MAVRL

Learn to Fly in Cluttered Environments With Varying Speed

Yu, Hang; Wagter, Christophe De; De Croon, Guido C.H.E.

DOI
10.1109/LRA.2024.3522778

Publication date
2025

Document Version
Final published version

Published in
IEEE Robotics and Automation Letters

MAVRL: Learn to Fly in Cluttered Environments With Varying Speed

Hang Yu, ChristopheDe Wagter, and Guido C. H. E de Croon

***Abstract*—Autonomous flight in unknown, cluttered environments is still a major challenge in robotics. Existing obstacle avoidance algorithms typically adopt a fixed flight velocity, overlooking the crucial balance between safety and agility. We propose a reinforcement learning algorithm to learn an adaptive flight speed policy tailored to varying environment complexities, enhancing obstacle avoidance safety. A downside of learning-based obstacle avoidance algorithms is that the lack of a mapping module can lead to the drone getting stuck in complex scenarios. To address this, we introduce a novel training setup for the latent space that retains memory of previous depth map observations. The latent space is explicitly trained to predict both past and current depth maps. Our findings confirm that varying speed leads to a superior balance of success rate and agility in cluttered environments. Additionally, our memory-augmented latent representation outperforms the latent representation commonly used in reinforcement learning. Furthermore, an extensive comparison of our method with the existing state-of-the-art approaches Agile-autonomy and Ego-planner shows the superior performance of our approach, especially in highly cluttered environments. Finally, after minimal fine-tuning, we successfully deployed our network on a real drone for enhanced obstacle avoidance.**

***Index Terms*—Collision avoidance, reinforcement learning, vision-based navigation.**

I. Introduction

OBSTACLE avoidance is a fundamental challenge in autonomous drone technology. While the past decades have seen a proliferation of obstacle avoidance algorithms [1], [2], [3], [4], [5], particularly those based on learning methods, their application within reinforcement learning (RL) frameworks [6], [7] presents unique challenges.

In drone obstacle avoidance, most research sets a fixed or expected speed for drones, leading to low flight efficiency in simple environments [8] and inadequate reaction times in complex ones [1]. In [9], published after our preprint, the concept of adaptive speed for enhanced obstacle avoidance safety was also utilized. However, this study relied solely on reinforcement learning for speed determination and maintained dependence on mapping and traditional path optimization. In autonomous drone RL, choosing between end-to-end learning from raw data [2], [10] and using a more efficient fixed latent space approach [11], [12] significantly influences training efficiency and policy performance. While end-to-end RL offers a thorough learning approach, it demands considerable computational resources and large datasets. Alternatively, condensing high-dimensional image data into low-dimensional latent spaces enhances learning efficiency. However, using only the current state as input, without the mapping module found in traditional obstacle avoidance

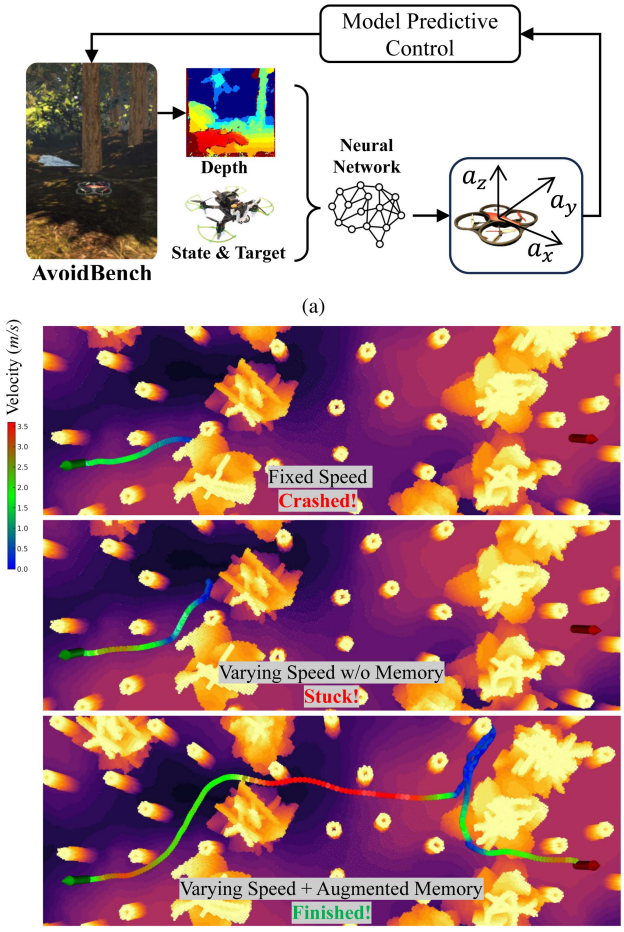

Fig. 1. (a) is the basic framework of MAVRL. (b) illustrates drone's trajectories in a Cluttered Environment. Fixed-speed flight often results in collisions with large obstacles. Absence of augmented memory leads to frequent entrapment in such obstacles. In contrast, MAVRL-equipped flights demonstrate safe and efficient navigation through complex terrains.

Received 5 August 2024; accepted 4 December 2024. Date of publication 25 December 2024; date of current version 3 January 2025. This article was recommended for publication by Associate Editor Wil Thomason and Editor Aniket Bera upon evaluation of the reviewers' comments. This work was supported by the European Commission Horizon project SPEAR under Grant Agreement 101119774. *(Corresponding author: Hang Yu.)*

The authors are with the Faculty of Aerospace Engineering, Delft University of Technology, 2628 Delft, The Netherlands (e-mail: h.y.yu@tudelft.nl; c.dewagter@tudelft.nl; g.c.h.e.deCroon@tudelft.nl).

Code: https://github.com/tudelft/mavrl.git

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3522778, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3522778

algorithms, often causes drones to become stuck in front of complex obstacles.

In our study, we introduce a novel obstacle avoidance pipeline named Memory-Augmented Varying-speed Reinforcement Learning (MAVRL). As shown in Fig. 1(a), MAVRL utilizes depth maps, along with the drone and target's states as inputs, and generates acceleration commands. Following the acceleration generation, model predictive control (MPC) from Agilicious [13] is employed to derive body rates and thrust commands for the drone. We train the pipeline in a simulated environment, featuring randomly generated obstacles of various complexities. Additionally, our approach introduces a novel latent space design that explicitly integrates memory. This latent space induces the drone to remember obstacles it has seen within a certain period of time, even if they are already outside of the field of view. As shown in Fig. 1(b), our memory-augmented latent representation and varying speed policy enable the drone to fly in a safe and efficient way instead of getting stuck in front of large obstacles or colliding.

Our main contributions are as follows:

- A latent space that retains past depth observations by predicting both past and current depths gives the drone a more explicit and structured memory. Extensive ablation studies show that this latent representation outperforms existing solutions in obstacle avoidance tasks.
- A thorough comparison in simulation with state-of-the-art approaches to obstacle avoidance, Agile-autonomy [1] and Ego-planner [8], shows superior performance of our proposed method with varying-speed strategy, especially in cluttered environments.
- The network is effectively deployed on a real drone with minimal post-simulation fine-tuning, demonstrating the practicality of our solution.

II. Related Work

A. Learning-Based Obstacle Avoidance

Recent studies [1], [2], [3] have shown significant advances in learning-based methods for obstacle avoidance. In supervised learning, Agile-Autonomy [1] uses the Metropolis-Hastings method for generating collision-free trajectories, with a neural network learning optimal policies. Reinforcement learning (RL) studies [6], [14] have demonstrated that training in high-fidelity simulators can exceed optimal control performance, with [6] developing 'Swift,' a system surpassing champion-level human pilots. However, such methods often rely on prior knowledge or additional detection modules, restricting their effectiveness in unfamiliar environments. [5] employed reinforcement learning to create a vision-based policy from a teacher policy with comprehensive state information. [15] proposed a 2D navigation planner using LiDAR-based costmaps and the Soft Actor Critic (SAC) algorithm, but the additional weight of LiDAR sensors limits practical deployment. Other methods [7], [10], [16] utilize RGB images as input, transforming them into a latent format conducive to RL training. Beyond supervised and reinforcement learning, self-supervised approaches have been used for obstacle avoidance [4], [17], [18], [19]. Notably, [19] introduced a robotic system that adapts its flying speed to obstacle density, drawing inspiration from flies and bees. We enable drones to perform reinforcement learning across variously complex environments, allowing for a wide range of adaptive speeds.

B. Latent Representations

Given the high dimensionality of visual inputs, the role of latent representation is pivotal in effectively processing this intricate data. Studies like [1], [2], [10] demonstrate the use of depth or RGB images to orchestrate aerial vehicle motions in an end-to-end fashion. However, such methods are not full-proof, particularly in cluttered environments, as evidenced by a sub-optimal success rate [11]. Latent representation is integral to numerous applications, including image classification [20] and vision-based navigation [2], [12], [21]. Study [21] introduced a latent representation for sampling-based motion planning. This representation incorporates AutoEncoders to encapsulate high-dimensional states like images, a dynamics network for predicting the next state, and a collision checker network. Mihir et al. [11] developed a unique collision encoding method for depth images, adept at preserving information about thin objects. When compared with a standard Variational Autoencoder (VAE) [22], their method demonstrated an ability to retain more details with the same latent dimensions. Further, [12] unveiled a learning-based pipeline for local navigation with quadrupedal robots in cluttered settings, featuring a pre-trained state representation. This representation combines a VAE to process depth images and a Long Short-Term Memory (LSTM) network [23] to predict the next latent state.

Our work is inspired by [12], but we focus on enhancing the latent representation to embody more explicit past memories, rather than predicting future states. We have validated that our approach yields superior performance, particularly in cluttered environments with large obstacles.

III. Memory-Augmented Representation

In this section, we present our approach to learning a latent space using a 256-dimensional vector to represent depth. Our method encodes a sequence of depth images, allowing the drone to retain memory of obstacles over time.

As shown in Fig. 2(a), the process starts by using a VAE to convert the current depth image into a latent representation $\mathbf{z}_t^{vae}$. This is processed by an LSTM to generate a final latent state representing past, present, or future depth images. The LSTM output, $\mathbf{z}_t$, is merged with a vector $\mathbf{x}_t$ containing the drone's state and target information, which is then fed into the Proximal Policy Optimization (PPO) algorithm [24] to compute acceleration commands. The pipeline consists of three components: VAE, LSTM, and PPO. The training process is as follows:

- Train a initial PPO policy with fixed and random VAE and LSTM components. This initial policy can navigate the drone to the target without considering obstacles.
- Collect a dataset of depth image sequences using the initial policy and train the VAE, skipping the LSTM phase in this step.
- Train the LSTM with a frozen encoder using the dataset collected from the initial policy.
- Retrain the PPO to obtain an adaptive speed policy for environments of varying complexity.

A. Encoding Depth Images

Our pipeline is based on AvoidBench [25], a high-fidelity simulator with photo-realistic scenes. Instead of directly acquiring depth images, AvoidBench uses a semi-global matching algorithm (SGM) [26] from a virtual stereo camera to replicate

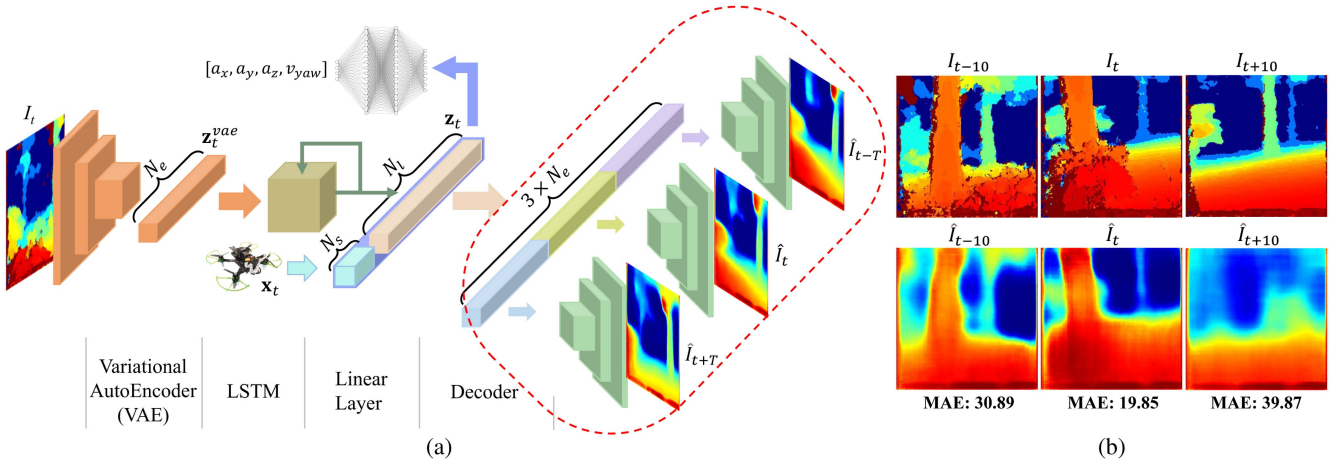

Fig. 2. (a) depicts MAVRL's network architecture. The depth image, encoded into a latent space by VAE, is processed by LSTM to create a memory-augmented representation. This, combined with the drone's state and target data, informs the acceleration command via PPO. The network within the red dotted box, used for LSTM training, is not for reference. (b) compares original and reconstructed depth images from latent space $\mathbf{z}_t$, showing better quality for past and current images than for future ones(highest MAE loss).

realistic depth errors, reducing the gap between simulation and reality. We use AvoidBench to generate depth images for training the VAE.

Consider a depth image at time t, denoted by $I_t \in \mathbb{D}$, where $\mathbb{D}$ is the set of all depth images. We use a VAE to encode each image into a latent space, $\mathbf{z}_t^{vae} \in \mathbb{Z}^{N_e}$, where $\mathbb{Z}^{N_e}$ represents all possible latent spaces and $N_e = 64$ is the dimension of VAE latent space. The VAE training employs an encoder-decoder framework without recurrent structures, using convolutional neural networks for both the encoder and decoder.

The encoder includes six convolutional layers, each followed by a ReLU activation function. The output from the final convolutional layer is flattened and then split into two components by two fully connected layers, representing the mean μ and variance σ^2. The latent space $\mathbf{z}_t^{vae}$ is sampled from a Gaussian distribution characterized by μ and σ^2. The decoder, mirroring the encoder, comprises six deconvolutional layers, each also followed by a ReLU activation function. The output of the last deconvolutional layer passes through a sigmoid activation function to yield the reconstructed depth image I_t^{recon}. The loss function for the VAE is detailed in (1).

$$\begin{aligned} \mathscr{L}_{VAE} &= \mathscr{L}_{recon} + \beta_{norm}\mathscr{L}_{KL} \\ \mathscr{L}_{recon} &= \text{MSE}(I_t, I_t^{recon}) \\ \mathscr{L}_{KL} &= \frac{1}{2}\sum_{i=1}^{N_e}(1 - \mu_i^2 - \sigma_i^2 + \log(\sigma_i^2)) \end{aligned} \tag{1}$$

where β_{norm} is the weight of Kullback-Leibler (KL) loss, I_t^{recon} is the reconstructed depth image from latent space $\mathbf{z}_t^{vae}$. The MSE loss is used to calculate the reconstruction loss $\mathscr{L}_{recon}$, while KL loss is used to calculate the KL divergence between the latent space and the Gaussian distribution.

B. Memory-Augmented Latent Representation

As shown in Fig. 2(a), the VAE output, $\mathbf{z}_t^{vae}$, is input to a single-layer LSTM network. During training, the LSTM output $\mathbf{z}_t \in \mathbb{Z}^{N_l}$ ($N_l = 256$) will be concatenated with the vector $\mathbf{x}_t$ consisting of the drone's state and target information. The combined vector is then passed through a fully connected layer to produce a vector of dimension $3 \times N_e$, which will be split into three segments corresponding to the past, current, and future depth images, $\hat{I}_{t-T}$, $\hat{I}_t$, and $\hat{I}_{t+T}$, all decoded by the same decoder. We use T to represent the number of time steps to predict forward or backward. The LSTM employs a specific loss function described in (2).

$$\mathscr{L}_{\text{LSTM}} = \sum_{i=-1,0,1} \lambda_i \cdot \text{MSE}(I_{t+iT}, \hat{I}_{t+iT}) \quad (\lambda_i \in \{0, 1\}) \tag{2}$$

Where $\hat{I}_{t+iT}$ denotes the reconstructed depth image from the latent space $\mathbf{z}_t$. The coefficient λ_i determines whether the past, current, or future depth image will be reconstructed during training. In the section V, we will show the impact of different λ_i configurations.

Fig. 2(b) shows depth images reconstructed from the latent space $\mathbf{z}_t$. The past and current images are more detailed than future ones, indicating the LSTM module's better encoding of past and present over future depth images. This is likely due to the unpredictability of future events and unseen environmental aspects.

IV. Reinforcement Learning for Obstacle Avoidance

We detail the reinforcement learning algorithm used to train our obstacle avoidance policy. We utilize PPO, a policy gradient method, to optimize the policy network by maximizing expected rewards. We treat obstacle avoidance as a Markov Decision Process (MDP), which structures decision-making in stochastic environments. Our approach, distinct from other RL-based strategies [5], [15], is tailored to handle environments of varying complexity, allowing the policy to adaptively respond to environmental challenges.

A. Problem Formulation

We use the AvoidBench simulator [25] for RL environment setup. Our drone, equipped with a stereo camera, uses PPO for training. For enhanced efficiency, we replace RotorS [27] dynamics model with a simpler kinematics model to allow parallel data generation with multiple drones. The control command includes 3D acceleration and a 1D yaw rate. The kinematics

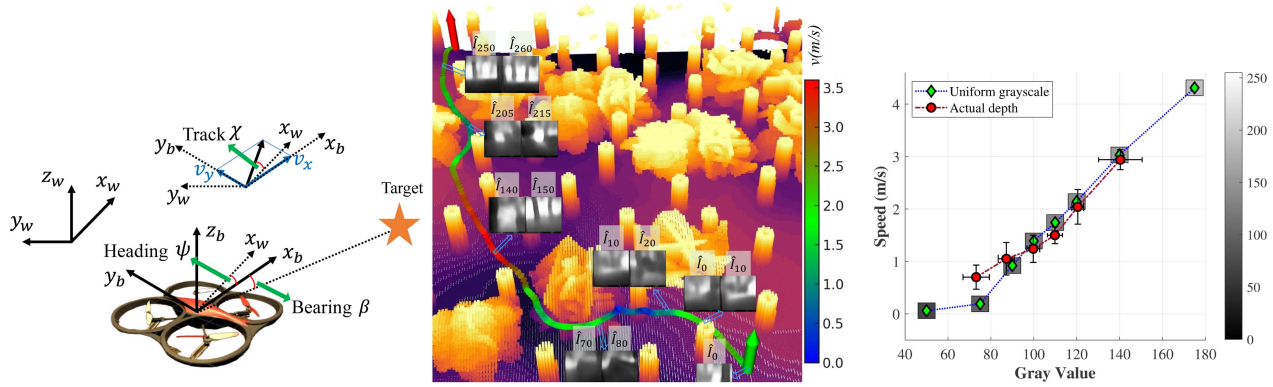

Fig. 3. (a) illustrates the drone's coordinate system, with the bearing angle β between body axis x_b and the target vector, and the track angle χ as the horizontal velocity's direction in the world frame. (b) presents an adaptive drone trajectory in a cluttered environment. The blue hollow arrows point to the reconstructed depth maps at the corresponding positions in the trajectory. The drone decelerates when navigating complex obstacles and accelerates in simpler scenarios, demonstrating dynamic speed adjustment based on obstacle density. (c) displays the drone's average speed in response to uniformly bright gray images (green diamond), and the mean and standard deviation of the speed when the input are actual depth images from (b) (red circles).

model is:

$$\dot{p} = R_b^w v, \quad \dot{v} \quad = a, \tag{3}$$

where p is the drone's position in the world frame, v is its velocity in the body frame, R_b^w is the rotation matrix from the body to the world frame, and a is the body-frame acceleration. This simplified kinematics model is used solely for policy training. For benchmarking against other methods, the RotorS dynamics model is employed.

The Markov Decision Process (MDP) for our model is defined as a tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, \mathscr{R}, \gamma)$, where $\mathscr{S}$ represents the state space, $\mathscr{A}$ denotes the action space, $\mathscr{P}$ defines the transition probability, $\mathscr{R}$ is the reward function, and γ signifies the discount factor. The state space $\mathscr{S}$ comprises the current latent representation $\mathbf{z}_t$, along with the drone's state and target information $\mathbf{x}_t$. The action space $\mathscr{A}$ includes the acceleration in the body frame and the yaw rate. The transition function is deterministic, governed by (3). Thus, for any state s and action a, the transition probability $\mathscr{P}(s'|s,a)$ is 1 for the unique next state s' and 0 for others.

As illustrated in Fig. 2(a), the drone's state and target information at time t, denoted as $\mathbf{x}_t$, is represented by a vector of length N_s. In our specific case, N_s is equal to 7. The coordinate system is shown in Fig. 3(a): the drone's position and velocity in world frame at time t are denoted as $\mathbf{p}(x_t, y_t, z_t)$ and $\mathbf{v}(v_{xt}, v_{yt}, v_{zt})$, respectively, with the target position represented as $\mathbf{p_g}(x_g t, y_g t, z_g t)$. The vector from the drone to the target is $\mathbf{d} = \mathbf{p_g} - \mathbf{p}$, which represented as (d_{xt}, d_{yt}, d_{zt}), and the drone's heading angle is ψ. The bearing angle β is defined as the angle between the body frame's x_b axis and the target vector, while the track angle χ represents the direction of horizontal velocity in the world frame. The drone's state and target information $\mathbf{x}_t$ is defined as:

$$\begin{aligned}
\mathbf{x}_t &= [d_{\text{hor}}, v_{\text{hor}}, \beta', d_{zt}, v_{zt}, \chi', \psi], \\
d_{\text{hor}} &= \ln\left(\sqrt{d_{xt}^2 + d_{yt}^2} + 1\right), \quad v_{\text{hor}} = \sqrt{v_{xt}^2 + v_{yt}^2}, \\
\beta' &= \beta + \psi = \arctan\left(d_{yt}/d_{xt}\right), \\
\chi' &= \chi - \psi = \arctan\left(v_{yt}/v_{xt}\right),
\end{aligned} \tag{4}$$

Here, d_{hor} is the log horizontal distance to the target, v_{hor} the drone's horizontal velocity, v_{zt} the vertical velocity, β' the direction to the target in the world frame, χ' the velocity direction in the body frame, and d_{zt} the vertical distance to the target.

B. Reward Functions

The reward function is designed to ensure that the drone flies safely and efficiently. As reaching the target and avoiding collisions are sparse rewards, we introduce a progressive reward to efficiently guide the drone. The progressive reward is defined as follows:

$$\begin{aligned}
r_{\text{progress}} = \lambda_d \cdot d_{\text{hor}} + \lambda_b \cdot |\chi' + \psi - \beta'| \\
+ \text{sign}(v_{\text{hor}} - v_{\max}) \cdot \lambda_v \cdot v_{\text{hor}} + \lambda_z \cdot d_z \\
+ \lambda_f \cdot |\chi'| + \lambda_a \cdot \|\mathbf{a}_{t-1} - \mathbf{a}_t\|,
\end{aligned} \tag{5}$$

where λ_d, λ_b, λ_v, λ_z, λ_f, and λ_a are weights for each term. $\mathbf{a}_t$ is the acceleration from the policy at time t. The first two terms guide the drone towards the target by penalizing horizontal distance and promoting correct directionality. The third term penalizes high horizontal velocity (activated when $v_{\text{hor}} > v_{\max}$ and $\lambda_v = 0$ for $v_{\text{hor}} < v_{\max}$, where $v_{\max}$ represents the threshold for penalizing horizontal velocity). The fourth term addresses vertical distance, the fifth encourages forward flight, and the last penalizes jerk for smoother flight.

Then the whole reward function is defined as:

$$r = \begin{cases} r_{\text{exceed}} & \text{if } (p_t < p_{\min} \text{ or } p_t > p_{\max}) \\ & (p \in \{x, y, z\}) \\ \dfrac{r_{\text{arrive}}}{TRAV} & \text{if } \|\mathbf{d}\| < d_{\min} \\ r_{\text{collision}} & \text{if collision} \\ r_{\text{progress}} & \text{otherwise} \end{cases} \tag{6}$$

where $p_{\min}$ and $p_{\max}$ are the minimum and maximum values of the coordinates at the boundary, respectively. We define r_{exceed} as the boundary-exceed penalty, r_{arrive} as the target arrival reward which can be obtained when the distance from the drone to the target point is less than $d_{\min}$, and $r_{\text{collision}}$ as the collision penalty. $TRAV$, introduced by Nous et al.[28], measures environmental clutter, accounting for drone's size and complex obstacle shapes. Higher values indicate easier navigation. PPO, trained within a fixed time window, incentivizes faster flight for higher arrival rewards. To balance safety and agility, the arrival reward inversely

correlates with $TRAV$, while the horizontal velocity penalty in (5), moderates speed.

The episode terminates once any of the previously mentioned conditions are met, following which the drone is reset to a new random starting point. In our setup, the values are configured as follows: $r_{\text{exceed}} = -2.0$ for exceeding boundaries, $r_{\text{arrive}} = 10.0$ for reaching the target, and $r_{\text{collision}} = -2.0$ for collisions. The traversability range, $TRAV$, is set between 3 and 13. The progressive reward, r_{progress}, ranges from -0.2 to 0 as defined in (5). Notably, this progressive reward is considerably smaller than the other rewards.

C. Training in Varying Complexity Environments

We use AvoidBench [25] to set up the RL environment, which is designed to test vision-based obstacle avoidance algorithms. AvoidBench builds on Flightmare [29] but includes larger bushes as obstacles, offering more environmental complexity than Flightmare's thin red trees. This complexity is adjustable via the radius of the Poisson distribution.

To enhance training efficiency, we start with a warm-up in a simpler environment (12-meter Poisson radius) to facilitate learning of basic navigation and achieving high rewards. We then increase the complexity (Poisson radius between 3.0 and 5.4 meters) to train the drone on speed adjustment relative to environmental density—speeding up in simpler settings and slowing in denser ones. This adaptive speed feature is crucial for balancing agility and safety in cluttered environments.

As illustrated in Fig. 3(b), the task involves the drone flying from the green arrow (start point) to the red arrow (target). The trajectory is color-coded to represent the drone's velocity. Additionally, we display some predicted depth images generated by our memory-augmented latent representation. These images are presented as pairs of $(\hat{I}_{t-10}, \hat{I}_t)$. For instance, in the pairs $(\hat{I}_0, \hat{I}_{10})$ and $(\hat{I}_{10}, \hat{I}_{20})$, it is evident that the drone retains memory of the depth image $\hat{I}_{10}$ seen 10 timestamps earlier. Observations from pairs $(\hat{I}_{10}, \hat{I}_{20})$, $(\hat{I}_{70}, \hat{I}_{80})$, and $(\hat{I}_{205}, \hat{I}_{215})$ demonstrate the drone's tendency to decelerate when encountering complex obstacles and to accelerate when observed distances in the flight direction are larger, as seen in $(\hat{I}_{140}, \hat{I}_{150})$ and $(\hat{I}_{250}, \hat{I}_{260})$.

We collected depth images and their corresponding speeds along the trajectory in Fig. 3(b) at 0.1 s intervals. The images were divided into six groups based on their average grayscale value, with each group containing an equal number of images. The average grayscale and speed, along with their standard deviations, were then calculated to plot the curve of red circles shown in Fig. 3(c). Speed responses to uniformly bright gray images, represented by green diamonds, are measured during stable zero acceleration periods, closely matching real navigation speeds. The plots show an inverse relationship between drone speed and obstacle proximity, with faster speeds when obstacles are distant and slower speeds as they get closer.

V. Experiments

To assess MAVRL's effectiveness, we conducted a series of experiments. We trained various latent representations to predict past, current, and future depths, then compared their performance using a policy network with consistent parameters.

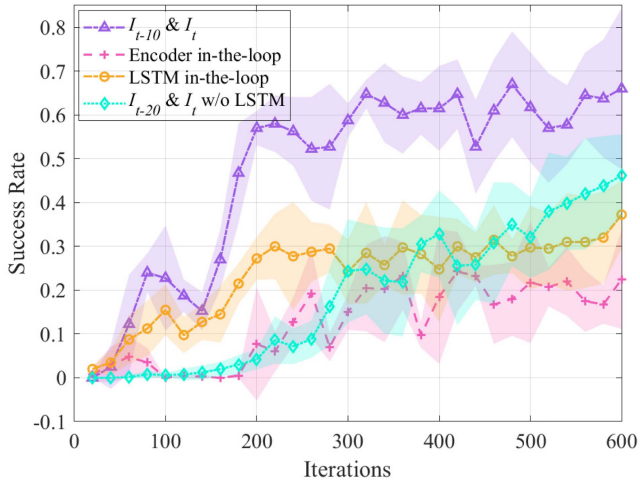

Fig. 4. Average success rates of I_{t-10} & I_t supervised LSTM latent space policy (purple line), encoder in-the-loop without LSTM policy (pink line), LSTM in-the-loop policy (yellow line), and embedded I_{t-20} & I_t without LSTM policy (cyan line). The shaded area represents the standard deviation.

We also evaluated MAVRL's varying speed feature by comparing a fixed-speed training regime to Agile-Autonomy [1] and Ego-planner [8], focusing on success rate and the balance between safety and agility. Finally, we implemented our network on an actual drone with minimal fine-tuning. It should be noted that all training and testing are performed in static obstacle environments.

All simulation experiments are run on a server with an Intel Core i7-13700 K CPU and an NVIDIA GeForce RTX 4090 GPU. To get enough training results for statistical analysis, we create 5 docker containers in the server and run per configuration 5 parallel, independent training processes.

A. Latent Representation

To assess the effectiveness of our latent representation, we conducted several ablation studies. Initially, we trained the network end-to-end, followed by training the policy network with encoder in the loop (no decoder and reconstruction loss). Then, we trained the policy with LSTM in the loop while the encoder was pre-trained as a VAE model.

We trained five policies with the same parameters but different random seeds for each ablation study, conducting 600 iterations with checkpoints every 20. Each policy was tested on four maps, performing 25 trials per map, resulting in $5 \times 4 \times 25$ trials per study. As depicted in Fig. 4, our memory-augmented latent representation (purple line) surpassed both the encoder in-the-loop (pink line) and LSTM in-the-loop (yellow line) policies in obstacle avoidance, with shaded areas showing standard deviations.

Further, to assess the necessity of LSTM, we experimented with inputs of embedded I_{t-20} and I_t from the data buffer directly without an LSTM policy. The cyan line in Fig. 4 indicates that the success rate is significantly higher when using the memory-augmented latent representation with LSTM compared to using embedded inputs alone. This confirms that LSTM's continuous memory capabilities substantially enhance performance in obstacle avoidance tasks.

To evaluate the effectiveness of the memory-augmented latent representation, we conducted an experiment where the LSTM

TABLE I
INFLUENCE OF DIFFERENT T

T	0	1	5	10	15	20	30
success rate	0.50	0.58	0.61	0.64	0.67	**0.74**	0.61

TABLE II
COMPARISON OF DIFFERENT TYPES OF LATENT SPACE

Latent space	Mean $\pm$ std	P-Value (for I_t)	P-Value (for I_{t+10})
I_t	0.500 ± 0.037	-	-
I_{t+10}	0.601 ± 0.083	0.0015	-
I_{t+10} with actions	0.650 ± 0.042	0.0	0.0686
$I_t \& I_{t-10}$	0.636 ± 0.089	0.0001	0.1913
$I_t \& I_{t-20}$	$\mathbf{0.707 \pm 0.051}$	0.0	0.0016
$I_t \& I_{t+10}$	0.692 ± 0.080	0.0	0.0110
$I_t \& I_{t+20}$	0.664 ± 0.100	0.0001	0.0680
$I_{t-20} \& I_t \& I_{t+10}$ with actions	0.700 ± 0.101	0.0	0.0120

was trained using various reconstruction configurations. These configurations were then employed to train the policy network. We investigated eight distinct types of latent representations for this study:

- Current depth image prediction I_t,
- Future depth prediction I_{t+10} with only embedded depth as LSTM's inputs,
- Future depth prediction I_{t+10} with embedded depth, current actions $\mathbf{a}_t$ and states $\mathbf{x}_t$ as LSTM's inputs,
- Current depth I_t, with short-term future I_{t+10},
- Current depth I_t, with long-term future I_{t+20},
- Current depth I_t, and past depth I_{t-10},
- Current depth I_t, with more distant past I_{t-20},
- Current I_t, past I_{t-20}, and future I_{t+10} depth maps, also with current actions $\mathbf{a}_t$ and states $\mathbf{x}_t$ as LSTM's inputs,

where predicting current depth [2] and predicting future depth [12] separately are the most common in the literature. Considering the onboard computing constraints, we set the high-level control frequency to 10 Hz. To assess the impact of different memory lengths, we tested various T values to reconstruct current and past depths ($I_t \& I_{t-T}$). Success rates of obstacle avoidance in the same evaluation environments are shown in Table I. For the ablation studies, we selected $T = 20$, the best value from the fine-tuned hyperparameter results, and used $T = 10$ as a comparison reference.

To evaluate how various latent representations affect policy network performance and the benefits of augmented memory, we conducted an extensive testing regimen. The policy network, using consistent latent representations, was trained ten times, each with a unique random seed, saving checkpoints every 20 iterations for a total of 600 iterations. The evaluation was conducted the same as the ablation study, and the results are detailed in Fig. 5 and Table II.

In Table II, we compared the highest success rate checkpoints for each latent representation, detailing average success rates and standard deviations. Fig. 5 displays the success rates of I_t, I_{t+10} without actions, I_{t+10} with actions, the superior combinations $I_t \& I_{t-20}$ (the better one compared with $I_t \& I_{t-10}$), and $I_t \& I_{t+10}$ (the better one compared with $I_t \& I_{t+20}$). The P-values from permutation tests [30] for each latent representation compared to I_t and I_{t+10} are provided in the last two columns of Table II. A P-value below 0.05 signifies significant differences.

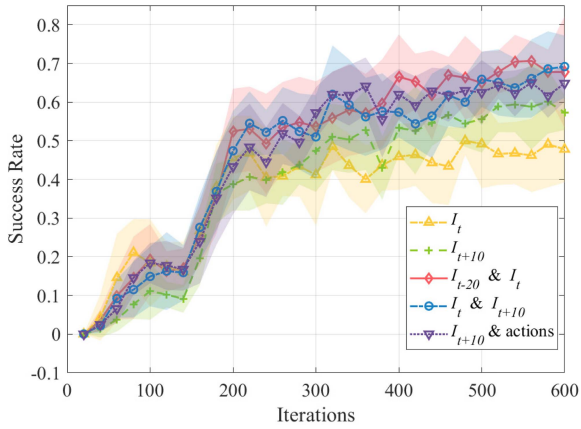

Fig. 5. Success rates of I_t, I_{t+10} without actions, I_{t+10} with actions, and the superior combinations $I_t \& I_{t-20}$ and $I_t \& I_{t+10}$. The shadow area represents the standard deviation.

TABLE III
ERRORS OF DIFFERENT PREDICTION ITEMS

Predicted Item	Mean $\pm$ std (Before PPO retraining)	Mean $\pm$ std (After PPO retraining)
I_{t-20}	26.96 ± 12.01	34.07 ± 12.48
I_{t-10}	22.35 ± 10.09	30.19 ± 12.24
I_t	19.31 ± 8.96	22.66 ± 8.69
I_{t+10}	41.44 ± 22.04	49.46 ± 20.05
I_{t+20}	49.67 ± 25.51	52.68 ± 21.20
I_{t+10} with actions	36.34 ± 19.04	47.55 ± 18.18

For example, the P-value for $I_t \& I_{t-20}$ versus I_{t+10} is 0.0016, suggesting only a 0.16% chance that the results are from the same distribution. Comparatively, the P-value for $I_t \& I_{t-20}$ versus I_t is 0.0, indicating a significant difference.

Thus, we deduce that augmenting current depth with past or future information consistently outperforms predictions based solely on current depth. The combination of I_t and I_{t-20} emerged as the most effective, closely followed by the combination of current and future depth $I_t \& I_{t+10}$ which are both much better than only predicting future depth. Adding actions as input when predicting the future also improves the performance. Predicting I_{t-20}, I_t, and I_{t+10}, along with auxiliary predictions of the drone's state and action, performs similarly to the latent representation obtained by predicting only I_t and I_{t-20}. This leads us to conclude that predicting past depth can be more beneficial in our task.

Since the LSTM training dataset was gathered using an initial policy, we compiled Table III to show the variations in predicting past, current, and future depths across different latent spaces, both before and after retraining with PPO. The mean and standard deviation, calculated from the grayscale values of the depth images, reveal that future prediction errors are the highest, though they slightly improve when actions are incorporated for I_{t+10}. Lower errors indicate that the latent space retains more features (2(b)). Reconstruction accuracy decreases slightly after retraining PPO.

Memory-augmented latent spaces significantly enhance drone navigation in environments with large obstacles, as shown in Fig. 1(b). Drones with memory navigate around large obstacles more effectively, opting for longer, safer paths, while those without memory often become entrapped. This highlights the crucial

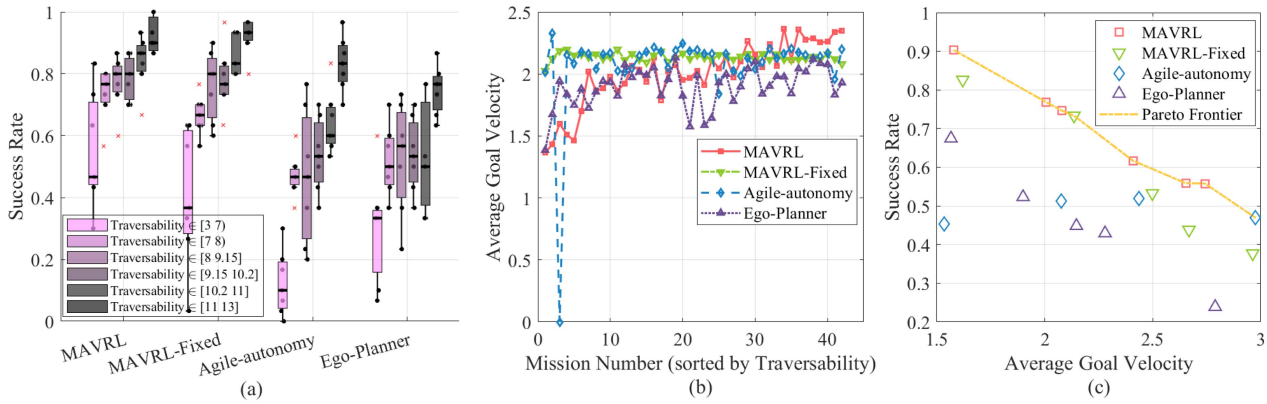

Fig. 6. (a) is the success rate of 2 different MAVRL versions, Agile-Autonomy and Ego-Planner. (b) is the average goal velocity (AGV) of 2 different MAVRL versions, Agile-Autonomy and Ego-Planner. (c) is the Pareto frontier of success rate versus average flight speed.

role of memory in improving obstacle avoidance, particularly with larger obstructions.

B. Benchmarking for Varying Speed Policy

Since predicting past and future depth images significantly improves policy performance, we use the combination of I_t and I_{t-20} as the latent representation for the following experiments. To evaluate the impact of a varying speed policy, we compared a policy network trained with variable speeds against one with a fixed speed. For the fixed speed setup, we modified the velocity penalty in the reward formula (5) to $\lambda_v \cdot |v_{\text{hor}} - v_{\text{desire}}|$, where v_{desire} is the desired velocity. While λ_v was set high, it remained lower than the collision penalty, allowing consistent speed learning. Both of the fixed speed model and varying speed model were then benchmarked against the learning-based Agile-Autonomy method [1] and the optimization-based Ego-planner [8] using the AvoidBench framework [25].

For MAVRL, it utilized the MPC controller from Agilicious [13] when outputting acceleration commands, which then generated body rate and thrust commands for the drone. Fig. 6(a) shows the success rates over six groups, each with seven maps and 30 trials per map (1260 runs per method), indicating MAVRL with varying speed often performs best. Fig. 6(b) explores the link between average goal velocity (AGV) [25] and map complexity, revealing MAVRL with varying speed tends to have higher AGV in less complex environments, while all algorithms maintain a similar AGV (around 2.0 m/s).

To validate MAVRL's superior performance across various agility levels when employing varying speeds, we fine-tuned the reward function parameters of both MAVRL variants to achieve different average flight speeds. This led to the construction of a Pareto frontier of success rate versus average flight speed, as shown in Fig. 6(c). The results confirm that MAVRL with varying speed forms the Pareto frontier, dominating the results of the other methods. However, its average speed could not exceed 3.0 m/s due to flight distance limitations, although the maximum speed reached 5.5 m/s.

C. Real World Tests

To validate MAVRL's real-world efficacy, we implemented our network on a real drone, maintaining the same architecture and hyperparameters as in our simulation experiments. Our test

Fig. 7. Real world test of MAVRL.

setup included a quadrotor equipped with 5-inch propellers and a Realsense D435i camera, powered by a Jetson Xavier NX featuring a 384-core GPU, 48 Tensor Cores, and a 6-core ARM CPU. On the RTX 4090 server, the network inference speed reaches 275 Hz, while on the Xavier NX it reaches 15 Hz. To ensure sufficient onboard computing power for MPC and data recording, we set the acceleration control frequency to 10 Hz on both the simulator and the real drone, with the MPC generating a high-frequency low-level control command at 100 Hz.

Depth images were captured with a Realsense D435i stereo camera facing forward on the drone, which has the same resolution and field of view setups as simulator. The simulation improved depth map accuracy by using the SGM algorithm without distortion, while the Realsense D435i achieved centimeter-level accuracy up to 3 meters in real world. Indoor positioning information is provided by Optitrack, while outdoor positioning information is provided by a RealSense T265 tracking camera.

Due to the real scene's environmental background shown in Fig. 7 being too close to the obstacles, the VAE and LSTM trained in simulation struggled to differentiate between obstacles and background effectively. We collected approximately 1,200 real depth maps and fine-tuned the VAE and LSTM using a smaller learning rate, consistent with the training method described in Section III for these components. When using the PPO network trained in the original simulation environment, the latent space generated by the fine-tuned VAE and LSTM was still able to effectively perform obstacle avoidance navigation tasks.

As shown in Fig. 7, the drone successfully navigated a cluttered environment, utilizing a latent representation augmented with past memory. In the supplementary video, we show that a network fine-tuned indoors can operate outdoors but is more prone to collisions in forests due to small branches and leaves being underrepresented in the latent space.

VI. Conclusion

Our approach leverages memory-augmented latent representations to endow the drone with a recollection of past scenarios. Experimental results demonstrated that reconstructing a more extensive history of past and current depth information significantly enhances the drone's performance in reinforcement learning-based obstacle avoidance tasks. Additionally, we established that adopting a varying speed strategy not only improves success rates but also strikes an optimal balance between safety and agility. The successful deployment of our network on a real drone, requiring minimal fine-tuning, marks a significant achievement. Looking forward, we will focus on improving the prediction and avoidance of dynamic obstacles while ensuring the retention of information about small obstacles, such as branches and leaves.

References

[1] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," in *Sci. Robot.*, vol. 6, Oct. 2021, Art. no. eabg5810.

[2] M. Kulkarni, H. Nguyen, and K. Alexis, "Semantically-enhanced deep collision prediction for autonomous navigation using aerial robots," in *Proc. 2023 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2023, pp. 3056–3063.

[3] H. Nguyen, S. H. Fyhn, P. De Petris, and K. Alexis, "Motion primitives-based navigation planning using deep collision prediction," in *Proc. 2022 Int. Conf. Robot. Automat.*, 2022, pp. 9660–9667.

[4] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1312–1319, Apr. 2021.

[5] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1989–1995.

[6] 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.

[7] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.

[8] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.

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

[10] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," in *Proc. Robot.: Sci. Syst. XIII*, 2017.

[11] M. Kulkarniet al., "Task-driven compression for collision encoding based on depth images," in *Proc. Int. Symp. Vis. Comput.*, 2023, pp. 259–273.

[12] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter, "Learning a state representation and navigation in cluttered and dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5081–5088, Jul. 2021.

[13] P. Foehn et al., "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Sci. Robot.*, vol. 7, no. 67, 2022, Art. no. eabl6259.

[14] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Sci. Robot.*, vol. 8, no. 82, 2023, Art. no. eadg1462.

[15] K. Nakhleh et al., "Sacplanner: Real-world collision avoidance with a soft actor critic local planner and polar state representations," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9464–9470.

[16] M. Kim, J. Kim, M. Jung, and H. Oh, "Towards monocular vision-based autonomous flight through deep reinforcement learning," *Expert Syst. Appl.*, vol. 198, 2022, Art. no. 116742.

[17] K. Lamers, S. Tijmons, C. De Wagter, and G. de Croon, "Self-supervised monocular distance learning on a lightweight micro air vehicle," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1779–1784.

[18] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3948–3955.

[19] T. Schoepe, E. Janotte, M. B. Milde, O. J. Bertrand, M. Egelhaaf, and E. Chicca, "Finding the gap: Neuromorphic motion-vision in dense environments," *Nature Commun.*, vol. 15, no. 1, 2024, Art. no. 817.

[20] A. Sellami and S. Tabbone, "Deep neural networks-based relevant latent representation learning for hyperspectral image classification," *Pattern Recognit.*, vol. 121, 2022, Art. no. 108224.

[21] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2407–2414, Jul. 2019.

[22] C. Doersch, "Tutorial on variational autoencoders," 2016, *arXiv:1606.05908*.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[25] H. Yu, G. C. H. E. de Croon, and C. De Wagter, "Avoidbench: A high-fidelity vision-based obstacle avoidance benchmarking suite for multi-rotors," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9183–9189.

[26] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[27] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—A modular gazebo MAV simulator framework," in *Robot Operating System (ROS)*. Berlin, Germany: Springer, 2016, pp. 595–625.

[28] C. Nous, R. Meertens, C. De Wagter, and G. De Croon, "Performance evaluation in obstacle avoidance," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3614–3619.

[29] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proc. Conf. Robot Learn.*, 2021, pp. 1147–1157.

[30] R. J. Boik, "The fisher-pitman permutation test: A non-robust alternative to the normal theory f test when variances are heterogeneous," *Brit. J. Math. Stat. Psychol.*, vol. 40, no. 1, pp. 26–42, 1987.