a dense forest ($0.18\ m^{-2}$). (e) Onboard visualization displaying the input LiDAR point cloud alongside the corresponding velocity command (blue arrow) generated by the policy.

TABLE II: COMPUTATIONAL LATENCY ON DESKTOP AND ONBOARD PLATFORMS

| | Desktop* | | Onboard† | |
|---|---|---|---|---|
| **Module** | **GPU** | **CPU** | **GPU** | **CPU** |
| ***Base Module Cost (Time in ms)***§ | | | | |
| **Perception Total** | **0.40**‡ | **0.80**‡ | **3.99**‡ | **1.15**‡ |
| - PC Pre-proc. | 0.26 | 0.54 | 2.51 | 0.80 |
| - Perc. Network | 0.14 | 0.26 | 1.48 | 0.35 |
| **Control Policy** | **0.39** | **0.21** | **1.72** | **0.27** |

* Desktop: Intel Core i5-14600K CPU, NVIDIA RTX 4060 Ti GPU.
† Onboard: Intel NUC 13 i7-1360P CPU, NVIDIA Jetson Orin NX GPU.
§ Timings reflect pure computational latency, excluding overhead from data I/O and transmission.
‡ Totals assume sequential execution.

reactive avoidance against dynamic objects, with key moments documented in Fig. 5(a-b). Similarly, in a dense outdoor forest ($0.18\ m^{-2}$ density), the drone consistently and autonomously maneuvers through the trees. Its flight and onboard decision-making process are visualized from multiple perspectives in Fig. 5(c-e), and a complete representative trajectory is presented in Fig. 1(b).

The computational performance that underpins this reliable flight behavior is detailed in Table II. The analysis shows the asynchronous architecture leverages the high efficiency of the Control Policy network by executing it multiple times for each slower perception update. This design proves effective in practice, sustaining a stable $100\ Hz$ control loop during physical deployment on both an Intel NUC 13 and an NVIDIA Jetson Orin NX.

The physical flight tests demonstrate our framework's viability for autonomous navigation on resource-constrained AAVs. The policy's robustness is evidenced by its successful zero-shot transfer to complex, real-world environments. This success stems from three key factors. First, the inherent robustness of our perception front-end stems from its high degree of environmental abstraction. Second, a lightweight network architecture enables real-time onboard inference. Third, the asynchronous design preserves a high-frequency control loop even under low-rate perception. These results indicate that our approach is a practical and effective solution for deployment in real-world robotic systems.

### D. Limitations and Future Work

While the proposed framework demonstrates robust performance, two key limitations exist. First, as a purely reactive method, the current policy lacks an explicit trajectory prediction module for dynamic agents. Consequently, while it handles static or slowly moving obstacles effectively, reliably avoiding high-speed dynamic obstacles remains a challenge. Second, regarding the sim-to-real gap at high speeds, the discrepancy between the simulated and physical platform dynamics becomes pronounced as velocity increases. Achieving robust flight at higher velocities requires precise system identification to align the low-level controller's response, which is a critical direction for future work.

## V. CONCLUSIONS

This paper introduces an end-to-end framework that successfully mitigates the temporal conflict between low-

frequency perception and high-frequency control in AAV navigation. The success of our approach is built upon three key innovations: a lightweight end-to-end architecture with a computationally efficient LiDAR processing module; a principled asynchronous design featuring a Temporal Encoding Module to manage the Age of Information; and a two-stage curriculum that ensures effective policy training. The framework's robustness is validated through successful zero-shot sim-to-real deployment, where the policy sustains a $100\ Hz$ control rate on an onboard computer while navigating a high-density forest. By enabling high-frequency control with low-frequency sensors, this work directly addresses key limitations of end-to-end models, significantly enhancing their agility and robustness for deployment in cluttered, real-world environments.

## REFERENCES


[1] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[2] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[3] G. Lu, W. Xu, and F. Zhang, "On-manifold model predictive control for trajectory tracking on robotic systems," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 9, pp. 9192–9202, 2023.

[4] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[5] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[6] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[7] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 164–10 183, 2024.

[8] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *2012 Proceedings IEEE INFOCOM*. IEEE, 2012, pp. 2731–2735.

[9] Y. Ren, F. Zhu, G. Lu, Y. Cai, L. Yin, F. Kong, J. Lin, N. Chen, and F. Zhang, "Safety-assured high-speed navigation for mavs," *Science Robotics*, vol. 10, no. 98, p. eado6187, 2025.

[10] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[11] W. Liu, Y. Ren, and F. Zhang, "Integrated planning and control for quadrotor navigation in presence of suddenly appearing objects and disturbances," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 899–906, 2023.

[12] J. Zhang, Z. Zhou, W. Xia, Y. Gong, and J. Mei, "Storm: Spatial-temporal iterative optimization for reliable multicopter trajectory generation," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 13 266–13 273.

[13] J. Tordesillas and J. P. How, "Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1399–1406, 2023.

[14] J. Lu, X. Zhang, H. Shen, L. Xu, and B. Tian, "You only plan once: A learning-based one-stage planner with guidance learning," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6083–6090, 2024.

[15] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar, "Deep learning for optimization of trajectories for quadrotors," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2479–2486, 2024.

[16] G. Zhao, T. Wu, Y. Chen, and F. Gao, "Learning speed adaptation for flight in clutter," *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7222–7229, 2024.

[17] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "Navrl: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3668–3675, 2025.

[18] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Learning vision-based agile flight via differentiable physics," *Nature Machine Intelligence*, pp. 1–13, 2025.

[19] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *8th Conference on Robot Learning (CoRL)*, 2024.

[20] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1989–1995.

[21] Y. Hu, Y. Zhang, Y. Song, Y. Deng, F. Yu, L. Zhang, W. Lin, D. Zou, and W. Yu, "Seeing through pixel motion: Learning obstacle avoidance from optical flow with one camera," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5871–5878, 2025.

[22] H. Yu, C. Wagter, and G. C. H. E. de Croon, "Mavrl: Learn to fly in cluttered environments with varying speed," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1441–1448, 2025.

[23] R. Han, S. Wang, S. Wang, Z. Zhang, J. Chen, S. Lin, C. Li, C. Xu, Y. C. Eldar, Q. Hao, and J. Pan, "Neupan: Direct point robot navigation with end-to-end model-based learning," *IEEE Transactions on Robotics*, vol. 41, pp. 2804–2824, 2025.

[24] G. Xu, T. Wu, Z. Wang, Q. Wang, and F. Gao, "Flying on point clouds with reinforcement learning," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 7231–7238.

[25] X. Fan, M. Lu, B. Xu, and P. Lu, "Flying in highly dynamic environments with end-to-end learning approach," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3851–3858, 2025.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[28] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.

[29] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[30] M. Pantic, I. Meijer, R. Bähnemann, N. Alatur, O. Andersson, C. Cadena, R. Siegwart, and L. Ott, "Obstacle avoidance using raycasting and riemannian motion policies at khz rates for mavs," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1666–1672.

[31] S. Ramstedt and C. Pal, "Real-time reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] W. Heemels, K. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *2012 IEEE 51st IEEE Conference on Decision and Control*, 2012, pp. 3270–3285.

[33] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[34] P.-W. Chou, D. Maturana, and S. Scherer, "Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution," in *International Conference on Machine Learning*. PMLR, 2017, pp. 834–843.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[36] B. Xu, F. Gao, C. Yu, R. Zhang, Y. Wu, and Y. Wang, "Omnidrones: An efficient and flexible platform for reinforcement learning in drone control," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2838–2844, 2024.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[38] Livox, "Mid-360 Specs," https://www.livoxtech.com/mid-360/specs, accessed: Sep. 02, 2025.