Learning Minimum-Time Flight in Cluttered Environments

Penicka, Robert ; Song, Yunlong ; Kaufmann, Elia ; Scaramuzza, Davide

DOI: https://doi.org/10.1109/LRA.2022.3181755

Learning Minimum-Time Flight in Cluttered Environments

Robert Penicka, Yunlong Song, Elia Kaufmann, Davide Scaramuzza

***Abstract*—We tackle the problem of minimum-time flight for a quadrotor through a sequence of waypoints in the presence of obstacles while exploiting the full quadrotor dynamics. Early works relied on simplified dynamics or polynomial trajectory representations that did not exploit the full actuator potential of the quadrotor, and, thus, resulted in suboptimal solutions. Recent works can plan minimum-time trajectories; yet, the trajectories are executed with control methods that do not account for obstacles. Thus, a successful execution of such trajectories is prone to errors due to model mismatch and in-flight disturbances. To this end, we leverage deep reinforcement learning and classical topological path planning to train robust neural-network controllers for minimum-time quadrotor flight in cluttered environments. The resulting neural network controller demonstrates substantially better performance of up to 19% over state-of-the-art methods. More importantly, the learned policy solves the planning and control problem simultaneously online to account for disturbances, thus achieving much higher robustness. As such, the presented method achieves 100% success rate of flying minimum-time policies without collision, while traditional planning and control approaches achieve only 40%. The proposed method is validated in both simulation and the real world, with quadrotor speeds of up to $42\,\mathrm{km\,h^{-1}}$ and accelerations of 3.6g.**

***Index Terms*—Integrated Planning and Learning, Motion and Path Planning, Reinforcement Learning**

Supplementary Material

Video: https://youtu.be/wR1niZvI3pI

I. Introduction

QUADROTORS are among the most agile and maneuverable flying machines [1] and have recently shown a substantial increase in autonomy capabilities [2]. This renders quadrotors the ideal platform for first responders to search for survivors as quickly as possible after natural disasters like earthquakes, forest fires, or floods. Though the astonishing agility of autonomous quadrotors has been demonstrated in many research labs [2]–[10], planning minimum-time trajectories in cluttered environments and navigating them without collision remains an open problem. To push research in the field of agile navigation and minimum-time planning, autonomous drone racing has emerged as a research field, with international competitions being organized, such as the Autonomous Drone Racing series at the recent IROS and NeurIPS conferences [11]–[13] and the AlphaPilot challenge [10], [14]. Drone racing requires flying a drone through a sequence of gates or doorways in minimum time while avoiding collisions with the environment, which is an ideal benchmark scenario for autonomous quadrotors being deployed in search and rescue scenarios.

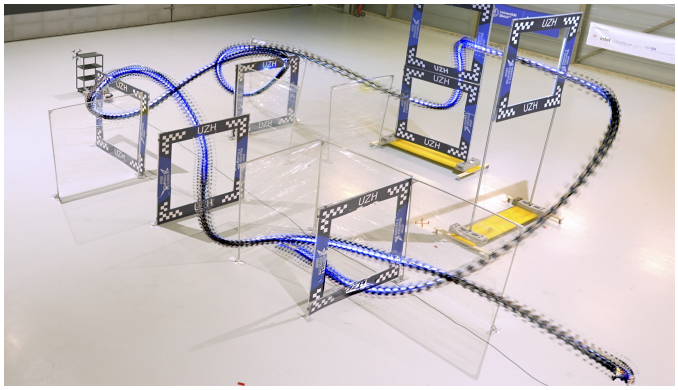


Fig. 1: Our quadrotor races through a complex race track in the real world while avoiding a set of obstacles made from transparent foil. While racing, the quadrotor reaches speeds of up to $42\,\mathrm{km\,h^{-1}}$ and accelerates with up to 3.6g.

By definition, the minimum-time objective requires the navigation and planning algorithm to constantly push the platform to its limits and operate the vehicle near the boundary of its physical envelope. Furthermore, the presence of 3D obstacles results in a highly nonconvex optimization problem that quickly becomes intractable to solve with traditional methods. These two aspects render minimum-time flight in cluttered environments very challenging as any slight disturbance or model mismatch could lead to a catastrophic crash. To this end, a planning and control method that tackles this task has to be robust against disturbances and needs to be able to adapt the trajectory online.

Previous work in the field of trajectory planning and control for autonomous quadrotors has solved only a subset of the problems imposed by minimum-time flight in cluttered environments. Existing methods either do not consider obstacles in time-optimal planning [4], [15], or cannot exploit the full actuation of the platform due to a simplification of the quadrotor dynamics [16]. Other methods do not support multi-waypoint scenarios in combination with a time-optimal objective [17], or rely on polynomial trajectory representations [18], that cannot represent time-optimal maneuvers due to their inherent smoothness. A recently proposed sampling-based method [19]

Manuscript received: February, 24, 2022; Revised May, 20, 2022; Accepted May, 31, 2022. This paper was recommended for publication by Editor Tetsuya Ogata upon evaluation of the Associate Editor and Reviewers' comments. The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, and Department of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland (https://rpg.ifi.uzh.ch). This work was supported as a part of NCCR Robotics, a National Centre of Competence in Research, funded by the Swiss National Science Foundation (grant number 51NF40_185543), the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE) and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT). Digital Object Identifier (DOI): 10.1109/LRA.2022.3181755

can plan minimum-time trajectories in cluttered environments; however, it is decoupled from the model predictive controller (MPC) [20] used to track the planned trajectory. This makes flying such a trajectory vulnerable to disturbances or model mismatches as the MPC does not account for the obstacles.

In light of recent successes in deep reinforcement learning (RL) [15], [21]–[24], we propose to address the problem using deep RL. RL has the advantage of automatically optimizing a parametric controller via trial and error and the ability to handle highly nonlinear dynamical systems and nonconvex objectives that are otherwise intractable to solve by conventional robotics methods. However, successful applications of RL have been largely limited to video games [21], [22], ground robots [23], or simple hovering of a quadrotor [24]. Applying RL to our problem remains a significant challenge due to the high sample complexity and nonconvexity of the task.

This work contributes a novel learning algorithm for minimum-time flight in cluttered environments. The key is to combine classical path planning with model-free deep reinforcement learning to optimize a neural network policy. The resulting policy directly outputs optimal control commands from high-dimensional observations. We show that the neural network policy outperforms state-of-the-art methods in terms of flight time in all tested scenarios that feature complex geometries. Our results indicate that the learned policy has obtained an implicit knowledge about the risk of navigating in close proximity of obstacles when being exposed to disturbances. This ability leads to more robust control performance and higher success rates when dealing with model mismatches. Our proposed approach is validated in real-world flights at speeds beyond $42\,\mathrm{km\,h^{-1}}$ and accelerations up to 3.6g.

II. Related Work

State-of-the-art methods for agile quadrotor flight mainly use the conventional approach to decouple trajectory planning and control. Given a planned trajectory, accurate trajectory tracking by the controller is instrumental for the vehicle to navigate through the environment safely. Hence, the final performance and success rate depend highly on both the quality of the planned trajectory as well as the robustness of the controller. One of the most popular paradigms to quadrotor trajectory generation exploits the differential flatness [25] of the platform using polynomial [10], [18], [26], [27] or B-spline [5], [28], [29] representations. However, those representations are suboptimal for minimum-time flight, since they are inherently smooth and cannot represent the rapid state or input changes at a reasonable order, and only reach the input limits for infinitesimal short durations [4].

Search-based planning methods [30], [31] use discrete-time and discrete-state representations and convert the trajectory planning to a graph search problem. These methods can optimize time up to discretization, however, they suffer from the curse of dimensionality and utilize simplified point-mass models instead of the full quadrotor dynamics. Furthermore, the existing search-based methods support planning only between two states. Algorithms like RRT* [17] can be used for a linearized quadrotor model around hover conditions, but the linearization around the hover operating point prohibits planning minimum-time trajectories.

More advanced trajectory planning methods frame the task as a constrained optimization problem and solve it via nonlinear programming. As such, trajectory optimization can be used for offline trajectory planning [4] or online tracking of a fixed reference path in a receding horizon fashion [32]. Optimization-based methods have the advantage of being able to incorporate nonlinear dynamics and constraints into the optimization framework. However, those methods either require long computation times (in the order of hours [4]) or rely on a series of approximations for the solver, which in turn results in sub-optimal performance. Neither of [4], [32] can solve the problem for environments that contain obstacles. In contrast, the sampling-based method [19] can find minimum-time trajectories also for cluttered environments. However, also this method plans a trajectory offline and therefore relies on a controller for tracking.

Modern control frameworks for trajectory tracking include nonlinear model predictive control (MPC) and differential flatness control [33]. However, most approaches struggle to handle disturbances during high-speed flight such as aerodynamic drag, thrust mismatches, and system delays. When the platform is at its actuation limit, the slightest deviation from the pre-planned trajectory may result in a suboptimal flight path, and even catastrophic crashes due to the presence of obstacles. There exist several MPC approaches that can handle obstacles adaptively online. However, they either consider simplified spherical obstacles [34], or a limited number of obstacles [35], [36] at a control frequency of only 10-20 Hz.

Learning-based methods address the aforementioned issues by learning an end-to-end policy that predicts control commands directly from high-dimensional observations. For example, imitation learning (IL) methods [2], [7] train neural network policies that can achieve agile flight in the wild using only onboard sensing and computing. IL is data-efficient, but not scalable since it requires designing an expert system for data collection. Recent works have demonstrated the usage of reinforcement learning to achieve superhuman performance in car racing [22], near-time-optimal flight in drone racing [15], and high-speed trajectory tracking using a learned policy [37]. Inspired by [15], [19], this work combines the topological path planning approach with deep RL to achieve minimum-time flight in complex cluttered environments.

III. Problem Statement

A. Quadrotor Dynamics

The quadrotor is modeled with state $\boldsymbol{x} = \left[\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\omega}, \boldsymbol{\Omega}\right]^T$ which consists of position $\boldsymbol{p} \in \mathbb{R}^3$, velocity $\boldsymbol{v} \in \mathbb{R}^3$, unit quaternion rotation $\boldsymbol{q} \in \mathbb{SO}(3)$, body rates $\boldsymbol{\omega} \in \mathbb{R}^3$, and the rotors' rotational speed $\boldsymbol{\Omega}$. The dynamics equations are

$$\begin{aligned} \dot{\boldsymbol{p}} &= \boldsymbol{v} & \dot{\boldsymbol{q}} &= \frac{1}{2}\boldsymbol{q} \odot \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \\ \dot{\boldsymbol{v}} &= \frac{R(\boldsymbol{q})(\boldsymbol{f}_T + \boldsymbol{f}_D)}{m} + \boldsymbol{g} & \dot{\boldsymbol{\omega}} &= \boldsymbol{J}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega}) \end{aligned} \tag{1}$$

where $\odot$ denotes the quaternion multiplication, $R(\boldsymbol{q})$ is the quaternion rotation, m is the mass, $\boldsymbol{J}$ is diagonal inertia ma-

trix, and $\boldsymbol{g}$ denotes Earth's gravity. The speeds of the propellers $\boldsymbol{\Omega}$ are modeled as a first-order system, $\dot{\boldsymbol{\Omega}} = \frac{1}{k_{mot}}(\boldsymbol{\Omega}_c - \boldsymbol{\Omega})$ with $\boldsymbol{\Omega}_c$ being the commanded speed and k_{mot} the time constant.

The collective thrust $\boldsymbol{f}_T$ and torque $\boldsymbol{\tau}_b$ are calculated as:

$$\boldsymbol{f}_T = \begin{bmatrix} 0 \\ 0 \\ \sum f_i \end{bmatrix}, \boldsymbol{\tau} = \begin{bmatrix} l/\sqrt{2}(f_1 - f_2 - f_3 + f_4) \\ l/\sqrt{2}(-f_1 - f_2 + f_3 + f_4) \\ \kappa(f_1 - f_2 + f_3 - f_4) \end{bmatrix}, \quad (2)$$

where κ is the torque constant and l is the arm length. Here, individual motor thrusts f_i are functions of the motor speeds using the thrust coefficient c_f as in (3),

$$f_i(\Omega) = \left[c_f \cdot \Omega^2\right]. \quad (3)$$

The drag force $\boldsymbol{f}_D$ is modeled as a linear function of velocity in body frame $\boldsymbol{v}_{\mathcal{B}}$ [38] with drag coefficients (k_{vx}, k_{vy}, k_{vz}):

$$\boldsymbol{f}_D = -\begin{bmatrix} k_{vx} v_{\mathcal{B},x} & k_{vy} v_{\mathcal{B},y} & k_{vz} v_{\mathcal{B},z} \end{bmatrix}^T. \quad (4)$$

The motors have a limited thrust range $[f_{min}, f_{max}]$:

$$f_{min} \leq f_i \leq f_{max}, \text{ for } i \in \{1, \dots, 4\}. \quad (5)$$

B. Minimum-time Planning Problem

The minimum-time planning problem is defined using the classical notion of configuration space $\mathcal{C}$ [39]. We assume an environment $\mathcal{W} = \mathbb{R}^3$, which contains obstacles $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_m\} \subset \mathcal{W}$. The quadrotor with state $\boldsymbol{x}$ and geometry $\mathcal{A}(\boldsymbol{x}) \subset \mathcal{W}$ has to find a collision-free trajectory in $\mathcal{C}$. To this end, it can move in free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, where $\mathcal{C}_{obs} = \{\boldsymbol{x} \in \mathcal{C} | \delta(\mathcal{A}(\boldsymbol{x}), \mathcal{O}) \leq d_c\} \subseteq \mathcal{C}$ is a set of configurations (quadrotor states) where the robot is in collision, i.e., having shortest distance $\delta(\cdot, \cdot)$ to any obstacle below a given threshold d_c.

We formulate the multi-waypoint minimum-time planning problem as an optimization problem (6) to find trajectories τ_i and their durations t_i for the dynamics (1)-(5).

$$\begin{aligned} \underset{\tau_0 \dots \tau_N}{\text{minimize}} \ & T = \sum_{i=0}^{N} t_i \\ \text{s.t.} \ & \tau_i \in \mathcal{C}_{free} \text{ for } i \in \{0, \dots, N\}, \\ & \tau_0(0) = \boldsymbol{x}_s, \ \tau_N(1) = \boldsymbol{x}_e, \\ & \|\tau_i(0)_p - \boldsymbol{p}_{wi}\| \leq r_{tol} \text{ for } i \in \{1, \dots, N\}, \\ & \tau_{i-1}(1) = \tau_i(0) \text{ for } i \in \{1, \dots, N\}, \\ & (1), (2), (3), (4), (5). \end{aligned} \quad (6)$$

For the multi-waypoint scenario, the quadrotor has to fly through a given sequence of waypoints $P_w = (\boldsymbol{p}_{wi}, i \in [1, \dots, N])$ while reaching their position $\boldsymbol{p}_{wi}$ with a certain proximity r_{tol}. The whole multi-waypoint trajectory can be described as a continuous sequence of N trajectories $\tau_i : [0, 1] \to \mathcal{C}_{free}$ for $i \in \{0, \dots, N\}$. The trajectory is assumed to have a given start $\tau_0(0) = \boldsymbol{x}_s$ and end $\tau_N(1) = \boldsymbol{x}_e$. Furthermore, the initial positions of the trajectories $\tau_i(0)_p$ have to be in the waypoints' proximity $\|\tau_i(0)_p - \boldsymbol{p}_{wi}\| \leq r_{tol}$ for $i \in \{1, \dots, N\}$, and the sequence has to be continuous $\tau_{i-1}(1) = \tau_i(0)$ for $i \in \{1, \dots, N\}$. The goal of the planning problem is then to minimize the final time $T = \sum_{i=0}^{N} t_i$ of reaching $\boldsymbol{x}_e$, where t_i denotes the time duration of τ_i.

IV. Methodology

The key ingredients of our approach to minimum-time flight in cluttered environments are three-fold: 1) generation of a topological guiding path using a probabilistic roadmap [40], 2) a novel task formulation that combines progress maximization along the guiding path with obstacle avoidance, and 3) a curriculum training strategy to train a neural network policy using deep reinforcement learning.

A. Topological Path Planning

We first find the topological paths that connect individual waypoints to guide the subsequent learning process. The topological guiding paths are found using a variant of the Probabilistic Roadmap [40] described in [19]. The algorithm searches for multiple distinct paths with different homotopy classes, e.g., going around obstacles from different sides. It uses random sampling in ellipsoids between individual waypoints and keeps enlarging each ellipsoid and number of samples until at least one path between waypoints is found.

The created roadmaps between waypoints are then searched for the shortest path using Dijkstra's algorithm. Samples within the shortest path with the smallest distance to obstacles are then removed from the roadmap and the shortest path search is repeated to obtain multiple distinct paths. Such distinct paths are then shortened, and the paths within the same homotopy class are removed. The resulting topological paths then represent the connectivity of the $\mathcal{C}_{free}$ between waypoints (as shown in Fig. 2(a)). For more details about the topological path search, we refer to [19].

B. Reinforcement Learning for Minimum-time Flight

In the following, we present the policy architecture, reward formulation, and strategy employed in our approach to train a policy for high-speed flight in cluttered environments.

Policy Architecture. The neural network policy uses an observation space that consists of three main parts: the quadrotor state, the next waypoint position, and the relative position of the farthest collision-free position on the guiding path. Specifically, we denote the observation vector as $\boldsymbol{o} = [\boldsymbol{p}(t), R(\boldsymbol{q}(t)), \boldsymbol{v}(t), \mathcal{W}, \boldsymbol{\gamma}(t)]$, where $\boldsymbol{p}(t)$, $R(\boldsymbol{q}(t))$, $\boldsymbol{v}(t)$ are the quadrotor's position, rotation matrix and velocity, respectively. Matrix $\mathcal{W} \in \mathcal{R}^{4,3}$ contains four positions corresponding to the bounding box of the currently targeted waypoint with edge size of $2r_{tol}$. The bounding box is used to convey information not only about the waypoint position, but also about the size and orientation of the target gate (see Fig. 1). Finally, the $\boldsymbol{\gamma}(t)$ is the farthest point on the guiding path connectable using a collision-free line segment from the current position $\boldsymbol{p}(t)$ (see Fig. 4). The $\boldsymbol{\gamma}(t)$ is used for indicating the flight direction of the quadrotor to maximize the progress. Throughout the development of the method, we found by ablating the observation components that all the components are necessary to learn minimum-time flight in all tested scenarios.

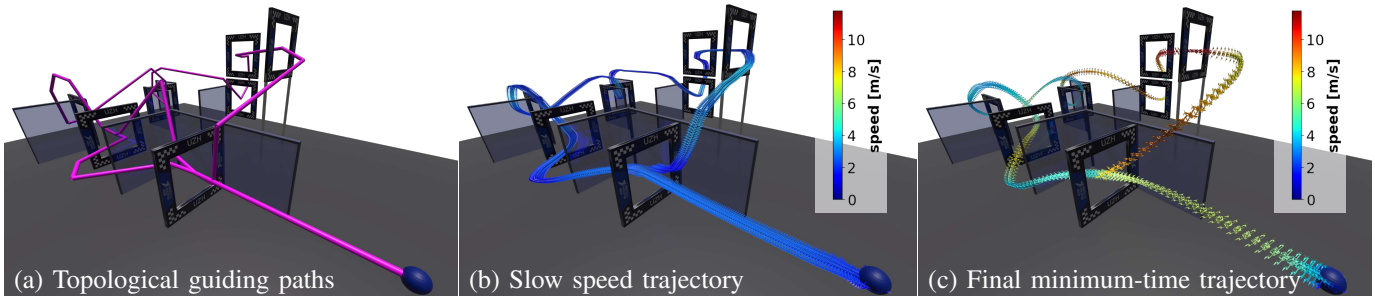


Fig. 2: Three main steps of our method (shown in Slalom environment) include: (a) finding topological guiding paths between the waypoints (gates), (b) learning slow policy that flies through all the waypoints, and (c) learning minimum-time policy.

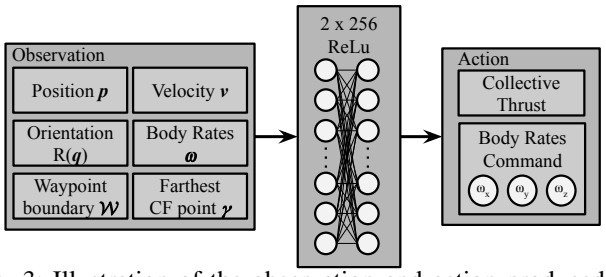


Fig. 3: Illustration of the observation and action produced by the neural network policy.

The action produced by the policy $\boldsymbol{a}(t) = [f_T, \boldsymbol{w}]$ is the collective thrust f_T in body-z axis and the commanded body rates $\boldsymbol{w}$. This action modality has been identified in [37] as the best performing for learning-based control policies. A low-level controller then tracks the desired collective thrust and body rates to produce the speed command for each rotor. Figure 3 illustrates the observation and action spaces, as well as the neural network architecture, which is a 2-layer multilayer perception (MLP).

Path Progress Maximization and Obstacle Avoidance. The objective of the studied problem is to minimize the time of reaching the last waypoint, which however, represents a very sparse signal that is difficult to optimize. This is why prior works opted for a dense proxy reward using the projected progress along the center line of a race track in car racing [22], or progress along the straight line segments between gates [15] for the task of drone racing. We further extend this progress-based reward for the task of minimum-time planning in cluttered environments by calculating the progress along the topological guiding path.

Figure 4 illustrates how the progress is computed between two consecutive states with positions $\boldsymbol{p}(t-1)$ and $\boldsymbol{p}(t)$. We assume the guiding topological path between start waypoint and end waypoint consists of a sequence of n points $(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n)$ that form a sequence of line segments $(l_1, \ldots, l_{n-1})$. To calculate the progress at position $\boldsymbol{p}$, we need to find the closest point $\psi(\boldsymbol{p})$ on the guiding path and its line segment index $l(\boldsymbol{p})$ as:

$$
\begin{aligned}
l(\boldsymbol{p}), \boldsymbol{\psi}(\boldsymbol{p}) = & \underset{l(\boldsymbol{p}), \psi(\boldsymbol{p})}{\arg\min} \|\boldsymbol{p} - \boldsymbol{\psi}(\boldsymbol{p})\| \\
\text{s.t. } & \boldsymbol{\psi}(\boldsymbol{p}) = \boldsymbol{g}_{l(\boldsymbol{p})} + t(\boldsymbol{g}_{l(\boldsymbol{p})+1} - \boldsymbol{g}_{l(\boldsymbol{p})}), \\
& t = \frac{(\boldsymbol{p} - \boldsymbol{g}_{l(\boldsymbol{p})}) \cdot (\boldsymbol{g}_{l(\boldsymbol{p})+1} - \boldsymbol{g}_{l(\boldsymbol{p})})}{\left\|\boldsymbol{g}_{l(\boldsymbol{p})+1} - \boldsymbol{g}_{l(\boldsymbol{p})}\right\|^2}, \\
& l(\boldsymbol{p}) \in \{1, \ldots, n-1\}\ ,\ t \in [0, 1].
\end{aligned} \tag{7}
$$

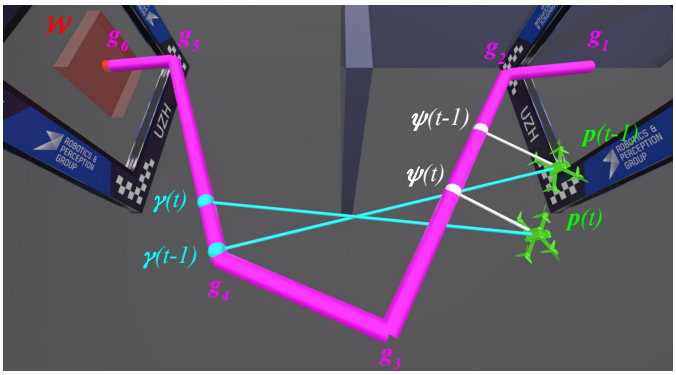


Fig. 4: Illustration of the collision-free topological guiding path between waypoints. The nearest point on the guiding path $\boldsymbol{\psi}$ from quadrotor position $\boldsymbol{p}$ is used to calculate progress reward. The farthest collision-free point $\boldsymbol{\gamma}$ and waypoint bounding box $\mathcal{W}$ are used as a part of observation.

The reached distance $s(\boldsymbol{p})$ along the guiding path is then calculated using (8) as the length of the topological path until the closest point $\boldsymbol{\psi}(\boldsymbol{p})$. The progress reward $r_p(t)$ at time t is then computed using equation (9) as a difference in reached distance between the current and previous time step.

$$
s(\boldsymbol{p}) = \sum_{i=1}^{l(\boldsymbol{p})-1} \|\boldsymbol{g}_{i+1} - \boldsymbol{g}_i\| + \left\|\boldsymbol{\psi}(\boldsymbol{p}) - \boldsymbol{g}_{l(\boldsymbol{p})}\right\| \tag{8}
$$

$$
r_p(t) = s(\boldsymbol{p}(t)) - s(\boldsymbol{p}(t-1)) \tag{9}
$$

The total reward $r(t)$ at time t then equals

$$
r(t) = k_p r_p(t) + k_s s(\boldsymbol{p}(t)) + k_{wp} r_{wp} + r_T - k_\omega \|\boldsymbol{\omega}\|\,, \tag{10}
$$

where k_p, k_ω, and k_{wp} are hyperparameters that define the contribution of each reward component. The reached distance $s(\boldsymbol{p}(t))$ with parameter k_s is used as part of the reward mainly to counteract the fact that the progress along the line segments can have many singularities. Such singularities emerge due to the sharp corners of the guiding path (see Fig. 4) and minimum-distance projection, which can block learning a policy that flies through all waypoints when negative progress occurs in the singularities.

The total reward also values passing of a waypoint $k_{wp} r_{wp}$ and discourage high body rates $k_\omega \|\boldsymbol{\omega}\|$. When a new waypoint is passed within distance $d_w \leq r_{tol}$ a positive reward of $r_{wp} = e^{-d_{wp}/r_{tol}}$ is added to prioritize passing close to the waypoint center and thus to increase robustness during real flight with disturbances. The terminal reward $r_T = -10$ is only added when the quadrotor collides with an obstacle.

During the development of the method, we tested several reward components and their ablation, rendering only the presented variant able to learn policies for all tested scenarios.

Training Strategy. Naive optimization of the reward formulation specified in (10) results in suboptimal performance due to local minima and a higher probability of collisions in high-speed flight. This is caused by the decoupled nature of the topological path planning and the reinforcement learning of minimum-time flight for the dynamic quadrotor model. Such decoupling makes the learning of high-speed flight along the topological paths a challenging problem.

We overcome this limitation by employing a curriculum strategy, where the racing policy is trained in two stages. In the first stage, a slow policy (see Fig. 2(b)) is trained to fly closely along the guiding path while the minimal v_{min} and maximal v_{max} speeds are limited through a scaled reward. This helps to find a policy that flies through the waypoints without collision as the guiding paths are known to be collision free and the high velocities would decrease the relative size of narrow passages. Specifically, in the initial slow flight training stage, the parameters of progress reward k_p and reached distance reward k_s are scaled down by a factor of s computed as:

$$s = s_{v_{max}} s_{v_{min}} s_{gd}, \tag{11}$$

$$s_{v_{max}} = \begin{cases} 10^{v_{max} - \|\boldsymbol{v}\|} & \text{if } \|\boldsymbol{v}\| > v_{max}, \\ 1 & \text{otherwise,} \end{cases} \tag{12}$$

$$s_{v_{min}} = \begin{cases} 10^{\|\boldsymbol{v}\| - v_{min}} & \text{if } \|\boldsymbol{v}\| < v_{min}, \\ 1 & \text{otherwise,} \end{cases} \tag{13}$$

$$s_{gd} = \begin{cases} e^{-\|\boldsymbol{p} - \boldsymbol{\psi}(\boldsymbol{p})\| + d_{max}} & \text{if } \|\boldsymbol{p} - \boldsymbol{\psi}(\boldsymbol{p})\| > d_{max}, \\ 1 & \text{otherwise.} \end{cases} \tag{14}$$

This adapted reward formulation forces the policy to find only slow trajectories with speed $v_{min} < \|\boldsymbol{v}\| < v_{max}$ and distance from the guiding paths $\|\boldsymbol{p} - \boldsymbol{\psi}(\boldsymbol{p})\| < d_{max}$, which in turn helps finding a trajectory around an already known collision-free guiding path.

The limited maximal speed in the slow flight training stage is also used to calculate the initial value of the parameter k_s as $k_s = 2(v_{max} \cdot d_t) / \sum_{i=1}^{n-1} \|\boldsymbol{g}_{i+1} - \boldsymbol{g}_i\|$, where d_t is the simulation time step. This limits the collected reward from reached distance to be approximately the same as the progress reward in the slow flight learning phase. While during the later minimum-time learning phase, the progress reward is, in comparison, significantly more prominent.

After the trained slow flight policy is able to navigate through all waypoints, the speed limits and the constrained distance to the guiding path are removed to enable training a minimum-time policy as shown in Fig. 2(c).

Training Details. The policy is trained using Proximal Policy Optimization (PPO) [41], which has demonstrated good performance in benchmarks for continuous control tasks. We utilize 100 parallel agents to train the policy entirely in simulation, which increases the speed of collecting data and diversifies the experienced states and observations among agents. The simulation uses the dynamics 1-5 and forward integrates them using a 4th order Runge-Kutta scheme. Additionally, the linear drag coefficients (k_{vx}, k_{vy}, k_{vz}) are randomized with normal distributions $N(0, k_{vx}), N(0, k_{vy}), N(0, k_{vz})$ for each agent after restart, to make the policy robust against unknown and possibly random aerodynamic effects.

Initialization of the agents is randomized among all specified waypoints and guiding paths to encourage diversity of experienced states. Each agent keeps a vector of valid states, i.e., states that are reached by running the policy from the start position without collision. In the first learning stage, the valid states are required to be within the speed limit $v_{min} < \|\boldsymbol{v}\| < v_{max}$ and close to the guiding path $\|\boldsymbol{p} - \boldsymbol{\psi}(\boldsymbol{p})\| < d_{max}$. The guiding paths are discretized into $1\,\mathrm{m}$ parts based on the distance along the path $s(\boldsymbol{p})$, and each such part is mapped to one valid state in the vector of agent's valid states. These valid states are then used to randomly initialize the state after an agent collides with an obstacle or reaches the final waypoint.

All policies are trained using 12 threads on a laptop featuring an Intel Xeon W-10885M CPU and a Quadro RTX 4000 Mobile GPU.

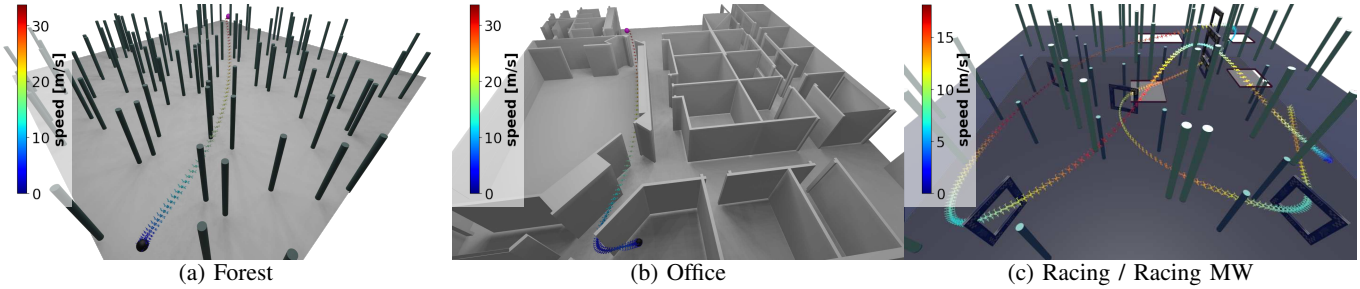
V. Results

The proposed method has been evaluated concerning the following performance criteria: (i) lap time performance of the planned trajectory in a simple simulation scenario without model mismatch, (ii) success rates of navigating the vehicle through a cluttered environment using high-fidelity simulation, and (iii) validation of the learned policy in the real world.

We identify a powerful race drone using real world flight data. The physical properties of the quadrotor, along with hyperparameters of the proposed and baseline methods, are summarized in Table I. The hyperparameters of the reward components were tuned such that the collected rewards from the individual components follow a particular priority, i.e. $-r_T \gg k_p r_p(t) \approx k_{wp} r_{wp} \gg k_s s(\boldsymbol{p}(t)) \approx -k_\omega \|\boldsymbol{\omega}\|$. The evaluation of the method is done in four environments. The Slalom environment is shown in Figure 2. The Forest, Office, and Racing environments, including the Multi-Waypoint (MW) Racing environment, are illustrated in Figure 5. The environments are represented as an Euclidean Signed Distance Field [42] with precision of $0.05\,\mathrm{m}$. We use Flightmare [43] for the simulation and the Stable Baselines [44] for policy training.

TABLE I: Parameters of the quadrotor and the algorithms.

	Variable	Value	Variable	Value
Quadrotor	m [kg]	0.85	l [m]	0.15
	f_{min} [N]	0	f_{max} [N]	7
	diag(J) [g m^2]	[1, 1, 1.7]	κ [-]	0.05
	w_{max} [rad s^{-1}]	15	c_f	1.563×10^{-6}
	k_{vx} [N s m^{-1}]	0.26	k_{vy} [N s m^{-1}]	0.28
	k_{vz} [N s m^{-1}]	0.42		
[18]	k_T	105000	N_{poly}	10
[30]	a_{max} [m s^{-2}]	31.4		
RL	k_p[-]	5.0	k_ω[-]	0.01
	k_{wp}[-]	5.0	d_t [s]	0.02
	v_{max} [m s^{-1}]	2	v_{min} [m s^{-1}]	1
	d_{max} [m]	0.3	d_c [m]	0.15



(a) Forest (b) Office (c) Racing / Racing MW

Fig. 5: Example trajectories found by the proposed method in the selected environments used for evaluation.

A. Lap Time Performance of Planned Trajectories in Simple Simulation

In the simple simulation, we analyze the quality, such as duration, of the planned trajectories and compare it with related baseline algorithms. The first baseline algorithm is the polynomial method [18] that jointly minimizes snap and time (using time penalty k_T) of a trajectory that is represented by an N_{poly}-th order polynomial. The search-based method [30] for quadrotor planning uses a discretized state of a point-mass up to acceleration to plan minimum-time trajectories using a graph-search algorithm. The search-based method uses an acceleration limit of $a_{max} = 31.4\ \mathrm{m\,s^{-2}}$, which represents the optimistic, yet infeasible, limit corresponding to achievable acceleration along one axis in horizontal motion. The optimization method presented in [4] plans truly time-optimal trajectories between given waypoints, however, the original variant had to be extended to allow planning in cluttered environments. Lastly, the sampling-based method [19] uses a hierarchical approach of increasing model complexity to plan minimum-time trajectories for the full quadrotor model in cluttered environments.

To allow for a fair comparison, the motor dynamics $\dot{\Omega}$ and the aerodynamic drag forces $\boldsymbol{f}_d$ are not considered in the simple simulation. This corresponds to the baseline methods [4], [18], [19], [30], which all do not account for both phenomena, except for [4] that can include a simple linear drag model. The comparison of the planning methods is presented in Table II. The trajectory duration is reported as the best-found duration T_b. Additionally, we show the average duration with standard deviation T_a from 30 different runs for the methods that are randomized and have non-zero deviation. We report the computation times for the baseline methods. For the proposed method, we show the inference time (including the time for preprocessing observations) of the trained neural network during evaluation. For each scenario, a different policy is learned with training time of approximately 45 minutes with a standard laptop.

Table II shows that the polynomial method has the highest trajectory duration while being the fastest in computation. In contrast, the search-based method generates substantially faster trajectories; however, it requires longer computation times. The search-based method is severely limited by the state space discretization, which limits finding truly time-optimal continuous-space trajectories. Furthermore, the method only returns valid solutions in the 2D Forest and Office environments, while it failed to find valid solutions for the 3D Racing environment and multi-waypoint scenarios. The optimization-based method can find the time-optimal trajectories only for the three simple Forest test cases. It fails in the other environments due to the introduced non-convex collision avoidance constraints. The sampling-based method is capable of finding high-quality solutions for all considered test cases; however, the computational time is comparable to [4], [30].

Finally, our learning-based method can achieve on par, mostly better, solutions compared to the conventional methods. In test cases where the baselines slightly outperform our approach, the difference is within, or close to, the time precision $d_t = 0.02\,\mathrm{s}$ of the RL policy. We observe that the performance margin of our approach compared to the baselines increases with increasing environment complexity.

In simplistic environments, our learning-based method has lower performance than the best baseline. This is due to the trade-off between flying safe and flying fast. As a result, our approach opts for safer, however slower, actions in cases where a time-optimal trajectory almost touches the obstacles. This effect is difficult to be modeled when using the optimization-based or the sampling-based method. The risk-awareness property of our neural network policy plays an essential role in achieving a high success rate in the presence of a model mismatch.

In more complex Office and Racing environments, our approach finds significantly faster trajectories than all baselines. This is because the sampling-based method uses a hierarchical approach where a point-mass trajectory guides the final trajectory for the quadrotor. It leads to rather bang-bang body rate behavior, which is favorable for the simple Forest scenarios. In contrast, for the Office scenarios, smooth body rates produced by RL are favorable. Finally, the inference time of the neural network policy is on average less than one millisecond, which allows fast online adaptation.

B. Success Rate of Minimum-time Flight in High Fidelity Simulation

To validate the success rate of navigating through given waypoints in a cluttered environment, we utilize a high-fidelity simulation which is based on Bade-Element-Momentum (BEM) theory [45]. Compared to the simple simulation, the BEM simulation can accurately model lift and drag produced by each rotor from the current ego-motion of the platform and the individual rotor speeds. Our proposed approach is compared with trajectories planned using the sampling-based method and tracked at $100\,\mathrm{Hz}$ using Model Predictive Control (MPC) [46] that outputs the same thrust and body rates commands as the RL policy. The output of the MPC is then

TABLE II: Comparison of baseline algorithms and our learning-based method.

Environment	Test case	Polynomial [18]			Search-based [30]		CPC [4]		Sampling-based [19]			**RL (ours)**	
		c. time[s]	T_a[s]	T_b[s]	c. time[s]	T_b[s]	c. time[s]	T_b[s]	c. time[s]	T_a[s]	T_b[s]	i. time[s]	T_b[s]
Forest	0	0.41	4.67±0.63	3.86	17.52	1.60	70.23	**0.95**	14.92	1.10±0.13	0.96	0.00042	0.98
	1	0.30	3.43±0.00	3.43	9.34	1.40	67.40	**0.96**	2.85	0.97±0.00	**0.96**	0.00042	1.00
	2	0.30	3.74±0.93	3.22	3.12	1.40	65.49	**0.95**	8.03	0.98±0.01	0.96	0.00042	1.00
	3	1.47	7.20±1.17	5.25	41.90	1.80	-	-	135.83	1.50±0.17	1.30	0.00052	**1.28**
Office	0	1.36	8.64±1.03	7.34	28.35	2.60	-	-	139.19	2.38±0.28	1.93	0.00040	**1.62**
	1	0.65	7.50±0.44	6.49	100.16	2.20	-	-	103.64	1.74±0.06	1.69	0.00043	**1.64**
	2	0.89	9.01±0.77	6.38	47.08	2.20	-	-	155.23	2.20±0.13	1.93	0.00045	**1.56**
	3	0.47	5.26±0.35	5.14	48.02	2.00	-	-	223.64	1.81±0.11	1.58	0.00040	**1.40**
Racing	0	1.98	6.56±0.66	5.79	-	-	-	-	365.91	1.61±0.29	1.34	0.00039	**1.20**
	1	2.06	6.21±0.88	5.13	-	-	-	-	428.02	1.63±0.15	1.36	0.00042	**1.20**
	2	1.87	6.26±0.42	4.94	-	-	-	-	138.17	1.45±0.12	**1.37**	0.00039	1.38
	3	1.91	5.27±0.69	4.73	-	-	-	-	604.55	2.14±0.74	1.57	0.00039	**1.34**
Racing MW	0	8.12	27.54±0.62	26.98	-	-	-	-	734.65	7.10±0.06	7.01	0.00037	**6.92**

TABLE III: Comparison of success rates of navigating through waypoints while avoiding obstacles.

Env.	Test case	SB [19] + MPC [4]			**RL (ours)**		
		success[%]	T_a[s]	T_b[s]	success[%]	T_a[s]	T_b[s]
Forest	0	25	**1.22**	**1.13**	**100**	1.23	1.21
	1	0	**-**	-	**100**	**1.21**	**1.18**
	2	27	**1.14**	**1.11**	**100**	1.20	1.18
	3	16	1.70	**1.42**	**100**	**1.57**	1.56
Office	0	41	2.38	2.12	**100**	**1.91**	**1.89**
	1	28	1.86	**1.78**	**100**	**1.84**	1.82
	2	56	2.29	1.97	**100**	**1.74**	**1.72**
	3	70	2.16	**1.70**	**100**	**1.99**	1.96
Racing	0	57	1.61	1.46	**100**	**1.41**	**1.39**
	1	51	1.64	1.45	**100**	**1.47**	**1.44**
	2	76	1.72	1.51	**100**	**1.51**	**1.49**
	3	54	1.80	1.62	**100**	**1.46**	**1.43**
Racing MW	0	25	**7.22**	**7.17**	**100**	**7.22**	7.18

tracked using the same low-level BetaFlight controller as used for tracking the policy actions. The sampling-based method is the only other method capable of computing collision-free trajectories for all test cases. We use MPC for the trajectory tracking, since it has been shown to successfully track truly time-optimal trajectories [4]. The MPC has been tuned to maximize position tracking performance to stick to the planned trajectory collision-free positions and thus avoid obstacles. The thrust limit of the considered platform is increased from the $f_{max} = 7\,\mathrm{N}$ used for planning and learning the policies, to $f_{max} = 8.5\,\mathrm{N}$ to have control margins for the MPC. Furthermore, the BEM simulation uses, in contrast to the simple simulation, both the motor dynamics and the aerodynamics.

Table III shows the comparison of the success rate, measured over 30 runs per test case. A successful run is defined if the quadrotor passes all waypoints within given tolerance r_{tol} and avoids all obstacles. The obstacle tolerance d_c is decreased by the ESDF precision to remove the influence of the discretized ESDF map representation. We also show the average T_a and best T_b duration of all tested flights for both methods.

Table III indicates that our learned policy can be transferred to a different simulator without fine-tuning. The policy successfully navigates the quadrotor through all environments without collisions while achieving faster flight trajectories. By contrast, the sampling-based method combined with MPC has significantly lower success rates. This is because the MPC struggles to handle disturbances during high-speed flight. When the platform is at its actuation limit, the slightest deviation from the pre-planned trajectory results in a suboptimal flight path, and eventually catastrophic crashes due to the presence of obstacles.

C. Real-world Validation

We validate our policy in the real world, where the quadrotor has to navigate through the Slalom environment. Figure 1 shows a successful deployment of the policy. The used platform is based on the open-hardware and open-source Agilicious quadrotor framework [46]. We use the BetaFlight controller to track the commanded collective thrust and body rates. We conducted our experiment in the world's largest indoor drone-testing arena ($30 \times 30 \times 8\ \mathrm{m}^3$) equipped with a motion capture system with $400\,\mathrm{Hz}$ operation frequency. The attached video shows the flight with thrust limits of $f_{max} = 4\,\mathrm{N}$ and $f_{max} = 7\,\mathrm{N}$ where in the second case the maximal velocity reached $42\,\mathrm{km\,h^{-1}}$. Our policy achieved a flight duration of $7.68\,\mathrm{s}$ in simulation, while the duration of the real flight was $7.90\,\mathrm{s}$, when using the thrust limit of $f_{max} = 7\,\mathrm{N}$.

VI. Conclusions

This paper introduced a novel method that combines deep reinforcement learning and classical topological path planning to train robust neural network controllers for minimum-time quadrotor flight in cluttered environments. We showed that the proposed method outperforms existing state-of-the-art approaches in the majority of the test cases, with improved trajectory quality of up to 19%. More importantly, we showed that the trained neural network controller can adapt online to counteract disturbances and model mismatches, and thus fly robustly. The presented method achieves 100% success rate of flying minimum-time policies without collision, while approaches relying on the traditional planning and control pipeline achieve only 40%. Our findings suggest that model-free deep RL is a promising method for addressing challenging tasks in agile flight, such as dynamic obstacle avoidance or vision-based minimum-time flight in cluttered environments.

ACKNOWLEDGMENT

The authors would like to thank Leonard Bauersfeld for helping with the multimedia material.

REFERENCES

[1] E. Ackermann, "Ai-powered drone learns extreme acrobatics," accessed 24.2.2022. [Online]. Available: https://spectrum.ieee.org/ai-powered-drone-extreme-acrobatics

[2] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE ICRA*, 2011, pp. 2520–2525.

[4] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, 2021.

[5] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[6] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE TRO*, vol. 36, no. 1, pp. 1–14, 2020.

[7] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *RSS*, July 2020.

[8] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[9] G. Ryou, E. Tal, and S. Karaman, "Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers," *The International Journal of Robotics Research*, vol. 40, no. 12-14, pp. 1352–1369, 2021.

[10] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for $\mathrm{SE}(3)$ planning in autonomous drone racing," *IEEE RA-L*, vol. 6, no. 4, pp. 8631–8638, 2021.

[11] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intell. Service Robotics*, vol. 12, no. 2, 2019.

[12] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *IROS*. IEEE, 2019.

[13] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," *PMLR post-proceedings of the NeurIPS 2019's Competition Track*, 2020.

[14] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Robotics: Science and Systems*, 2020.

[15] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *IROS*, 2021, pp. 1205–1212.

[16] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[17] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *ICRA*, 2013, pp. 5054–5061.

[18] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, 2016, pp. 649–666.

[19] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE RA-L*, 2022.

[20] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *European Control Conference (ECC)*, 2021, pp. 1556–1563.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[22] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE RA-L*, vol. 6, no. 3, pp. 4257–4264, 2021.

[23] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, 2020.

[24] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[25] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *IJRR*, vol. 31, no. 5, pp. 664–674, 2012.

[26] M. Burri, H. Oleynikova, , M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *IROS*, Sept 2015.

[27] 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.

[28] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *ICRA*, 2020, pp. 1208–1214.

[29] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE RA-L*, vol. 3, no. 4, pp. 3725–3732, 2018.

[30] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *IROS*, 2017, pp. 2872–2879.

[31] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," *IEEE RA-L*, vol. 3, no. 3, pp. 2439–2446, 2018.

[32] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for near-time-optimal quadrotor flight," *arXiv:2108.13205*, 2021.

[33] 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," *arXiv:2109.01365*, 2021.

[34] B. Lindqvist, S. S. Mansouri, A.-a. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear mpc for collision avoidance and control of uavs with dynamic obstacles," *IEEE RA-L*, vol. 5, no. 4, pp. 6001–6008, 2020.

[35] E. Small, P. Sopasakis, E. Fresk, P. Patrinos, and G. Nikolakopoulos, "Aerial navigation in obstructed environments with embedded nonlinear model predictive control," in *ECC*, 2019, pp. 3556–3563.

[36] G. Garimella, M. Sheckells, and M. Kobilarov, "Robust obstacle avoidance for aerial platforms using adaptive model predictive control," in *IEEE ICRA*, 2017, pp. 5876–5882.

[37] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *IEEE ICRA*, 2022.

[38] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE RA-L*, vol. 3, no. 2, pp. 620–626, 2017.

[39] S. M. LaValle, *Planning algorithms*. Cambridge univer. press, 2006.

[40] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[42] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019, pp. 165–174.

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

[44] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *JMLR*, vol. 22, no. 268, pp. 1–8, 2021.

[45] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *RSS*, 2021.

[46] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," 2021, under review.