LiMPNet: Lightweight Multi-sensor Perception and DRL Navigation for Tiny Drones in Mapless Environments

Omer Kurkutlu, Arman Roohi

Department of Electrical and Computer Engineering, University of Illinois Chicago, Chicago, IL
okurku2@uic.edu, aroohi@uic.edu

Abstract

Autonomous tiny drones face significant challenges in navigation due to strict constraints on size, weight, power, and onboard computational capacity. This paper presents a lightweight navigation framework that integrates basic multi-sensor perception with deep reinforcement learning (DRL) to enable safe, mapless flight in cluttered environments. We employ the Crazyflie 2.1 nano-drone, equipped with a grayscale camera and a multi-ranger deck, a laser-based distance sensor, for real-time obstacle detection and avoidance. A Proximal Policy Optimization (PPO) agent is trained within a ROS and Gazebo simulation environment to generate collision-free trajectories using fused visual and range data. The system is evaluated in two environments: a simple obstacle field, where the drone achieves a 100% success rate (112/112 episodes), and a densely cluttered map, where it reaches the target in 35% of trials (7/20). These results demonstrate that effective autonomous navigation is achievable using minimal sensing and low-computation models, making it well-suited for resource-constrained aerial platforms.

Introduction

Tiny drones, also known as micro aerial vehicles (MAVs), are ultra-lightweight and compact flying robots, typically weighing less than 100 grams and measuring under 10 centimeters in size (Palossi et al. 2019; Floreano and Wood 2015). Their miniature form factor offers unique advantages such as agility, ease of deployment, and the ability to operate in confined or hazardous spaces that are inaccessible to larger drones. As a result, they have become increasingly relevant in domains like indoor inspection, infrastructure monitoring, search-and-rescue missions in collapsed buildings, precision agriculture (Zhao et al. 2021), wildlife observation, and urban surveillance (Loquercio et al. 2018).

Despite their potential, developing autonomous navigation systems for tiny drones remains a significant challenge. These platforms operate under strict size, weight, and power constraints, which limit their onboard computational resources, sensor payload capacity, and battery life (Banbury et al. 2021; Floreano and Wood 2015). While some systems address these limitations by offloading computation to external ground stations or cloud infrastructure, such solutions are often impractical for autonomous missions in communication-constrained or GPS-denied environments (Tai, Paolo, and Liu 2017).

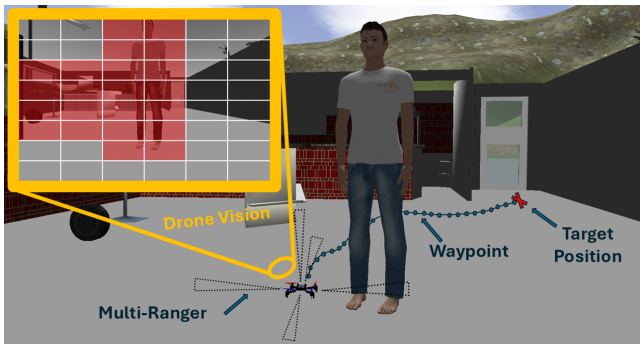

Figure 1: Simulation setup in Gazebo showing the Crazyflie 2.1 nano-drone navigating an indoor environment. The drone utilizes a grayscale camera to generate a 6×8 obstacle matrix and a multi-ranger deck for multi-direction laser ranging. The PPO agent utilizes these fused inputs for reactive, mapless navigation toward a predefined target. The planned path is visualized as a sequence of waypoints leading to the target.

In addition, tiny drones must operate in real time and in dynamic, unstructured environments. This makes mapless navigation with reactive obstacle avoidance a critical capability (Gandhi et al. 2017). However, achieving such autonomy while maintaining low latency and fitting within sub-watt power envelopes remains a difficult balance. To address these challenges, there is growing interest in lightweight, learning-based approaches that combine minimal sensing with compact control policies. In particular, the integration of DRL and Tiny Machine Learning (TinyML) enables drones to learn adaptive and computationally efficient navigation strategies (Liu et al. 2022; Loquercio et al. 2018). DRL agents demonstrate strong navigation performance, achieving high success rates in structured environments (Gandhi et al. 2017; Tai, Paolo, and Liu 2017). TinyML techniques have enabled inference times under 50 ms on microcontroller-class processors (Palossi et al. 2019; Liu et al. 2022), making these strategies practical for edge deployment.

In this work, we propose a DRL-based navigation system tailored to the constraints of tiny drones. Our architecture integrates compact vision-based perception with a multi-ranger deck, a sensor capable of measuring distances in five directions (left, right, front, back, and up), to enhance environmental awareness, as illustrated in Fig. 1. Specifically, we incorporate a YOLO-based obstacle detector (Jocher et al. 2023) for grid-based spatial representation, a PID controller for precise motion execution (Furrer et al. 2016b), and an onboard IMU sensor for position control and lightweight collision detection. A PPO agent (Schulman et al. 2017) governs high-level decision-making to enable mapless adaptive navigation. The framework is developed and trained within the ROS and Gazebo simulation environment. Using TinyML principles, the framework is optimized for deployment on embedded platforms with sub-watt power consumption and minimal memory footprint, while maintaining robust autonomous performance in cluttered, unknown environments.

Background and Related Works

Recent research on MAVs has focused on overcoming the severe size, weight, and power (SWaP) constraints of ultra-light platforms (Navardi and Mohsenin 2023). Examples include the Crazyflie 2.1 with the size of 92 mm $\times$ 92 mm, and 27 grams of weight (including battery) (Bitcraze AB 2023; Palossi et al. 2019). Similarly, platforms like PULP-DroNav operate within sub-watt envelopes and weigh under 50 g (Palossi et al. 2019), typically achieving flight times of less than 7 minutes due to limited battery capacity (Floreano and Wood 2015). These extreme constraints have extensively explored lightweight mechanical structures, minimal sensor configurations, and efficient control strategies (Sajjadi and D'Andrea 2022; Banbury et al. 2021).

A significant limitation for such MAVs lies in perception (Navardi et al. 2022). While larger drones often rely on rich sensor suites, including LiDAR, stereo cameras, radar, and depth sensors, for accurate mapping and navigation, these are generally too bulky and power-hungry for drones under 100 grams. For example, a Velodyne VLP-16 LiDAR weighs 830 grams and consumes over 8 watts, while compact alternatives like the RPLIDAR A1 still weigh 248 grams and draw 0.5 watts. Stereo cameras like the Intel RealSense D435 exceed 70 grams and require external compute modules. As a result, tiny drones are restricted to low-weight alternatives such as monochrome cameras, infrared and ultrasonic sensors, and IMUs (Palossi et al. 2019; Banbury et al. 2021; Zhao et al. 2021).

Given these constraints, researchers have explored the combination of lightweight visual and range-based sensors to improve robustness in obstacle-rich or dynamic environments (Suzuki and Amano 2011; Pikalov et al. 2021). However, monocular vision remains the most practical sensing modality due to its efficiency, despite its limitations in field-of-view and sensitivity to lighting conditions (Zhao et al. 2021). To extract control-relevant features from visual inputs, lightweight convolutional neural networks (CNNs) have been trained to predict control actions directly from RGB or grayscale imagery (Tabrizchi et al. 2024a; Loquercio et al. 2018; Tabrizchi et al. 2024b). These models achieve real-time inference on embedded systems, but typically require supervised training on curated datasets, which may not generalize to unseen environments. To address this limitation, additional sensing modalities (Ren et al. 2025; Gaire and Roohi 2025) such as the multi-ranger deck can be integrated. This sensor also offers complementary information that enhances spatial awareness in multiple directions. DRL methods have been employed for mapless navigation to address generalization and enable learning directly from interaction. Algorithms such as PPO enable drones to learn control policies in simulation without explicit mapping (Gandhi et al. 2017; Tai, Paolo, and Liu 2017). Simulators like Gazebo and AirSim provide realistic flight dynamics and modular sensor models, facilitating policy learning and evaluation (Furrer et al. 2016b). Moreover, domain randomization and abstraction techniques help bridge the sim-to-real gap (Sadeghi and Levine 2017; Tobin et al. 2017).

Complementary to DRL, TinyML advances have enabled neural network inference on ultra-low-power hardware. Through quantization and pruning, models with sub-2MB footprints and less than 50ms latency have been deployed on processors like STM32 and GAP8 (Palossi et al. 2019; Banbury et al. 2021). These techniques have enabled real-time onboard inference for applications such as object detection and environmental monitoring in nano-drones (Zhao et al. 2021). Together, these efforts demonstrate the viability of autonomous perception and navigation on resource-constrained MAVs using learning-based methods. However, most prior work remains limited to single-sensor pipelines or relies on high-end offline computation during training. Our work builds on these foundations by proposing a multi-modal, low-latency perception architecture designed for onboard inference and DRL-based policy learning in simulation.

Proposed System

This work aims to develop a lightweight, multi-modal perception and control framework that enables real-time obstacle detection and autonomous navigation on ultra-constrained MAVs using DRL and onboard inference. Since the drone is trained using DRL, the proposed system is developed, trained, and tested within a digital twin of the Crazyflie 2.1 in the Gazebo simulation environment. ROS facilitates communication between different modules (as shown in Fig. 2), ensuring seamless integration and execution of DRL, obstacle detection algorithms, and position control mechanisms.

Obstacle Detection

Obstacle detection is a critical component of our system. To enable fast and interpretable spatial reasoning on resource-constrained platforms, the high-resolution grayscale images are transformed into compact 6×8 binary grids representing obstacle presence, as shown in Fig. 3. We use YOLOv8n, a nano-variant optimized for minimal parameter size and memory footprint, trained on a custom dataset tailored for drone-based obstacle detection. The model processes grayscale input from the onboard camera and outputs

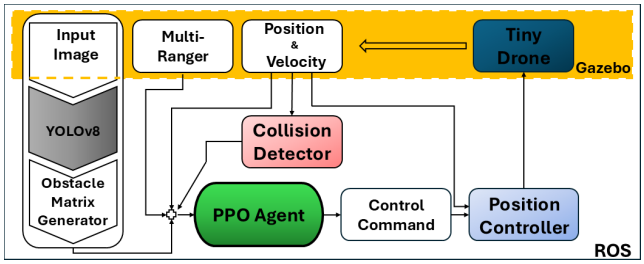

Figure 2: ROS-based system architecture for the autonomous drone. This diagram illustrates the communication flow between various modules, including DRL, obstacle detection, and position control, ensuring seamless execution within the Gazebo simulation environment.

bounding boxes around detected objects. However, since object classification is unnecessary for our application, the system only determines whether an obstacle is present or not. To further structure the perception output, the captured image is divided into a 6×8 grid, where each grid cell is assigned a binary value: 0 indicates no obstacle, while 1 indicates an obstacle within that region, and this 6×8 binary obstacle grid, referred to as the 'obstacle matrix'.

The trained model and obstacle matrix generator are tested in the Gazebo simulation environment, where images are captured directly from the simulation world. Figure 3 illustrates the obstacle detection process. Figure 3(a) shows a snapshot of the simulation environment, including the drone and various obstacles. Figure 3(b) presents the raw grayscale image captured from the drone's onboard camera. Since the physical camera used in Crazyflie 2.1 is monochrome, our system processes grayscale images. Finally, Fig. 3(c) demonstrates the object detection output, where the 6×8 grid representation is overlaid on the captured image. Grid cells containing detected obstacles are highlighted in red. Notably, the tree and the truck are not marked as obstacles due to their distance from the drone, and therefore are not considered immediate hazards for navigation. The processed obstacle matrix is then passed as input to the DRL algorithm.

In addition to detecting obstacles, the trained model can also infer obstacle distance, as demonstrated in Fig. 4. In

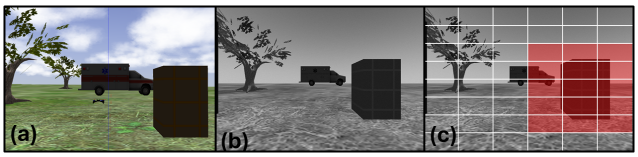

Figure 3: Obstacle detection process in the Gazebo simulation environment. (a) The simulation environment displays the drone and surrounding obstacles. (b) The raw grayscale image was captured from the drone's onboard camera. (c) The processed grid representation, where each cell indicates obstacle presence (highlighted in red). The binary obstacle matrix is subsequently passed to the DRL algorithm for decision-making.

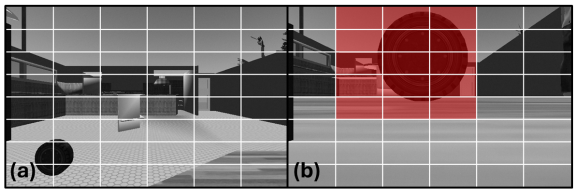

Figure 4: Obstacle detection with distance awareness. (a) A tire positioned at a farther distance remains undetected. (b) The same tire moved closer is identified as an obstacle, demonstrating the model's ability to consider only obstacles within a critical proximity threshold.

Fig. 4(a), a tire placed far from the drone is not detected as an obstacle, while in Fig. 4(b), the same tire is moved closer to the drone and is successfully identified as an obstacle. This distance-awareness capability ensures that the system only considers objects within a critical proximity threshold, improving the drone's navigation efficiency.

While image-based obstacle detection is effective, native depth estimation from images alone is unreliable. To enhance distance measurement, we integrate the multi-ranger deck, a laser sensor that provides precise distance readings. The multi-ranger deck enables the Crazyflie to detect objects with millimeter precision up to a range of four meters. This additional sensing capability significantly improves the drone's ability to navigate complex environments by providing accurate spatial awareness.

Position Control and Trajectory Generation

The position control mechanism in our system is based on the RotorS framework (Furrer et al. 2016a), a widely used open-source simulator for aerial robotics. This controller accurately tracks desired trajectories by considering the full dynamics of the drone, ensuring stable and precise flight. Using this control strategy, our drone can effectively follow waypoints and navigate with minimal deviation from the intended path. The controller translates high-level position and yaw commands into low-level thrust and moment inputs for the Crazyflie 2.1, as illustrated in Fig. 5. The position commands along the x, y, and z axes are generated by the DRL policy, while the yaw angle is kept constant, as the drone does not perform rotational maneuvers in our scenario. The control system relies on a state estimator that fuses onboard IMU data to provide accurate and continuous state feedback, ensuring stable and responsive flight behavior.

Additionally, the controller facilitates smooth transitions between waypoints, minimizing energy consumption and enhancing flight efficiency. Using this robust control method, we optimize drone's flight time while maintaining precise trajectory tracking and real-time adaptability. This makes the position control approach particularly well-suited for the resource-constrained aerial platforms.

Collision Detection

To enable real-time collision detection on a tiny autonomous drone, we implemented a lightweight method based on IMU

data, specifically focusing on the linear acceleration readings. At each timestep t, the total acceleration magnitude a_t is computed from the IMU's three-axis measurements as:

$$a_t = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

where a_x, a_y, and a_z represent the linear accelerations along the drone's principal axes. A sliding window of the most recent N acceleration magnitudes is maintained to capture recent motion behavior. The standard deviation σ over this window is calculated as:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(a_i - \bar{a})^2} \tag{2}$$

where the mean acceleration $\bar{a}$ is given by:

$$\bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i \tag{3}$$

A collision event is inferred when the standard deviation σ exceeds an intuitively predefined threshold, indicating abrupt fluctuations in motion consistent with impact or contact. This method provides an efficient and robust solution for collision detection without requiring additional sensors, making it particularly suitable for the limited computational and energy budgets of tiny drones like the Crazyflie 2.1.

DRL: Obstacle Avoidance and Navigation

To achieve adaptive and autonomous navigation in cluttered environments, we implement a DRL approach based on the PPO algorithm. PPO, a widely used on-policy optimization method, is known for its clipped objective function and entropy regularization, which promote stable policy updates and robust exploration. The algorithm is deployed using the Stable-Baselines3 library within the Gymnasium framework and is tightly integrated with ROS for real-time communication with the simulation environment. The agent, represented by a Crazyflie 2.1 nano-drone, operates within a Gazebo environment populated with static obstacles. At each timestep, the drone receives observations consisting of

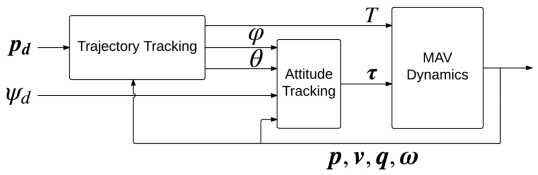

Figure 5: Controller sketch illustrating the position control mechanism. The desired position p_d and yaw angle ψ_d are used as inputs to the controller. The control strategy consists of an outer trajectory tracking controller that generates attitude and thrust references, which are then tracked by an inner attitude tracking controller. The figure also includes the full dynamics of the MAVs.

Num	Observation	Unit
1–3	Relative goal position (x, y, z)	Normalized (-1 to 2)
4–8	Distance measurements (front, back, right, left, up)	Normalized (0 to 1)
9–56	Grid-based obstacle detection (6×8 grid)	Binary (0 or 1)

Table 1: Observations in the Observation Space.

three components: a normalized relative goal position vector, proximity readings from the multi-ranger deck, and a flattened 6×8 binary grid representing obstacle occupancy around the drone, as detailed in Table 1. ROS subscribers continuously update the drone's internal state by listening to odometry, sensor data, collision events, and simulation time.

The action space is discretized into 26 possible movements, each corresponding to a small displacement along the x, y, and z directions. Specifically, for each axis, the drone can move in three ways: a positive step (+1), no step (0), or a negative step (–1). This results in a total of $3^3 = 27$ possible movement combinations across three dimensions. However, the trivial (0, 0, 0) action, which corresponds to no movement at all, is excluded to ensure the drone always attempts to make progress. Thus, the final discrete action space consists of $27 - 1 = 26$ unique directional movements. Each selected action is scaled by 0.001, meaning the drone moves 1 mm in each specified direction per step. Then each selected action translates into a local waypoint adjustment relative to the drone's current position, as described in Table 2. These target waypoints are then sent to the drone's position control module, allowing fine-grained, incremental navigation through complex environments.

Num	Action
1–26	3D movement direction (Δx, Δy, Δz)

Table 2: Actions in the Action Space

The policy network is a fully connected neural network composed of two hidden layers with 256 neurons each, employing ReLU activation functions. The network processes the multi-modal input observations and outputs a probability distribution over the discrete action space, from which the following movement command is sampled.

The reward function is designed to guide the drone toward efficient and collision-free navigation. A dense reward is computed at each timestep based on the change in Euclidean distance to the target; a positive reward is given when the drone moves closer to the target, while moving away from the target results in a small penalty. Upon successfully reaching the target region (within 10 cm), a significant reward is awarded to encourage goal completion. Conversely, collisions with obstacles trigger an immediate negative reward, terminating the episode early. Additionally, if the drone deviates excessively from the mission area or exceeds a maximum number of allowed steps without

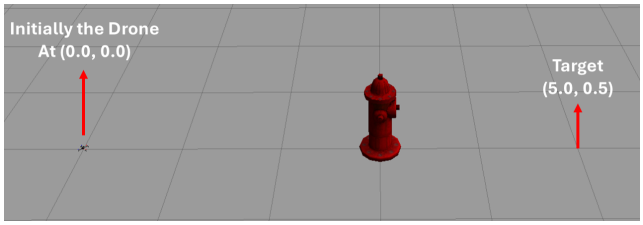

Figure 6: Simulation in the basic environments with a single obstacle.

progress, the episode is also terminated. This reward shaping strategy balances exploration and exploitation, encouraging smooth, goal-directed trajectories while penalizing unsafe behaviors. This reinforcement learning strategy enables the Crazyflie drone to continuously refine its trajectory planning policy, leading to highly adaptive, safe, and efficient navigation behavior even in densely cluttered environments.

Simulation and Training Results

The proposed navigation framework was developed, trained, and evaluated entirely within a ROS and Gazebo simulation environment across two distinct scenarios: a simple environment containing a single obstacle as shown in Fig. 6, and a cluttered environment with multiple randomly placed obstacles as shown in Fig. 7.

At the beginning of each episode, the drone was initialized at a takeoff height of 16cm and tasked with reaching a predefined target position while avoiding collisions. Based on the current observations, the model selected the most appropriate action, which was then executed using a position control interface to update the drone's state. An episode terminated when one of the following conditions was met, the drone successfully reached the target, collided with an obstacle, or exceeded a maximum number of steps without making meaningful progress toward the target for ensuring efficient and focused training.

In the basic environment, a single fire hydrant was positioned approximately 3 m in front of the drone, directly along its path to a target located at coordinates $(5\,\mathrm{m}, 0\,\mathrm{m}, 0.5\,\mathrm{m})$ on the x, y, z axes, respectively (Fig. 6). The PPO model was trained for 20 million steps in this environment. During the early stages of training, the drone's actions were largely random, often resulting in collisions with the obstacle. As the agent experienced repeated failures, it began to associate collisions with penalties and adjusted its behavior accordingly. Gradually, it learned to approach the target while avoiding the obstacle by shifting laterally or adjusting its altitude. At the end of training, the drone consistently executed smooth avoidance strategies, either flying around or above the hydrant, to reach the target successfully.

To test generalization, we evaluated the model trained in the basic environment within the cluttered environment. However, it failed to navigate effectively, likely due to the increased complexity and multi-directional obstacles. We therefore continued training the same policy for an additional 20 million steps in the cluttered environment where

Figure 7: Simulation in the cluttered environment with multiple obstacles.

the target is located at coordinates $(9\,\mathrm{m}, 0\,\mathrm{m}, 0.5\,\mathrm{m})$ on the (x, y, z) axes, respectively. Additionally, a table is placed to make the navigation even harder, as shown in the Fig. 7. Initially, the drone struggled to find safe directions due to surrounding obstacles, but it gradually learned to navigate by leveraging its lateral and vertical movement capabilities. Notably, the agent discovered the shortest path by flying under a table, which demonstrates its ability to reach the target efficiently.

Discussion

To further evaluate the robustness of our model, we conducted experiments in a simulated outdoor environment with a distant target located at $(100\,\mathrm{m}, 0\,\mathrm{m}, 0.5\,\mathrm{m})$ along the (x, y, z) axes, as shown in Fig. 8. In the best-case scenario, the drone was able to travel over 90 meters. However, it frequently chose suboptimal paths, often flying over trees rather than navigating around them, resulting in longer trajectories.

This behavior stems from the YOLOv8n model classifying entire trees (both trunks and leaves) as obstacles, which led to large bounding boxes covering significant portions of the image. As a result, the drone perceived large areas as non-traversable. A potential solution is to retrain the detector on a more granular dataset that separates tree trunks and foliage into distinct classes. This would allow for more precise spatial reasoning and potentially shorter, safer paths.

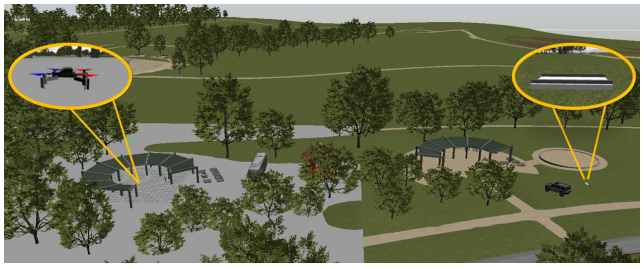

Figure 8: Simulation in the outdoor environment. The drone is targeted to fly $(100\,\mathrm{m}, 0\,\mathrm{m}, 0.5\,\mathrm{m})$ along the (x, y, z) axes and land on the platform magnified in the figure.

Additionally, the drone occasionally failed to recognize narrow or nearby obstacles such as poles and tree trunks, resulting in collisions. These limitations highlight the importance of tailoring obstacle representations to the specific requirements of the application. In practice, the types of obstacles a drone may encounter vary significantly across use cases, requiring datasets and perception models that reflect the operational context of the deployment scenario.

Conclusion and Future Work

In this work, we presented a lightweight, multi-sensor navigation framework for autonomous tiny drones operating under strict SWaP constraints. By combining a grayscale camera, multi-ranger deck, and onboard IMU, and using a YOLOv8n-based grid representation with a PPO-trained policy, the system enables reliable navigation in unknown environments. It leverages compact and complementary sensor inputs while maintaining low latency operation, making it suitable for deployment on microcontroller-class hardware. Simulation results demonstrate that the agent successfully learns collision avoidance behaviors using minimal sensing and discrete control. In the basic environment with a single obstacle, the drone achieved a 100% success rate across 112/112 trials. In a more cluttered environment, it reached the target in 35% of 7/20 trials. These results demonstrate the feasibility of achieving autonomous navigation in mapless environments using lightweight sensor configurations.

Future work will focus on enhancing the system's robustness and reducing flight time during navigation. Improving obstacle detection accuracy and refining the model's ability to distinguish between obstacles and non-obstacles are key priorities. To achieve this, we plan to expand the training dataset with more diverse and representative samples of objects relevant to drone navigation. We plan to deploy on physical drones to validate performance in real-world conditions, accounting for sensing noise and environmental variability.

Acknowledgments

This work is supported in part by the National Science Foundation under Grant Nos. 2504839 and 2448133.

References

Banbury, C.; et al. 2021. Micronets: Neural network architectures for deploying TinyML applications. In *Proceedings of Machine Learning and Systems (MLSys)*.

Bitcraze AB. 2023. Crazyflie 2.1 - Open Source Nano Quadcopter. https://www.bitcraze.io/products/crazyflie-2-1/. Accessed: 2025-07-21.

Floreano, D.; and Wood, R. J. 2015. Science, technology and the future of small autonomous drones. *Nature*, 521(7553): 460–466.

Furrer, F.; et al. 2016a. *RotorS – A Modular Gazebo MAV Simulator Framework*, volume 625, 595–625. ISBN 978-3-319-26054-9.

Furrer, F.; et al. 2016b. RotorS—A modular gazebo MAV simulator framework. In *Robot Operating System (ROS)*, 595–625. Springer.

Gaire, R.; and Roohi, A. 2025. CARN: Complexity-Aware Routing Network for Efficient and Adaptive Inference. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 3318–3326.

Gandhi, D.; et al. 2017. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3948–3955. IEEE.

Jocher, G.; et al. 2023. YOLO by Ultralytics. https://github.com/ultralytics/ultralytics. Accessed: 2025-07-14.

Liu, H.; et al. 2022. TinyML: A survey of state-of-the-art learning frameworks, applications, and future trends. *ACM Computing Surveys (CSUR)*, 55(4).

Loquercio, A.; et al. 2018. DroNet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2): 1088–1095.

Navardi, M.; and Mohsenin, T. 2023. MLAE2: Metareasoning for Latency-Aware Energy-Efficient Autonomous Nano-Drones. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

Navardi, M.; et al. 2022. An Optimization Framework for Efficient Vision-Based Autonomous Drone Navigation. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 304–307.

Palossi, D.; et al. 2019. Ultra-low power deep learning-powered autonomous nano drones. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 1–8. IEEE.

Pikalov, S.; et al. 2021. Vision-Less Sensing for Autonomous Micro-Drones. *Sensors*, 21(16): 5293.

Ren, Y.; Zhu, F.; Lu, G.; Cai, Y.; Yin, L.; Kong, F.; Lin, J.; Chen, N.; and Zhang, F. 2025. Safety-assured high-speed navigation for MAVs. *Science Robotics*, 10(98): eado6187.

Sadeghi, F.; and Levine, S. 2017. CAD2RL: Real Single-Image Flight without a Single Real Image. In *Robotics: Science and Systems*.

Sajjadi, M.; and D'Andrea, R. 2022. NavBench: Benchmarking Visual Navigation for Resource-Constrained Micro Aerial Vehicles. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2897–2904. IEEE.

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

Suzuki, T.; and Amano, Y. 2011. Vision-based localization of a small UAV for generating a large mosaic image. In *2011 IEEE/SICE International Symposium on System Integration (SII)*, 123–128. IEEE.

Tabrizchi, S.; Gaire, R.; Morsali, M.; Liehr, M.; Cady, N.; Angizi, S.; and Roohi, A. 2024a. APRIS: Approximate Processing ReRAM In-Sensor Architecture Enabling Artificial-Intelligence-Powered Edge. *IEEE Transactions on Emerging Topics in Computing*.

Tabrizchi, S.; Reidy, B. C.; Najafi, D.; Angizi, S.; Zand, R.; and Roohi, A. 2024b. ViTSen: Bridging Vision Transformers and Edge Computing With Advanced In/Near-Sensor Processing. *IEEE Embedded Systems Letters*, 16(4): 341–344.

Tai, L.; Paolo, G.; and Liu, M. 2017. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 31–36. IEEE.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; and Abbeel, P. 2017. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23–30. IEEE.

Zhao, Z.; et al. 2021. A tiny drone-based solution for plant monitoring in precision agriculture. *IEEE Transactions on Industrial Informatics*, 17(6): 4043–4052.