2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Monocular Reactive Collision Avoidance for MAV Teleoperation with Deep Reinforcement Learning

Raffaele Brilli, Marco Legittimo, Francesco Crocetti,
Mirko Leomanni, Mario Luca Fravolini, Gabriele Costante

***Abstract*— Enabling Micro Aerial Vehicles (MAVs) with semi-autonomous capabilities to assist their teleoperation is crucial in several applications. Remote human operators do not have, in general, the situational awareness to perceive obstacles near the drone, nor the readiness to provide commands to avoid collisions. In this work, we devise a novel teleoperation setting that asks the operator to provide a simple high-level signal encoding the speed and the direction they expect the drone to follow. We then endow the MAV with an end-to-end Deep Reinforcement Learning (DRL) model that computes control commands to track the desired trajectory while performing collision avoidance. Differently from State-of-the-Art (SotA) works, it allows the robot to move freely in the 3D space, requires only the current RGB image captured by a monocular camera and the current robot position, and does not make any assumption about obstacle shape and size. We show the effectiveness and the generalization capabilities of our strategy by comparing it against a SotA baseline in photorealistic simulated environments.**

I. INTRODUCTION

The teleoperation of MAVs in unstructured environments characterized by a significant presence of obstacles poses important technical challenges [1], [2], [3]. In many applications, including surveillance, monitoring, and exploration, MAVs are often remotely flown by human operators that provide real-time motion commands. However, it is often difficult for the remote operator to perceive obstacles around the drone and understand which of those are an immediate threat. Furthermore, it is even more challenging to timely react and send proper commands in order to avoid collisions and move back the MAV on the planned path. To ease the workload of the operator, in this work we propose a simpler and more feasible setting. We require the operator to send a command specifying only the direction and the velocity to be followed (*e.g.*, by using the remote controller stick) and leave to the MAV the job of autonomously following the desired trajectory while preventing collisions until the next command is received.

Note that, the remote command determines a straight trajectory starting from the drone initial position in the world reference frame and not in the robot body one. Therefore, avoiding obstacles causes the MAV to deviate from the requested trajectory and realignment maneuvers are necessary (see Figure 1). This implies that the MAV must be endowed with two capabilities: *Collision Avoidance (CA)* [4] and *Trajectory Following* (TF).

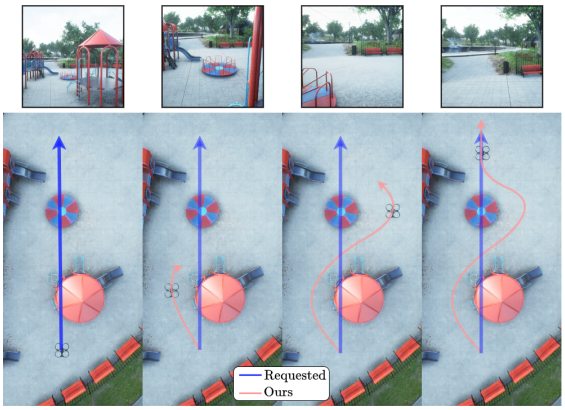

Fig. 1: Overview of our approach. The solution we propose is able to track the trajectory requested by the remote operator while avoiding collisions.

Most of the state-of-the-art (SotA) strategies rely on modular solutions [5] [6], *i.e.*, the MAV is equipped with different and separate functional components. These modules include localization [7], [8], [9], object detection [10], map reconstruction [11] and trajectory re-planning [12].

Despite the promising results achieved, modular solutions have some important drawbacks. First, most of them either focus on CA or on TF, and they rarely address both tasks simultaneously. Second, each of their components makes different assumptions regarding the environment characteristics and the shape and size of the objects. Furthermore, there are usually many hyper-parameters that need to be tuned for each specific environment, which further burdens the deployment phase. In addition, most of the SotA solutions make the simplifying assumption of limiting the MAV motions on a 2D plane (*i.e.*, fixing the height) and require expensive or power-demanding sensors, such as LIDARs or RGB-D cameras.

Contributions: To address the aforementioned issues, we draw inspiration from the results reported in recent works that are based on the Deep Reinforcement Learning (DRL) paradigm [13], [14] and propose a novel end-to-end solution to enhance MAV teleoperation. In particular:

i) We devise a new teleoperation setting where the operator is asked to provide a simple command encoding the desired direction, leaving to the MAV the task to avoid obstacles and track the trajectory.

ii) We propose a novel DRL-based end-to-end approach that

All the authors are with the Department of Engineering, University of Perugia, 06125 Perugia, raffale.brilli@studenti.unipg.it, {marco.legittimo, francesco.crocetti, mirko.leomanni, mario.fravolini, gabriele.costante}@unipg.it

Work in part supported by HiPeRT s.r.l. within a research project with the Autonomous Robotics Research Center (ARRC) of the UAE Technology Innovation Institute (TII), and in part by the University of Perugia, Fondi di Ricerca di Ateneo 2021, Project "AIDMIX—Artificial Intelligence for Decision Making: Methods for Interpretability and eXplainability".

directly computes motion commands to control the drone in order to simultaneously perform TF and CA.

iii) Differently from other SotA approaches, our solution is monocular, does not constrain the MAV at fixed heights, and, besides images, requires only the drone pose information.

iv) Through simulated experiments, we show that our method can generalize over environments and command velocities different from those provided during the training phase.

II. RELATED WORK

As introduced in the previous section, the setting we consider in this work requires that the MAV guidance and control systems are endowed with trajectory following (TF) and collision avoidance (CA).

Trajectory following is a long-standing problem in the robotics community and the number of related works in the literature is very rich (see, *e.g.*, , [15]). In general, the reference trajectory is computed at the beginning of the navigation under the assumption of a known and static map [16]. Otherwise, TF capabilities need to be coupled with a collision avoidance procedure to re-plan the trajectory.

Several approaches have been proposed for Collision Avoidance (CA) [17], [18], and they can be divided into two main categories: *deliberative CA* and *reactive CA* [4]. Deliberative CA algorithms start by computing an initial optimal collision-free trajectory towards the desired goal location. Whenever a new obstacle is detected and added to the map, the trajectory is updated to avoid collisions. These approaches require localization and global mapping capabilities and need to be coupled with a TF component for trajectory tracking [19], [20]. Their modular architecture is subject to error propagation since the performance drop of one module might significantly affect those of another one and, eventually, cause the failure of the entire system. In addition, the construction and the storage of a global map require a considerable amount of memory resources.

On the other hand, reactive CA methods follow a different logic: they perform local avoidance maneuvers to let the robot temporarily deviate from the optimal trajectory to avoid collisions. This strategy typically makes use of lightweight algorithms that do not require global mapping or localization modules (although the latter is still required to perform TF).

The effectiveness of reactive CA strategies significantly depends on the capability to collect and process information from the robot surroundings. Different sensor setups have been explored, including sensors that directly provide 3D information about the obstacles, *e.g.*, LIDAR [21], [22], SONAR [23] or RGB-D camera [24], [1]. Alternatively, nearby objects can be indirectly detected from RGB images by using Convolutional Neural Networks (CNNs) [2] or by running clustering algorithm on raw sensor data [25].

Once obstacles are identified, they are processed by a separate avoidance module that computes reactive maneuvers to prevent collisions. Among the different paradigms proposed for this purpose, one of the most popular is the *Artificial Potential Fields (APF)* [26], [22], which is based on the generation of virtual force fields around obstacles that "push" the robot away to avoid collisions [27].

All the aforementioned approaches involve separate detection and avoidance modules. This modular structure usually relies on several parameters that require accurate fine-tuning procedures. In addition, they are exposed to error propagation, *i.e.*, failures or poor estimations of one component could affect the functionality of others and cause system failures.

To surpass the modular philosophy in favor of end-to-end strategies, data-driven solutions have been recently considered [28]. However, it is only after the advent of Deep Reinforcement Learning (DRL) that perception-to-action strategies have flourished. Over the past few years, several reactive CA based on DRL solutions for ground robots have been presented [29], [30], while, more recently, also aerial platforms have been considered. Under the assumption of flying at a fixed height, the authors in [31] combine a CNN-based monocular depth estimation block with a DRL model for MAV navigation. Generative Adversarial Networks (GANs) [32] are, instead, exploited in [33] to improve the depth map estimation and learn a discrete navigation policy. However, in addition to assuming fixed height, both [31] and [32] focus only on CA and do not consider settings where the MAV has to realign to the trajectory commanded by the operator. Instead, a scenario similar to ours is considered in [34], where the authors designed a continuous action policy to avoid collisions while keeping the MAV close to the desired trajectory (*i.e.*, a straight path). However, as in the previous works, this approach assumes that the drone can only move on a 2D plane and, additionally, requires ground-truth depth maps and GPS data.

The restriction of the fixed flight height is removed in [35], where a discrete-action DQN model is trained to maximize coverage while avoiding collisions. Continuous actions are instead considered in [36], where *Soft Actor-Critic (SAC)* [37]-based DRL model is proposed. Differently from our setting, the aforementioned approaches [35], [36] only consider CA and do not guarantee trajectory following. Additionally, they require precise depth maps provided either by the simulation engine or by a simulated stereo rig.

III. METHODOLOGY

A. Problem Statement

In this section, the MAV teleoperation setting we consider is described. Let $\mathbf{p} = [x\ y\ z]^T$ be the MAV position in the inertial coordinate system and $\mathbf{p}(t_0)$ be its initial position. At time t_0, the operator transmits to the MAV the desired velocity command vector $\mathbf{v}_d$, expressed in the inertial frame (*i.e.*, the operator fixed frame). $\mathbf{p}(t_0)$ and $\mathbf{v}_d$ are then used to plan a 3D straight trajectory $\mathbf{t}_r(t)$, defined as:

$$\mathbf{t}_r(t) = \mathbf{p}(t_0) + \mathbf{v}_d t \tag{1}$$

We refer to this trajectory as *requested* to highlight that this is the one demanded by the operator. $\mathbf{t}_r(t)$ is sampled at discrete times $t_i = i/f_s$, where f_s is the controller rate of operation and with $i \in \mathbb{N}_0$. This results in a sequence of points in the inertial frame $P_r = \{\mathbf{p}_r(t_0),\ \mathbf{p}_r(t_1),\ \dots\ \}$,

with $\mathbf{p}_r(t_i) = \mathbf{t}_r(t_i)$. The objective of the MAV is to avoid collisions while tracking the requested trajectory, *i.e.*, limiting, as much as possible, the deviation between the current MAV position $\mathbf{p}(t_i)$ and the corresponding requested position $\mathbf{p}_r(t_i)$ at each time step. We assume that the robot has access only to RGB images provided by a front-looking monocular camera and to the position vectors $\mathbf{p}_t$, which can be recovered, for instance, by a visual odometry module [7]. To compute the control actions, we devise a DRL-based approach (see Section III-C). The learning mechanism of a DRL model requires continuous interaction between the agent (*i.e.*, the MAV) and the environment to generate sequences of observations, rewards, and actions. To this aim, we exploited a simulation engine, which allows us to create multiple 3D photorealistic scenarios and eliminates the risk of damaging a physical platform. In the following, we introduce the MAV dynamic model used in simulation.

B. Quadrocopter Dynamic Model

The quadrocopter is modeled as a rigid body. Let $\mathbf{v} = [v_x \ v_y \ v_z]^T$ be the quadrotor velocity in the inertial frame, $\mathbf{R}$ be the rotation matrix that describes the rotation from the quadrotor body frame to the inertial frame, and $\boldsymbol{\omega} = [\omega_1 \ \omega_2 \ \omega_3]^T$ be the angular velocity of the body frame relative to the inertial frame. The equations describing the MAV dynamics are given by [38]

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \frac{f}{m}\mathbf{R}_3 - \mathbf{g} \\ \dot{\mathbf{R}} &= \mathbf{R}\,[\boldsymbol{\omega}]_\times \end{aligned} \tag{2}$$

where f is the total thrust, m is the mass of the quadrotor, $\mathbf{g}$ is the gravitational acceleration ($\mathbf{g} = [0\ 0\ 9.8]^T$ m/s^2), $\mathbf{R}_j$ is the j-th column of matrix $\mathbf{R}$, and $[\boldsymbol{\omega}]_\times$ is the skew-symmetric representation of $\boldsymbol{\omega}$. In order to enforce the quadrotor thrust constraint $f > 0$, we adopt a change of variable

$$f = me^{\lambda} \tag{3}$$

where λ is the new thrust input. By differentiating $\mathbf{p}(t)$ thrice with respect to time and using (2)-(3), one obtains

$$\dddot{\mathbf{p}} = e^{\lambda}\mathbf{R}\mathbf{q} \tag{4}$$

where $\mathbf{q} = [\omega_2 \ -\omega_1 \ \dot{\lambda}]^T$. Let

$$\mathbf{q} = e^{-\lambda}\mathbf{R}^T\mathbf{u} \tag{5}$$

By substituting (5) into (4), one obtains the jerk dynamics

$$\dddot{\mathbf{p}} = \mathbf{u} \tag{6}$$

Herein, the vector $\mathbf{u}$ is considered as the control input and, as clarified in the following section, it is the output of our DRL network. Note that our approach is model-free, *i.e.*, it does not need nor has access to the MAV model, which is only used to generate the experience during the learning phase.

C. Reinforcement Learning Formulation

Our objective is to learn a controller that computes suitable control commands able to track the reference trajectory and avoid obstacles. To this aim, we rely on the reinforcement learning framework that allows one to train iteratively such a controller by applying actions and collecting the corresponding rewards and observations from the environment. In the following, we formally define these entities in our setting:

Action: At each time step, the model outputs the control command $\mathbf{u}(t_i) \in \mathbb{R}^3$ in Eq. 6. Each component of the vector is constrained in the interval $[-j_{max}, j_{max}]$, where j_{max} is the maximum allowed jerk. The actual physical control inputs $f(t_i)$, $\omega_1(t_i)$, $\omega_2(t_i)$ of the system (2) can be immediately recovered from $\mathbf{u}(t_i)$ through the equations (3) and (5). As we do not require the MAV to have a specific orientation, we set $\omega_3(t_i) = 0$ to maintain the initial heading.

Observations: The environment provides the RGB image $\mathcal{I}(t_i)$ and a 12-dimensional vector $\mathbf{s}(t_i) = [\mathbf{e}_\mathbf{p}(t_i), \mathbf{e}_\mathbf{v}(t_i), \mathbf{v}(t_i), \dot{\mathbf{v}}(t_i)]$ containing the position error $\mathbf{e}_\mathbf{p}(t_i)$, the velocity error $\mathbf{e}_\mathbf{v}(t_i) = \mathbf{v}(t_i) - \mathbf{v}_d$, the MAV velocity $\mathbf{v}(t_i)$ and the MAV acceleration $\dot{\mathbf{v}}(t_i)$. All these signals are derived from the known position vectors $\mathbf{p}(t_i)$. We define the observation tuple as $\mathbf{o}(t_i) = \{\mathcal{I}(t_i), \mathbf{s}(t_i)\}$.

Reward: The RL-based solution we devise relies on an episodic formulation. Hence, in order to define the reward signals, we first specify the maximum episode duration T_{max} as follows:

$$T_{max} = \frac{\|\mathbf{p}_{max} - \mathbf{p}_r(t_0)\|}{\|\mathbf{v}_d\|}, \tag{7}$$

where $\mathbf{p}_{max}$ is a given point along the requested trajectory $\mathbf{t}_r(t)$. Then, we introduce i_{max} as the smallest integer such that $i_{max} \geq f_s T_{max}$. Hence, the stopping conditions of an episode are defined as follows:

- a collision has occurred;
- $\|\mathbf{e}_\mathbf{p}(t_i)\| > \mathbf{e}_{\mathbf{p},max}$, where $\mathbf{e}_\mathbf{p}(t_i) = \mathbf{p}(t_i) - \mathbf{p}_r(t_i)$ is the position error and $\mathbf{e}_{\mathbf{p},max}$ is a user defined threshold;
- $i = i_{max}$, *i.e.*, the time limit has been reached.

The Reward function is designed to enforce the MAV to minimize the position error $\mathbf{e}_\mathbf{p}$ while preventing collisions with the surrounding objects:

$$r(t_i) = \begin{cases} 1 & \|\mathbf{e}_\mathbf{p}(t_i)\| \leq 1 \\ 0.5 - \dfrac{\|\mathbf{e}_\mathbf{p}(t_i)\|}{2\mathbf{e}_{\mathbf{p},max}} & 1 < \|\mathbf{e}_\mathbf{p}(t_i)\| < \mathbf{e}_{\mathbf{p},max} \\ -i_{max} & \|\mathbf{e}_\mathbf{p}(t_i)\| > \mathbf{e}_{\mathbf{p},max} \text{ or } \textit{coll}, \end{cases} \tag{8}$$

where *coll* indicates that a collision has occurred. In case the MAV reaches the timestep i_{max} without colliding or deviating too much from the desired trajectory, the cumulative reward of the current episode is positive in the interval $(0, i_{max}]$. Otherwise in case this is negative, the cumulative reward takes values in the interval $[-i_{max}, 0)$.

D. Actor-Critic Network Design

The request to operate with continuous action signals and the need to process high-dimensional observations (*i.e.*, images), implies an infinite number of observation-action pairs.

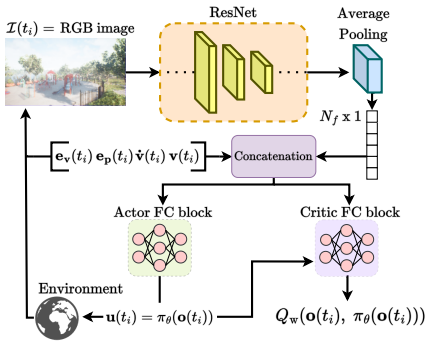

Fig. 2: Overall architecture of the proposed approach. Both the actor and the critic networks take as input the current RGB image $\mathcal{I}(t_i)$ and the MAV pose information provided by the environment. The feature extraction CNN is shared between the two models, while their fully connected layers have different parameters.

In this context, classical tabular RL approaches cannot be employed. Therefore, we exploit the Deep RL paradigm based on deep function approximators to estimate the observation-action values and the optimal policy.

Specifically, we rely on the popular Actor-Critic formulation and introduce the actor (A-DNN) and critic (C-DNN) deep neural networks (see Figure 2). We choose a symmetric structure for the two models, *i.e.*, at each timestep both of them take as input the same observation tuple $\mathbf{o}(t_i) = \{\mathcal{I}(t_i), \mathbf{s}(t_i)\}$. The image $\mathcal{I}(t_i)$ is then processed by a shared ResNet CNN [39], pre-trained on the ImageNet dataset [40]. The ResNet feature extractor computes N_f feature maps, each one reduced by an average pooling layer to obtain a $N_f \times 1$ vector that is concatenated with the MAV position information $\mathbf{s}(t_i)$. The resulting vector is fed to two fully connected blocks. They share the same structure, *i.e.*, three hidden layers with respectively 256, 128 and 64 neurons and ReLU activations, but have different parameters. Both the networks produce continuous outputs: the A-DNN computes the continuous action policy $\pi_\theta(\mathbf{o}(t_i))$, while the C-DNN estimates Q-values $Q_\mathbf{w}(\mathbf{o}(t_i), \pi_\theta(\mathbf{o}(t_i)))$, where θ and $\mathbf{w}$ indicate the parameter of A-DNN and C-DNN.

The models are trained by taking advantage of the off-policy Soft Actor-Critic (SAC) optimization framework [37].

At test time, the A-DNN network takes the role of the MAV controller, which processes the state $\mathbf{o}(t_i)$ and returns the action $\mathbf{u}(t_i) = \pi_\theta(\mathbf{o}(t_i))$.

IV. EXPERIMENTAL SETUP

A. Environments and Implementation Details

The training and evaluation of our controller is performed using the Unreal Engine (UE4) software[1]. The simulator enables us to generate photorealistic scenarios and allows to efficiently and safely train the DRL controller. We also exploit the AirSim plugin [2] to generate randomized obstacles, retrieve RGB images and robot poses, and collect the depth maps (the latter are only used for the baseline, as detailed in Section IV-B).

[1]https://www.unrealengine.com/
[2]https://microsoft.github.io/AirSim/

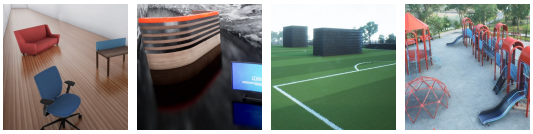

Fig. 3: Examples of the environments used for training and test.

At the beginning of each episode, the MAV is assumed to be hovering at a fixed position, which, without loss of generality, is taken coincident with the origin of the inertial frame, *i.e.*, $\mathbf{p}(t_0) = 0$. Moreover, it is assumed that the operator transmits a command signal of the form $\mathbf{v}_d = [v_{d,x}\ 0\ 0]^T$ and therefore $\mathbf{p}_r(t_i) = [v_{d,x}t_i\ 0\ 0]^T$. Accordingly, the terminal point on the reference trajectory takes on the form $\mathbf{p}_{max} = [x_{max}\ 0\ 0]^T$. The MAV is equipped with an RGB camera with a resolution of 256x144 and a Field of View (FoV) equal to 90°. In each training step we use the physics model in III-B to evolve the MAV dynamics in the simulation environments.

1) Environments used for Training: The MAV agent is trained in an indoor 3D corridor with length, width, and height of 200 m, 16 m, and 10 m, respectively. At the start of each episode, we create a new instance of the environment by randomly selecting the texture of walls and floors from a collection of different materials (*e.g.*, concrete, wood, metal, etc.) and spawn different objects inside it. Objects are randomized by changing: i) type (*e.g.*, chair, desk, column, etc.), ii) texture, iii) position, iv) z-axis angle, and v) size. We set $f_s = 5$ Hz, the output jerk saturation to $j_{max} = 2.0$ m/s^3 and the MAV mass (2) to $m = 1$ kg. Differently from the evaluation experiments, the requested velocity is fixed to $v_{d,x} = 2.0$ m/s. Finally, we set $x_{max} = 180$ m, which implies that $T_{max} = 90$ s and $i_{max} = 450$.

2) Environments used for Evaluation: To assess the generalization performance of our approach and compare it against the baseline (see Section IV-B), we generate three categories of test environments (see Figure 3):

Training-like: four corridor environments (TL1, TL2, TL3, TL4) with objects and textures similar to the training ones;

Training-unlike: four corridor environments (TU1, TU2, TU3, TU4) populated by objects and textures different from those experienced during the training phase;

Outdoor: one outdoor football pitch environment (FP) with concrete boxes as obstacles, and an outdoor playground environment (PG); in these cases x_{max} is set to 90 m and 35 m respectively to fit the structure of the environments.

In addition, in order to evaluate the capability of our approach to operate with requested velocities different from the training one, we also run a set of tests with two additional $\mathbf{v}_d$ values, *i.e.*, $v_{d,x} = 3.0$ m/s and $v_{d,x} = 3.5$ m/s.

3) Deep Neural Network and Training details: We adopt the SAC implementation provided by the *Stable-Baselines3* [41] framework. We use the 34-layers version of ResNet as the CNN-based feature extractor used by A-DNN and C-DNN, which produces $N_{feat} = 512$ feature maps for each frame. We optimize the Actor and Critic weights for 1000000 simulations steps using the Adam optimizer. At each time step, we randomly sample 256 samples from the Replay

Buffer. Training is performed on a workstation equipped with an Nvidia GTX TITAN Xp GPU with 12 GB of VRAM and requires about 32 hours to reach convergence.

B. Baseline

As discussed in Section II, to the best of our knowledge there are no SotA approaches that consider a MAV teleoperation setting as the one considered in our study. Hence, we choose to compare it against a modular strategy. Specifically, we develop a control mechanism for the system (2) based on three sequential modules that perform TF and vision-based reactive CA, and exploit Artificial Potential Fields (APFs) (see Figure 4). We refer to this baseline as APF-CATF.

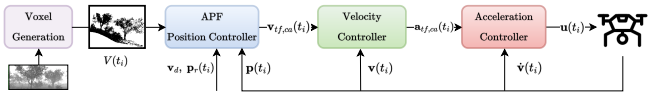

Fig. 4: Architecture of the APF-CATF baseline.

The first module takes inspiration from the vision-based APF-based strategy method presented in [2]. It is responsible for detecting obstacles and computing the reference velocity to achieve avoidance and trajectory tracking. This module computes a reference velocity signal $\mathbf{v}_{tf,ca}(t_i)$ and its control law is given by:

$$\mathbf{v}_{tf,ca}(t_i) = \begin{cases} -K_p(\mathbf{p}(t_i) - \mathbf{p}_r(t_i)) + \mathbf{v}_d & dist_{min} > \delta_2 \\ \mathbf{v}_{avcff}(\mathbf{p}(t_i), V(t_i)) & dist_{min} \leq \delta_2 \end{cases} \tag{9}$$

where $dist_{min}$ is the peripheral distance between the nearest voxel and the MAV, and δ_2 is the safety radius of the keep-out zone. When $dist_{min} > \delta_2$ the layer acts as a proportional position controller while preserving the $\mathbf{v}_d$ command requested by the operator by summing it as a feedforward element. When instead $dist_{min} \leq \delta_2$, $\mathbf{v}_{avcff}$ is selected as the velocity to be controlled. This is obtained by using the *Adaptive Virtual Cushion Force Field (AVCFF)* function (see [22] and [2] for more details) and its purpose is to repel the MAV from the nearby obstacles. The second module acts as a proportional velocity controller and outputs the acceleration signal using the following law:

$$\mathbf{a}_{tf,ca}(t_i) = K_v(\mathbf{v}_{tf,ca}(t_i) - \mathbf{v}(t_i)) \tag{10}$$

Finally, the third module is a proportional acceleration controller and computes the jerk command as follows:

$$\mathbf{u}(t_i) = K_a(\mathbf{a}_{tf,ca}(t_i) - \dot{\mathbf{v}}(t_i)) \tag{11}$$

We found through experiments that suitable gains for the baseline controller are $K_p = 0.1$, $K_v = 1.0$ and $K_a = 3.0$, while the avoidance zone radius is set to $\delta_2 = 5.0$ m.

Notice that the first module requires the $V(t_i)$ voxel-grid representation of the obstacles. To compute such representation, we decided to provide the baseline with ground truth depth maps (from the AirSim plugin) instead of using a depth estimation block to process RGB images. Although this clearly advantages the baseline (voxels are computed without possible estimation errors), surprisingly, we observe through experiments that our approach, which instead has access only to RGB images, is able to guarantee better performance.

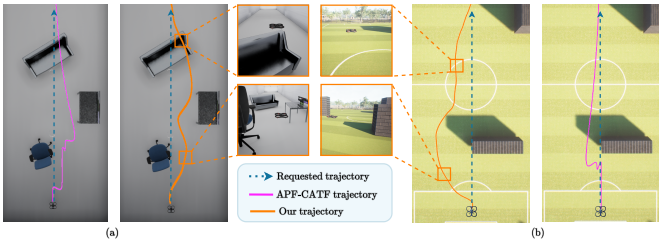

Fig. 5: Qualitative evaluation in an indoor (a) and an outdoor (b) environment. Orange trajectories refer to our approach, while the APF-CATF baseline is associated with the purple color. In both cases, we provide two third-person view snapshots of the drone.

C. Performance Evaluation metrics

To quantitatively compare our approach with the baseline, we defined different metrics. Each metric is averaged over N_{tr} trials (in our experiments we set $N_{tr} = 10$). To test the robustness of the approaches, each trial differs from the others due to two sources of randomness: i) random MAV initial position, *i.e.*, $\mathbf{p}(t_0) = [0\ y_n\ 0]^T$, where y_n is sampled from $\mathcal{N}(0,\ 3)$; ii) random additive noise in the jerk input vector $\mathbf{u}(t_i)$, sampled from $\mathcal{N}(0,\ 0.01)$. Details of the metrics are provided below:

Success Rate (SR): Success rate is defined as the average of the successes from each trial. The success S_n is equal to 0, in case, during the trial, the MAV collides or the condition $\|\mathbf{e}_\mathbf{p}(t_i)\| > \mathbf{e}_{\mathbf{p},max}$ is verified, with $\mathbf{e}_{\mathbf{p},max} = 25$ m. Otherwise S_n is set equal to 1.

Success-weighted Deviation from Optimal Displacement (S-DOD): By using the Dijkstra algorithm on a ground truth voxel-grid representation of the environment, we compute the optimal obstacles-free path that connects $\mathbf{p}_r(t_0)$ and $\mathbf{p}_r(T_{max})$ while minimizing the deviation from the desired trajectory $\mathbf{t}_r$. We call $\bar{d}_{opt}$ the mean displacement from $\mathbf{t}_r$ of the points of the aforementioned path. The (S-DOD) is, therefore, defined as follows:

$$S\text{-}DOD = \frac{1}{N_{tr}} \sum_{n=1}^{N_{tr}} S_n \frac{\bar{d}_{opt}}{\bar{d}_n} \tag{12}$$

where $\bar{d}_n$ is the mean displacement between the points of $\mathbf{t}_r$ and those of the trajectory performed by the MAV at trial n. This metric is maximized (*i.e.*, it is equal to 1.0) when the MAV trajectory has the same average displacement as the Dijkstra-computed one, *i.e.*, the one that minimally deviates from the requested trajectory.

Average Control Effort (ACE): This is defined as:

$$ACE = \frac{1}{N_{tr}} \int_0^{T_n} \|\mathbf{u}(t)\| dt \tag{13}$$

where T_n is the episode length and $\mathbf{u}(t)$ is the jerk command.

V. RESULTS AND DISCUSSION

The quantitative scores achieved in test environments are reported in Table I, while some qualitative results are shown in Figure 5. Their analysis highlights that our approach outperforms the baseline in the majority of the scenarios against both SR and S-DOD metrics. This is remarkable

	APF-CATF			Ours		
Scenario	SR	$S\text{-}DOD$	ACE	SR	$S\text{-}DOD$	ACE
$v_{d,x} = 2.0$ m/s						
TL1	0.0	0.00	3.91	**1.0**	**0.15**	4.72
TL2	0.1	0.01	3.65	**1.0**	**0.16**	4.83
TL3	0.4	0.09	3.53	**0.8**	**0.16**	5.42
TL4	**1.0**	**0.31**	0.51	**1.0**	0.13	5.13
TU1	**0.7**	**0.12**	1.87	**0.7**	0.11	5.85
TU2	0.0	0.00	2.13	**1.0**	**0.11**	5.50
TU3	0.2	0.02	3.30	**0.9**	**0.12**	5.44
TU4	0.4	**0.07**	3.34	**0.6**	0.06	6.38
FP	0.6	**0.04**	1.82	**0.8**	0.01	6.07
PG	0.5	**0.08**	2.95	**0.8**	0.03	6.53
$v_{d,x} = 3.0$ m/s						
TL1	0.2	0.04	11.96	**1.0**	**0.12**	4.52
TL2	0.7	0.11	5.81	**1.0**	**0.14**	5.39
TL3	0.2	0.03	9.98	**1.0**	**0.26**	5.83
TL4	**0.9**	**0.22**	2.45	0.6	0.06	5.33
TU1	**1.0**	**0.16**	1.81	0.7	0.09	6.09
TU2	0.0	0.00	5.25	**0.9**	**0.08**	5.98
TU3	0.3	0.03	7.94	**0.7**	**0.09**	5.96
TU4	0.2	0.03	8.56	**0.5**	**0.04**	7.14
FP	**0.9**	**0.04**	2.02	**0.9**	0.01	6.15
PG	0.1	0.01	9.69	**0.9**	**0.03**	6.54
$v_{d,x} = 3.5$ m/s						
TL1	0.2	0.02	12.00	**1.0**	**0.12**	4.92
TL2	0.9	**0.12**	3.29	**0.9**	0.09	5.95
TL3	0.3	0.04	12.00	**0.6**	**0.25**	5.91
TL4	**1.0**	**0.16**	2.27	0.4	0.04	5.74
TU1	**0.8**	**0.10**	5.31	**0.8**	0.09	6.38
TU2	0.0	0.00	8.19	**0.3**	**0.03**	6.36
TU3	0.1	0.01	12.00	**0.5**	**0.07**	6.32
TU4	**0.2**	**0.02**	12.00	**0.2**	0.01	7.65
FP	0.6	**0.02**	5.61	**1.0**	0.01	6.33
PG	0.5	0.02	12.00	**1.0**	**0.03**	6.72

TABLE I: Quantitative comparison between APF-CATF and our method. Cells highlighted in green refer to the approach that obtained the higher value for the metric.

considering that APF-CATF has a significant advantage over our strategy since it employs the error-free (ideal) ground truth depth maps instead of raw RGB images. The success rate difference between the two methods is considerable in some scenarios (*e.g.*, see TL1, TL2, and PG). This can be ascribed to an inherent limitation of APF strategies: the repulsive obstacle fields and the "attraction" force exerted by the trajectory tracking task can possibly generate local minima, which causes the MAV to get stuck.

Another interesting consideration can be drawn by observing the S-DOD metric, which is low for both approaches. This is expected since the trajectories computed with the Dijkstra algorithm are very close to the obstacles. Nonetheless, our approach outperforms APF-CATF in most of the cases.

The results for the indoor scenario in Figure 5.a highlight the capability of our approach to take advantage from the whole 3D space to avoid obstacles while trying to stay close to the trajectory commanded by the operator. In fact, in this example, our method correctly understands that avoiding the couch by flying over it minimizes the displacement with respect to the reference trajectory.

Table I also provides interesting insights about the generalization capabilities of our approach. As expected, the best performance have been achieved in the *training-like* (TL) scenarios. Nonetheless, the performance drop on the TU and outdoor environments (that are not used to train the model) is limited and, in most of the cases, we still outperform the baseline. Notably, the SR metric in the outdoor scenario is close to 1 (its maximum value), which is impressive considering that these environments are very different from the training ones (see Figure 3). Similar considerations on the generalization capacity can be drawn in case the MAV is flown with different requested velocities. Specifically, although in all the training episodes $v_{d,x} = 2.0$ m/s, we do not experience a significant performance drop for our DRL model by varying its value during tests. This result is remarkable, considering also that, besides generalization issues, increasing the desired speed poses computational challenges and demands for faster reactions. Nonetheless, the experimental results prove that our approach is robust also to these aspects.

Finally, we can also draw some intuitions by analyzing the ACE metric. At lower speeds, *i.e.*, $v_{d,x} = 2.0$ m/s, the APF baseline guarantees lower control efforts. This is expected since model-based solutions are able to provide smoother trajectories when compared to those computed with DRL models. In addition, the advantage given by the availability of ground truth depth maps removes the risk of spurious obstacles that could cause spikes in the control commands. Interestingly, in many of the tests at higher speeds, instead, our approach provides a lower CE value. This can be explained considering that the presence of local minima in the force field computed by APF-CATF is even more problematic if the MAV is required to fly faster. The repulsive and attractive forces might, in fact, cause bigger oscillations in proximity of those points.

VI. CONCLUSION

In this paper, we present a novel assisted teleoperation setting for MAV that requires the operator to provide solely the intent velocity command. For this purpose, a novel end-to-end monocular DRL-based approach is proposed to enable the MAV to track the trajectory specified by the remote pilot and simultaneously avoid collisions. Through several experiments in photorealistic simulated scenarios, we show the generalization capabilities of our solution and verify that, compared to a SotA baseline, it guarantees better performance. Future work will explore the recurrent neural network paradigm to model temporal dependencies and remove the need for an external positioning system. We will also exploit the recurrent architecture in order to supply our system with the capability to also elude moving obstacles. Furthermore, the performance on real-world scenarios will also be evaluated.

REFERENCES

[1] F. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero, "Semi-autonomous teleoperation of uavs in search and rescue scenarios," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 1066–1074.

[2] R. Brilli, M. Pozzi, F. Giorgetti, M. L. Fravolini, P. Valigi, D. Prattichizzo, and G. Costante, "Monocular reactive collision avoidance based on force fields for enhancing the teleoperation of mavs," in *2021 20th International Conference on Advanced Robotics (ICAR)*, 2021, pp. 91–98.

[3] T. M. Lam, H. W. Boschloo, M. Mulder, and M. M. van Paassen, "Artificial force field for haptic feedback in uav teleoperation," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, no. 6, pp. 1316–1330, 2009.

[4] J. N. Yasin, S. A. S. Mohamed, M.-H. Haghbayan, J. Heikkonen, H. Tenhunen, and J. Plosila, "Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches," *IEEE Access*, vol. 8, pp. 105 139–105 155, 2020.

[5] W. L. Leong, P. Wang, S. Huang, Z. Ma, H. Yang, J. Sun, Y. Zhou, M. R. Abdul Hamid, S. Srigrarom, and R. Teo, "Vision-based sense and avoid with monocular vision and real-time object detection for uavs," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2021, pp. 1345–1354.

[6] X.-Z. Peng, H.-Y. Lin, and J.-M. Dai, "Path planning and obstacle avoidance for vision guided quadrotor uav navigation," in *2016 12th IEEE International Conference on Control and Automation (ICCA)*, 2016, pp. 984–989.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[8] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4666–4672.

[9] S. Felicioni, M. Legittimo, M. L. Fravolini, and G. Costante, "Goln: Graph object-based localization network," in *2021 20th International Conference on Advanced Robotics (ICAR)*. IEEE, 2021, pp. 849–856.

[10] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[12] Y. Lin and S. Saripalli, "Sampling-based path planning for uav collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[13] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[14] A. Dionigi, A. Devo, L. Guiducci, and G. Costante, "E-vat: An asymmetric end-to-end approach to visual active exploration and tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4259–4266, 2022.

[15] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022.

[16] L. Filippis, G. Guglieri, and F. Quagliotti, "Path planning strategies for uavs in 3d environments," *J. Intell. Robotics Syst.*, vol. 65, no. 1–4, p. 247–264, jan 2012.

[17] X. Yu and Y. Zhang, "Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects," *Progress in Aerospace Sciences*, vol. 74, pp. 152–166, 2015.

[18] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annual Reviews in Control*, vol. 48, pp. 147–164, 2019.

[19] D. Sislak, P. Volf, A. Komenda, J. Samek, and M. Pechoucek, "Agent-based multi-layer collision avoidance to unmanned aerial vehicles," in *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2007, pp. 365–370.

[20] G. Wu, D. Shi, and J. Guo, "Deliberative collision avoidance for unmanned surface vehicle based on the directional weight," *Journal of Shanghai Jiaotong University (Science)*, vol. 21, pp. 307–312, 06 2016.

[21] D. Bareiss, J. Bourne, and K. Leang, "On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles," *Autonomous Robots*, vol. 41, 10 2017.

[22] M. Wang and H. Voos, "An integrated teleoperation assistance system for collision avoidance of high-speed uavs in complex environments," in *2020 17th International Conference on Ubiquitous Robots (UR)*, 2020, pp. 290–296.

[23] B. Giovanini, H. De Oliveira, and P. Rosa, "An automatic collision avoidance approach to assist remotely operated quadrotors," vol. 531, 02 2017, pp. 213–225.

[24] M. Odelga, P. Stegagno, N. Kochanek, and H. H. Bülthoff, "A self-contained teleoperated quadrotor: On-board state-estimation and indoor obstacle avoidance," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7840–7847.

[25] G. Tartaglione and M. Ariola, "Obstacle avoidance via landmark clustering in a path-planning algorithm," in *2018 Annual American Control Conference (ACC)*, 2018, pp. 2776–2781.

[26] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[27] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[28] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2759–2764.

[29] L. Xie, S. Wang, A. Markham, and A. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *ArXiv*, vol. abs/1706.09829, 2017.

[30] L. Gao, J. Ding, W. Liu, H. Piao, Y. Wang, X. Yang, and B. Yin, "A vision-based irregular obstacle avoidance framework via deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9262–9269.

[31] M. Kim, J. Kim, M. Jung, and H. Oh, "Towards monocular vision-based autonomous flight through deep reinforcement learning," *Expert Systems with Applications*, vol. 198, p. 116742, 2022.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[33] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge," *Trans. Intell. Transport. Syst.*, vol. 22, no. 1, p. 107–118, jan 2021.

[34] S. Song, Y. Zhang, X. Qin, K. Saunders, and J. Liu, "Vision-guided collision avoidance through deep reinforcement learning," in *NAECON 2021 - IEEE National Aerospace and Electronics Conference*, 2021, pp. 191–194.

[35] J. Roghair, A. Niaraki, K. Ko, and A. Jannesari, "A vision based deep reinforcement learning algorithm for uav obstacle avoidance," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2021, pp. 115–128.

[36] Z. Xue and T. Gonsalves, "Vision based drone obstacle avoidance by deep reinforcement learning," *AI*, vol. 2, no. 3, pp. 366–380, 2021.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[38] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[40] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[41] 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.