robotics

MDPI

Article

Sim-to-Real Deep Reinforcement Learning for Safe End-to-End Planning of Aerial Robots

Halil Ibrahim Ugurlu, Xuan Huy Pham and Erdal Kayacan *

Artificial Intelligence in Robotics Laboratory (AiR Lab), The Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark
* Correspondence: erdal@ece.au.dk

Abstract: In this study, a novel end-to-end path planning algorithm based on deep reinforcement learning is proposed for aerial robots deployed in dense environments. The learning agent finds an obstacle-free way around the provided rough, global path by only depending on the observations from a forward-facing depth camera. A novel deep reinforcement learning framework is proposed to train the end-to-end policy with the capability of safely avoiding obstacles. The Webots open-source robot simulator is utilized for training the policy, introducing highly randomized environmental configurations for better generalization. The training is performed without dynamics calculations through randomized position updates to minimize the amount of data processed. The trained policy is first comprehensively evaluated in simulations involving physical dynamics and software-in-the-loop flight control. The proposed method is proven to have a 38% and 50% higher success rate compared to both deep reinforcement learning-based and artificial potential field-based baselines, respectively. The generalization capability of the method is verified in simulation-to-real transfer without further training. Real-time experiments are conducted with several trials in two different scenarios, showing a 50% higher success rate of the proposed method compared to the deep reinforcement learning-based baseline.

Keywords: deep reinforcement learning; obstacle avoidance; quadrotors; sim-to-real transfer

Citation: Ugurlu, H.I.; Pham, X.H.; Kayacan, E. Sim-to-Real Deep Reinforcement Learning for Safe End-to-End Planning of Aerial Robots. *Robotics* **2022**, *11*, 109. https://doi.org/10.3390/robotics11050109

Academic Editors: Xiaowei Huang, Wenjie Ruan and Xingyu Zhao

Received: 30 August 2022
Accepted: 10 October 2022
Published: 13 October 2022

1. Introduction

Autonomous aerial robots are increasingly deployed in applications that require safe path planning in dense environments, such as a greenhouse covered with dense plants, search and rescue operation in an unstructured collapsed building or navigation in a forest. Traditionally, autonomous navigation is solved under separate problems such as state estimation, perception, planning, and control [1]. This approach may lead to higher latency combining individual blocks and system integration issues. On the other hand, recent developments in machine learning, particularly in reinforcement learning (RL) and deep reinforcement learning (DRL), enable an agent to learn various navigation tasks end-to-end with only a single neural network policy that generates required robot actions directly from sensory input. These methods are promising for solving navigation problems faster computationally since they do not deal with the integration of subsystems that are tuned for their particular goals.

This study attempts to address the end-to-end planning problem of a quadrotor UAV in dense indoor environments. The quadrotor deployed with a depth camera is required to find its way around the global trajectory. We propose a DRL-based safe navigation methodology for quadrotor flight. The learned DRL policy, utilizing the depth images and the knowledge of a global trajectory, generates safe waypoints for the quadrotor. We develop a Webots-based simulation environment where the DRL agent is trained with obstacle tracks where the obstacle locations, shapes, and sparsity are randomized for every episode of policy training for better generalization. Furthermore, we introduce safety

boundaries to be considered during training in addition to collision checks. The safety boundaries enable the agent to prevent risky situations that make the method more robust to uncertainties.

Contributions

The contributions of this paper are fourfold:

- A novel DRL simulation framework is proposed for training an end-to-end planner for a quadrotor flight, including a faster training strategy using non-dynamic state updates and highly randomized simulation environments.
- The impact of continuous/discrete actions and proposed safety boundaries in RL training are investigated.
- We open-source the Webots-based DRL framework, including all training and evaluation scripts (the code, trained models, and simulation environment will be available at https://github.com/open-airlab/gym-depth-planning, accessed on 1 July 2022).
- The method is evaluated with extensive experiments in Webots-based simulation environments and multiple real-world scenarios, transferring the network from simulation to real without further training.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 explains the end-to-end planning methodology for a quadrotor UAV with the formalization of the RL problem. Section 4 provides the experimental setup and the comprehensive tests of the proposed method in the simulation environment. The section also provides the results of the real-time tests. Finally, Section 5 concludes this work with future research directions.

2. Related Work

As a machine learning paradigm, RL aims to solve sequential decision-making problems through the interaction of a learning agent with its environment [2]. With the success of the deep learning models in machine learning, it is also applied to RL, which brings about the DRL field with success in several benchmark problems such as video games [3] or continuous control tasks [4]. Several methods are proposed to optimize deep neural networks to learn the value function [3], policy function [5], or both [4,6] in the RL domain, such as the proximal policy optimization (PPO) [7] algorithm, a state-of-the-art method utilized in this work. RL and its successor DRL have gained attention in robotics applications as it is encouraging a complete framework for intelligent robots to learn by interacting with their environment.

Since deep learning-based methods require plenty of data, they have emphasized using simulation data as an alternative to expensive real-world data. The usefulness of simulations becomes more crucial for DRL considering potential hardware failures during exploration in the real-world [8]. However, there is a gap between simulation and real-world data, as sensor signal qualities may not be preserved due to the lack of realistic noise. Earlier works have shown that certain data modalities provide a better abstraction for sim-to-real transfer, such as using depth images [9] or applying morphological filters [10]. Another gap between simulation and reality comes from the limitations in modeling real-world dynamics, which are generally countered by domain randomization, e.g., randomizing physical parameters [11] or randomizing observations gathered by visual sensors [12].

Deep neural network-based methods are utilized in the control and navigation of several robotics applications, including real-world demonstrations. Those applications can be classified into two categories considering the input to the neural network: the state information, such as positions and velocities, or raw sensory data, such as color or depth images. Using state information directly, neural network policies have similar functionality with a controller block in quadrotor UAVs, such as in attitude control [13] or position control [11,14] level. Furthermore, various output configurations from motion primitives [15] to lowest-level motor voltage commands [16] for the learned policies are also

investigated. Compared to conventional control theoretic approaches, those methods are lacking in providing mathematical guarantees such as stability analysis [17]. However, it is an active research area where the most recent works promisingly show that DRL-based cascaded control outperforms classical proportional-integral-derivative (PID) controllers [18] and demonstrates challenging control tasks such as high-speed flight control [19].

Prior to deep learning-based methods, the planning methods for robotics have been extensively studied. In particular, graph-based (e.g., A* [20] and D* [21]), potential field-based [22], and sampling-based [23] methods can be counted as subfields of conventional planning algorithms, which require a graph or map representation of the configuration space. Conventional planning algorithms are also an active research area for the application of quadrotor flight [24], as well as other fields, such as collision avoidance of near-Earth space systems [25]. Unlike conventional planning algorithms, DRL enables the learning of so-called neural network end-to-end planners or visuomotor controllers that can generate actions directly from sensory input without any map. Although several applications for ground robots utilize lidar sensors for obstacle avoidance tasks [26,27], visual sensors are more commonly used in aerial applications such as color or depth images. End-to-end navigation is broadly investigated for quadrotor UAVs in several domains such as corridor following [28], drone racing [1,29], aerial cinematography [30], autonomous landing [31,32] or obstacle avoidance [33,34], which is the application in this paper. A recent study demonstrates the capabilities of DRL in a safety critic mission, leveraging the depth and semantic images for an emergency landing [32]. Similarly, a high-speed quadrotor flight with obstacle avoidance has been shown with an imitation learning-based neural network policy recently [35]. In the context of the present study, safe navigation is considered rather than agility. Furthermore, instead of imitation learning, DRL is studied. More similar to the present study, Camci et al. [36] utilize a quadrotor with a depth camera for obstacle avoidance but with discrete actions. Dooraki et al. [37] also propose a similar application with continuous actions in the position domain. The present research differs by proposing safety boundaries and enabling heading angle steps together with position steps.

3. End-to-End Motion Planning of UAV

3.1. Reinforcement Learning Formalization of the Environment

An RL problem is generally formalized as a Markov decision process (MDP) with state, action, and reward components with discrete timesteps, t. The common variables are shown in Appendix A. For the problem of end-to-end local planning, the state is defined as multi-modal data containing the depth image and the vector representing the moving target point,

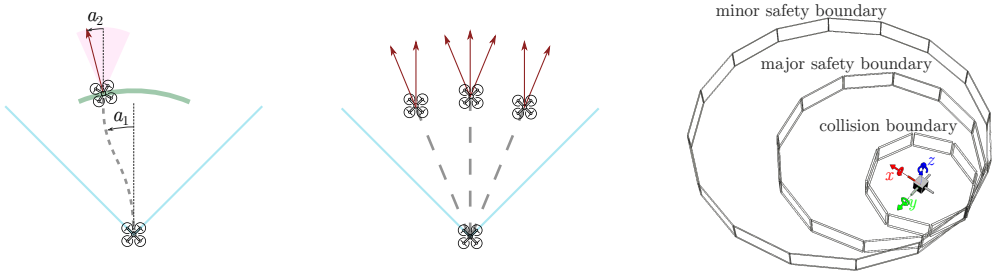
$$s_t = (I_{depth,t}, \mathbf{p}_t), \tag{1}$$

where $I_{depth,t}$ is 64×64 matrix representing depth image and $\mathbf{p}_t = [x_t, y_t]^T$ is 2×1 vector representing the position of the target point with respect to the body frame. As shown in Figure 1c, x-axis and y-axis represent the forward and the left direction, respectively.

The MDP environment is constructed for both continuous and discrete action spaces for comparison purposes. For the formation of continuous action space, a vector of length two is selected,

$$\mathbf{a}_t \in \{[a_1, a_2]^T | -\pi/8 \leq a_1, a_2 \leq \pi/8, a_1, a_2 \in \mathbb{R}\} \tag{2}$$

where a_1 defines the direction of 1 m position step and a_2 defines the rotation in yaw angle with respect to the current body frame. The distribution of actions is illustrated in Figure 1a. The boundaries of the action space are selected to match the information from the single-depth camera by assuring that all the actions are taken into a known area. On the other hand, yaw angle change enables a sharper turn around an obstacle, as well as a change in point of view if required.



(**a**) Continuous actions. (**b**) Discrete actions. (**c**) Safety boundaries.

Figure 1. The actions and the safety boundaries of the end-to-end planning agent are illustrated. (**a**) Continuous actions from the top view are defined by two angles: a_1 represents the direction of the position step (dashed lines), and a_2 represents the heading angle (red arrow). FOV of the depth camera is presented as blue lines. (**b**) Seven possible discrete actions: position and yaw angle steps. (**c**) Illustration of quadrotor body reference frame where the x-axis is the forward-looking direction and circular safety and collision boundaries considered in the simulation environment. The diameter of collision, major and minor safety boundaries are 1, 2, and 3 m, respectively.

The discrete action space, which is a subset of the aforementioned continuous action set, is comprised of seven actions, defined as a combination of a 1 m position step in three possible directions and a turn in yaw angle with respect to the drone's reference frame, as shown in Table 1. The possible actions are also illustrated in Figure 1b. Both the direction of position step and heading angle are a combination of the spatial limits of a continuous action set and forward direction while moving and opposite turning sides are neglected.

The discrete action set is mainly constructed for comparison with previous work [36], with some modifications. First, the position step direction, a_1, is kept small in order to fit with the field-of-view (FOV) of the depth camera so that the UAV does not hit an unseen object. Second, yaw angle change is enabled to match the capabilities of continuous actions. Finally, since we restrict the problem definition for constant altitude flight, we disable the actions that change altitude. We believe these updates facilitate a fair comparison between continuous and discrete action selections in such a problem domain.

Table 1. Discrete actions: each action is applied as a position step and a turn in heading angle with respect to the drone's reference frame.

Choice	Corresponding Continuous Action $[a_1, a_2]$
Action 1	$[\pi/8, \pi/8]$
Action 2	$[\pi/8, 0]$
Action 3	$[0, \pi/8]$
Action 4	$[0, 0]$
Action 5	$[0, -\pi/8]$
Action 6	$[-\pi/8, 0]$
Action 7	$[-\pi/8, -\pi/8]$

An episode begins when the UAV is at the beginning of a track defining a global trajectory of length L and obstacles placed. At each timestep, an action is applied to the UAV, then the depth image and next target point are obtained as the new state. Figure 2 illustrates the selection of the target point projected on the global path and 5 m ahead of the drone for consecutive timesteps. The episode is terminated under three conditions: crashing into an obstacle, deviating from the global trajectory, and finalizing the route.

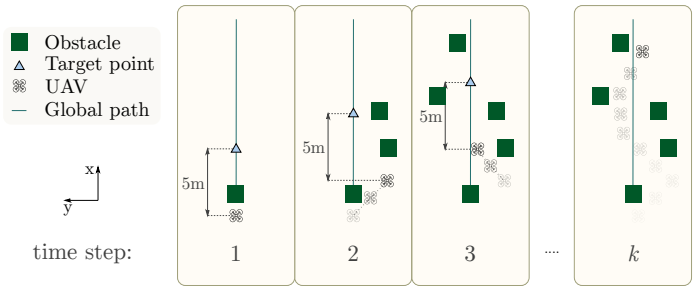


Figure 2. Moving target generation for the end-to-end agent. The target point is located 5 m ahead of the current location of the UAV projected onto the global path. The obstacles, generated target points, UAV, and global path are represented as shown in the legend. The UAV takes a position step at each timestep and is informed by the vector showing the generated target point. The overall trajectory is demonstrated at the k'th timestep.

A circular boundary is defined around the quadrotor with a diameter of 1 m to encounter collisions. Whenever a part of this boundary is violated by an obstacle, the collision is counted, and the episode is terminated. In addition to the collision boundary, two safety boundaries, major and minor, are defined with diameters 2 m and 3 m, centered at $[0.5, 0, 0]$ and $[1, 0, 0]$ on the quadrotor body frame. These safety boundaries are located toward the frontal area of the drone to detect risky objects in the short-term action path, as shown in Figure 1c. Unlike the collision, the violation of safety boundaries is not resulting in the termination of the episode, but it adds a negative reward to avoid being close to obstacles. Since the quadrotor motion is considered in the forward direction, these safety boundaries are chosen to be tangent with the collision boundary from the reverse direction. The diameters of major and minor safety boundaries are chosen to occupy the regions in two and three consecutive action steps, respectively.

The reward signal is based on the UAV's relative motion and the occupation of safety boundaries at every timestep if the episode is not terminated. For termination of an episode, both the collision and excessive deviation are punished with constant values. On the other hand, finishing a route without a crash is rewarded. The reward signal is defined as,

$$r_t = \begin{cases} 2\Delta x - d_y - 0.3d_\theta - 10\mathbb{1}_{major-safety} - 2\mathbb{1}_{minor-safety}, & \text{for non-terminal steps,} \\ R_{dp}, & \text{for } d_y > 5 \text{ m,} \\ R_{cp}, & \text{for collision,} \\ R_{fr}, & \text{for finishing normally,} \end{cases} \tag{3}$$

where Δx, d_y and d_θ are the distance traveled forward, the distance to the global trajectory and the yaw angle difference from forward-looking; $R_{dp} = -10$, $R_{cp} = -20$ and $R_{fr} = 20$ are punishment for excessive deviation, punishment for collision, and reward for finishing an episode without any crash; $\mathbb{1}_{major-safety}$ and $\mathbb{1}_{minor-safety}$ are indicator functions returning one or zero when corresponding safety boundary is occupied or not. This reward logic enables the agent to learn to avoid obstacles while quickly navigating toward the goal, as well as keeping a distance from obstacles thanks to safety boundaries.

3.2. Randomization of the Environment

For every episode of training, the obstacles in the environment are randomized. The randomization strategy is summarized in Algorithm 1. The algorithm randomly creates a corridor and places obstacles with varying shapes, sizes, orientations, and locations.

Algorithm 1 Randomized obstacle environment.

$with\_wall \sim \mathcal{U}(\{True, False\})$
if *with_wall* **then**
  $wall\_width \sim \mathcal{U}(4, 10)$
end if
$\#obstacles \sim \mathcal{U}(2, 7)$
for each obstacle **do**
  $obstacle\_shape \sim \mathcal{U}(\{Box, Sphere, Cylinder\})$
  $obstacle\_position.x \sim \mathcal{U}(3, L)$
  $obstacle\_position.y \sim \mathcal{N}(0, 2.5)$
  $obstacle\_position.z \sim \mathcal{U}(2, 3)$
  $randomize\_obstacle\_orientation()$
  if *obstacle_shape* is cylinder **then**
    $radius \sim \mathcal{U}(0.5, 1.5)$
    $height \sim \mathcal{U}(1, 3)$
  end if
  if *obstacle_shape* is box **then**
    $length \sim \mathcal{U}(0.5, 2.5)$
  end if
  if *obstacle_shape* is sphere **then**
    $radius \sim \mathcal{U}(0.5, 1.5)$
  end if
end for

3.3. Deep Reinforcement Learning: Actor and Critic Network Architecture

The actor and critic networks trained by PPO [7] rely on the same feature extractor. PPO is a policy gradient algorithm that optimizes the parameterized policy (actor) function, $\pi_\theta(a_t|s_t)$, with parameters, θ, using the clipped objective [7],

$$J^{CLIP} = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)], \quad (4)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ measures how different the new and old policy parameters. $\hat{A}_t$ is the advantage estimate in the given timestep, which measures how much a certain action acquires an extra long-term reward return using the parameterized value (critic) function. The optimization runs after every rollout of n_{steps} number of timesteps. Using this objective function, PPO increases the probability of good action decisions while suppressing bad decisions, similar to a previous DRL method trust region policy optimization (TRPO) [5], which only uses the expected value of $r_t(\theta)\hat{A}_t$. PPO introduces the clipped objective, which prevents large policy updates with only first-order optimization.

The actor–critic neural network feeds the depth image to three convolutional layers with the number of filters, kernel size, stride, and activation functions, as given in Figure 3. The convolutional layer is flattened and then reduced to a tensor of 256 neurons by a fully connected layer. This tensor is concatenated with the moving target input to create the feature vector that is shared by both actor and critic networks. The critic network utilizes two fully connected layers with 64 neurons each and a tangent hyperbolic (tanh) activation to regress the value function. The actor (policy) network has similar hidden layers to the critic network, but the output layer consists of n_a neurons where n_a is equal to two for continuous actions and seven for discrete actions.

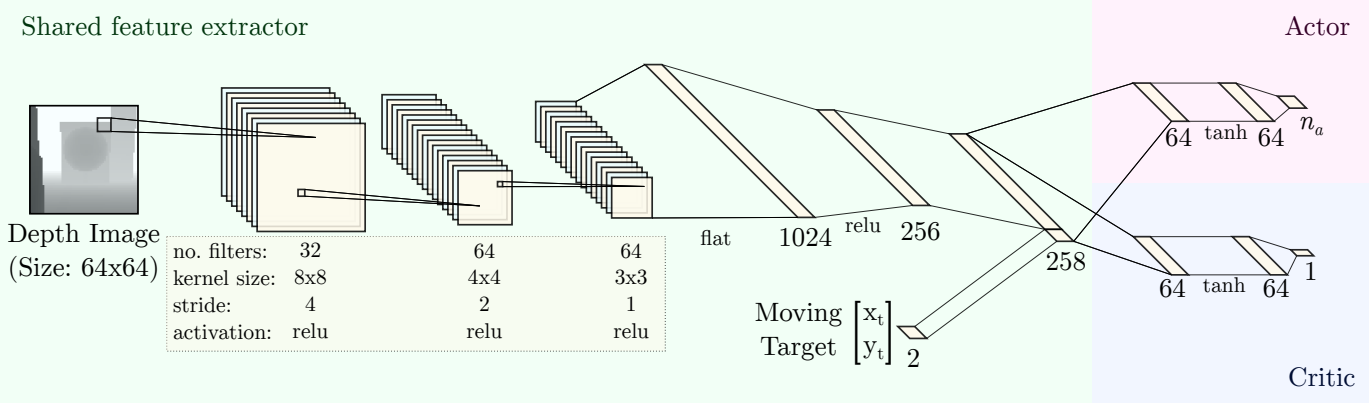


Figure 3. Actor–critic network structure of the end-to-end local planner. The actor network takes inputs as depth images through convolutional layers and the moving target through fully connected layers and outputs the next actions as position and yaw angle. The critic network shares the same feature extractor and outputs the value estimation.

4. Experiments and Results

4.1. Simulation Setup

A Webots-based simulation environment has been developed to train and test the proposed algorithms. Webots [38] is an open-source robot simulator that allows different programming interfaces, such as python, or robot operating system (ROS), for several kinds of robots. The 2021a release of Webots has been utilized to develop the cluttered environment and deploy the UAV with the required sensory equipment. A third-party software package, ArduPilot, is selected to implement a quadrotor UAV in Webots to benefit from its MAVLink extendable communication-featured stable and reliable UAV with its Webots SITL extension. The UAV robot is then equipped with a depth camera to provide the required information to carry out end-to-end planning operations. The environment needs to be reset every time a collision occurs in training, which is a benefit we can have in simulation throughout the trial-and-error process.

The simulation environment is wrapped as an OpenAI gym environment [39] to allow the required communication between the DRL algorithm and the environment. ROS [40] handles this communication between the gym wrapper and the simulation. Specifically, the MAVROS package is used to acquire the state estimation of the quadrotor UAV and send position commands. The remaining information, such as depth images and collision, is communicated directly by individual Webots ROS topics. The gym environment interfaces with the simulation environment as an MDP for the DRL algorithm, as explained in Section 3.1.

4.2. Training in Simulation

The agent is trained in Webots with randomized obstacle environments to present a variety of data for the deep network. The agent is subject to different obstacle shapes, sizes, locations, and densities for every episode of training. The randomization enables the agent to generalize the experience during RL training. Each episode begins on a randomly created route and terminates either at the end of the route or in a collision. Sample environment configurations used for evaluation purposes are shown in Figure 4.

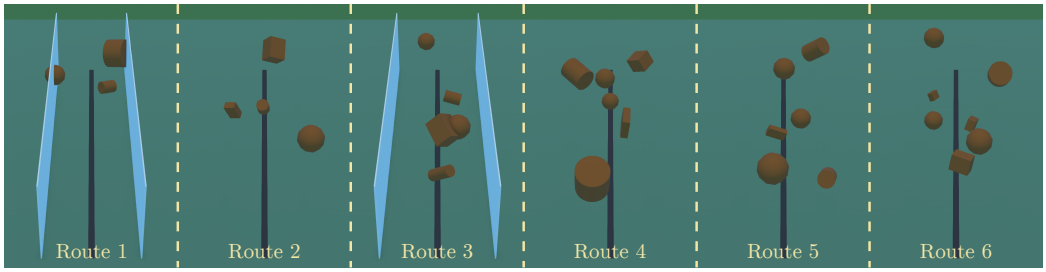


Figure 4. Six evaluation routes generated by the proposed environment randomizer. The black lines represent the projection of a 30 m global trajectory on the ground.

Since the policy network generates waypoints to travel, the robot is transported in the training simulations to acquire a new observation. This method reduces the computational burden for physical dynamics in each update step, thus fastening the overall training time. The transportation is also randomized in position and orientation in order to improve the variety of the training data as well as to address the possible poor performance of the controller in following the waypoints.

The policy network is trained with the PPO algorithm implementation in stable-baselines3 [41]. The 'number of steps to run per update' hyperparameter, n_{steps}, is set to 1024, while other hyperparameters are kept as default. The algorithm is trained through 100,000 timesteps, and the best network is stored during training based on the reward performance in the recent 20 episodes.

4.3. Simulation Results

The same randomization method is used for creating the evaluation routes. A set of six unique routes has been determined to test and compare the methods fairly, as shown in Figure 4. The routes are numbered with increasing order of the number of obstacles contained, which roughly makes the route more challenging. Furthermore, each route starts with a random offset of ± 0.5 m in the horizontal positioning of the drone for evaluation purposes. Each method is evaluated ten times in each lane. The success and the distance traveled without collision are recorded for every trial. In Table 2, the success rate and average traveled distance are listed for ten trials in each route. In addition, a safety cost is measured based on the inverse distance of the objects closer than 3 m, and the average of this safety cost over all runs is reported.

The proposed method, the safe continuous depth planner (SCDP), is compared with two DRL-based versions and a potential field-based planner. The continuous depth planner (CDP) considers the same method without introducing safety boundaries. The discrete depth planner (DDP) is considered as a baseline, which is a modified version of previous research [36] using a discrete action domain, as explained in Section 3.1. An artificial potential field-based planner (APF) is also implemented as a conventional baseline method [22]. The chosen baselines represent two important classes of motion planning algorithms for quadrotors: learning-based and model-based methods.

In the implementation of APF, each pixel in the middle row of the depth image creates a repulsive force, and the moving target creates an attractive force. As such, APF uses the same observation for the end-to-end planner. The action is also selected from the continuous action set according to the direction of the common artificial force in the reference frame of the UAV. The angle of the artificial force is mapped to the action set. When the angle is above $\pi/8$ in magnitude, it also activates yaw angle turning actions. The parameters of attractive and repulsive forces are tuned on the training routes and then tested to compare with the proposed method fairly.

As can be seen from Table 2, the proposed safety boundaries demonstrate better performance than the plain case in terms of the success rate. It is also observed that the final policy avoids becoming closer to the obstacles, considering the reported safety cost, because the rewards encountered with safety boundaries help the agent avoid dangerous situations.

The same observations are valid when the continuous action set is compared against the discrete set. The continuous depth planner is significantly more capable of handling dense obstacle scenarios, such as routes #5 and #6, since it can generate finer trajectories. Lastly, the learning-based end-to-end planners perform better than the baseline artificial potential field method because of their learning capabilities to handle uncertainties.

Table 2. Average travel distance (in meters) and success rate (in percentage) of methods—safe continuous depth planner (SCDP), continuous depth planner (CDP), discrete depth planner (DDP), and artificial potential field (APF)—over 10 runs at 6 test routes.

		Route 1	Route 2	Route 3	Route 4	Route 5	Route 6	Overall	Safety Cost
SCDP	**distance**	30	30	30	25.8	30	30	**29.7**	0.51
	success rate	100	100	100	70	100	90	**93**	
CDP	**distance**	30	30	30	19.4	30	27.4	28.0	0.52
	success rate	100	100	100	0	100	80	80	
DDP	**distance**	30	28.8	8.93	16.6	28.0	25.8	23.1	0.57
	success rate	100	90	0	0	80	60	55	
APF	**distance**	27.5	24.7	29.0	10.7	9.7	20.7	20.4	0.87
	success rate	90	10	90	10	0	60	43	

In order to provide a qualitative comparison, sample trajectories obtained from route #6 are presented in Figure 5. In parallel with the observations in the comparison table, SCDP tries to avoid risky situations. Additionally, the generated path by SCDP is smoother, implying consistency in the sequential actions. Intuitively having a smoother trajectory reduces the controller effort to follow provided waypoints, which is another advantage of SCDP against other methods.

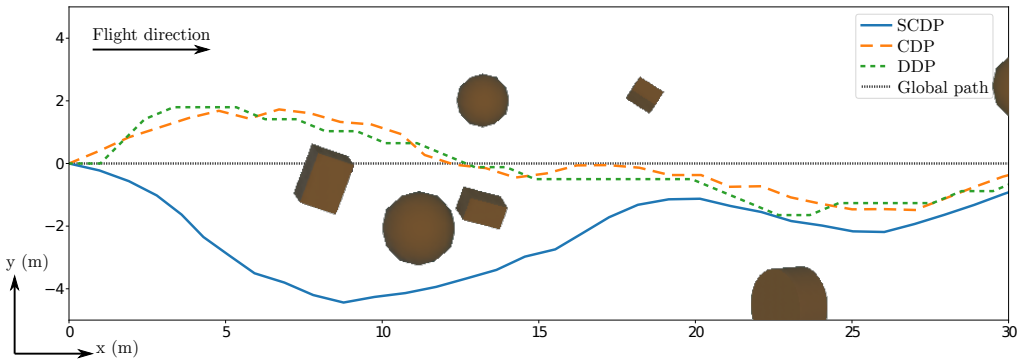


Figure 5. Comparison of the sample trajectories collected in route #6.

4.4. Real-Time Experiments

The trained model is deployed for real-time experiments in a custom quadrotor carrying an Intel Realsense D435i depth camera, as shown in Figure 6. The drone is controlled by a Pixhawk autopilot [42]. The overall framework runs entirely onboard on an NVIDIA Jetson TX2 computer, except that the robot's localization is provided by a motion capture system. The overall pipeline can run up to 8 Hz. A geometric controller [43] is used to track the poses generated by the policy accurately, with a linear speed of around 1 m/s.

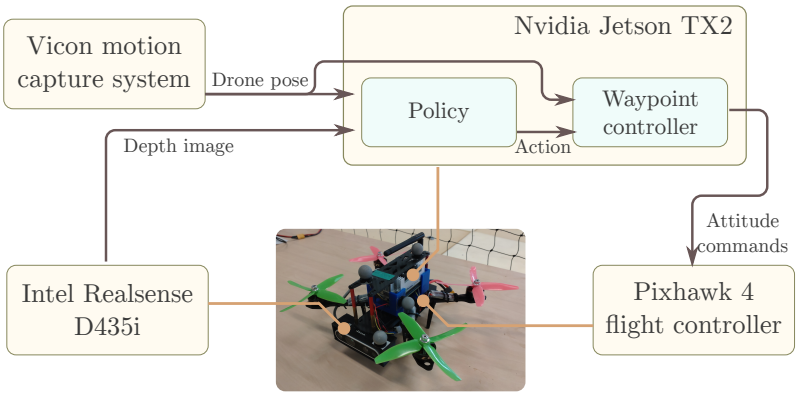


Figure 6. Custom drone used in real-time experiments. The depth images are acquired from an Intel Realsense D435i depth camera. The end-to-end planner is running onboard by an NVIDIA Jetson TX2. Pixhawk 4 flight controller is utilized for following the waypoints.

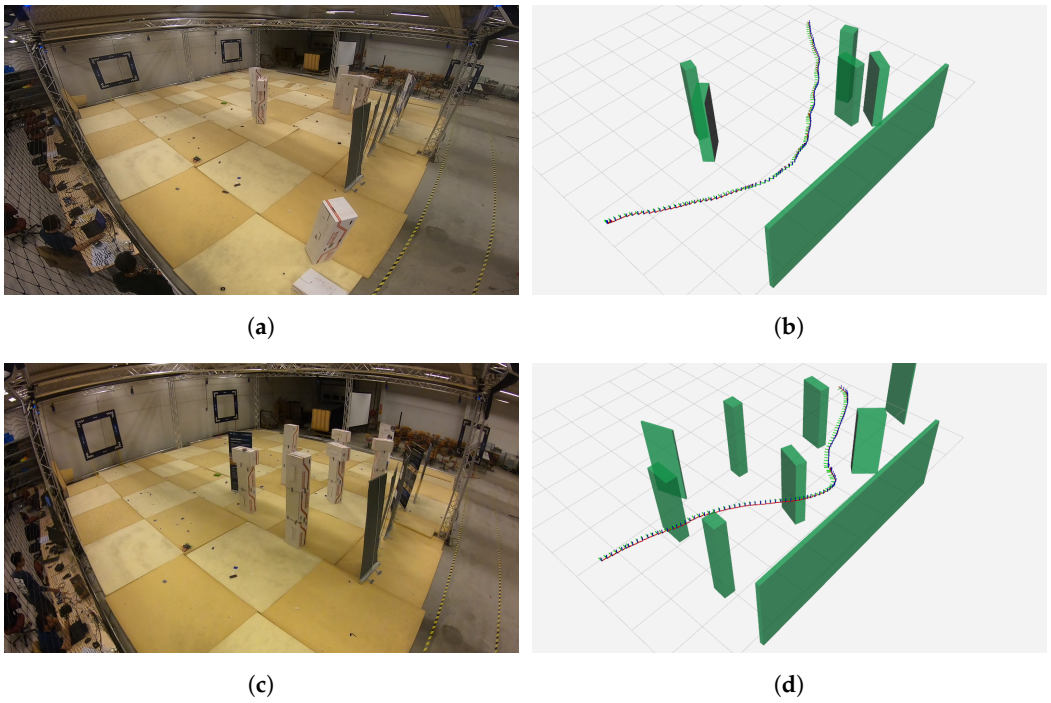
To cope with noisy real depth data, input images are enhanced by using a fast depth dilation algorithm [44] and then resized by cropping the top part of the image to 64×64 to feed the policy network. We find that processed depth images help to bridge the gap between simulation and the real world. Unlike the simulations, the generated actions are applied at the same frequency before reaching the waypoint to prevent the quadrotor from stopping after each action, which causes a lot of noise due to pitch movements. Additionally, the quadrotor can track the trajectory faster and smoother. Furthermore, the applied action is calculated as the mean of the recent two actions generated, which prevents the robot from applying oscillating actions, which might cause a failure due to noise. The consecutive oscillating actions are especially expected in narrow passages, where the drone successively observes the obstacles on the right and left and thus decides to switch directions.

The real-time experiments are conducted in two different scenarios, as shown in Figure 7. For the first scenario, a moderate-level experimental setup is designed by grouping obstacles into two groups with wider free space and making the obstacles larger and, hence, easier to observe. The second scenario is denser and more complex, using eight obstacles distributed around the global trajectory. The obstacles are created with cardboard boxes grouped in various configurations. Additionally, a wall-like structure is created using banners on the right side of the flight route (the video can be found: https://youtu.be/HPXXc_R3re8, accessed on 12 July 2022).

SCDP and DDP methods are executed five times each in both moderate and difficult-level scenarios. Table 3 presents the comparison of the two methods in both scenarios. Similar to the simulations, the drone successfully navigates through obstacles, with the narrowest passage being approximately three times the drone's size. The SCDP method succeeds in all trials in the moderate scenario, while one collision is observed with DDP. Similarly, SCDP outperforms in the difficult scenario yet encounters one collision. Although the DDP method also successfully avoids obstacles in most cases, the track cannot be finalized successfully; instead, the drone exits the global trajectory, which shows our framework handles the noisy and complicated inputs better by learning confident actions.

Table 3. Comparison of the SCDP and DDP in real-time experiments. The moderate and difficult-level scenarios are evaluated five times for both methods.

		Moderate Scenario	Difficult Scenario
SCDP	**success rate**	100%	80%
	collision rate	0%	20%
	distance (meters)	8	7.4
DDP	**success rate**	80%	0%
	collision rate	20%	20%
	distance (meters)	7.7	7.1

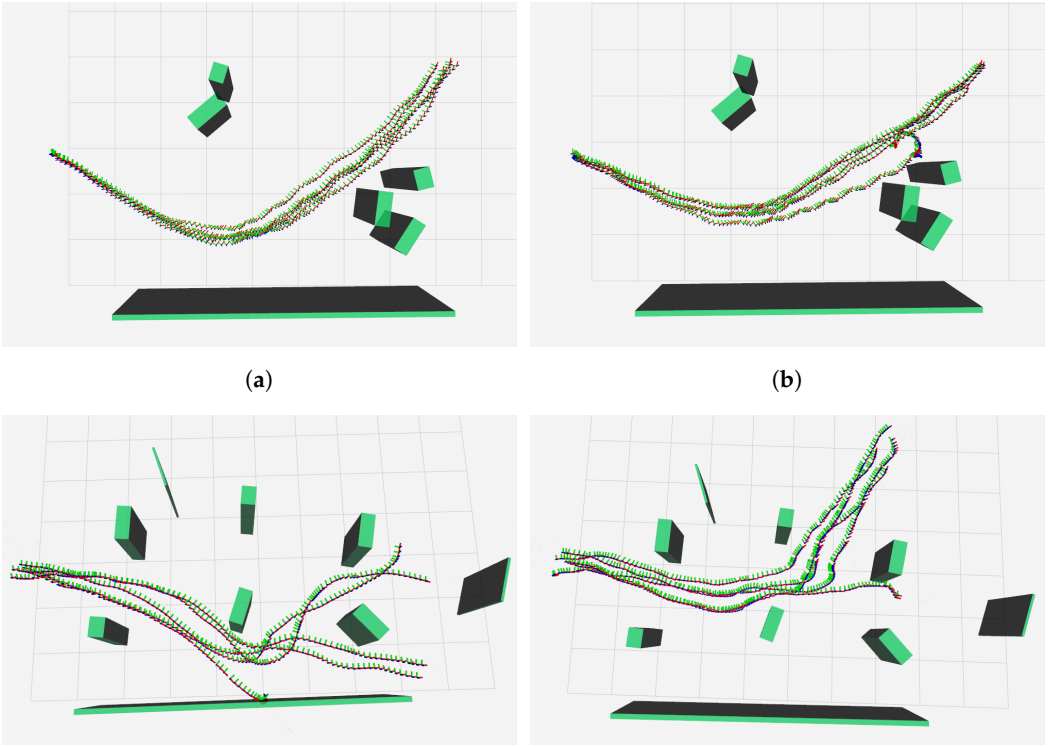


(**a**) (**b**)

(**c**) (**d**)

Figure 7. Moderate and difficult-level real-time experimental setups. (**a**) Real-time evaluation track with moderate-level obstacle configuration. (**b**) Visualization of the moderate-level obstacle configuration and a sample trajectory in RVIZ. (**c**) Real-time evaluation track with difficult-level obstacle configuration. (**d**) Visualization of the difficult-level obstacle configuration and a sample trajectory in RVIZ.

The trajectories obtained with each method and each scenario are visualized in Figure 8. Although it is practically more challenging to obtain the variety of obstacle configurations in real experiments than in simulation, the difficult scenario is observed to contain significant challenges to benchmark algorithms considering the variation of the resulting five trajectories. In contrast, in the moderate scenario, all trajectories follow a similar pattern. Together with the challenge of higher maneuverability, the difficult scenario also introduces more diverse, in-depth observations. Similar to the simulation results, the trajectories obtained by SCDP are smoother than the baseline method.



(**a**) (**b**)

(**c**) (**d**)

Figure 8. Trajectories acquired by five runs in the moderate and difficult scenario by SCDP and DDP methods. (**a**) SCDP in the moderate scenario. (**b**) DDP in the moderate scenario. (**c**) SCDP in the difficult scenario. (**d**) DDP in the difficult scenario.

5. Conclusions

In this work, an end-to-end planner is trained with DRL for safe navigation in cluttered obstacle environments. The end-to-end planning algorithm is trained and tested in comprehensive simulations developed in Webots. While the training of the policy network is handled without dynamics and control to save time, it is successfully sim-to-real transferred for physical evaluations. Moreover, safety boundaries for training are introduced, which successfully prevents the quadrotor from being in hazardous situations. The method is also deployed in real-world indoor environments successfully. The end-to-end planner outperforms a baseline implementation based on the artificial potential field method, which has a lower success rate, especially in cluttered obstacle settings. This shows that SCDP has learned to make better long-term decisions. The real-world experiments demonstrate that the proposed UAV planner trained solely with simulation can work directly in a real environment.

There are also certain limitations of the proposed method to be addressed in future work. First, although the proposed planning method does not require the computation of a map, the neural network-based method still requires significant computational resources in training and also in deployment. Currently, the inference time of the used network is not suitable for real-time robot control. If the algorithm can run continuously in real-time, there is a possibility to provide lower-level control commands, instead of waypoints, to the UAV, which can improve the tracking performance of the robot. Second, due to the black box characteristics of neural networks, the planner cannot be theoretically analyzed similarly to conventional planning methods, such as its completeness.

Author Contributions: Conceptualization, H.I.U. and E.K.; methodology, H.I.U.; software, H.I.U.; validation, H.I.U.; formal analysis, H.I.U.; investigation, H.I.U. and X.H.P.; resources, E.K.; data curation, H.I.U.; writing—original draft preparation, H.I.U.; writing—review and editing, H.I.U.; visualization, H.I.U.; supervision, E.K.; project administration, E.K.; funding acquisition, E.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European Union's Horizon 2020 Research and Innovation Program (OpenDR) grant number 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All of the codes and data will be available at https://github.com/open-airlab/safe-continuous-depth-planning.git, accessed on 2 July 2022.

Acknowledgments: The authors would like to thank Abdelhakim Amer for his technical support in conducting real-time experiments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

DRL	deep reinforcement learning
UAV	unmanned aerial vehicle
PPO	proximal policy optimization
FOV	field of view
PID	proportional-integral-derivative
MDP	Markov decision process
SITL	software-in-the-loop
ROS	robot operating system
SCDP	safe continuous depth planner
CDP	continuous depth planner
DDP	discrete depth planner
APF	artificial potential field

Appendix A

The following variables are used in this manuscript:

Table A1. Common variables in the manuscript.

t	discrete timestep
s_t	state at timestep t
a_t	action at timestep t
r_t	reward at timestep t
I_{depth}	matrix representing the depth image
L	length of the global trajectory
$\mathcal{U}$	uniform distribution
$\mathcal{N}$	normal distribution

References

1. Pham, H.X.; Ugurlu, H.I.; Le Fevre, J.; Bardakci, D.; Kayacan, E. Deep learning for vision-based navigation in autonomous drone racing. In *Deep Learning for Robot Perception and Cognition*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 371–406.
2. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
3. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]

4. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
5. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6 July–11 July 2015; pp. 1889–1897.
6. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
7. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
8. Muratore, F.; Ramos, F.; Turk, G.; Yu, W.; Gienger, M.; Peters, J. Robot learning from randomized simulations: A review. *Front. Robot. AI* **2022**, *9*, 799893. [CrossRef] [PubMed]
9. Hoeller, D.; Wellhausen, L.; Farshidian, F.; Hutter, M. Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5081–5088. [CrossRef]
10. Pham, H.X.; Sarabakha, A.; Odnoshyvkin, M.; Kayacan, E. PencilNet: Zero-Shot Sim-to-Real Transfer Learning for Robust Gate Perception in Autonomous Drone Racing. *arXiv* **2022**, arXiv:2207.14131.
11. Molchanov, A.; Chen, T.; Hönig, W.; Preiss, J.A.; Ayanian, N.; Sukhatme, G.S. Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 59–66.
12. Morales, T.; Sarabakha, A.; Kayacan, E. Image generation for efficient neural network training in autonomous drone racing. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
13. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement learning for UAV attitude control. *ACM Trans. Cyber Phys. Syst.* **2019**, *3*, 22. [CrossRef]
14. Ugurlu, H.I.; Kalkan, S.; Saranli, A. Reinforcement Learning versus Conventional Control for Controlling a Planar Bi-rotor Platform with Tail Appendage. *J. Intell. Robot. Syst.* **2021**, *102*, 77. [CrossRef]
15. Camci, E.; Kayacan, E. Learning motion primitives for planning swift maneuvers of quadrotor. *Auton. Robot.* **2019**, *43*, 1733–1745. [CrossRef]
16. Dooraki, A.R.; Lee, D.J. An innovative bio-inspired flight controller for quad-rotor drones: Quad-rotor drone learning to fly using reinforcement learning. *Robot. Auton. Syst.* **2021**, *135*, 103671. [CrossRef]
17. Brunke, L.; Greeff, M.; Hall, A.W.; Yuan, Z.; Zhou, S.; Panerati, J.; Schoellig, A.P. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annu. Rev. Control Robot. Auton. Syst.* **2022**, *5*, 411–444. [CrossRef]
18. Han, H.; Cheng, J.; Xi, Z.; Yao, B. Cascade Flight Control of Quadrotors Based on Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 11134–11141. [CrossRef]
19. Kaufmann, E.; Bauersfeld, L.; Scaramuzza, D. A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 10504–10510.
20. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
21. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 203–220.
22. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 25–28 March 1985; Volume 2, pp. 500–505.
23. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: https://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=FDD7D4058FECC1206F4FA333A3286F56?doi=10.1.1.35.1853 (accessed on 14 July 2022)
24. Zhou, D.; Wang, Z.; Schwager, M. Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures. *IEEE Trans. Robot.* **2018**, *34*, 916–923. [CrossRef]
25. Raigoza, K.; Sands, T. Autonomous Trajectory Generation Comparison for De-Orbiting with Multiple Collision Avoidance. *Sensors* **2022**, *22*, 7066. [CrossRef] [PubMed]
26. Feng, S.; Sebastian, B.; Ben-Tzvi, P. A Collision Avoidance Method Based on Deep Reinforcement Learning. *Robotics* **2021**, *10*, 73. [CrossRef]
27. Dooraki, A.R.; Lee, D.J. An end-to-end deep reinforcement learning-based intelligent agent capable of autonomous exploration in unknown environments. *Sensors* **2018**, *18*, 3575. [CrossRef] [PubMed]
28. Kang, K.; Belkhale, S.; Kahn, G.; Abbeel, P.; Levine, S. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 6008–6014.
29. Bonatti, R.; Madaan, R.; Vineet, V.; Scherer, S.; Kapoor, A. Learning visuomotor policies for aerial navigation using cross-modal representations. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 1637–1644.
30. Bonatti, R.; Wang, W.; Ho, C.; Ahuja, A.; Gschwindt, M.; Camci, E.; Kayacan, E.; Choudhury, S.; Scherer, S. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *J. Field Robot.* **2020**, *37*, 606–641. [CrossRef]

31. Polvara, R.; Patacchiola, M.; Hanheide, M.; Neumann, G. Sim-to-Real Quadrotor Landing via Sequential Deep Q-Networks and Domain Randomization. *Robotics* **2020**, *9*, 8. [CrossRef]
32. Bartolomei, L.; Kompis, Y.; Pinto Teixeira, L.; Chli, M. Autonomous Emergency Landing for Multicopters Using Deep Reinforcement Learning. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022), Kyoto, Japan, 23–27 October 2022.
33. Muñoz, G.; Barrado, C.; Çetin, E.; Salami, E. Deep reinforcement learning for drone delivery. *Drones* **2019**, *3*, 72. [CrossRef]
34. Doukhi, O.; Lee, D.J. Deep reinforcement learning for end-to-end local motion planning of autonomous aerial robots in unknown outdoor environments: Real-time flight experiments. *Sensors* **2021**, *21*, 2534. [CrossRef] [PubMed]
35. Loquercio, A.; Kaufmann, E.; Ranftl, R.; Müller, M.; Koltun, V.; Scaramuzza, D. Learning High-Speed Flight in the Wild. In Proceedings of the Science Robotics, New York, NY, USA, 27 June–1 July 2021.
36. Camci, E.; Campolo, D.; Kayacan, E. Deep reinforcement learning for motion planning of quadrotors using raw depth images. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–7.
37. Dooraki, A.R.; Lee, D.J. A Multi-Objective Reinforcement Learning Based Controller for Autonomous Navigation in Challenging Environments. *Machines* **2022**, *10*, 500. [CrossRef]
38. Michel, O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, 5. [CrossRef]
39. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
40. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
41. Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; Dormann, N. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
42. Meier, L.; Tanskanen, P.; Fraundorfer, F.; Pollefeys, M. Pixhawk: A system for autonomous flight using onboard computer vision. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2992–2997.
43. Faessler, M.; Franchi, A.; Scaramuzza, D. Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robot. Autom. Lett.* **2017**, *3*, 620–626. [CrossRef]
44. Ku, J.; Harakeh, A.; Waslander, S.L. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. In Proceedings of the 2018 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, Canada, 9–11 May 2018; pp. 16–22.