Vision-Based Autonomous Navigation of Drone Swarms in Unknown Cluttered Environments via Reinforcement and Imitation Learning

Tom Fähndrich, Charbel Toumieh, Dario Floreano, *Fellow, IEEE,*

***Abstract*—This paper presents a novel approach for vision-based autonomous navigation of drone swarms in cluttered and unknown environments by combining Reinforcement Learning (RL) and Imitation Learning. The proposed dual-policy framework utilizes a state-based teacher policy, trained with privileged information, to guide a vision-based student policy relying on depth images. The student policy can navigate swarm configurations, effectively avoiding both obstacles and inter-agent collisions. Notably, the student policy achieves this without using Convolutional Neural Networks (CNNs), instead leveraging a computationally efficient minpooling technique to process visual inputs. Experimental results demonstrate that the vision-based model achieves a high success rate (above 86%), particularly at lower speeds, despite the complexity of the task. While the teacher policy outperforms the student policy, the latter shows strong generalization capabilities to unseen environments, opening new possibilities for real-world applications for aerial swarms in areas such as search and rescue or environmental monitoring. Future work will focus on improving obstacle representation, incorporating dynamic obstacles, and closing the sim-to-real gap to enhance deployment in practical scenarios.**

***Index Terms*—Reinforcement Learning, Aerial Swarms, Autonomous Navigation, Vision-based Flight.**

I. Introduction

In recent years, the demand for autonomous aerial vehicles capable of navigating complex and cluttered environments while achieving agile maneuvers has grown significantly. This is particularly true for applications involving swarms of drones, such as search and rescue, environmental monitoring, or even the filmmaking industry. The ability to maneuver dynamically, effectively, and safely through unknown terrains is critical. However, developing a navigation policy that not only enables these swarms to operate efficiently in cluttered environments but also generalizes well to new, unknown environments remains a challenging problem. Vision-based navigation, where drones rely on visual inputs to interpret their surroundings, offers a promising solution. Yet, this approach introduces significant challenges due to the high-dimensional nature of visual data and the need for real-time processing in dynamic and unpredictable environments.

Traditional approaches for vision-based aerial navigation typically involve a sequential pipeline consisting of localization, mapping, planning, and control [1], [2], [3], [4]. These methods have shown impressive results even in forest environments [5] where they enable precise and reliable navigation in unknown terrains. They rely on the construction of a detailed map of the environment, followed by the computation of an optimal path through this map and the subsequent control of the aerial vehicles to follow this path precisely. Some approaches also consider swarms [6], [7], [8], [9] but these techniques can lack scalability, especially to effectively achieve decentralized control or do not consider visual inputs. Their heavy computational cost limits the effective scalability of swarm applications, where multiple agents must coordinate their movements in real-time. In addition, the splitting of the problem into sub-tasks, while making it more comprehensible, also makes it more sensitive to unmodeled uncertainties.

Recent advances in Reinforcement Learning (RL) approaches have shown promise in addressing these limitations. In aerial robotics, state-based RL approaches, where the policy is trained using precise state information about the agent and the environment, have demonstrated superior performance in complex tasks, such as learning to fly in seconds [10] or drone racing [11], [12], [13], where RL-trained policies have even

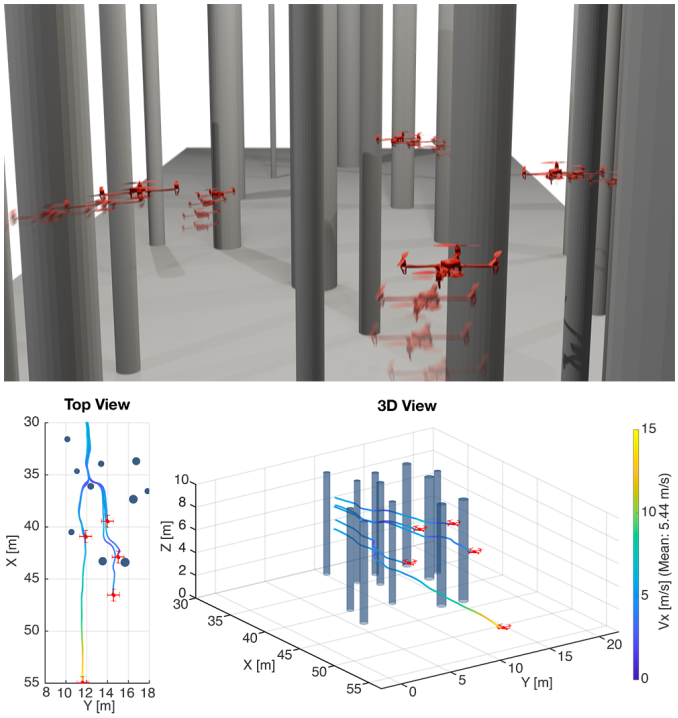

Fig. 1: A swarm of quadrotors navigating autonomously and agilely in a forest-like cluttered environment, while avoiding collisions.

The authors are with the Laboratory of Intelligent Systems, Ecole Polytechnique Federale de Lausanne (EPFL), CH1015 Lausanne, Switzerland.

outperformed human pilots [14]. These works have shown that by learning directly from interactions with the environment, model-free RL can develop policies that are generalizable and capable of handling dynamic and agile navigation without the need to model the high complexity of the task. These achievements also apply to other areas of mobile robotics [15], [16]. However, extending these successes to vision-based RL for autonomous navigation in unknown cluttered environments and obstacle avoidance in drone swarms remains a significant challenge. Vision-based RL requires the processing of visual inputs, which drastically increases the dimensionality of the observations, making exploration more difficult due to the vast and complex state space that needs to be navigated to discover optimal policies. This also increases the computational cost required for rendering images during training thus slowing down the learning process, as millions of interactions with the environment are typically required to achieve a robust policy. The approaches presented in [17], [18] successfully tackle these issues using asymmetric actor-critic [19] but lack generalizability to new environments not seen in training and does not consider swarms.

Imitation learning [20], [21] has been used to address some of these challenges by enabling policies to be trained through the demonstration of expert behavior, thus reducing the number of interactions required during training. This method can significantly speed up the training process and alleviate the computational demands associated with Reinforcement Learning. For example, studies such as [22] and [23] have demonstrated high-speed, agile obstacle avoidance using vision or performing acrobatic maneuvers in natural environments, guided by a sensorimotor policy learned from an optimal controller. Similarly, [24] presents a method for learning vision-based agile flight by learning from a Reinforcement Learning teacher policy that leverages privileged information about obstacles. However, imitation learning often suffers from poor generalization to new, unseen environments, as the policy tends to mimic the specific scenarios encountered during training and can lack exploration. Additionally, most of the presented imitation learning studies are designed for individual agents, making it difficult to extend these methods to swarm scenarios.

In this work, we propose an approach inspired by [24] that combines the strengths of RL and imitation learning to tackle the challenges of vision-based navigation for aerial swarms. We introduce a dual-policy framework in which a teacher policy, trained using state-based RL with privileged information, guides the learning of a student policy that relies on visual depth features. The teacher policy is designed to track a reference speed while navigating efficiently through cluttered environments using exact obstacle locations, providing a robust benchmark for the student policy. The student policy, trained via imitation learning, learns to replicate the teacher's decisions based on visual depth observations. We demonstrate that this method is sample efficient and has generalization capacities to unseen environments for both policies achieving success rates of over 98% in the state-based configuration and over 86% in the vision-based configuration at 3 m/s. Additionally, our approach is scalable to swarm scenarios, allowing multiple aerial vehicles to coordinate their movements and avoid collisions thanks to onboard vision and shared positions (Fig. 1).

II. Method

A. Overview

We tackle the task of vision-based autonomous navigation of swarms of quadrotors in cluttered environments using an imitation learning framework that divides the problem in two parts. First, using a Deep Reinforcement Learning (DRL) framework, we train a state-based policy that leverages omniscient information about the exact position of obstacles close to the agents to produce a controller capable of navigating efficiently through obstacles while tracking a given speed. Secondly, a student policy, which no longer benefits from privileged information but instead uses a depth camera, is trained to replicate the teacher's actions based on sensory observations (Fig. 2 shows an overview of the framework).
The agents are modeled as simple discrete-time systems where the evolution is described by $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \Delta t \cdot f(\boldsymbol{x}_t, \boldsymbol{u}_t)$ with $\boldsymbol{x}_t, \boldsymbol{x}_{t+1} \in \mathcal{X}$ the states of the system, $\boldsymbol{u}_t \in \mathcal{U}$ the input, and $\Delta t = 0.02$s the discrete time step. The physics considered for the simulation is a simple point mass model that expresses the agents' motion as:

$$\begin{aligned} \boldsymbol{v}_{t+1} &= \boldsymbol{v}_t + \boldsymbol{a}_t \cdot \Delta t \\ \boldsymbol{p}_{t+1} &= \boldsymbol{p}_t + \boldsymbol{v}_t \cdot \Delta t \\ \psi_{t+1} &= \arctan\left(\frac{v_{y,t+1}}{v_{x,t+1}}\right) \end{aligned} \tag{1}$$

where $\boldsymbol{p}_t = [p_{x,t}, p_{y,t}, p_{z,t}]^T \in \mathcal{X}$, $\boldsymbol{v}_t = [v_{x,t}, v_{y,t}, v_{z,t}]^T \in \mathcal{X}$ and $\boldsymbol{a}_t = [a_{x,t}, a_{y,t}, a_{z,t}]^T \in \mathcal{U}$ are respectively the position, velocity and acceleration vectors of the quadrotors in the World frame at time t. The yaw angle $\psi_t \in \mathcal{X}$ of the quadrotors is defined such that the agent always aligns with the motion to ensure the depth camera keeps track of obstacles ahead. The maximum acceleration allowed is 15 m/s^2 in each axis.

B. State-based teacher policy

The teacher policy is a two-layer MLP (512 $\times$ 512) based on a DRL framework with the privileged knowledge of the exact relative position of the agent's closest obstacles.

1) Deep Reinforcement Learning: The task proposed is described by several agents navigating the same environment and sharing relative positions between their observation spaces. The proposed framework is a Markov Decision Process (MDP) formulation represented by the tuple $(\mathcal{O}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. The agent collects an observation $\boldsymbol{o}_t \in \mathcal{O}$, takes an action $\boldsymbol{a}_t \in \mathcal{A}$ drawn from a stochastic policy $\pi_\theta(\boldsymbol{a}_t \mid \boldsymbol{o}_t)$, then transitions to a new observation $\boldsymbol{o}_{t+1}$ with the probability $\mathcal{P}(\boldsymbol{o}_{t+1} \mid \boldsymbol{o}_t, \boldsymbol{a}_t)$ given by the probability function $\mathcal{P} : \mathcal{O} \times \mathcal{A} \times \mathcal{O} \to \mathbb{R}$ and collects a direct reward $r_t \in \mathcal{R}$. The goal for the agent is to interact with the environment by selecting the actions that maximize future expected discounted reward and therefore optimize the

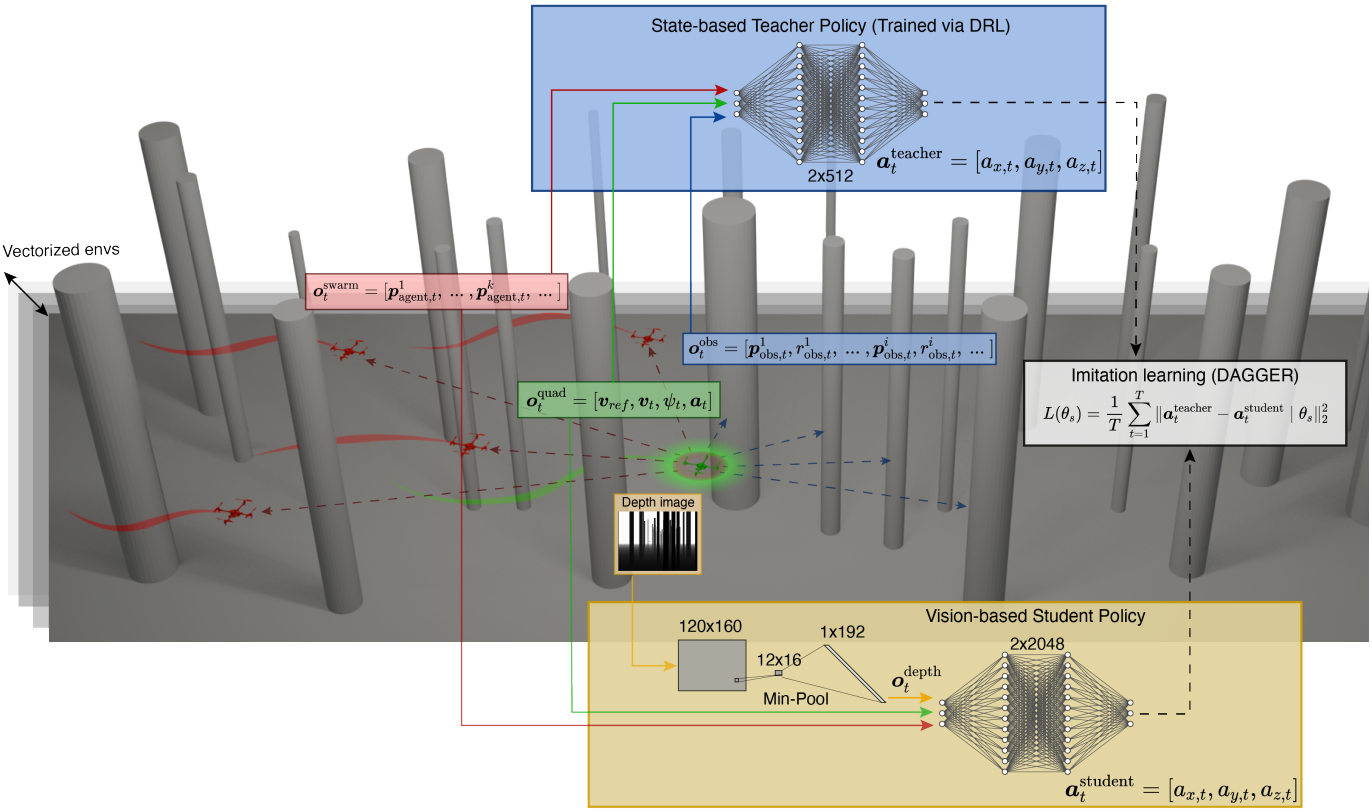

Fig. 2: **Method overview: Imitation learning framework**. The custom Flightmare [25] simulator updates the quadrotor observations (o_t^{swarm}, o_t^{quad}, o_t^{obs}) and depth image from the onboard camera at each time step and for each of the parallelized environments. Privileged information about the exact relative position of the obstacles is provided to the teacher policy which has been trained using DRL and outputs the teacher action (a_t^{teacher}). The depth image (160x120) is supplied to the student network, which first downsamples it to 16x12 by minpooling. It then directly feeds the flattened depth features (o_t^{depth}) to the student network to output the student action (a_t^{student}). A DAGGER [26] imitation learning framework is used to regress the student's action.

parameters θ of the policy to achieve this objective. It can be formulated as:

$$\pi^*_{\boldsymbol{\theta}} = \underset{\pi}{\text{argmax}}\ \mathbb{E}_{\boldsymbol{o}_t, \boldsymbol{a}_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{2}$$

With $\lambda \in [0, 1[$ the discount factor reducing the weight of long-term versus short-term rewards. The algorithm chosen in this work to identify the optimal policy π^*_θ leading to maximization of the discounted expected reward is Proximal Policy Optimization (PPO) [27]. This method offers a balance of stability and efficiency, providing reliable performance with simpler implementation and tuning.

Action Space: At each time step t the policy outputs an action $\boldsymbol{a}_t = [a_{x,t}, a_{y,t}, a_{z,t}] \in \mathbb{R}^3$ based on the observation $\boldsymbol{o}_t$ and corresponding to the acceleration of the drone in the World frame. This high-level command is chosen because it simplifies the learning process by allowing the policy to focus on the navigation and path-planning of the quadrotors. This command could then be mapped into a thrust command for each motor by a low-level controller embedded in real hardware.

State-based Observation Space: To output actions enabling agents to navigate in cluttered environments, the teacher policy has access to three main components: the state vector of the quadrotor being controlled $\boldsymbol{o}_t^{\text{quad}}$, the position and size of nearby obstacles $\boldsymbol{o}_t^{\text{obs}}$, and the positions of other agents in the swarm $\boldsymbol{o}_t^{\text{swarm}}$. The state vector representing the quadrotor $\boldsymbol{o}_t^{\text{quad}} = [\boldsymbol{v}_{\text{ref}}, \boldsymbol{v}_t, \psi_t, \boldsymbol{a}_t] \in \mathbb{R}^{10}$ is composed of the linear reference velocity vector, the quadrotor's linear velocity vector, the yaw in Euler angles and the linear acceleration vector. This observation brings essential informations on the actual state of the drone to achieve smooth trajectories and velocity tracking. The obstacles observation vector $\boldsymbol{o}_t^{\text{obs}} = [\boldsymbol{p}^1_{\text{obs},t}, r^1_{\text{obs},t}, \dots, \boldsymbol{p}^i_{\text{obs},t}, r^i_{\text{obs},t}, \dots] \in \mathbb{R}^{18}$ with $i \in [1, \dots, 6]$ represents the relative position $\boldsymbol{p}^i_{\text{obs},t} = [x^i_{\text{obs},t}, y^i_{\text{obs},t}] \in \mathbb{R}^2$ and radius $r^i_{\text{obs},t} \in \mathbb{R}$ of the six closest obstacles to the agent. If the relative distance to some of the obstacles is superior to 10 m, the position vector is set to $\boldsymbol{p}^i_{obs,t} = [10.0, 10.0]$ to indicate their non-proximity and if there is not enough obstacles in the environment the radius is in addition set to 0. The privileged observations about the ground truth position and radius of the closest obstacles allow the emergence of a seamless obstacle-avoiding policy. The swarm observation vector $\boldsymbol{o}_t^{\text{swarm}} = [\boldsymbol{p}^1_{\text{agent},t}, \dots, \boldsymbol{p}^k_{\text{agent},t}, \dots] \in \mathbb{R}^{12}$ with $k \in [1, \dots, 4]$ encodes the relative position to the four other agents composing the swarm with $\boldsymbol{p}^i_{\text{agent},t} = [x^i_{\text{agent},t}, y^i_{\text{agent},t}, z^i_{\text{agent},t}] \in \mathbb{R}^3$ avoiding inter-agent

collisions.

Reward Function: The reward function r_t is designed to encode the task of agile and coordinated navigation of the agents while avoiding collisions (against both obstacles and agents) and tracking the predefined reference velocity $\boldsymbol{v}_{ref}$. The reward is computed at each time step as:

$$r_t = \begin{cases} -1000, & \text{if } \mathbf{p} \notin [\mathbf{p}_{\text{min}}, \mathbf{p}_{\text{max}}] \\ -450, & \text{if collision} \\ 10, & \text{if } p_{x,\text{goal}} \text{ reached} \\ r_t^{\text{tot}}, & \text{otherwise} \end{cases} \tag{3}$$

$$r_t^{\text{tot}} = r_t^{\text{vel}} + r_t^{\text{obs}} + r_t^{\text{swarm}} + r_t^{\text{acc}} + r_t^{\text{surv}} \tag{4}$$

With individual rewards encrypting each task as:

$$\begin{aligned} r_t^{\text{vel}} &= \lambda_1 \left\| \boldsymbol{v}_t - \boldsymbol{v}_{\text{ref}} \right\|^2 \\ r_t^{\text{obs}} &= \begin{cases} \sum_{i=1}^{6} \lambda_2 \, \exp(-d_{\text{obs},t}^i), & \text{if } d_{\text{obs},t}^i < 0.5 \\ 0, & \text{otherwise} \end{cases} \\ r_t^{\text{swarm}} &= \begin{cases} \sum_{k=1}^{4} \lambda_3 (d_{\text{quad},t}^k - 3.0), & \text{if } d_{\text{quad},t}^k > 3.0 \\ \sum_{k=1}^{4} \lambda_4 \, \exp(-d_{\text{quad},t}^k), & \text{if } d_{\text{quad},t}^k < 0.7 \\ 0, & \text{otherwise} \end{cases} \\ r_t^{\text{acc}} &= \lambda_5 \left\| \boldsymbol{a}_t \right\| + \lambda_6 \left\| \boldsymbol{a}_t - \boldsymbol{a}_{t-1} \right\|^2 \\ r_t^{\text{surv}} &= \begin{cases} 0, & \text{if collision} \\ 0.06, & \text{if } 2.0 < p_z < 8.0 \\ 0.03, & \text{otherwise} \end{cases} \end{aligned} \tag{5}$$

With $d_{\text{obs},t}^i = \left\| \boldsymbol{p}_{\text{obs},t}^i \right\|$ the distance between the center of the quadrotor and the center of the obstacle i, $d_{\text{quad},t}^k = \left\| \boldsymbol{p}_{\text{quad},t}^k \right\|$ the distance between the center of the quadrotor and the center of the quadrotor k and $[\mathbf{p}_{\text{min}}, \mathbf{p}_{\text{max}}]$ the boundaries of the map. The reward component r_t^{vel} penalizes for not tracking the reference speed vector, r_t^{obs} penalizes for passing too close to the obstacle i, r_t^{swarm} encourages swarm cohesion and avoids inter-agent collisions, r_t^{acc} penalizes huge acceleration commands and jerk, and r_t^{surv} rewards agent survivability as well as appropriate flight altitude. The coefficients λ_1 = -0.065, λ_2 = -1.5, λ_3 = -0.0007, λ_4 = -1.0, λ_5 = -0.0009, λ_6 = -0.00007 are chosen empirically to ensure proper agents' actions and a performance trade-off between the main tasks.

2) Policy Training: The teacher policy is trained in simulation using a custom version of Flightmare [25] adapted from [28]. The adapted version allows us to include multiple quadrotors equipped with depth cameras in the same simulation environment, to share the position of the agents within the swarm, to detect inter-agent collisions, and to generate random and customized obstacle configurations. The policy is trained using 50 parallelized environments, each containing a swarm of five quadrotors collecting interactions simultaneously and the PPO implementation is based on [29]. This parallelized sampling strategy allows to considerably accelerate the training by collecting up to 110'000 interactions per second on an Intel® Core™ i9-13900K and NVIDIA GeForce RTX 4090. The PPO parameters used for training are presented in Tab. I.

TABLE I: Hyperparameters used for PPO

Parameter	Value
learning rate	3e-4
discount factor	0.99
GAE-λ	0.95
n_steps	500
clip range	0.2
entropy coefficient	0.0
value function	0.5
batch size	50000
policy network MLP	[512, 512]
value network MLP	[512, 512]

Domain randomization: With the aim of making the policy generalizable to trajectories never seen during training, we implemented a domain randomization strategy to maximize the potential of different scenarios and exploration capabilities. The training environments consist of 100 random configurations of cylinders (x and y locations) with random radii [0.4, 1.0] m over a 80×20×12 m map (Fig. 3). At each reset, and for every vectorized environment, a new configuration is randomly chosen. Each cylinder's exact position is additionally offset by an amount in the range [-1, 1] m in x and y positions. In addition, the initialization position of each agent in the swarm ($p_{x,0}$, $p_{y,0}$) is also offset by a random amount between [-0.5, 0.5] m to randomize the swarm configuration and the trajectories taken.

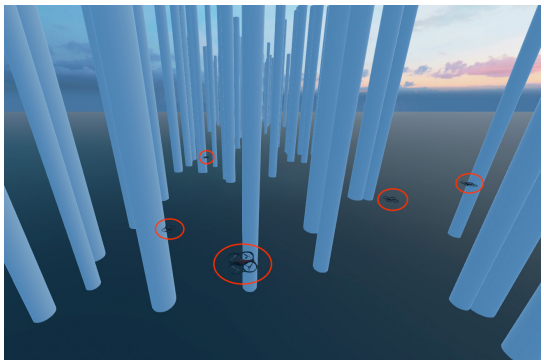

Fig. 3: Overview of a training environment with the swarm of quadrotors and randomly placed cylindrical obstacles.

C. Vision-based student policy

The student policy no longer benefits from omniscient information on the position and size of obstacles, but instead relies on the output of a depth camera mounted on each quadrotor. The action space remains the same as for the teacher (see Sec. II-B1), but the observation space changes. It now consists of the current quadrotor state vector ($\boldsymbol{o}_t^{\text{quad}} = [\boldsymbol{v}_{\text{ref}}, \boldsymbol{v}_t, \psi_t, \boldsymbol{a}_t] \in \mathbb{R}^{10}$), the relative position of the other agents composing the swarm ($\boldsymbol{o}_t^{\text{swarm}} = [\boldsymbol{p}_{\text{agent},t}^1, \ \dots \ , \boldsymbol{p}_{\text{agent},t}^k, \ \dots \ ] \in \mathbb{R}^{12}$) and the depth features extracted from the image ($\boldsymbol{o}_t^{\text{depth}} \in \mathbb{R}^{192}$) for obstacle avoidance. Moreover, as a result of the increase in the observation dimensionality associated with the replacement of privileged information on obstacles by depth features, we

also increased the student MLP network to 2048 $\times$ 2048.

1) Depth features: Most of the existing similar works tackle the problem of extracting relevant depth features with a neural network architecture using multiple layers of CNN and then concatenating the output with the other considered states. Some of them directly add CNN layers to the neural controller and thus to the training loop [17], [30], while others pre-train or use a pre-trained autoencoder to extract features [24], [22]. In this work, we achieved the best performance by simply downsampling the images while ensuring to keep track of the closest objects. To achieve this, the depth images provided by the quadrotor's on-board camera (160x120 pixels) are passed through a minpooling filter to downsample them to 16x12 pixels, retaining information on the closest objects and thus maximizing caution (Fig. 4). The downsampled image is then flattened and the 192 depth features $\boldsymbol{o}_t^{\text{depth}}$ are concatenated with the other observations ($\boldsymbol{o}_t^{\text{quad}}$, $\boldsymbol{o}_t^{\text{swarm}}$).

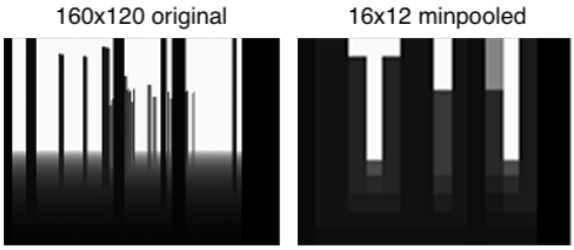

Fig. 4: On the left an original image (160x120 pixels) outputted by the depth camera placed on a quadrotor. On the right, the same image downsampled by minpooling (16x12 pixels) to keep track of the closest objects.

The absence of CNN layers in our approach can be explained by the simplicity of the obstacle shapes, which are cylindrical and do not require complex feature extraction across multiple channels. The downsampled minpooled image effectively serves as an occupancy map for the scene ahead, making additional feature extraction unnecessary.

D. Imitation learning

The imitation learning framework used to teach the state-based teacher policy (π^t) actions to the vision-based student policy (π^s) is a modified Dataset Aggregation (DAGGER) method (Alg. 1) inspired by [26]. This allows to iteratively refine the student policy by aggregating data from both the teacher and the student. This approach reduces the compounding errors that typically arise in classical supervised learning, where the student is trained on a static dataset and may not learn to handle bad or unfamiliar states effectively. By exposing the student to a wider range of scenarios, including those resulting from its own mistakes, it helps create a more robust and adaptable policy.

The algorithm is structured as follows: The student policy is initialized and pre-trained using DRL and PPO until a straight-line flying behavior is achieved. This step is important to facilitate and accelerate the convergence of DAGGER, as it allows the exploration of a large number of states and the pre-tuning of the policy. This pre-trained student policy π_1^s is then used in weighted combination with the teacher policy (line 6) to collect T = 250 steps trajectories for each agent in the swarm. The student's observations ($\boldsymbol{o}^s$) and expert actions ($\pi^t(\boldsymbol{o}^t)$) aggregates the dataset $\mathcal{D}$. The student policy is then re-trained and updated ($\pi_i^s \rightarrow \pi_{i+1}^s$) on the newly expanded dataset. This process is iterated until convergence is reached.

Algorithm 1 Custom DAGGER

1: Initialize π_1^s to any vision-based student policy
2: Initialize π^t the state-based teacher policy
3: Pre-train π_1^s using DRL until forward flight capacities
4: Initialize $\mathcal{D} \leftarrow \emptyset$
5: **for** $i = 1$ to N **do**
6: Let $\pi_i = \beta^i \cdot \pi^t + (1 - \beta^i) \cdot \pi_i^s$
7: Sample T-step trajectories using π_i
8: Get dataset $\mathcal{D}_i = \{(\boldsymbol{o}^s, \pi^t(\boldsymbol{o}^t))\}$ of collected observations by π_i and actions given by teacher
9: Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
10: Train student policy π_{i+1}^s on $\mathcal{D}$
11: **end for**
12: **return** best π_i^s on validation

The coefficient $\beta \in [0, 1]$ allows to balance the weight of teacher and student policies in the modified policy π_i. This increases datatest quality at the start and its importance decreases exponentially with iterations as the student policy learns. We decided to choose β = 0.99 which delivers smooth learning in most cases. The loss function used to regress student action is a standard L2 loss:

$$L(\theta_s) = \frac{1}{T} \sum_{t=1}^{T} \| \boldsymbol{a}_t^{\text{teacher}} - \boldsymbol{a}_t^{\text{student}} \mid \theta_s \|_2^2 \tag{6}$$

With θ_s, the parameters of the student policy network.

III. Results

The experiment consists of making a swarm of five agents evolve in a cluttered environment similar to those observed during training (randomly placed cylinders simulating a forest (Fig. 3)) but unknown i.e. never seen during training. To quantify performance, we will compare the results obtained by the vision-based policy to those obtained by the state-based teacher policy, which constitutes the baseline. The questions we want to answer are as follows: (i) What is the performance degradation of the vision-based model compared to the baseline? (ii) Is the vision-based policy capable of generalizing to new, unknown environments? The metrics considered to assess these performances are the following: (i) The Success Rate (SR) [%] defined by the percentage of drones able to cross the environment without colliding (with another agent or an obstacle). (ii) The average velocity [m/s] achieved along the trajectory for a given reference velocity.

The evaluation is carried out at three discrete target velocities (3, 5, and 7 m/s) in unknown, 45$\times$20$\times$12 m cluttered environments, and results are averaged over 100 experiments. To carry out these experiments, we trained three state-based teacher policies and three vision-based student policies at

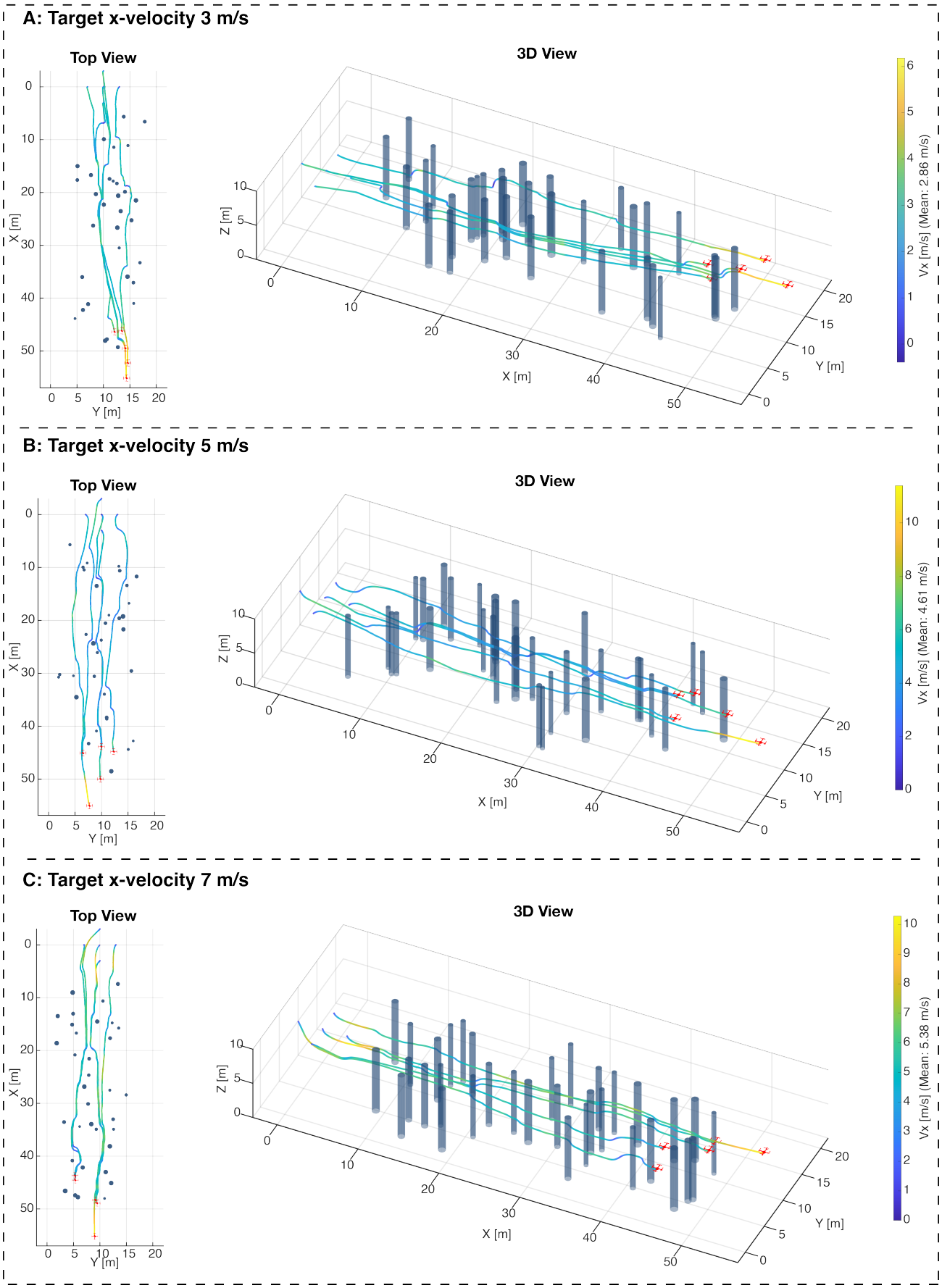

Fig. 5: Comparison of 5 quadrotors swarm trajectories (top view on the left and 3D on the right) obtained with the vision-based student policy on 3 different evaluation obstacles configurations for the 3 target v_x (**A**: 3 m/s, **B**: 5 m/s, **C**: 7 m/s). The trajectory is colored according to the agent's x-velocity profile.

the three considered benchmark speeds. Around 250 million environment interactions using PPO are required for teacher convergence, compared with roughly 1 million interactions using PPO plus 375'000 interactions using DAGGER for the student.

As expected, the state-based teacher performs better than the vision-based student for all three reference velocities. An SR of 98.4% is obtained for the lowest speed and decreases to 94.6% for the highest speed. In comparison, the vision-based version achieves an SR of 86.6% for a reference velocity of 3 m/s, decreasing to 78.4% at the highest target velocity (Tab. II). Despite the reduction in SR, the vision-based policy remains highly capable of navigating through unknown obstacle configurations without any privileged information about the obstacles, especially for lower target velocities. The resulting trajectories demonstrate agile and appropriate maneuvers for reliable navigation, highlighting awareness of the surrounding obstacles while considering three velocities (Fig. 5). Regarding velocity tracking, the results obtained for the teacher and the student are fairly similar. There is no clear difference between the two policies. For the 3 and 5 m/s target speed, the controllers achieve an average speed close to the requested one. For a target velocity of 7 m/s the policies obtain a lower average speed. This is explained in particular by the fact that the acceleration is limited to 15 m/s^2 limiting the manoeuvrability while maintaining high speed.

TABLE II: Results comparison between state-based and vision-based policy considering swarms of five agents for the three target x-velocities (3 m/s, 5 m/s, 7 m/s) of mean v_x distribution [m/s], Success Rate [%] and max v_x averaged over 100 experiments on unknown environments.

Target v_x	Policy	Mean v_x [m/s]	SR [%]	Max v_x [m/s]
3 m/s	Vision-based	2.77 ± 0.49	86.6	6.14
	State-based	2.80 ± 0.54	98.4	8.81
5 m/s	Vision-based	4.47 ± 1.23	79.6	14.53
	State-based	4.67 ± 1.05	96.0	9.49
7 m/s	Vision-based	5.17 ± 1.68	78.4	13.40
	State-based	5.49 ± 1.54	94.6	13.38

We can observe policies having developed inherent obstacle avoidance strategies that minimize the risk of inter-agent collisions. To achieve this, quadrotors tend to operate at different cruising altitudes to avoid interfering with each other (Fig. 6), as it would be the case in the aviation industry. Obstacle avoidance is dynamic, without obstructing the trajectory of other agents. Moreover, agents naturally tend to modulate their speed in high obstacle density zones and to accelerate when the path is clear.

Finally, if we compare the trajectories followed by the state-based and vision-based models, we can see that they are fairly similar (Fig. 7). The routes taken are slightly different but remain comparable. The student's trajectories tend to be less smooth but beyond the fact that the SR is inferior demonstrate similar obstacle avoidance strategies. This indicates that the teacher was able to infuse the student with his motion planning strategies through the imitation learning method proposed.

IV. Discussion and Conclusion

The results presented in this study highlight the effectiveness of our dual-policy framework for vision-based autonomous navigation of quadrotor swarms in cluttered and unknown environments. The approach, which combines DRL for state-based teacher policy with imitation learning for a vision-based student policy has shown that good performances can be achieved even in the challenging task of navigating swarms through complex environments while avoiding collisions - a capability not commonly demonstrated in existing literature.

A key finding is that the vision-based model, despite relying solely on minpooling for depth image processing and not integrating CNNs, can still navigate effectively within a swarm configuration and generalize to new, unknown environments. This is particularly significant given the difficulty of the task, where real-time decision-making based on high-dimensional visual data is required. The Success Rate of the vision-based policy, although lower than that of the state-based teacher, remains within a reasonable range, particularly at lower speeds (above 86% at 3 m/s and above 78% at 7 m/s). The teacher policy, benefiting from privileged information, achieved a very high SR, highlighting its relevance as a baseline for training the student policy and assessing its performance. Additionally, the results indicate that the framework's scalability to

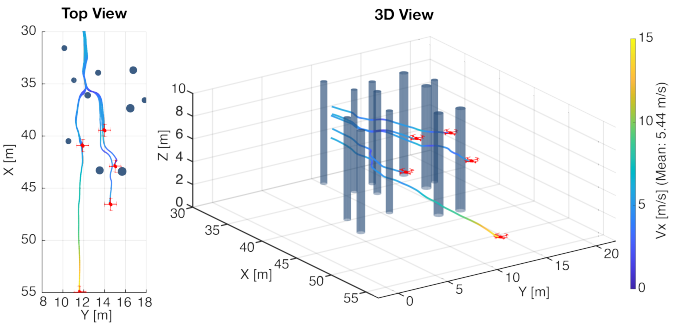

Fig. 6: Obstacle avoidance strategy at 7 m/s target velocity in swarm configuration (state-based). Top view on the left and 3D view on the right with x-velocity profile colormap.

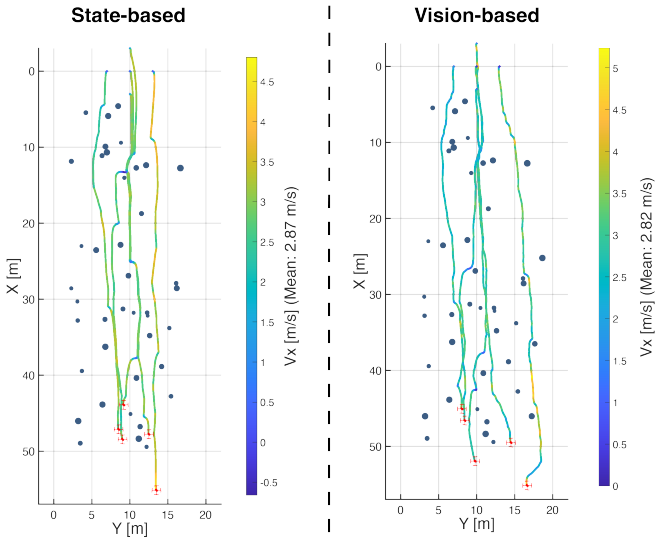

Fig. 7: Trajectory comparison at 3 m/s target velocity in swarm configuration for the same obstacle environment (Top view). State-based policy on the left and vision-based policy on the right with x-velocity profile colormap.

swarm scenarios is effective. The ability of the vision-based student policy to coordinate in order to avoid both inter-agent collisions and obstacles validates the approach's potential for real-world applications involving swarms.

The simplicity of using only minpooling as a downsampling technique instead of relying on more complex CNNs has proven advantageous in this context, as it reduces the computational cost while still capturing essential information for obstacle avoidance. This approach proved sufficient for the relatively simple shapes of obstacles used in this study (cylinders), suggesting that for environments with similarly structured obstacles, simpler image processing techniques may be adequate. However, in more varied or complex environments, where obstacle shapes are less regular, more sophisticated feature extraction methods might be necessary to maintain performance.

The proposed framework can be further improved. For instance, replacing MLP architecture with a Long Short-Term Memory (LSTM) network could enhance the policy's ability to handle visual temporal sequences, potentially increasing the SR at higher speeds and allowing more complex maneuvers. Moreover, introducing a variety of obstacle shapes (not only cylinders) and more realistic renderings would challenge the model further and provide insights into its robustness and generalization capacities. Adding dynamic obstacles could further complexify the task and bring it one step closer to real-world applications.

Future work should therefore focus on closing the sim-to-real gap, which is critical for deploying these models in real-world applications. Improving the policy's ability to handle model mismatches and uncertainties, particularly in more dynamic and varied environments, would make it more applicable to real-world tasks such as search and rescue, environmental monitoring, and other scenarios requiring agile, robust and coordinated swarm navigation. Overall, this study opens up new possibilities for the use of vision-based navigation in quadrotor swarms, offering a scalable, efficient and generalizable solution for complex environments.

REFERENCES

[1] F. Gao, L. Wang, B. Zhou, L. Han, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," 2019. [Online]. Available: https://arxiv.org/abs/1907.00520

[2] X. Zhou, Z. Wang, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *CoRR*, vol. abs/2008.08835, 2020. [Online]. Available: https://arxiv.org/abs/2008.08835

[3] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1484–1491.

[4] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion: Lin et al." *Journal of Field Robotics*, vol. 35, 07 2017.

[5] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, 05 2022.

[6] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," 2021. [Online]. Available: https://arxiv.org/abs/2011.04183

[7] C. Toumieh and A. Lambert, "Decentralized multi-agent planning using model predictive control and time-aware safe corridors," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 110–11 117, 2022.

[8] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multi-agent and dynamic environments," 2021. [Online]. Available: https://arxiv.org/abs/2010.11061

[9] C. Toumieh and D. Floreano, "High-speed motion planning for aerial swarms in unknown and cluttered environments," *IEEE Transactions on Robotics*, vol. 40, pp. 3642–3656, 2024.

[10] J. Eschmann, D. Albani, and G. Loianno, "Learning to fly in seconds," 2024. [Online]. Available: https://arxiv.org/abs/2311.13081

[11] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," *CoRR*, vol. abs/2103.08624, 2021. [Online]. Available: https://arxiv.org/abs/2103.08624

[12] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, p. 7209–7216, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3181755

[13] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.adg1462

[14] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Mueller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, 08 2023.

[15] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[16] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, Jan. 2022. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abk2822

[17] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," 2024. [Online]. Available: https://arxiv.org/abs/2406.12505

[18] A. Devo, J. Mao, G. Costante, and G. Loianno, "Autonomous single-image drone exploration with deep reinforcement learning and mixed reality," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5031–5038, 2022.

[19] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," 2017. [Online]. Available: https://arxiv.org/abs/1710.06542

[20] A. Devo, J. Mao, G. Costante, and G. Loianno, "Autonomous single-image drone exploration with deep reinforcement learning and mixed reality," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5031–5038, 2022.

[21] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, 2019.

[22] A. Loquercio, E. Kaufmann, R. Ranftl, M. Mueller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science robotics*, vol. 6, p. eabg5810, 10 2021.

[23] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *CoRR*, vol. abs/2006.05768, 2020. [Online]. Available: https://arxiv.org/abs/2006.05768

[24] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," 2023. [Online]. Available: https://arxiv.org/abs/2210.01841

[25] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[26] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *CoRR*, vol. abs/1011.0686, 2010. [Online]. Available: http://arxiv.org/abs/1011.0686

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[28] Robotics and P. G. U. of Zurich, "ICRA 2022 DodgeDrone Challenge: Vision-based Agile Drone Flight," https://uzh-rpg.github.io/icra2022-dodgedrone/, 2022, accessed: 2024-08-13.

[29] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[30] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.