Autonomous Drone Racing: A Survey

Drew Hanover, Antonio Loquercio, Leonard Bauersfeld, *Graduate Student Member, IEEE*, Angel Romero, *Graduate Student Member, IEEE*, Robert Penicka, Yunlong Song, Giovanni Cioffi, *Student Member, IEEE*, Elia Kaufmann, *Member, IEEE*, and Davide Scaramuzza, *Senior Member, IEEE*

(Survey Paper)

***Abstract*—Over the last decade, the use of autonomous drone systems for surveying, search and rescue, or last-mile delivery has increased exponentially. With the rise of these applications comes the need for highly robust, safety-critical algorithms that can operate drones in complex and uncertain environments. In addition, flying fast enables drones to cover more ground, increasing productivity and further strengthening their use case. One proxy for developing algorithms used in high-speed navigation is the task of autonomous drone racing (ADR), where researchers program drones to fly through a sequence of gates and avoid obstacles as quickly as possible using onboard sensors and limited computational power. Speeds and accelerations exceed over 80 km/h and 4 g, respectively, raising significant challenges across perception, planning, control, and state estimation. To achieve maximum performance, systems require real-time algorithms that are robust to motion blur, high dynamic range, model uncertainties, aerodynamic disturbances, and often unpredictable opponents. This survey covers the progression of ADR across model-based and learning-based approaches. In this article, we provide an overview of the field, its evolution over the years, and conclude with the biggest challenges and open questions to be faced in the future.**

***Index Terms*—Autonomous robots, autonomous aerial vehicles, drones.**

I. Introduction

THROUGHOUT history, humans have been obsessed with racing competitions, where physical and mental fitness are put to the test. The earliest mention of a formal race dates all the way back to 3000 BC in ancient Egypt, where the Pharaoh was thought to have run a race at the Sed festival to demonstrate his physical fitness, indicating his ability to rule over the kingdom [1], [2]. As time has progressed, humans have moved from racing on-foot to using chariots, cars, planes, and more recently, quadcopters [3]. Although the vessel frequently changes, one thing that has remained constant since the early days of racing has been the recurring theme of using the task as a catalyst for scientific and engineering development. Recently, we have seen a push to remove humans from the loop, automating the highly complex task of racing in order to push vehicle performance beyond what humans can achieve [4], [5].

Manuscript received 1 February 2024; accepted 3 March 2024. Date of publication 14 May 2024; date of current version 6 June 2024. This paper was recommended for publication by Associate Editor Giuseppe Loianno and Editor Paolo Robuffo Giordano upon evaluation of the reviewers' comments. This work was supported in part by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics, the Czech Science Foundation (GACR) under research project 23-06162M, in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 871479 (AERIAL-CORE), and in part by the European Research Council (ERC) under Grant 864042 (AGILEFLIGHT). *(Corresponding author: Leonard Bauersfeld.)*

Drew Hanover, Leonard Bauersfeld, Angel Romero, Yunlong Song, Giovanni Cioffi, Elia Kaufmann, and Davide Scaramuzza are with the Robotics and Perception Group, University of Zürich, 8050 Zürich, Switzerland (e-mail: bauersfeld@ifi.uzh.ch).

Antonio Loquercio is with the University of California, Berkeley (UC Berkeley), Berkeley, CA 4720 USA.

Robert Penicka is with the Multi-Robot Systems Group, Czech Technical University in Prague, 160 00 Prague, Czech Republic.

Digital Object Identifier 10.1109/TRO.2024.3400838

A. Why Autonomous Drone Racing (ADR)?

Drone racing is a popular sport with high-profile international competitions. In a traditional drone race, each vehicle is controlled by a human pilot, who receives a first-person-view (FPV) live stream from an onboard camera and flies the drone via a radio transmitter. An onboard image from the drone can be seen in Fig. 1(b). Having access to an FPV feed sets drone racing apart from remote-controlled fixed-wing aircraft racing, where pilots typically control the vehicle in a line-of-sight fashion. Human drone pilots need years of training to master the advanced navigation and control skills required to succeed in international competitions. Such skills would also be valuable to autonomous systems that must fly through complex environments in applications such as disaster response, aerial delivery, and inspection of complex structures. For example, automating inspection tasks can save lives while being more productive than manual inspection. According to a recent survey on unmanned aerial vehicle (UAV) use in bridge inspection [6], most drones used for inspection tasks rely on GPS navigation with the biggest limiting factor on inspection efficiency being the drones' endurance and mobility. In addition, the most popular drones used for surveying by several US Departments of Transportation are not fully autonomous and require expert human pilots [6]. In these applications, an increase in autonomy and operational speed will offer gains in utility as faster flight increases the operating radius achievable with a given battery [7]. Drone racing research has made significant progress in bringing the skills of autonomous drones closer to those of professional human pilots [5]. This required advances on all parts of the flight stack, i.e., estimation, planning, control, and hardware [8], which we cover in length in this survey. However, several challenges remain to bridge the

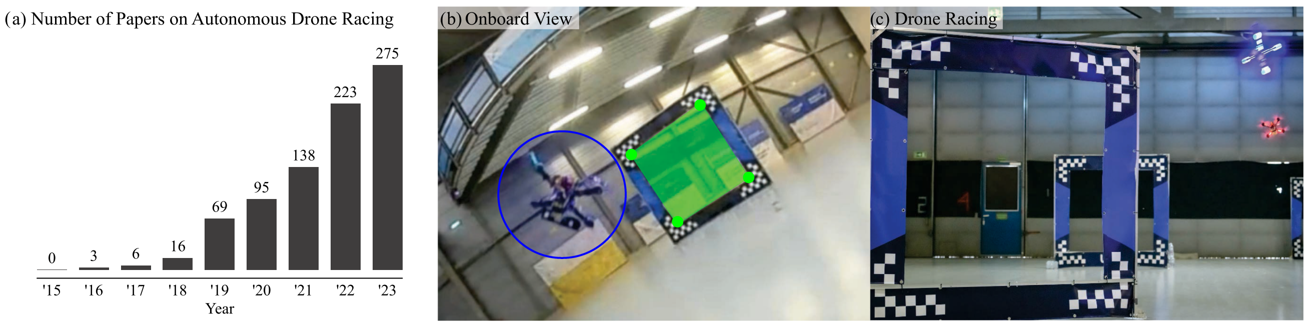
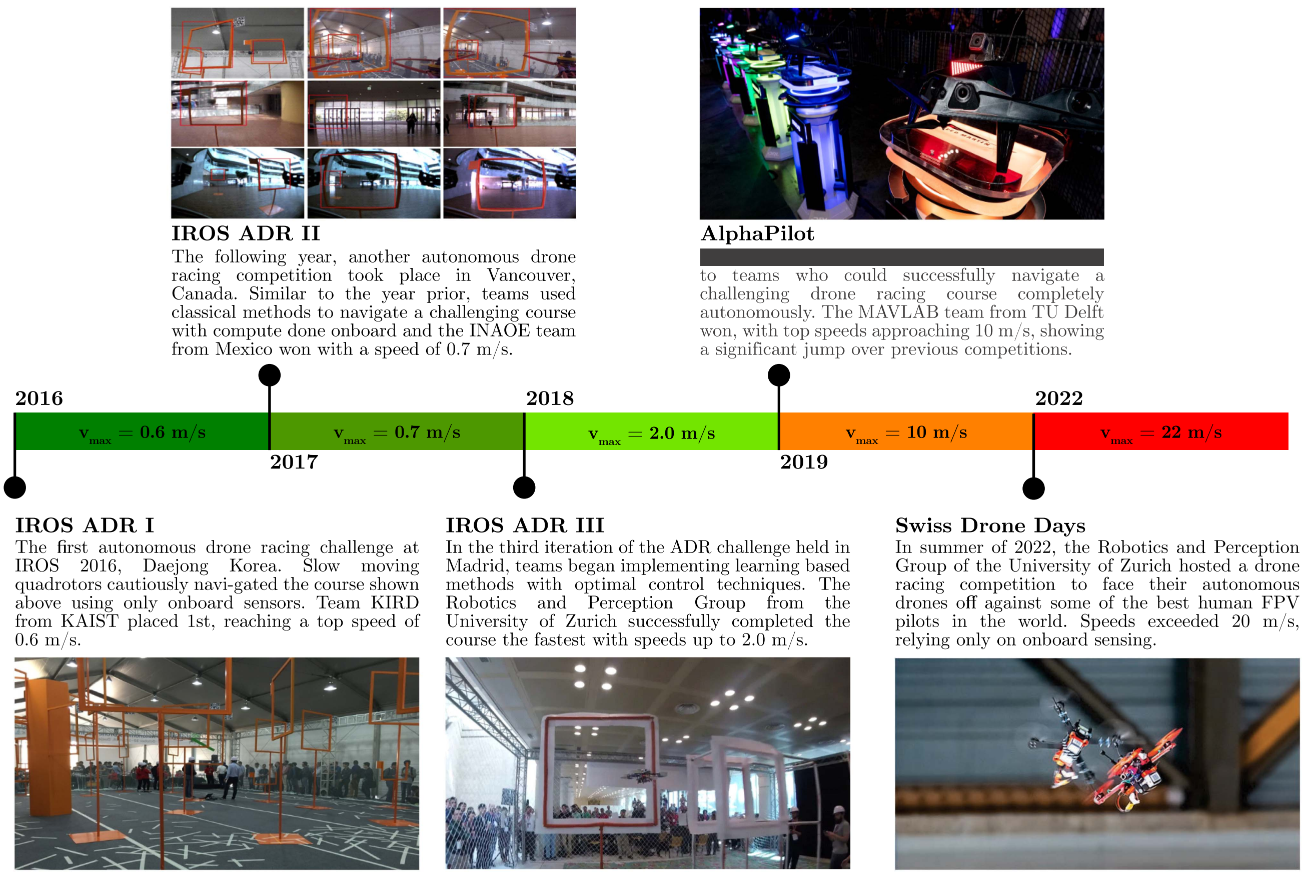


Fig. 1. Drone racing is a sport rapidly gaining popularity where opponents compete on a predefined race track consisting of a series of gates. ADR research aims to build algorithms that can outperform human pilots in such competitions. (a) Task of ADR has gained a substantial amount of interest from the research community in the last few years, as indicated by the increasing number of related publications per year, as evidenced by a Google Scholar search for "ADR." (b) Autonomous drones rely on visual and inertial sensors to estimate their own states, as well as their opponents' states. (c) Agile maneuvers are required to overtake opponents and win the race.



IROS ADR II
The following year, another autonomous drone racing competition took place in Vancouver, Canada. Similar to the year prior, teams used classical methods to navigate a challenging course with compute done onboard and the INAOE team from Mexico won with a speed of 0.7 m/s.

AlphaPilot
[redacted]
to teams who could successfully navigate a challenging drone racing course completely autonomously. The MAVLAB team from TU Delft won, with top speeds approaching 10 m/s, showing a significant jump over previous competitions.

IROS ADR I
The first autonomous drone racing challenge at IROS 2016, Daejong Korea. Slow moving quadrotors cautiously navi-gated the course shown above using only onboard sensors. Team KIRD from KAIST placed 1st, reaching a top speed of 0.6 m/s.

IROS ADR III
In the third iteration of the ADR challenge held in Madrid, teams began implementing learning based methods with optimal control techniques. The Robotics and Perception Group from the University of Zurich successfully completed the course the fastest with speeds up to 2.0 m/s.

Swiss Drone Days
In summer of 2022, the Robotics and Perception Group of the University of Zurich hosted a drone racing competition to face their autonomous drones off against some of the best human FPV pilots in the world. Speeds exceeded 20 m/s, relying only on onboard sensing.

Fig. 2. History of drone racing competitions that use real drones for navigating the race track, IROS ADR II photo credit [10].

gap between drone racing and real-world applications, such as safety [9] and generalization over tasks and environments.

Over the last five years, several projects have been launched to encourage rapid progress within the field, such as DARPA's Fast Lightweight Autonomy [11], European Research Council's AgileFlight [12], and the AutoAssess project [13]. With million-dollar funding for each of these projects and significant commercial potential, a large incentive exists for researchers and entrepreneurs alike to achieve autonomous operation in GNSS-denied and confined critical infrastructure. Competitions such as the International Conference on Intelligent Robots and Systems (IROS 2016–2019) ADR series [14], NeurIPS 2019's Game of Drones [15], and the 2019 AlphaPilot Challenge [16], [17] provided further opportunity for researchers to compare their methodologies against one another in a competitive fashion. A depiction of the progress made from these competitions can be seen in Fig. 2. However, we are far from having solved ADR—A notion reflected by the recently announced competition scheduled for 2025 and to be hosted by the Abu Dhabi Autonomous Racing League [18].

Drone racing is a challenging benchmark that can help researchers to gauge progress on complex perception, planning, and control algorithms. Autonomous drones in a racing setting must be able to perceive, reason, plan, and act on the tens of milliseconds scale, all onboard a computationally limited platform. Apart from being very challenging to solve, the drone racing task offers a single measure of the progress of the state-of-the-art in autonomous flying robotics: lap time. Solving this problem requires efficient, lightweight algorithms to provide optimal decision and control behaviors in real time. Just a few years back, it was unclear whether such a problem was feasible in the first place, even given perfect information. Drone racing research has advanced much since its early stages [19]. Such advances required radically new algorithmic ideas, e.g., training learning-based sensorimotor controllers only in simulation, together with engineering advances to the platform and the overall system [8]. Now that superhuman performance has been achieved [5] (despite being in controlled conditions), we predict that more work will be done on safety and generalization over tasks and environments to bridge the gap between drone racing and real-world applications. This research effort is already evident today, as shown by the increasing number of articles in the field over the years [see Fig. 1(a)].

This is the first survey on the state of the art in ADR. This overview will be useful to researchers who wish to make connections between existing works, learn about the strengths and weaknesses of current and past approaches, and identify directions moving forward which should progress the field in a meaningful way.

B. Task Specification

The drone racing task is to fly a quadrotor through a sequence of gates in a given order in minimum time while avoiding collisions. Humans are astonishingly good at this task, flying at speeds well over 100 km/h with only an FPV camera as their sensory input. Beyond this, expert pilots can adapt to new race tracks quickly in a matter of minutes, however, the sensorimotor skills required by professional drone pilots take years of training to acquire.

For an autonomous drone to successfully complete this task, it must be able to detect opponents and waypoints along the track, calculate their location and orientation in 3-D space, and compute an action that enables navigation through the track as quickly as possible while still controlling a highly nonlinear system at the limits. This is challenging in three different aspects: Perception, Planning, and Control. Poor design in any of these aspects can make the difference between winning and losing the race, which can be decided by less than a tenth of a second.

The rest of this article is organized as follows. First, the modeling procedure of the drone, including aerodynamics, batteries, motors, cameras, and the system nonlinearities, are discussed in detail in Section II. A classical robotics pipeline is then introduced in Section III, with a deep dive into literature relevant to agile flight split into perception, planning, and control subsections. All of these components are equally important because, at the edge of a drone's performance envelope, all parts—perception, state estimation, planning, and control—need to meticulously work together. Next, we delve into learning-based methods for perception, planning, and control, which rely on recent advancements from the machine learning community in Section IV. Then, a discussion of the development of simulation tools that can enable rapid development for agile flight applications in Section V. A history of drone racing competitions and the methods used for each are included in Section VI. Next, a summary of open source code bases, hardware platforms, and datasets for researchers is provided in Section VII. A forward-looking discussion on the opportunities and challenges for future researchers interested in ADR is presented in Section VIII. Finally, Section IX concludes this article.

II. Drone Modeling

To further advance research on fast and agile flight, it is important to have accurate models that capture the complex nonlinear dynamics of multicopter vehicles at the limit of their performance envelope.

This section reviews different dynamics modeling approaches from classic, first-principles modeling to pure data-driven models in the context of drone racing. For the vehicle dynamics, the key aspects that need to be modeled are the kinematics, aerodynamics, the electric motors, and the battery. In addition to the vehicle dynamics models discussed in this section, many difficulties for ADR models are introduced by the onboard sensors, whose characteristics need to be modeled. For example, inertial measurement units (IMUs) are subject to bias and noise, and the intrinsic as well as extrinsic parameters of onboard sensors change over time as hard crashes may lead to miscalibration.

A. Kinematics

Typically, the vehicle is assumed to be a 6-degree-of-freedom (DoF) rigid body of mass m with a (diagonal) inertia matrix $\boldsymbol{J} = \mathrm{diag}(J_x, J_y, J_z)$. Given a dynamic state $\boldsymbol{x} \in \mathbb{R}^{17}$, the equations describing its evolution in time are commonly written as

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) = \begin{bmatrix} \dot{\boldsymbol{p}}_{WB} \\ \dot{\boldsymbol{q}}_{WB} \\ \dot{\boldsymbol{v}}_W \\ \dot{\boldsymbol{\omega}}_B \\ \dot{\boldsymbol{\Omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_W \\ \boldsymbol{q}_{WB} \begin{bmatrix} 0 \\ \boldsymbol{\omega}_B/2 \end{bmatrix} \\ \frac{1}{m}\left(\boldsymbol{q}_{WB} \odot \boldsymbol{f}\right) + \boldsymbol{g}_W \\ \boldsymbol{J}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{\omega}_B \times \boldsymbol{J}\boldsymbol{\omega}_B\right) \\ \frac{1}{\tau_\Omega}(\boldsymbol{\Omega}_{\mathrm{ss}} - \boldsymbol{\Omega}) \end{bmatrix} \tag{1}$$

where $\boldsymbol{p}_{WB} \in \mathbb{R}^3$ is the position of the center of mass given in the world frame, $\boldsymbol{q}_{WB} \in \mathrm{SO}(3)$ is a quaternion defining the rotation of the body frame relative to the world (vehicle attitude), $\boldsymbol{v}_W \in \mathbb{R}^3$ is the velocity of the vehicle in the world frame, $\boldsymbol{\omega}_B \in \mathbb{R}^3$ is the body rate of the vehicle, $\boldsymbol{\Omega} \in \mathbb{R}^4$ is the motor speed, $\boldsymbol{g}_W = [0, 0, -9.81\,\mathrm{m/s^2}]^\intercal$ denotes Earth's gravity, and $\boldsymbol{u} \in \mathbb{R}^4$ is the input. Depending on the control modality, the input can be single rotor thrusts or collective thrust and body rates. In this setting, the task of the model is to calculate the total force $\boldsymbol{f}$ and total torque $\boldsymbol{\tau}$ that acts on the drone as accurately as possible. Note the quaternion-vector product denoted by $\odot$ representing a rotation of the vector by the quaternion as in $\boldsymbol{q} \odot \boldsymbol{f} = \boldsymbol{q} \cdot [0, \boldsymbol{f}^\intercal]^\intercal \cdot \bar{\boldsymbol{q}}$, where $\bar{\boldsymbol{q}}$ is the quaternion's conjugate. Those forces and torques, collectively referred to as wrench, are determined by the aerodynamics of the platform as well as the vehicles' actuators, e.g., the propellers.

B. Aerodynamics

This section discusses the different approaches to modeling the aerodynamics of the drone and its propellers. The most widely used modeling assumption is that the propeller thrust and drag torque are proportional to the square of the rotational speed [20], [21], [22], [23], [24] and that the body drag is negligible. These assumptions quickly break down at high speeds encountered in drone racing as this model neglects the following:

1) linear rotor drag [25], [26];
2) dynamic lift [25];
3) rotor-to-rotor [27], [28], [29];
4) rotor-to-body [27], [28], [29] interactions;
5) aerodynamic body drag [26], [28].

The accuracy of the propeller model can be improved by leveraging the blade-element-momentum (BEM) theory, where the propeller is modeled as a rotating wing. Such first-principle approaches [30], [31], [32], [33], [34] have been shown to provide very accurate models of the wrench generated by a single propeller as they properly capture effects 1) and 2). Implemented efficiently, a BEM model can be run in real time [35] and has been successfully used to test algorithms in simulation [36], [37].

Accounting for the remaining open points 3)–5), the aerodynamics of the drone body as well as any interaction effects need to be calculated, which requires a full computational fluid dynamics (CFD) simulation [27], [28], [29], [38], [39]. Due to the extreme computational demands, this is impractical in drone racing. To still get close to the accuracy of CFD methods while retaining the computational simplicity of the previously mentioned methods, data-driven approaches are employed [35], [40], [41], [42], [43], [44]. In the early works [42], [43], the whole vehicle dynamics model was learned from data. In a similar fashion, Sun et al. [41] use a combination of polynomials—identified from wind-tunnel flight data—to represent the vehicle dynamics. In [35] and [40], it has been shown that higher modeling accuracies can be achieved when combining a first-principle model with a data-driven component. Such a combination of first-principle and data-driven models also leads to improved generalization performance, as shown in [35], which combines a BEM model with a temporal convolutional network [45] to regress the residual wrench. Recently, a similar hybrid modeling approach has been applied to moving-horizon estimation [46].

C. Motor and Battery Models

The previous section outlines different approaches to how the aerodynamic wrench can be estimated based on the state of the vehicle. However, for all such models, the rotational speed of the propeller is assumed to be known. On most multicopters, the motors are not equipped with closed-loop motor speed control but are controlled by a "throttle" command, which controls the duty cycle of a pulsewidth-modulation signal applied to the motors. The actual rotational speed that the motor achieves is a function of the throttle command as well as other parameters such as the battery voltage and the drag torque of the rotor [7]. Therefore, in order to have a dynamics model for the motors, we need a model of the battery to calculate the voltage applied to the motors. Most literature on battery modeling relies on so-called Peukert models [47], but for lithium-polymer batteries in drone racing, this is hardly applicable because the battery discharge current often exceeds 100 A (e.g., 50–100C) [48], [49]. Graybox battery models for the voltage that are based on a one-time-constant (OTC) equivalent circuit [50], [51] are much more suitable for drone racing tasks as shown in [7], because they are applicable to the extremely high loads experienced during a racing scenario. In combination with either a polynomial or a constant-efficiency motor model, such OTC models can be used to accurately simulate the battery voltage during agile flight [7]. Given a simulation of the battery voltage, one can measure the performance characteristics of a given motor-propeller combination to determine the mapping of throttle command and voltage to resulting steady-state propeller speed Ω_{ss}. When the highest model fidelity is desired, a more sophisticated motor simulation [52] can further improve the accuracy, which can be desirable if the controller directly outputs single-rotor thrusts instead of the more commonly used collective-thrust and body rates control modality.

D. Camera and IMU Modeling

Drone racing pushes not only the mechanical and electrical components of drones to their limits, but is also highly demanding in terms of sensor performance. For an in-depth overview of the many different sensor options for drone racing, the reader is referred to [53]. The most common sensors aboard autonomous drones are monocular or stereo cameras combined with IMUs thanks to their low cost, low weight, and mechanical robustness.

For vision-based drone racing, having an accurate simulation of the perception pipeline is critical for validation and controller development. In terms of modeling and simulation of the camera, it is common to use a pinhole model [54] and estimate the focal length, image center, and distortion parameters from measurements. Combined with accurate information on how far the camera is displaced from the center of gravity of the vehicle, this allows simulating observations. Either low-level sensory observations (e.g., images) are simulated using a rendering engine [22], [55] or more abstract visual features (e.g., landmark positions) are simulated using the projection equations.

In the context of using a simulation to test approaches before attempting real-world deployment, an accurate model of the IMU characteristics is important, as the bias and noise strongly influence the performance of many methods. The IMU intrinsic calibration estimates the noise characteristic of the sensor. The camera-IMU extrinsic calibration estimates the relative position and orientation of the two sensors as well as the time offset. Kalibr [56] is a widespread tool to perform these calibrations.

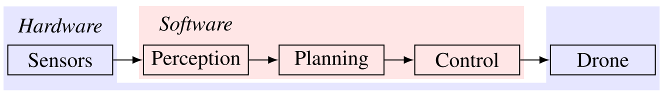


Fig. 3. Architecture 1: A classic architecture for an autonomous system programmed using model-based approaches.

However, the biggest source of measurement error of the inertial sensors onboard a drone is not the sensors themselves but the strong high-frequency vibrations introduced by the fast-spinning propellers. The vibrations lead to aliasing effects on the IMU measurements and introduce additional motion blur on the camera images. The structural vibrations and their effect on the measurements are extremely difficult to model and correct for. Therefore, a suitable hardware design is imperative that dampens the mount of the camera and the IMU with respect to the vehicle frame.

III. Classical Perception, Planning, and Control Pipeline

Since the inception of mobile robotics, a common architecture has been primarily used to achieve autonomous navigation capabilities across various systems. In a traditional robotics software stack, the navigation task is broken into three main components: perception, planning, and control. A diagram of this architecture can be seen in Fig. 3. This section covers recent research in these areas relating specifically to agile flight and ADR. All approaches detailed in this section rely on first principles modeling and optimization techniques.

A. Perception

The perception block estimates the vehicle state and perceives the environment using onboard sensors. The most common solution for state estimation of flying vehicles is visual-inertial odometry (VIO), thanks to its low cost and low weight requirements. VIO uses camera and IMU measurements to estimate the state $\hat{x}$ (position, orientation, and velocity) of the drone platform. The inertial measurements are integrated to obtain relative position, orientation, and velocity estimates in a short time, e.g., between two camera images. However, the integration for a longer time, e.g., a few seconds, accumulates large drift due to scale factor errors, axis misalignment errors, and time-varying biases [57] that commonly affect off-the-self IMU measurements. The camera measurements provide rich information about the environment at a lower rate, usually around 30 Hz, than IMU measurements. Unlike the IMU measurements, the camera measurements are affected by environmental conditions. The quality of information they provide for state estimation degrades in the case of poor illumination conditions, textureless scenes, and motion blur. For this reason, the camera and inertial measurements complement each other and are the standard choice for state estimation of flying vehicles [58]. In this section, we first give an overview of VIO with a focus on the methods that can be applied for online state estimation of a racing drone. Second, we give an overview of possible additional sensor modalities that integrated into the classical VIO pipeline have the potential to improve state estimation at high speed. Finally, we conclude with a discussion on the application of classical VIO methods to drone racing tasks.

1) Visual-Inertial Odometry (VIO): VIO is the most common solution for state estimation of aerial vehicles [58] using only onboard sensing and computing, thanks to its favorable tradeoff between accuracy and computational requirements. VIO algorithms usually comprise two main blocks: the frontend and the backend.

The frontend uses camera images to estimate the motion of the sensor. Two main approaches exist in the literature: direct methods [59], [60] and feature-based methods [61], [62], [63]. Direct methods work directly on the raw pixel intensities. These methods commonly extract image patches and estimate the camera trajectory by tracking the motion of such patches through consecutive images. The tracking is achieved by minimizing a photometric error defined on the raw pixel intensities [59]. This tracking method is particularly interesting for drone racing because of its robustness in featureless scenarios. In fact, a direct frontend [60] is used to estimate the state of a racing drone in [16]. On the contrary, feature-based methods [61], [62], [63] extract points of interest, commonly known as visual features or keypoints, from the raw image pixels. The camera trajectory is estimated by tracking these points through consecutive images. High-speed motions make it difficult (e.g., due to motion blur) to track features on many consecutive; consequently, feature-based methods struggle in drone racing scenarios. However, feature-based methods exhibit higher robustness than direct methods to brightness changes. The VIO methods used in [64] and [65] demonstrate that a hybrid frontend, combining the benefits of direct and feature-based methods, is beneficial for drone racing tasks.

The backend fuses the output of the fronted with the inertial measurements. Two categories exist in the literature according to how the sensor fusion problem is solved: filtering methods [61] and fixed-lag smoothing methods [62], [63]. Filtering methods are based on an extended Kalman filter (EKF). These methods propagate the system's state using the inertial measurements and fuse the camera measurements in the update step. The pioneer filter-based VIO algorithm is the multistate constraint Kalman filter (MSCKF) originally proposed in [61]. Since then, many different versions of the MSCKF have been developed [66]. Fixed-lag smoothing methods, also called sliding window estimators, solve a nonlinear optimization problem where the variables to be optimized are a window of the recent robot states. The cost function to minimize contains visual, inertial, and past states marginalized residuals. Thanks to their favorable tradeoff between accuracy and computational cost, filter-based methods have been commonly used in drone racing [16], [64], [65].

2) Additional Sensor Modalities in VIO: Recently, classical VIO pipelines have been augmented with event cameras [67], [68], [69] or drone dynamics [70], [71], [72], to improve state estimation at high speed.

Low latency, high temporal resolution (in the order of microsecond) and high dynamic range (140 dB compared to 60 dB of standard cameras) are the main properties of event cameras [73], which make this novel sensor a great complementary sensor to standard cameras. Including event data in VIO algorithms onboard flying vehicles achieves increased robustness against motion blur as demonstrated in [67], [68], and [69]. Although applications of event cameras in drone racing tasks are yet to be explored, investigating the use of this sensor is a promising research direction to robustify VIO systems for agile flights.

The drone dynamics are used to define additional constraints in the estimation process. VIMO [70] is the first to integrate error terms related to the drone transitional dynamics in a VIO backend. VIMO inspired a few works [71], [74] that propose an improved noise model of the dynamics [74] and a learned component to account for unmodeled aerodynamics [71]. In particular, the results of [71] show that the learned aerodynamics effects help to improve the VIO estimates at high speeds.

Cioffi et al. [72] propose an odometry algorithm that relies on an IMU as the only sensor modality (no camera is needed), and leverages a learned dynamics component to estimate the state of the racing drone. Consequently, this method does not use a visual frontend.

3) Discussions: Delmerico and Scaramuzza [75] present a benchmark comparison between a number of VIO solutions on the EuRoC dataset [76]. The EuRoC dataset contains camera and IMU data recorded onboard a drone flying in indoor environments. The drone moves with average linear and angular velocities up to 0.9 m/s and 0.75 rad/s, respectively. These values are far below the ones reached in drone racing. The conclusions of [75] show that state-of-the-art VIO algorithms provide reliable solutions for estimating the state of the drone at limited speeds. However, these classical VIO methods cannot provide accurate state estimates for drone racing tasks. VIO methods accumulate large drift in scenarios characterized by motion blur, low texture, and high dynamic range [77]. These scenarios are the norm in drone racing.

To help research VIO algorithms for drone racing tasks, Delmerico et al. [78] propose the UZH-FPV Drone Racing dataset. This dataset contains images recorded from standard cameras, event camera data, and IMU data recorded onboard a quadrotor flown by a human pilot. All the flights include visual challenges similar to those in drone racing competitions.

Successful state estimation solutions for drone racing [77], [79] reduce the drift accumulated in VIO by localizing to a prior map of the track. In drone racing competitions, a map of the track in the form of gate positions is known beforehand. The localization process is based on the detection of the gates. Performing gate detection is challenging. Often during the race, none of the gates is visible in the camera's field of view. Moreover, motion blur makes gate corner detection difficult. For this reason, gate detection and VIO are complementary. In [80], a gate detector was proposed that uses an RGB camera to identify the gates based on their color. This detector, tailored to the IROS drone racing context [19], is aimed at extreme computational efficiency, which is particularly important for tiny drones. The method in [81] relies on detecting gates and using a model of the drone dynamics to estimate the position of the racing drone. Differently from [77] and [79], this method does not use a VIO but controls the drone based on a visual-servoing approach. All the other gate detection methods in the literature are based on deep learning techniques [82]. We review them in Section IV. The known gate positions and the detections in the onboard images are used to estimate the relative pose between the camera and the gate using the perspective-n-point algorithm (PnP) [83]. This relative pose is used to constrain the VIO estimates, and consequently, reduce the drift. There is significant room for innovation on this front, as the VIO-PnP paradigm has existed for several years with little innovation. However, one clear benefit of the VIO-PnP approach is its ability to use a monocular camera setup with a large FOV. While this comes with a lack of scale and higher uncertainty in motion estimation, both can be compensated using inertial sensors and localization with respect to known landmarks (e.g., gates). As evidenced by the rich literature, this makes a monocular setup the preferred solution for ADR practitioners. The choice of a monocular sensor is very much in agreement with how human pilots fly: while they have goggles with two monitors, the video stream they receive is from a monocular camera system on the drone. Other approaches used in early drone racing competitions relied on the technique of visual servoing via stereo cameras [10], but relying on a stereo camera pair comes with inherent difficulties. In the presence of motion blur, stereo-matching approaches degrade quickly. Furthermore, drones only allow for a very small baseline and require a wide-angle camera to perceive as much of the surroundings as possible. Both lead to very high depth estimation errors in the stereo setup. The solution proposed in [10] was found to be sensitive to indoor lighting changes and needed to be hand-tuned for every flight.

Recent works [84], [85], [86] proposed vision-based odometry algorithms that are learned end-to-end. Theoretically, these methods could be specialized to drone racing tasks and potentially outperform classical VIO approaches. However, they are in the early development phase, and how to customize them for the drone racing task is still an open research question. In addition, they currently have high computational costs that make them impractical for online state estimation onboard drones. We refer the reader to Setion IV for a detailed review of VIO methods based on deep learning.

B. Planning

Once a state estimate $\hat{\boldsymbol{x}}$ has been obtained from the perception module, the next step in the classical pipeline is to plan a feasible, time-optimal trajectory $\tau_{\text{ref}} = (\boldsymbol{x}_{\text{ref}}, \boldsymbol{u}_{\text{ref}})_k \ \ \forall k \in 0 \ldots N$, which respects the physical limits of the platform as well as the constraints imposed by the environment. This requires predicting the drone's future states such that minimum lap time is reached without crashing.

The planning for drones has matured over the last decade from works mostly verified in simulation to works shown in both controlled lab environments and unknown unstructured environments. In the classical pipeline, planning can include up to two distinct planning problems, path planning and trajectory planning. Path planning tackles the problem of finding a geometrical path between a given start and goal position while passing specified waypoints and avoiding obstacles. Trajectory planning then uses a found geometric path to either create a collision-free flight corridor [87], [88], to find new waypoints for the trajectory to avoid collisions [37], [89], to constrain the trajectory to stay close to the found path [90], [91], or directly finds time allocation for a given path [92], [93]. Therefore, path planning can be seen as a way to select the homotopy class of the collision-free space the drone flies through, while trajectory planning finds the full (or simplified) time-allocated drone state predictions to be followed by the control pipeline (see Section III-C). However, many works rely solely on trajectory planning as they assume no collision with the environment when a trajectory is found [94], [95], [96], [97], [98]. Other works directly find a collision-free trajectory [99], [100], [101], [102] without having a previously found path. On the other hand, some control approaches [103], [104] do not need a specified time-allocated trajectory and rely only on the geometrical path for controlling the drone.

In the following text, we first overview the most popular *path planning* approaches for drones that are used for further trajectory planning. Then, we categorize trajectory planning methods in *polynomial and spline trajectory* planning, *optimization-based* trajectory planning, *search-based* trajectory planning, and *sampling-based* trajectory planning.

1) Path Planning: Path planning approaches can be broadly divided into sampling-based planning and combinatorial planning [105]. Sampling-based methods do not construct the obstacle space explicitly but rather rely on random sampling of the configuration space together with collision detection. The most popular variants of the sampling-based methods with numerous modified versions are the probabilistic roadmaps (PRM) [106] and rapidly exploring random trees (RRT) [107]. Important variants of these algorithms named RRT* and PRM* [108], can find the optimal path given infinite time. The combinatorial planning methods, in contrast to the sampling-based methods, directly represent the obstacle or free space using e.g. polygonal maps or cell decomposition such as grid-based maps. With the help of a graph representation of the decomposed free space, classical path search algorithms such as A* [109] or Dijkstra's algorithm [110] can be used to find a path.

Variants of the aforementioned path planning approaches are used in many of the methods listed in Section III-B2–III-B5 to help find trajectories for either fast flight or even drone racing. The RRT* algorithm is used to find new waypoints for polynomial trajectory planning in [89], and the PRM* is used as a path planning part to guide sampling-based trajectory planning in [37]. The sampling-based planning in [101] and [102] directly performs both path and trajectory planning. While the trajectory planning objective can be to minimize time duration of a trajectory, the path planning typically tries to find the shortest paths. Therefore, some methods search for multiple distinct paths to enable the trajectory planning to search over multiple options on how to navigate around obstacles [37], [90], [91]. Other methods [87] use search-based algorithms to find an initial path and to create a convex flight corridor for constraining the collision-free trajectory planning. Similarly, the search-based methods [99], [100] use a variant of A* to perform both path and trajectory planning at the same time.

2) Polynomial and Spline Trajectory Planning: The polynomial and spline methods leverage the differential flatness property [111], [112] of quadrotors and represent a trajectory as a continuous-time polynomial or spline. This property simplifies the full-state trajectory planning to a variant where only four flat outputs need to be planned (typically 3-D position and heading). By taking their high-order derivatives, these flat outputs can represent a dynamically feasible trajectory with their respective control inputs. This property is used by many polynomial and spline methods that are nowadays among the most used for general quadrotor flight.

The widely used polynomial trajectories [111], [112] minimize snap (fourth-order position derivative) of a trajectory. Different methods opted for minimizing jerk (third-order position derivative) for planning a trajectory [113]. However, the trajectories that result from having jerk as the primary objective have been shown to minimize the aggressiveness of the control inputs [113], which is fundamentally different from minimizing the lap times, where extremely aggressive trajectories are generally required. Richter et al. [89], therefore, extended the objective by jointly optimizing both the snap of a trajectory and the total time through a user-specified penalty on time. Recently, Han et al. [87] proposed a polynomial-based trajectory planning method for drone racing. It jointly optimizes control effort and regularized time and penalizes the dynamic feasibility and collisions.

Because of their numerical stability, other methods use B-splines to represent trajectories [90], [91] instead of high-order polynomial representations that are numerically sensitive. These methods jointly optimize different objectives, simultaneously smoothness, dynamic feasibility, collision avoidance, safety [91], and vision-based target tracking [94]. Recently, Qin et al. [114] proposed a polynomial trajectory representation based on [115] and use it to plan time-optimal trajectories through gates of arbitrary shapes for drone racing, achieving close-to-time-optimal results while being more computationally efficient than [95].

Although both polynomial and spline trajectories are widely used due to their computational efficiency, polynomial-based trajectories (and their derivatives) are smooth by definition. Therefore, only smooth control inputs can be sampled from them. For this reason, the traditional polynomial planning [111] with a finite number of coefficients and one polynomial segment between every two waypoints (gates) cannot represent true time-optimal trajectories [95]. Yet, direct collocation methods [116] that rely on polynomials to approximate the input and state dynamics can achieve nearly optimal performance. This is mainly due to a larger number of polynomial segments between the waypoints in collocation methods, joint optimization of both polynomial coefficients and collocation points, and due to the approximation of the entire dynamics by polynomials. This allows to keep the acceleration at the possible maximum at all times similar to the optimization-based shooting method [95].

Therefore, while the classical polynomial and spline methods can be considered optimization-based, they only optimize coefficients of a single polynomial between every two waypoints to describe quadrotor position and heading, leveraging the differential flatness property [111], [112].

3) Optimization-Based Trajectory Planning: Optimization-based trajectory planning enables us to independently select the optimal sequence of states and inputs at every time step, which inherently considers time minimization while complying with quadrotor dynamics and input constraints. Optimization-based approaches have been extensively considered in the literature, ranging from exploiting point-mass models [96], simplified quadrotor models [97], [117], and full-state quadrotor models [88], [95].

Time-optimality of a trajectory could also be accomplished by using a specific path parameterization that maximizes velocity over a given path [92]. This method was shown for quadrotors in [93] for minimizing time of flight considering both translational and rotational quadrotor dynamics. However, the method only creates a velocity profile over a given path that is not further optimized.

Apart from time optimality, complying with intermediate waypoint constraints is another requirement for path planning in ADR. A common practice of solving a trajectory optimization problem with waypoint constraints is allocating waypoints to specific time steps and minimizing the spatial distance between these waypoints and the position at the corresponding allocated time steps on the reference trajectory (e.g., [98] and [118]). The time allocation of the waypoints is, however, nontrivial and difficult to determine. This is tackled in [88], but the work uses body rates and collective thrust as control inputs and does not represent realistic actuator saturation. Recent work [95] introduces a complementary progress constraints approach, which considers true actuator saturation, uses single rotor thrusts as control inputs, and exploits quaternions to create full, singularity-free representations of the orientation space with consistent linearization characteristics. While the aforementioned methods create time-optimal trajectories passing through given gates, they are computationally costly, and hence, intractable in real time.

4) Search-Based Trajectory Planning: Search-based planning methods [99], [100] rely on discretized state and time spaces. They solve the trajectory planning through graph search algorithms such as A*. The search graph is built using minimum-time motion primitives with discretized velocity, acceleration, or jerk input. The algorithms then use trajectories of a simpler model, e.g., with velocity input, as heuristics for the search with a more complex model. Search-based planning methods can optimize the flight time up to discretization, but they suffer from the curse of dimensionality, which renders them too computationally demanding for a complex quadrotor model. Furthermore, the employed per-axis dynamic limits (velocity, acceleration, and jerk) do not represent the true quadrotor model, further decreasing the quality of found plans. Finally, although searching for minimum time trajectories, the methods are currently limited to planning between two states, which is not suitable for multiwaypoint drone racing.

5) Sampling-Based Trajectory Planning: Sampling-based methods like RRT* [119] can be used for planning trajectories for linearized quadrotor models. Several time-minimizing approaches [77], [101] use a point-mass model for high-level time-optimal trajectory planning. In [101], an additional trajectory smoothing step is performed where the generated trajectory is connected with high-order polynomials by leveraging the differential flatness property of the quadrotor. Ichter et al. [120] use sampling-based approach with massive GPU parallelization and a 6-D double integrator system of UAV with additional single integrator yaw dynamics. However, these point-mass approaches need to relax the single actuator constraints and instead limit the per-axis acceleration, which results in trajectories that are conservative and suboptimal given a minimum time objective. Zhiling et al. [102] use minimum-jerk motion primitives for connecting randomly sampled states inside RRT* to plan a collision-free trajectory. Since the authors use polynomials, this approach can only generate smooth control inputs, meaning that they cannot rapidly switch from full thrust to zero thrust if required.

The first method for planning minimum-time trajectories in a cluttered environment for the full quadrotor model was proposed in [37]. It uses a hierarchical sampling-based approach with an incrementally more complex quadrotor model to guide the sampling. The authors showed that the method outperforms both polynomial and search-based methods in minimizing trajectory time. Yet, the method is offline and intractable in real time. Most recently, Romero et al. [104] proposed an online replanning approach that plans minimum-time trajectories for a point-mass model. The paths of replanned trajectories are then consequently used by model predictive contouring control [103] with a full quadrotor model to maximize the progress along the path. This method is capable of outperforming other classical approaches due to the replanning capability and progress maximization with a full quadrotor model.

6) Discussion: A planned trajectory can be understood as an intermediate representation that, given information about the robot's dynamics and the environment, helps guide the platform through the race track and ultimately perform the task at hand. One might argue if this intermediate representation is needed at all, since ultimately, what we are looking for is a policy that maps sensor information and current environment knowledge to the actuation space. This is generally achieved with learning-based approaches, discussed in Section IV, which bypass the planning stage and directly convert sensor observations to actuation commands [121], [122], [123].

One of the biggest benefits of explicit planning is *modularity*. This means that the developed algorithms can be used off-the-shelf for different drone tasks outside racing, such as search and rescue, which is not the case for single-purpose learned approaches. However, explicit planning suffers from the disconnection (or an open loop) between the planning and the deployment stage. Unexpected deviations from the plan, be it in the time domain (like unmodeled system delays) or in the state-space domain (like state estimation drifts or jumps in the VIO pipeline), can lead to compound errors, and ultimately, a complete system failure.

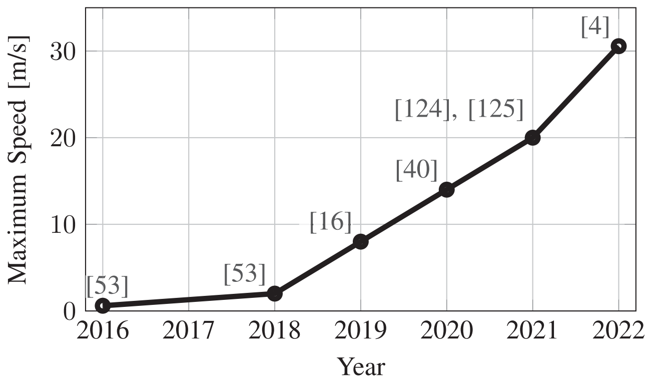


Fig. 4. Top speeds demonstrated on autonomous drones over time from both literature and competition data.

This can be tackled with more complex control approaches that do some part of the replanning online [104].

C. Control

Over the last decade, significant advancements have been made in agile multicopter control. Every year, increasing top speeds are demonstrated in the literature as shown in Fig. 4.

Controllers must be able to make real-time decisions in the face of poor sensor information and model mismatch. Control inputs, $u(t)$, can come in a variety of modalities for quadrotor control, such as velocity and heading, body rates and collective thrust, or direct rotor thrust commands [36]. Typically, a high-level controller computes a desired virtual input such as body rates and collective thrust, which is then passed down to a low-level flight controller that directly controls the individual rotors on the multicopter.

Commonly used open source controllers such as PixHawk[1] or BetaFlight[2] are widely available to the drone racing community. BetaFlight is the most commonly used low-level controller for agile drone flight and has been widely adopted by the FPV racing community.

In the following sections, we provide an overview of successful approaches to achieving high speeds in both simulation and real-world applications. We sort the approaches into *model-based control* and *coupled perception and control*.

1) Model-Based Control: In model-based control, an explicit model of the dynamic system is used to calculate control commands that satisfy a given objective such as minimizing time or tracking error. Models enable the prediction of future states of the drone and provide information about the system's stability properties. In [126], geometric tracking control is introduced on the special Euclidean group SE(3) and completely avoids singularities commonly associated with Euler angle formulations on SO(3). This nonlinear controller showed the ability to execute acrobatic maneuvers in simulation and was the first to demonstrate recovery from an inverted initial attitude. The dynamic model of a quadrotor is shown to be differentially flat when choosing its position and heading as flat outputs in [112]. In this work, many agile maneuvers are performed onboard real drones with speeds up to 2.6 $\mathrm{m/s}$.

The previous work is extended in [26], proving that the dynamics model of a quadrotor subject to linear rotor drag is also differentially flat. The inclusion of the aerodynamic model within the nonlinear controller led to demonstrated flight speeds up to 4 $\mathrm{m/s}$ while reducing tracking error by 50% onboard a real drone.

The differential flatness method is further extended in [127] by cascading an incremental nonlinear dynamic inversion (INDI) controller with the differential flatness controller described in [112] but neglects the aerodynamic model addition from [26]. The INDI controller is designed to track the angular acceleration commands $\dot{\Omega}$ from the given reference trajectory. Top speeds of nearly 13 $\mathrm{m/s}$ and accelerations over 2 g are demonstrated onboard a real quadrotor. The controller shows robustness against large aerodynamic disturbances in part due to the INDI controller.

An investigation of the performance of nonlinear model predictive control (NMPC) against differential flatness methods is available in [125]. Cascaded controllers of INDI-NMPC and INDI-differential flatness are shown to track aggressive racing trajectories that achieve speeds of around 20m/s and accelerations of over 4 g. While differential flatness methods are computationally efficient controllers and relatively easy to implement, they are outperformed on racing tasks by NMPC.

An excellent overview of model predictive control (MPC) methods applied to microaerial vehicles can be found in [128]. Because quadrotors are highly nonlinear systems, nonlinear MPC is often used as the tool of choice for agile maneuvers. The debate of linear versus nonlinear MPC is thoroughly discussed in [129]. Model predictive path integral (MPPI) control is a sampling-based optimal control method that has found excellent success on the AutoRally project, a 1/5th scale ground vehicle designed to drive as fast as possible on loose dirt surfaces [130], [131]. An introduction to MPPI can be found online.[3] The MPPI approach can be used on agile quadrotors to navigate complex forest environments, however, analysis was only performed in simulation [130]. Most of the successful demonstrations of MPPI come from ground robots [130], [131]. Because MPPI is a sampling-based algorithm, scaling to higher dimension state spaces of quadrotors can lead to performance issues as shown in [124].

Nonlinear MPC methods are also used in [40] where a nominal quadrotor model is augmented with a data-driven model composed of Gaussian processes and used directly within the MPC formulation. The authors found that the Gaussian-process model could capture highly nonlinear aerodynamic behavior, which is difficult to model in practice as described in Section II. The additional terms introduced by the Gaussian process added computational overhead to the MPC solve times, but it was still able to run onboard a Jetson TX2 computer.

Similar to [127], Hanover et al. [124] question whether or not it is necessary to explicitly model the additional aerodynamic

[1][Online]. Available: https://pixhawk.org/
[2][Online]. Available: https://github.com/betaflight/betaflight
[3][Online]. Available: https://autorally.github.io/

terms from [40] due to the added computational and modeling complexity. Instead, they propose to learn residual model dynamics online using a cascaded adaptive NMPC architecture. Aggressive flight approaching 20m/s and over 4 g acceleration is demonstrated on real racing quadrotors. In addition, completely unknown payloads can be introduced to the system, with minimal degradation in tracking performance. The adaptive inner loop controller added minimal computational overhead and improved tracking performance over the Gaussian process MPC by 70% on a series of high-speed flights of a racing quadrotor [40], [124].

Contouring control methods can deal with competing optimization goals such as trajectory tracking accuracy and minimum flight times [132]. These methods minimize a cost function, which makes tradeoffs between these competing objectives. In [133], nonlinear model predictive contouring control (MPCC) is applied to control small model race cars. MPCC was then extended to agile quadrotor flight in [103]. Although the velocities achieved by the MPCC controller were lower than that of [124] and [125], the lap times for the same race track were actually lower due to the ability of the controller to find a new time allocation that takes into account the current state of the platform at every timestep. The work is further extended to solve the time-allocation problem online, and to replan online [104] while also controlling near the limit of the flight system. Similar work uses tunneling constraints in the MPCC formulation in [134].

2) Perception Awareness: Other methods that lie in the intersection of perception, planning, and control include a perception objective in the cost function that helps improve the visibility of an objective or the quality of the state-estimation pipeline. The methods are called *perception aware*, and the first methods were proposed in [135], [136], and [137]. This is integral to the drone-racing problem because, to navigate a challenging race course, the gates that define the course layout must be kept in view of the onboard cameras as much as possible. In addition, coupling the perception with the planning and/or control problem can alleviate issues in state estimation because the racing gates are usually feature rich. Therefore, the use of perception-related objectives in both planning and control pipelines is commonplace [91], [93], [120], [137], [138]. For example, in [93] and [137], the authors tackle the problem of minimizing the time required by a quadrotor to execute a given path, while maintaining a given set of landmarks within the field of view of its on-board camera, or Tordesillas and How [139] include a perception-aware term in the cost function to maximize the visibility of the closest dynamic obstacle, in order to readily plan a path that avoids it. These methods are called *perception aware* [140], and in the following, we highlight their core characteristics.

The goal is as follows: navigate a trajectory with low tracking error while keeping a point of interest in view while minimizing motion blur for maximum feature detection and tracking. The first instance applied to agile quadrotors was perception-aware model predictive control (PAMPC) introduced in [140]. In this work, a nonlinear program is optimized using a sequential quadratic programming approximation in real time. The cost function contains both vehicle dynamic terms as well as perception awareness terms such as keeping an area of interest in the center of the camera frame.

This technique is applied to the drone racing problem in [141], where an MPPI controller is designed with a deep optical flow component that predicts the movement of relevant pixels (i.e., gates). The perception constraints are introduced into a nonlinear optimization problem and deployed in a drone-racing simulator. The approach was not demonstrated onboard real hardware. In [142], a perception-aware MPC based on differential flatness was used to ensure that a minimum number of features are tracked between control updates and thus guarantee localization. To achieve this, a perception chance constraint within the MPC formulation is introduced to ensure that at least n number of landmarks are within the field of view of the camera at all times with some bounded probability.

3) Discussion: The performance of model-based controllers degrades when the model they operate on is inaccurate [124]. For drones, defining a good enough model is an arduous process due to highly complex aerodynamic forces, which can be difficult to capture accurately within a real-time capable model. In addition, the tuning process of many model-based controllers can be arduous, and requires a high level of domain expertise to achieve satisfactory performance.

In any optimal control problem, a cost function that the user wants to optimize must be defined. Traditionally, convenient mathematical functions leveraging convex costs are used because these functions are easy to optimize and there is a large toolchain available for optimizing such problems such as Acados [143], CVXGEN [144], HPIPM [145], or Mosek [146]. In many drone racing articles, the optimal control problem is formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{u}} \boldsymbol{x}_N^T \boldsymbol{Q} \boldsymbol{x}_N + \sum_{k=0}^{N-1} \boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \\ &\text{subject to}: \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t) \\ &\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \boldsymbol{u}_{\text{min}} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{\text{max}} \end{aligned} \tag{2}$$

where the state is given by x_k, the control input is given by u_k, the state cost matrix is given by Q, and the control cost matrix is given by R. The optimization problem is constrained by the dynamics of the system given by $f(x_k, u_k, \delta t)$ where δt is a finite time step. The nonlinear dynamics are typically propagated forward using an integrator such as fourth-order Runge–Kutta, RK4. In addition, the problem is subject to the thrust limits of the platform, u_{min} and u_{max}, and some initial condition of the system x_0. In this formulation, a reference position and control are provided by a high-level planner and the goal of the controller is to track the given reference, but this objective is ill-defined for the drone racing problem: in drone racing, we wish to complete the track in as little time as possible; therefore, our objective can be better formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{u}} \sum_{k=0}^{T} \delta t \\ &\text{subject to}: \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{\text{RK4}}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t) \end{aligned}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}}, \quad \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{u} \in \mathcal{U} \tag{3}$$

where T is the number of discrete time steps it takes to complete the race, and the set $\mathcal{U}$ contains the input constraints (e.g., single-rotor thrust constraints). The set $\mathcal{X}$ encodes all state constraints, from possible limits in the state itself (e.g., attitude or velocity constraints), to more complex constraints such as the fact that the drone has to pass through a set of gates in a predetermined order without colliding. This approach requires a time-horizon that predicts all the way until the end of the task, which is intractable to optimize online.

Reinforcement learning (RL) methods [36], [122] can optimize a proxy of this cost function, however do so in an offline fashion, requiring large amounts of training experience to approximate the value function. RL methods do not necessarily depend on a high-level planner to provide a reference to track. We will discuss some recent approaches using RL methods in the following section.

IV. Learning-Based Approaches

In this section, we present various learning-based approaches for drone racing. These approaches replace the planner, controller, and/or perception stack with a neural network. Learning-based methods have gained significant traction in the last few years, given their ability to cope with both high-dimensional (e.g., images) or low-dimensional (e.g., states) input data, their representation power, and the ease of developing and deploying them on hardware.

The big advantage of these methods is that they require less computational effort than traditional methods, possibly enabling low-latency replanning and control. In addition, they are much more robust to system latencies and sensor noise, which can be easily accounted for by identifying them on physical drones, and then, adding them to the training environments [36]. However, the major limitation of these methods is their sample complexity. There are currently two possibilities for data gathering. The first, mostly popular in the initial stages of learning-based robotics [64], [79], [147], [148], [149] is to collect data in the real world. The data are then annotated by a human or an automated process, and used for training. The second, much more popular in recent years and currently achieving the best results, consists of using simulation for collecting training data [36], [65], [121], [150], [151]. However, significant simulation engineering is required to enable generalization if the training data come from a simulator. Conversely, generalization is easier if data come from the real world, but the data collection process is very slow, tedious, and expensive.

Surveys covering existing methods for learning-based flight already exist [152], [153]. In contrast to them, we cover the most recent advances and give a broader discussion on the comparison between learning-based and traditional methods for drone racing.

A. Learned Perception

For learned perception modules, the goal of the network is to use images from an RGB, depth, or event camera to detect landmarks within the environment and output useful representations such as waypoints, or the location of gates on the track. A depiction of this architecture can be seen in Fig. 5. An overview of deep learning methods for vision-based navigation specific to drone racing can be found in [153].

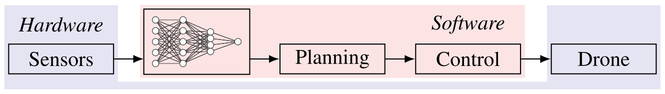


Fig. 5. Architecture 2: Learned perception.

In [64], a dataset of images is collected from a forward-facing camera mounted on a drone labeled with the relative position to the closest gate. This dataset is used to train a network that predicts from an image both the next gate location and its uncertainty. Predictions are then fused with a VIO system in an (EKF to predict the position of the drone on the track. Similarly in [16], a convolutional neural network (CNN) is used to detect gate corners in the AlphaPilot challenge. Once the gate corners are detected, classical computer vision algorithms like PnP can be used to find the coordinates of the gate in the camera frame. Using an EKF, the gate corner locations can be fused with a traditional VIO pipeline to improve the estimates of the drone's location and orientation [16].

Oftentimes, perception networks consume precious resources onboard computationally limited drones. To minimize the network processing time, the authors in [82] and [154] proposed optimized architectures for gate detection on real-world data. A similar optimization went into "GateNet" [155], a CNN to detect gate center locations, distance, and orientation relative to the drone. The same authors developed a follow-up work denoted as "Pencil-Net" to do gate detection using a lightweight CNN in [156]. Most learning-based perception networks can suffer from poor generalization when deployed in environments that were not included in the training data. To reduce deployment sensitivity to lighting conditions or background content, virtual gates can be added to real-world backgrounds [157].

Up until recently, RGB and depth cameras were used exclusively in the drone racing task, however, these sensor modalities can be sensitive to changes in the environment such as illumination changes. To overcome this, Andersen et al. [158] proposed using event cameras coupled with a sparse CNN, recurrent modules, and a you only look once object detector to detect gates. The use of event cameras overcomes potential issues with motion blur from the rapid movement of the drone and is a promising path forward for high-speed navigation.

Overall, deep learning methods for gate detection are the de facto standard in all drone racing systems. However, such gate detectors are always coupled with traditional VIO systems. which explicitly estimate the metric state of the drone. These approaches are discussed in Section III. It is interesting to notice that learning-based odometry systems, such as in [84], [85], and [86], have not yet replaced traditional methods. This is particularly surprising since deep visual odometry systems can specialize to a particular environment, which can be useful for drone racing since the race track is fixed and known in advance.

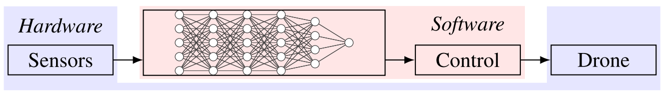


Fig. 6. Architecture 3: Learned planning and perception.

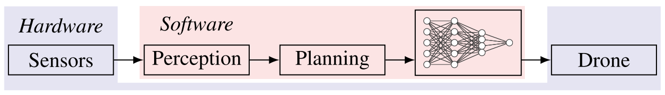


Fig. 7. Architecture 4: Learned control.

A disadvantage of these methods is the high computational cost that makes them impractical for online applications. However, research in end-to-end visual odometry is moving forward at a fast pace [86]. Recently, works proposing end-to-end VIO systems for drones have been published [159], [160], [161]. Sanket et al. [159] propose to learn global optical flow, which is then loosely fused with an IMU for full 6-DoF relative pose estimation. The method in [160] and its extension [161] proposes a CNN-based ego-motion estimator for fast flights. The performance of this method in the UZH-FPV dataset shows that although end-to-end VIO methods are a promising solution for agile flights, they are not yet mature for drone racing. We foresee that in the near future, researchers will be able to apply these methods to the drone racing task.

B. Learned Planning and Perception

A tightly coupled planning and perception stack (see Fig. 6) is a very attractive algorithmic perspective. First, it greatly simplifies the perception task: an explicit notion of a map or globally consistent metric state is not required. Second, it largely reduces computational costs, both in the pretraining and evaluation stages. Finally, it can leverage large amounts of data, collected either in simulation or the real world, to become robust against noise in perception or dynamics. Yet, an interesting observation is that these methods still work best when coupled with an explicit estimator of the metric state [5]. In contrast to traditional methods, a locally consistent odometry system is sufficient [65], [79], [150], waving away the complexities of full-slam methods (e.g., loop closure).

In [79], a coupled perception and planning stack for drone racing is trained using real-world flight demonstrations. While good performance is indicated on the racing task as well as robustness against drift in state estimation, the method requires retraining for each new environment. Therefore, in the follow-up work [65], data generated entirely in simulation are used to train the perception-planning stack, waiving the labor and time-consuming requirement of data collection in the real world. A similar pipeline was used for high-speed autonomous flight through complex environments in [150], which proposes to train a neural network in simulation to map noisy sensory observations to collision-free trajectories directly. This approach was later extended to nanoquadcopters [162], which won the authors the first position in the *IMAV 2022 Nanocopter AI Challenge*. Recent work [163], [164] has shown the possibility of training sensorimotor controllers for obstacle avoidance end-to-end using RL, paving the way toward a system that could solve drone racing completely end-to-end. However, these works still rely on explicit state estimation and a controller to execute velocity commands.

Several other works apply a similar stacked perception and planning pipeline for other ADR tasks [147], [148], [149], [165]. We point the interested reader to existing surveys on the role of learning in drone navigation [152].

A few works also studied the planning problem using data-driven methods, decoupling it from the perception problem. An interesting approach demonstrated in the NeurIPS Game of Drones competition [166] used an off-the-shelf RL algorithm in place of a classic model-based planner for drone racing [167].

C. Learned Control

Data-driven control (see Fig. 7), like RL, allows for overcoming many limitations of prior model-based controller designs by learning effective controllers directly from experience. For example, model-free RL was applied to low-level attitude control [168], in which a learned low-level controller trained with proximal policy optimization (PPO) outperformed a fully tuned PID controller on almost every metric. Similarly, Lambert et al. [169] used model-based RL for low-level control of an a priori unknown dynamic system. More related to drone racing, recent works showcased the potential of learning-based controllers for high-speed trajectory tracking and drone racing [36]. Imitation learning is more data efficient compared to model-free RL. In [170], aggressive online control of a quadrotor has been achieved via training a network policy offline to imitate the control command produced by a model-based controller. Similarly, Sánchez-Sánchez and Izzo [171] studied real-time optimal control via deep neural networks in an autonomous landing problem. Other work in this category has shown that RL can find optimal [122], [172] or highly adaptive controllers [173].

With a learning-based controller, it can be difficult to provide robustness guarantees as with traditional methods such as the linear quadratic regulator (LQR). While a learning-based controller may provide superior performance to classical methods in simulation, it may be that they cannot be used in the real world due to the inability to provide an analysis of the controller's stability properties. This is particularly problematic for tracking the time-optimal trajectories required by drone racing. Recent works have attempted to address this using the Lyapunov-stable neural network design for the control of quadrotors [174]. This work shows that it is possible to have a learning-based controller with guarantees that can also outperform classical LQR methods. Building upon this concept, reachability analysis and safety checks can be embedded in a learned safety layer [175].

D. Learned Planning and Control

The second paradigm of learned control is to produce the control command directly from state inputs without requiring a high-level trajectory planner, as shown in the architecture

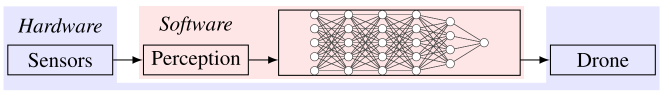


Fig. 8. Architecture 5: Learned planning and control.

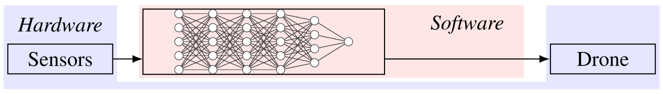


Fig. 9. Architecture 6: End-to-end learning.

diagram of Fig. 8. This approach enabled an autonomous drone with only onboard perception, for the first time, to outperform a professional human, and is state-of-the-art at the time of writing [5]. In ADR, this was proposed in [4] and [122], where a neural network policy is trained with RL to fly through a race track in simulation in near-minimum time. Major advantages of the RL-based method are its capability to handle large track changes and the scalability to tackle large-scale random track layouts while retaining computational efficiency. In [123], deep RL is combined with classical topological path planning to train robust neural network controllers for minimum-time quadrotor flight in cluttered environments. The learned policy solves the planning and control problem simultaneously, forgoing the need for explicit trajectory planning and control.

In this same category, another class of algorithms try to exploit the benefits of model-based and learning-based approaches using differentiable optimizers approaches [176], [177], [178], which leverage differentiability through controllers. For example, for tuning linear controllers by getting the analytic gradients [179], or for creating a differentiable prediction, planning, and controller pipeline for autonomous vehicles [180]. On this same direction, Romero et al. [181] equip the RL agent with a differentiable MPC [176], located at the last layer of the actor network that provides the system with online replanning capabilities and allows the policy to predict and optimize the short-term consequences of its actions while retaining the benefits of RL training.

All these methods inherit the classic advantage of policy learning. In addition, they do not require an external controller to track the plan. This eliminates the discrepancy between the planning and deployment stages, which is one of the main limitations of traditional planning methods (see Section III-B). Some of the limitations of traditional planning remain, such as the requirement of a globally consistent state estimation and a map of the environment. Also, they have not yet been demonstrated in sparse long-horizon planning problems, e.g., flying through a maze at high speeds, where their performance would likely drop due to sample complexity.

E. End-to-End Flight

Expert pilots take raw sensory images from an FPV camera stream and map directly to control commands. In this section, we explore approaches emulating this holistic navigation paradigm in autonomous drones (see Fig. 9).

Two families of approaches can be used to pursue an end-to-end navigation paradigm. The first is substituting each of the perception, planning, and control blocks with a neural network. This structure is followed by [182] and [183], where the authors train a perception-planning network and a control network using imitation learning. The perception network takes raw images as input and predicts waypoints to the next gate. The control network uses such predictions with ground-truth velocity and attitude information to predict control commands for tracking the waypoints. They showed improvements over pure end-to-end approaches, which directly map pixels to control commands and were able to show competitive lap times on par with intermediate human pilots within the Sim4CV simulator [184]. Yet, the division into independent blocks leads to compounding errors and latencies, which negatively affect performance when flying at high speeds [150].

The second family of approaches directly maps sensor observation to commands without any modularity. This design is used in [185], which to date remains the only example of the completely end-to-end racing system. Indeed, other end-to-end systems generally require an inner loop controller and inertial information to be executed. For instance, Rojas-Perez and Martinez-Carranza [186] train an end-to-end CNN to directly predict roll, pitch, yaw, and altitude from camera images. Similarly,the authors in [187] and [188] use a neural network to predict commands directly from vision. To improve sample complexity, they use contrastive learning to extract robust feature representations from images and leverage a two-stage learning-by-cheating framework.

Independently of the design paradigm they follow, end-to-end navigation algorithms are currently bound to simulation. The reasons why no method was successfully deployed in the real world include weak generalization to unseen environments, large computational complexity, and inferior performance to other modular methods. Another interesting observation is that humans can pilot a drone exclusively from visual observations. Conversely, except for [185], end-to-end systems still rely on the state extracted from other measurement modalities, e.g., an IMU. The question of whether autonomous drones can race in the real world at high-speed without any inertial information remains open. We provide more details on this question in Section VIII.

F. Discussion

Data-driven approaches are revolutionizing the research in ADR, ranging from improving the system model to end-to-end control. Currently, the best-performing algorithms for drone racing include a learning-based component [16], [17], and this trend is unlikely to change in the coming years. Indeed, compared to classical model-driven design, they can process high-dimensional sensory inputs directly, can be made robust to any modeling uncertainty (e.g., latency) by simply incorporating it in the training pipeline, and require far less engineering effort for tuning and deploying them [36].

Our analysis shows that the majority of learning-based approaches heavily rely on simulators. While simulators may get better and faster in the near future, recent advances in real-world training [189], [190] and fine tuning [191], [192] offer a potential alternative for zero-shot simulation to reality transfer for sensorimotor policies. However, so far, these works have been limited to legged locomotion. Extension to agile drones could lead to the successful deployment of end-to-end policies, possibly improving the state of the art in agile flight.

Another limitation of the approaches discussed in this section is their inability to adapt to new and uncertain environments quickly. The field of adaptive control has studied this problem extensively [193], [194], [195]. Inspired by these works, there has been a recent push to use advancements in machine learning within the adaptive control framework. A method to learn parametric uncertainty functions is introduced in [196]. These uncertainty functions could be learned offline using data captured from agile flight experiments, and then, embedded within an adaptive controller to adjust controller parameters online during flight. Results indicate that highly accurate trajectory tracking can be achieved with this approach, even in the face of strong wing gusts exceeding 6.5 m/s. More recently, learning-based controllers have shown the ability to adapt zero-shot to large variations in hardware and external disturbances [197]. We see this as a promising area of research and one that is integral for reliable performance in changing environmental conditions.

V. Drone Racing Simulators

One tool that has drastically accelerated the progress of research in autonomous drone flight is the use of simulation environments that attempt to recreate the conditions that real drones experience when flying. Over the years, several simulation environments have been developed for the use of general research.

In 2016, the widely used RotorS simulation environment was published, which extends the capabilities of the popular Gazebo simulation engine to multirotors [23]. Gazebo uses the Bullet physics engine for basic dynamic simulation and contact forces. Linear drag on the body of the multicopter is simulated based on the cross-sectional area and linear velocity of the simulated object. The RotorS extension features many easy-to-use plugins for developing multirotors, however, it distinctly lacks the photorealistic details needed to simulate accurate behavior of estimation and perception pipelines.

AirSim was introduced by Microsoft in 2018 as a photorealistic simulator for the control of drones [21]. It is built on the unreal graphics engine and features easy-to-use plugins for popular flight controllers such as PX4,[4] ArduPilot,[5] and others. It was used in the 2019 NeurIPS Game of Drones challenge [166]. Because of the photorealism of AirSim, it is possible to simulate the entire perception and estimation pipeline with a good possibility of transfer to real-world drone systems. In addition, AirSim comes prepackaged with an OpenAI-Gym environment for training RL algorithms. Organizations such as Bell, Airtonomy, and NASA are using AirSim to generate training data for learning-based perception models.

[4][Online]. Available: https://px4.io/

[5][Online]. Available: https://ardupilot.org/

FlightGoggles [55] was developed as another photorealistic simulator and was used as the primary simulation environment for the Lockheed Martin AlphaPilot challenge. FlightGoggles contains two separate components: a photorealistic rendering engine built with Unity3D and a dynamic simulation implemented in C++. FlightGoggles provides an interface with real-world vehicles using a motion capture system; such an interface allows the rendering of simulated images that correspond to the position of physical vehicles in the real world.

A recent simulator focused on safe RL was proposed in [198]. It uses Gazebo and the Pybullet physics engine as the backend. Leaderboards for several safety-focused training environments exist, encouraging researchers to submit their approaches and compete with other researchers around the world.

Flightmare [22] is a simulation environment featuring photorealistic graphics provided by the Unity engine. The physics engine is decoupled and can be swapped out with various engines for user-defined levels of simulation fidelity. Similar to FlightGoggles, Flightmare can also provide hardware-in-the-loop simulation functions where a virtual, synthetic camera image can be provided to the drone for use in control and estimation [8].

Finally, Aerial Gym [199] is a GPU-accelerated simulator that allows simulating millions of multirotor vehicles in parallel with nonlinear geometric controllers for attitude, velocity, and position tracking. In addition, the simulator offers a flexible interface for modeling a large number of obstacles and generating data such as RGB, depth, segmentation, and optical flow.

VI. Competitions

To gauge the progress of the field as a whole, several drone racing competitions have taken place since 2016. We include a graphical overview of these events in Fig. 2. The ADR competition was an annual competition that took place during the IROS conference between 2016 and 2019. In 2016, 11 teams competed in ADR and were tasked to navigate a series of gates in sequence. The positions of the gates were not known to the participating teams ahead of time, therefore, teams flew very cautiously identifying the next waypoints online. Each team was given 30 min prior to the official competition to fly the course as many times as they wished. The winning team, from korea advanced institute of science & technology (KAIST), made it through 10 of the 26 gates in 1 min and 26 s. For comparison, a human was able to complete the entire 26-gate course in 1 min 31 s. A survey summarizing the approaches used for these early competitions can be found in [19]. The following year, a similar competition took place during IROS in Vancouver, Canada, with better results. This time, 14 teams participated and were given a CAD drawing of the course prior to the event with locations and dimensions of all gates. Only five teams participated in the final in-person event, with the winning team making it through 9 out of 13 gates in over 3 min. A summary of the winning approaches can be found in [14]. Two more ADR competitions took place at IROS 2018 and 2019, with drones navigating courses faster and more reliably.

In 2019, Lockheed Martin sponsored the AlphaPilot AI Drone Racing Innovation Challenge where a 1 million dollar grand

TABLE I
Open Source Software and Datasets

Name and Reference	Category	Year	Link
PAMPC [140]	Controller	2018	https://github.com/uzh-rpg/rpg_mpc
Deep Drone Acrobatics [121]	Controller	2019	https://github.com/uzh-rpg/deep_drone_acrobatics
Data Driven MPC [40]	Controller	2020	https://github.com/uzh-rpg/data_driven_mpc
High MPC [203]	Controller	2022	https://github.com/uzh-rpg/high_mpc
AutoTune [204]	Controller Tuner	2022	https://github.com/uzh-rpg/mh_autotune
Blackbird [205]	Dataset	2018	https://github.com/mit-aera/Blackbird-Dataset
UZH-FPV [78]	Dataset	2019	https://fpv.ifi.uzh.ch/
NeuroBEM [35]	Dataset	2020	https://rpg.ifi.uzh.ch/NeuroBEM.html
Eye Gaze Drone Racing [206]	Dataset	2021	https://osf.io/gvdse/
TII Drone Racing Dataset [207]	Dataset	2024	https://github.com/tii-racing/drone-racing-dataset
Time-optimal Planning for Quadrotor Waypoint Flight [95]	Planner	2021	https://github.com/uzh-rpg/rpg_time_optimal
Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments [37]	Planner	2022	https://github.com/uzh-rpg/sb_min_time_quadrotor_planning
RotorS [23]	Simulator	2016	https://github.com/ethz-asl/rotors_simulator
AirSim [166]	Simulator	2018	https://microsoft.github.io/AirSim/
FlightGoggles [55]	Simulator	2019	https://github.com/mit-aera/FlightGoggles
Flightmare [22]	Simulator	2020	https://uzh-rpg.github.io/flightmare/
Learning to fly—a gym environment with pybullet physics for RL of multi agent quadcopter control [198]	Simulator	2021	https://github.com/utiasDSL/gym-pybullet-drones
Aerial Gym [199]	Simulator	2023	https://github.com/ntnu-arl/aerial_gym_simulator
Sim 2 Real Domain Randomization [65]	Sim2Real Transfer	2019	https://github.com/uzh-rpg/sim2real_drone_racing
RPG Quadrotor Control [26]	Software Stack	2017	https://github.com/uzh-rpg/rpg_quadrotor_control
Agilicious [8]	Software Stack	2022	https://github.com/uzh-rpg/agilicious
Kalibr [56]	Camera Calibration	2022	https://github.com/ethz-asl/kalibr
VID-Fusion [74]	Estimation	2021	https://github.com/ZJU-FAST-Lab/VID-Fusion
Fast-Racing [87]	Planner	2021	https://github.com/ZJU-FAST-Lab/Fast-Racing
Ego-planner [208]	Planner	2021	https://github.com/ZJU-FAST-Lab/ego-planner
GCOPTER [115]	Planner	2022	https://github.com/ZJU-FAST-Lab/GCOPTER
FASTER [209]	Planner	2021	https://github.com/mit-acl/faster
Panther [139]	Planner	2022	https://github.com/mit-acl/panther
Deep Panther [138]	Planner	2023	https://github.com/mit-acl/deep_panther
Raptor [91]	Planner	2021	https://github.com/HKUST-Aerial-Robotics/Fast-Planner

prize was awarded to the winning team [200]. The competition took place first in a virtual qualifying round that used the FlightGoggles simulation environment [55]. Nine teams out of more than 400 worldwide qualified for the final challenge, which included navigating a new track in a time-trial setting against an expert human pilot. Such competition took the form of a tournament, with three seasonal races and a final championship race. This made it very different from previous single-day competitions. Ultimately, professional pilot Gabriel Kocher, from the Drone Racing League, manually piloted his drone through the course in only 6 s. It took 11 s to the winner, MAVLab from TU Delft, and 15 s to the second-place winner, UZH-RPG from the University of Zürich, to complete the course autonomously. The two different approaches are documented in [16] and [17]. Further comments are provided by the winner in [201]. Rojas-Perez and Martinez-Carranza [53] provide an overview of the types of hardware used for some of the drone racing competitions mentioned so far.

In 2019, the Game of Drones competition took place at the NeurIPS conference. This competition was purely simulation-based and used the AirSim simulation environment built by Microsoft [15], [21], [166]. Participants in the Game of Drones competition raced against simulated opponents in a head-to-head fashion, similar to how humans compete in FPV drone racing. Teams raced against a single simulated opponent, navigating through a complex series of gates in three different tiers: Planning only, perception only, and perception with planning.

In 2022, at the Swiss Drone Days event in Zürich, Switzerland, three of the world's best human pilots competed against researchers from the Robotics and Perception Group of the University of Zürich. Flight speeds exceeding 100 km/h were demonstrated by the autonomous drones. When relying on motion capture, the autonomous drones were able to achieve significantly faster laptimes than the expert human pilots. They additionally demonstrated it was possible to win races without motion capture, using only onboard computing and sensors to navigate the race track. IEEE Spectrum author E. Ackermann discusses the multiday event in [202].

Looking into the future, the Abu Dhabi Autonomous Racing League recently announced plans for an ADR competition in 2025.

VII. Datasets, Hardware, and Open Source Code

In this section, we provide an overview of the existing open source code bases, useful datasets for ADR as well as hardware considerations. We first discuss datasets, and then, group the existing open source code bases by their use-cases in Table I and conclude with a brief overview over drone racing hardware.

A. Datasets

In 2018, researchers from MIT released a large-scale dataset for perception during aggressive UAV flight [205]. This dataset contains over 10 h of flight data, which include simulated stereo and downward-facing camera images at 120 Hz, real-world IMU data at 100 Hz, motor speed data at 190 Hz, and motion capture data at 360 Hz. The sensor suite was chosen such that algorithms like VIO or simultaneous localization and mapping could be evaluated on the dataset.

In 2019, the UZH-FPV Drone Racing Dataset was released, which contains many agile maneuvers flown by a professional racing pilot [78]. The dataset includes indoors and outdoors real-world camera images, inertial measurements, event camera data, and ground-truth poses provided by an advanced motion capture system (a total station) providing millimeter-level accuracy. In 2024, the dataset was extended with new data recorded onboard an autonomous racing drone flying in a racing track with peak speed exceeding 20 $\mathrm{m/s}$. These new data include large field-of-view camera images, inertial measurements, and ground truth from a motion capture system. Similar to [205], the authors of this dataset hope to push the state of the art in state estimation during aggressive motion and have created competitions to allow researchers to compete against one another on this agile flight benchmark.[6] A recent effort reported in [207] open-sourced high-quality data from both autonomous and human-piloted flights. This effort enables the study of both the perception and control problem without actual hardware, lowering the barrier of entry for studying drone racing.

Research on how expert human pilots focus on their targets during flying and provide a dataset that contains flight trajectories, videos, and data from the pilots is examined in [206]

NeuroBEM [35] is a hybrid aerodynamic quadrotor model that combines BEM theory models with learned aerodynamic representations from highly aggressive maneuvers. While the model is fit to the specific quadrotor platform defined in [8], the approach can be used for any quadrotor platform and provides over 50% reduction in model prediction errors compared to traditional, exclusively first-principles approaches.

B. Open-Source Code

A significant amount of ADR research has been open sourced to the community, making implementation less daunting for newcomers to the field. A collection of all known drone racing repositories has been provided to the reader in Table I. These code bases range across controllers, planners, sensor calibration, and even entire software stacks dedicated to drone racing. We encourage both newcomers and experienced researchers to check out the extensive amount of open source code bases available and contribute back to the community.

C. Hardware

This survey does not intend to cover the hardware design of racing drones rigorously. For an in-depth overview, see [8], where the hardware and software design for developing a very capable research platform are discussed. To make this survey self-contained, this section presents a brief overview of the hardware design of a racing drone nevertheless.

1) Racing Drone Design: A suitable hardware design should maximize the agility and acceleration of the drone, and hence, it needs to be as lightweight as possible [210]. For drones featuring onboard compute, the drone size is thus lower bounded by the size of the computer. Currently, the NVIDIA Jetson family is the smallest off-the-shelf hardware with sufficient compute to run complex neural networks, and it leads to drones built on 6-in frames. Carbon fiber offers an excellent compromise between the weight and durability of the frame, while other parts (such as holders for the computer) can be designed using a 3-D printer.

For actuation, fast-spinning brushless dc motors are ideal because of their high specific power output, often exceeding 500 W for a 50-g motor. In general, larger propellers will improve the energy efficiency of the drone [7] while smaller propellers lead to a faster motor response. On a 6-in frame, three-bladed 5-in propellers present a good compromise. To sustain the power demand of brushless drone-racing motors (often exceeding 2 kW at full throttle [7]), a lithium-polymer battery with a sufficiently high discharge current rating (e.g., 120 C) is required.

The Pixhawk PX4 flightstack, despite being commonly used for quadrotors [11], [211], fixed-wings [212], and hybrid vertical takeoff and landing (VTOL) platforms [213], is not optimized for agile flight. Conversely, agile autonomous research platforms [5], [8] use Betaflight as a low-level controller, similar to professional human racing pilots.

The design of a capable racing drone is important for researchers developing new technology. However, in many drone racing competitions, the hardware design is not left to the participants but is standardized. This approach is common in human drone racing, where thousands of identical drones are built before each competition. This concept was also adopted by the AlphaPilot [200] competition, where all participants used a given platform. Overall, this approach ensures fair competition.

2) Beyond Quadcopters: While this survey focuses on multicopter drones, future drone racing competitions will go beyond this platform. Indeed, FPV fixed-wing racing is already a popular sport among human pilots [214]. For example, VTOL drones might offer a great alternative to quadcopters. VTOL aircraft combine the high speeds achieved by fixed-wing drones with some of the maneuverability of multicopters. Pioneering works on this platform have already shown agile control [215] and trajectory generation for aerobatic VTOL flight [216]. Perhaps, once such research platforms are available off the shelf, VTOL aircraft racing will become a popular platform for ADR research.

VIII. Open Research Questions and Challenges

While a lot of progress has been made, there are still many challenges to be overcome in drone racing research. In the following, we discuss the most interesting challenges in detail.

A. Challenge 1: Reliable State Estimation at High Speeds

In its current form, online, robust, and accurate state estimation is highly beneficial when pushing autonomous drones to their limits. Currently, classical state estimation approaches based on VIO cannot cope with the perceptual challenges present in drone racing tasks. Motion blur, low texture, and high dynamic range are some reasons why classical VIO algorithms accumulate large errors in localization. The miscalibration of intrinsic

[6][Online]. Available: https://fpv.ifi.uzh.ch/uzh/uzh-fpv-leader-board/

and extrinsic camera parameters can lead to improper estimates of the camera pose on a drone. This is due to local movements of the camera frame relative to the drone body, as well as changes in temperature and pressure. VIO drift can render the state estimates unusable unless corrected through localizations to a prior map. New sensor modalities, such as event cameras, could potentially alleviate this issue. Although event-aided VIO algorithms for drones have been proposed to improve robustness to motion blur, they have not been demonstrated at high speeds as seen in drone racing. Future research in agile flight may focus on finding new event representations that are computationally efficient and compatible with classical VIO formulations. One example is to exploit direct methods [217]. Other promising sensor modalities are motor speed controllers and force sensors. These sensor measurements could be used to include more advanced drone models in VIO, e.g., modeling aerodynamics effects, in order to limit the drift that accumulates where camera measurements are degraded. One of the main consequences of motion blur, low texture, and high dynamic range is unreliable feature extraction and matching. This consequently degrades the performance of the visual frontend. Deep learning methods have the potential to solve this problem. What hinders the application of these methods to drone racing at the moment is their computational cost. Future research should work on lightweight neural networks that can provide inference at a high rate. Neural networks could also be used to remove nonzero mean noise and constant errors from the inertial measurements. A potentially fruitful area of research is in combining neural networks for input processing with a geometry-based VIO backend. This could lead to the next step in the research on VIO for drone racing. Current works [86], [218] have shown that this direction outperforms end-to-end visual-based odometry methods.

B. Challenge 2: Flying From Purely Vision

State-of-the-art autonomous navigation methods rely on visual and inertial information, usually combined with classic perception algorithms. Conversely, expert human pilots rely on nothing more than a FPV video stream, which they use to identify goals and estimate the ego-motion of the drone. Building systems that, similarly to human pilots, only rely on visual information is very interesting from a scientific perspective. Indeed, since simulating RGB is very challenging, solving this question might require lifelong learning algorithms operating in the real world. In addition, eliminating inertial information might have some engineering advantages too, e.g., data throughput, power consumption, and lower cost. Seminal works in this direction try to understand how humans solve this task [206], [219]. They found that expert pilots can control drones despite a 200-ms latency, which is compensated by the human brain. Taking inspiration from biology, a recent work [220] shows that it is possible to fly with camera images and an onboard gyroscope (e.g., removing the accelerometer), as long as the system never hovers. However, the aforementioned questions still remain mostly open and a good avenue for research at the intersection of computer vision, neuroscience, and biology.

C. Challenge 3: Multiplayer Racing

Much of the work done up until this point on ADR has focused on time-optimal flight without considering how a capable opponent might impact the competition dynamics. In FPV races, pilots can compete against up to five opponents simultaneously, bringing about the need to anticipate how their opponents might behave. Humans are astonishingly capable of recognizing opportunities for overtaking and executing complex maneuvers in the face of large aerodynamic disturbances caused by flying close to another drone. Achieving such capabilities requires an agent to estimate their opponent's state using only onboard visual sensors. However, these observations in drone racing are sparse because the camera faces forward along the heading axis, meaning that the only time an opponent is observable is when the ego-agent is behind them. Sophisticated motion and planning models that can propagate predictions of the opponents' states and racing lines through time are necessary to anticipate collisions or overtaking opportunities. One way to simplify the problem is combining classical vision with learning-based control, which has shown promising results in multiagent zero-sum games for locomotion [221]. An initial study [222] examined how game-theoretic planners can lead to highly competitive behavior in two-player drone racing, however, this work was confined to racing on a 2-D plane. The work was further extended to 3-D spaces in [223], but there is a significant opportunity for researchers to explore the competitive nature of drone racing and develop interesting racing strategies that lead to time-optimal agents that are able to deal with complex opponent behavior.

D. Challenge 4: Safety

ADR research has so far focused on demonstrating that superhuman performance in racing is possible in controlled conditions [5] but has put less emphasis on risk and safety. We predict that this trend will soon change. Adding safety to agile flight has gained much attention recently [9], [198]. Initial works focused on generating a collision-free trajectory [224], [225], [226] with less emphasis on performance. More geared toward agile flight, the authors in [209], [227], [228], and [229] have studied the problem of trading off safety and performance. All the aforementioned works rely on solving constrained optimization problems. Outside of drone racing, similar paradigms have been developed and have the potential to inspire future algorithms. Such methods are, for example, conformal analysis [230], chance-constrained dynamic programming [231], control barrier functions [232], or reachability analysis [233]. The latter has been successfully applied in the context of autonomous driving with collision avoidance [234], [235].

More modern, learning-based methods have been explored for risk-aware autonomous driving in the context of a map-prediction approach [236] and in combination with Tube MPC [237], a form of MPC that takes stochasticity into account. However, such approaches generally do not scale to high-dimensional perception but rely on robust state estimation for all involved agents. Combining such algorithms with the methods for vision-based, high-speed drone racing presented in this survey could solve both of these problems simultaneously.

As a first step in this direction, recent work [238] has shown that a learned control policy can be conditioned on an auxiliary input signal from a user. The signal regulates the maximally available thrust, leading to a single learned policy that can race at various speeds and risk levels.

E. Challenge 5: Transfer to Real-World Applications

Drone racing, while an extraordinarily challenging research environment, is ultimately not the end goal. Opportunities exist for technology transfer between the drone racing research community to real-world applications such as search and rescue, inspection, agriculture, videography, delivery, passenger air vehicles, law enforcement, and defense. However, applications that leverage the full agility of the platform have much to gain. Drones that fly fast, fly farther, therefore, increasing the productivity of drones in every commercial sector [7].

One of the major challenges to real-world application is generalization to conditions where the environmental knowledge before deployment is limited. For example, we often do not have a known map ahead of time for real-world applications, which requires simultaneous estimation of the state of the drone while mapping the environment. However, a central theme of drone racing research has been the development of adaptive control strategies and decision-making algorithms to enable drones to react rapidly to changes in the race track or the robot condition (see Section III-A and III-C). These strategies can be used to handle real-world applications where environmental knowledge is imperfect and to enable adaptation to unforeseen obstacles and challenges. In addition, learning-based sensorimotor controllers for drones, increasingly more popular due to research on racing, have been designed with the ability to generalize from limited data, adapt, and improve their performance over time (see Section IV). Such generalization and adaptation abilities have already been applied to cases where there is no previous knowledge of the environment [150].

Building algorithms that can continually improve from their experience is another alternative to favor this transfer. While recent advances in RL research point to the feasibility of this path [191], [192], [239], it is unclear when and how such recent approaches would be applicable to drones or similarly agile platforms in the real world. Collecting data for continual RL onboard a drone is notoriously difficult. This is because the drone does not have the luxury of remaining in contact with the ground like legged robots and cars, and thus, has to immediately know how to hover otherwise a crash will occur. One interesting area that may be useful for continual RL in drones is the notion of "safe-RL." The goal of safe RL is to enable exploration without ever incurring catastrophic failure of the system. Initial work on this topic can be found in [240]. A survey article covering safe RL methods can be found in [9]. Furthermore, a thorough review article on continual, or life-long RL can be found in [241].

IX. Conclusion

From racing at a pace comparable to walking speed [19], autonomous drones have advanced to surpassing world champions [5]. Such an exponential advance has been driven by both algorithmic innovations, e.g., learning sensorimotor controllers in simulation, and system engineering improvements. Such advances span the entire navigation pipeline: perception, planning, and control. This article comprehensively covered each of these topics. Methodologically, the dominant trend was a shift from conventional methods to data-driven solutions. However, in contrast to fields like computer vision and natural language processing, neural networks did not replace but coexist with traditional methods: no method with competitive performance in the real world was fully data driven. The most resilient part of the pipeline was state estimation, where strong prior knowledge about the dynamics and environment were still needed to cope with the lack of sensorimotor data. In the short term, we predicted that such a hybrid approach could be applied to other physical systems, e.g., autonomous ground vehicles and personal robots. However, in the long term, we predicted that, similarly to research in computer vision and natural language processing, neural networks will replace each part of the pipeline. This will require many innovations, e.g., computationally efficient architectures, offline pretraining strategies, and fast adaptation schemes to previously unseen conditions. While autonomous drones are already superhuman in controlled scenarios, many challenges are yet to be solved to outperform human champions in official drone racing leagues and transfer the findings to real-world applications.

Acknowledgment

The authors would like to thank M. Muglikar for her valuable inputs on event-camera methods for state estimation and perception.

Authors' Contributions: D. Hanover initiated the idea of this article, created the article structure, and contributed to all sections of this article while coordinating efforts among the co-authors. A. Loquercio contributed to the article structure and the learning-based sections. L. Bauersfeld authored the Drone Modeling section and created the graphics seen throughout. A. Romero contributed to the Classical Planning and Control sections. G. Cioffi contributed to the Classical Perception and Challenges sections. Y. Song contributed to the Simulators and Learning-Based Planning/Control sections. R. Penicka contributed to both Classical and Learning-Based Planning sections. E. Kaufmann contributed to the article structure and throughout the Learning-Based sections. D. Scaramuzza contributed to the general article structure and revised the article thoroughly and critically.

References

[1] T. A. Wilkinson, *Early Dynastic Egypt*. Evanston, IL, USA: Routledge, 2002.

[2] S. M. Arab, "The sed-festival (heb sed) renewal of the kings' reign," *Arab World Books*, Nov. 2017, Accessed: May 22, 2024. [Online]. Available: https://www.arabworldbooks.com/en/e-zine/the-sed-festival-heb-sed-renewal-of-the-kings-reign

[3] J. Betz et al., "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[4] Y. Song, A. Romero, M. Mueller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Sci. Robot.*, vol. 8, no. 82, 2023, Art. no. adg1462.

[5] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023.

[6] Z. Ameli, Y. Aremanda, W. A. Friess, and E. N. Landis, "Impact of UAV hardware options on bridge inspection mission capabilities," *Drones*, vol. 6, no. 3, 2022, Art. no. 64.

[7] L. Bauersfeld and D. Scaramuzza, "Range, endurance, and optimal speed estimates for multicopters," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2953–2960, Apr. 2022.

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

[9] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 5, pp. 411–444, 2022.

[10] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim, "A direct visual servoing-based framework for the 2016 IROS autonomous drone racing challenge," *J. Field Robot.*, vol. 35, no. 1, pp. 146–166, 2018.

[11] K. Mohta et al., "Fast, autonomous flight in GPS-denied and cluttered environments," *J. Field Robot.*, vol. 35, no. 1, pp. 101–120, 2018.

[12] AGILEFLIGHT: Low-latency Perception and Action for Agile Vision-based Flight, Accessed: May 22, 2024. [Online]. Available: https://cordis.europa.eu/project/id/864042

[13] AUTOASSES: Autonomous Aaerial Inspection of GNSS-Denied and Confined Critical Infrastructures, Accessed: May 22, 2024. [Online]. Available: https://cordis.europa.eu/project/id/101120732

[14] H. Moon et al., "Challenges and implemented technologies used in autonomous drone racing," *Intell. Serv. Robot.*, vol. 12, no. 2, pp. 137–148, 2019.

[15] Microsoft, "Game of drones", Accessed: May 22, 2024. [Online]. Available: https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/

[16] P. Foehn et al., "Alphapilot: Autonomous drone racing," *Auton. Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[17] C. De Wagter, F. Paredes-Vallés, N. Sheth, and G. de Croon, "The artificial intelligence behind the winning entry to the 2019 AI robotic racing competition," *Field Robot.*, vol. 2, pp. 1263–1290, 2022.

[18] The Motorsport Concept Building an Autonomous Mobility Ecosystem, [Online]. Available: https://a2rl.io/news/18/The-Motorsport-Concept-Building-an-Autonomous-Mobility-Ecosystem---ASPIRE-s-Executive-Director,-Dr-Tom-McCarthy

[19] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, "The IROS 2016 competitions [competitions]," *IEEE Robot. Automat. Mag.*, vol. 24, no. 1, pp. 20–29, Mar. 2017.

[20] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robot. Automat. Mag.*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[21] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics*. Berlin, Germany: Springer, 2018, pp. 621–635.

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

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

[24] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," in *Proc. Int. Conf. Simul., Model., Program. Auton. Robots*, 2012, pp. 400–411.

[25] R. W. Prouty, *Helicopter Performance, Stability, and Control*. Melbourne, FL, USA: Krieger Pub. Co, 1995.

[26] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.

[27] S. Yoon, H. C. Lee, and T. H. Pulliam, "Computational analysis of multi-rotor flows," in *Proc. 54th AIAA Aerosp. Sci. Meeting*, 2016, Art. no. 0812.

[28] P. V. Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *Proc. AIAA Aerosp. Sci. Meeting*, 2018, Art. no. 1266.

[29] S. Yoon, P. V. Diaz, D. D. Boyd, W. M. Chan, and C. R. Theodore, "Computational aerodynamic modeling of small quadcopter vehicles," in *Proc. Amer. Helicopter Soc. 73rd Annu. Forum*, Fort Worth, TX, USA, 2017, pp. 371–386.

[30] R. Gill and R. D'Andrea, "Propeller thrust and drag in forward flight," in *Proc. IEEE Conf. Control Technol. Appl.*, 2017, pp. 73–79.

[31] R. Gill and R. D'Andrea, "Computationally efficient force and moment models for propellers in UAV forward flight applications," *Drones*, vol. 3, no. 4, pp. 77–124, 2019.

[32] W. Khan and M. Nahon, "Toward an accurate physics-based UAV thruster model," *IEEE/ASME Trans. Mechatron.*, vol. 18, no. 4, pp. 1269–1279, Aug. 2013.

[33] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proc. AIAA Guid., Navigation Control Conf. Exhib.*, 2007, Art. no. 6461.

[34] M. Bangura and R. Mahony, "Thrust control for multirotor aerial vehicles," *IEEE Trans. Robot.*, vol. 33, no. 2, pp. 390–405, Apr. 2017.

[35] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," *RSS: Robot., Sci., Syst.*, 2021, pp. 1–10.

[36] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *Proc. Int. Conf. Robot. Automat.*, 2022,pp. 10504–10510.

[37] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5719–5726, Apr. 2022.

[38] P. V. Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *Proc. AIAA Aerosp. Sci. Meeting*, 2018, Art. no. 1266.

[39] J. Luo, L. Zhu, and G. Yan, "Novel quadrotor forward-flight model based on wake interference," *AIAA J.*, vol. 53, no. 12, pp. 3522–3533, 2015.

[40] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3769–3776, Apr. 2021.

[41] S. Sun, C. C. de Visser, and Q. Chu, "Quadrotor gray-box model identification from high-speed flight data," *J. Aircr.*, vol. 56, no. 2, pp. 645–661, 2019.

[42] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 4653–4660.

[43] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 3223–3230.

[44] G. Shi et al., "Neural lander: Stable drone landing control using learned dynamics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 9784–9790.

[45] A. van den Oord et al., "WaveNet: A generative model for raw audio," in *Proc. 9th ISCA Workshop Speech Synth. Workshop*, 2016, p. 125.

[46] B. Wang, Z. Ma, S. Lai, and L. Zhao, "Neural moving horizon estimation for robust flight control," *IEEE Trans. Robot.*, vol. 40, pp. 639–659, 2024.

[47] W. Peukert, "Über die abhängigkeit der kapazität von der entladestromstärke bei bleiakkumulatoren," *Elektrotechn. Zeitschr.*, vol. 20, , pp. 287–288 1897.

[48] N. Galushkin, N. Yazvinskaya, and D. Galushkin, "Generalized analytical model for capacity evaluation of automotive-grade lithium batteries," *J. Electrochem. Soc.*, vol. 162, pp. A308–A314, 2015.

[49] N. Galushkin, N. N. Yazvinskaya, and D. N. Galushkin, "A critical review of using the Peukert equation and its generalizations for lithium-ion cells," *J. Electrochem. Soc.*, vol. 167, no. 12, Aug. 2020, Art. no. 120516.

[50] X. Zhang, W. Zhang, and G. Lei, "A review of li-ion battery equivalent circuit models," *Trans. Elect. Electron. Mater.*, vol. 17, pp. 311–316, 2016.

[51] L. Zhang, S. Wang, D.-I. Stroe, C. Zou, C. Fernandez, and C. Yu, "An accurate time constant parameter determination method for the varying condition equivalent circuit model of lithium batteries," *Energies*, vol. 13, no. 8, 2020, Art. no. 2057.

[52] D. Bicego, J. Mazzetto, R. Carli, M. Farina, A. Franchi, and V. Arellano-Quintana, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *J. Intell. Robot. Syst.*, vol. 100, pp. 1213–1247, 2020.

[53] L. O. Rojas-Perez and J. Martinez-Carranza, "On-board processing for autonomous drone racing: An overview," *Integration*, vol. 80, pp. 46–59, 2021.

[54] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robot. Automat. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011.

[55] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 6941–6948.

[56] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 4304–4311.

[57] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online IMU intrinsic calibration: Is it necessary?," in *Proc. Robot., Sci. Syst.*, Corvallis, OR, USA, 2020, p. 1–10.

[58] D. Scaramuzza and Z. Zhang, "Visual-inertial odometry of aerial robots," in *Encyclopedia Robot.*, M. Ang, O. Khatib, and B. Siciliano, Ed., Berlin, Heidelberg: Springer, 2019, doi: 10.1007/978-3-642-41610-1_71-1.

[59] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 611–625, Mar. 2018.

[60] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 298–304.

[61] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2007, pp. 3565–3572.

[62] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 314–334, 2015.

[63] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[64] E. Kaufmann et al., "Beauty and the beast: Optimal methods meet learning for drone racing," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 690–696.

[65] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Trans. Robot.*, vol. 36, no. 1, pp. 1–14, Feb. 2019.

[66] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 4666–4672.

[67] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 994–1001, Apr. 2018.

[68] S. Sun, G. Cioffi, C. De Visser, and D. Scaramuzza, "Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 580–587, Apr. 2021.

[69] P. Chen, W. Guan, and P. Lu, "ESVIO: Event-based stereo visual inertial odometry," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3661–3668, Jun. 2023.

[70] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "VIMO: Simultaneous visual inertial model-based odometry and force estimation," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2785–2792, Jul. 2019.

[71] G. Cioffi, L. Bauersfeld, and D. Scaramuzza, "HDVIO: Improving localization and disturbance estimation with hybrid dynamics VIO," *Robot., Sci. Syst.*, 2023.

[72] G. Cioffi, L. Bauersfeld, E. Kaufmann, and D. Scaramuzza, "Learned inertial odometry for autonomous drone racing," *IEEE Robot. Automat. Lett.*, vol. 8, no. 5, pp. 2684–2691, May 2023.

[73] G. Gallego et al., "Event-based vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 154–180, Jan. 2022.

[74] Z. Ding, T. Yang, K. Zhang, C. Xu, and F. Gao, "VID-Fusion: Robust visual-inertial-dynamics odometry for accurate external force estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 14469–14475.

[75] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2502–2509.

[76] M. Burri et al., "The EuRoC micro aerial vehicle datasets," *Int. J. Robot. Res.*, vol. 35, no. 10, pp. 1157–1163, 2016.

[77] P. Foehn et al., "Alphapilot: Autonomous drone racing," *Robot., Sci. Syst.*, vol. 46, no. 1, pp. 307–320, 2020.

[78] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? The UZH-FPV drone racing dataset," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 6713–6719.

[79] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 133–145.

[80] S. Li, M. M. Ozo, C. De Wagter, and G. C. de Croon, "Autonomous drone race: A computationally efficient vision-based navigation and control strategy," *Robot. Auton. Syst.*, vol. 133, 2020, Art. no. 103621.

[81] S. Li, E. van der Horst, P. Duernay, C. De Wagter, and G. C. de Croon, "Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone," *J. Field Robot.*, vol. 37, no. 4, pp. 667–692, 2020.

[82] D. Zhang and D. D. Doyle, "Gate detection using deep learning," in *Proc. IEEE Aerosp. Conf.*, 2020, pp. 1–11.

[83] R. Szeliski, *Computer Vision: Algorithms and Applications*. Berlin, Germany: Springer Nature, 2022.

[84] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 2043–2050.

[85] W. Wang, Y. Hu, and S. Scherer, "TartanVO: A generalizable learning-based VO," in *Proc. Conf. Robot. Learn.*, 2020, pp. 1761–1772.

[86] Z. Teed, L. Lipson, and J. Deng, "Deep patch visual odometry," *Adv. Neural Inf. Process. Syst.*, vol. 36, p. 39033–39051, 2024.

[87] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 8631–8638, Oct. 2021.

[88] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 4, pp. 1335–1344, Jul. 2018.

[89] C. Richter, A. Bry, and N. Roy, "Polynomial Trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*. Berlin, Germany: Springer, 2016, pp. 649–666.

[90] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time UAV replanning using guided gradient-based optimization and topological paths," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1208–1214.

[91] B. Zhou, J. Pan, F. Gao, and S. Shen, "RAPTOR: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 1992–2009, Dec. 2021.

[92] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 645–659, Jun. 2018.

[93] I. Spasojevic, V. Murali, and S. Karaman, "Perception-aware time optimal path parameterization for quadrotors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3213–3219.

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

[95] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Sci. Robot.*, vol. 6, no. 56, 2021, Art. no. eabh1221.

[96] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload," in *Proc. Robot., Sci. Syst.*, 2017, p. 1–10.

[97] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Auton. Robots*, vol. 33, pp. 69–88, Mar. 2012.

[98] K. Bousson and P. F. Machado, "4D trajectory generation and tracking for waypoint-based aerial navigation," *WSEAS Trans. Syst. Control*, no. 3, pp. 105–119, 2013.

[99] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2872–2879.

[100] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.

[101] R. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," in *Proc. AIAA Guid., Navigation, Control Conf.*, 2016, Art. no. 1374.

[102] T. Zhiling, B. Chen, R. Lan, and S. Li, "Vector field guided RRT* based on motion primitives for quadrotor kinodynamic planning," *J. Intell. Robot. Syst.*, vol. 100, pp. 1325–1339, 2020.

[103] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3340–3356, Dec. 2022.

[104] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7730–7737, Jul. 2022.

[105] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[106] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[107] S. Lavalle and J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, Wellesley, MA, USA: A K Peteres, Jan. 2000, pp. 293–308.

[108] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[109] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[110] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Proc. Edsger Wybe Dijkstra: His Life, Work, Legacy*, 1959, pp. 269–271.

[111] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 2520–2525.

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

[113] M. W. Müeller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.

[114] C. Qin, M. S. Michet, J. Chen, and H. H.-T. Liu, "Time-optimal gate-traversing planner for autonomous drone racing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.

[115] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3259–3278, Oct. 2022.

[116] T. Fork and F. Borrelli, "Euclidean and non-Euclidean trajectory optimization approaches for quadrotor racing," 2023, *arXiv:2309.07262*.

[117] W. V. Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *Proc. IEEE Eur. Control Conf.*, 2013, pp. 1788–1792.

[118] T. R. Jorris and R. G. Cobb, "Three-dimensional trajectory optimization satisfying waypoint and no-fly zone constraints," *J. Guidance, Control, Dyn.*, vol. 32, no. 2, pp. 551–572, 2009.

[119] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 5054–5061.

[120] B. Ichter, B. Landry, E. Schmerling, and M. Pavone, "Perception-aware motion planning via multiobjective search on gpus," in *Robotics Research*. Cham: Springer, 2020, pp. 895–912.

[121] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Proc. Robot., Sci. Syst.*, Corvalis, OR, USA, 2020, pp. 1–10.

[122] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1205–1212.

[123] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7209–7216, Jul. 2022.

[124] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear MPC for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 690–697, Apr. 2022.

[125] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3357–3373, Dec. 2022.

[126] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV for extreme maneuverability," *IFAC Proc. Vol.*, vol. 44, no. 1, pp. 6337–6342, 2011.

[127] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 1203–1218, May 2021.

[128] H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in *Proc. Eur. Control Conf.*, 2021, pp. 1556–1563.

[129] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear MPC for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[130] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *J. Guid., Control, Dyn.*, vol. 40, no. 2, pp. 344–357, 2017.

[131] B. Goldfain et al., "Autorally: An open platform for aggressive autonomous driving," *IEEE Control Syst. Mag.*, vol. 39, no. 1, pp. 26–55, Feb. 2019.

[132] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *Proc. IEEE 49th Conf. Decis. Control*, 2010, pp. 6137–6142.

[133] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Appl. Methods*, vol. 36, no. 5, pp. 628–647, Jul. 2014.

[134] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 4044–4050.

[135] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza, "Perception-aware path planning," 2016, *arXiv:1605.04151*.

[136] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5774–5781.

[137] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor UAV," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 6199–6206.

[138] J. Tordesillas and J. P. How, "Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1399–1406, Mar. 2023.

[139] J. Tordesillas and J. P. How, "PANTHER: Perception-aware trajectory planner in dynamic environments," *IEEE Access*, vol. 10, pp. 22662–22677, 2022.

[140] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.

[141] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and pixelMPC," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 1207–1214, Apr. 2020.

[142] M. Greeff, T. D. Barfoot, and A. P. Schoellig, "A perception-aware flatness-based model predictive controller for fast vision-based multirotor flight," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9412–9419, 2020.

[143] R. Verschueren et al., "Acados—A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, no. 1, pp. 147–183, 2022.

[144] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optim. Eng.*, vol. 13, no. 1, pp. 1–27, 2012.

[145] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[146] MOSEK ApS, "MOSEK optimization toolbox for MATLAB," *User's Guide Reference Manual*, vol. 4, no. 1, 2019.

[147] A. Giusti et al., "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robot. Automat. Lett.*, vol. 1, no. 2, pp. 661–667, Jul. 2016.

[148] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

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

[150] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Sci. Robot.*, vol. 6, no. 59, 2021, Art. no. eabg5810.

[151] F. Sadeghi and S. Levine, "CAD 2 RL: Real single-image flight without a single real image," in *Proc. Robot., Sci. Syst.*, 2017, pp. 48–55.

[152] T. Lee, S. Mckeever, and J. Courtney, "Flying free: A research overview of deep learning in drone navigation autonomy," *Drones*, vol. 5, no. 2, 2021, Art. no. 52.

[153] H. X. Pham, H. I. Ugurlu, J. Le Fevre, D. Bardakci, and E. Kayacan, "Deep learning for vision-based navigation in autonomous drone racing," in *Deep Learning for Robot Perception and Cognition*. New York, NY, USA: Elsevier, 2022, pp. 371–406.

[154] A. A. Cabrera-Ponce, L. O. Rojas-Perez, J. A. Carrasco-Ochoa, J. F. Martinez-Trinidad, and J. Martinez-Carranza, "Gate detection for micro aerial vehicles using a single shot detector," *IEEE Latin Amer. Trans.*, vol. 17, no. 12, pp. 2045–2052, Dec. 2019.

[155] H. X. Pham, I. Bozcan, A. Sarabakha, S. Haddadin, and E. Kayacan, "GateNet: An efficient deep neural network architecture for gate perception using fish-eye camera in autonomous drone racing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4176–4183.

[156] H. X. Pham, A. Sarabakha, M. Odnoshyvkin, and E. Kayacan, "PencilNet: Zero-shot sim-to-real transfer learning for robust gate perception in autonomous drone racing," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11847–11854, Oct. 2022.

[157] T. Morales, A. Sarabakha, and E. Kayacan, "Image generation for efficient neural network training in autonomous drone racing," in *Proc. Int. Joint Conf. Neural Netw.*, 2020, pp. 1–8.

[158] K. F. Andersen, H. X. Pham, H. I. Ugurlu, and E. Kayacan, "Event-based navigation for autonomous drone racing with sparse gated recurrent network," in *Proc. Eur. Control Conf.*, 2022, pp. 1342–1348.
[159] N. J. Sanket, C. D. Singh, C. Fermüller, and Y. Aloimonos, "PRGFlow: Unified swap-aware deep global optical flow for aerial robot navigation," *Electron. Lett.*, vol. 57, no. 16, pp. 614–617, 2021.
[160] Y. Xu and G. C. de Croon, "CNN-based ego-motion estimation for fast MAV maneuvers," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 7606–7612.
[161] Y. Xu and G. C. de Croon, "CUAHN-VIO: Content-and-uncertainty-aware homography network for visual-inertial odometry," 2022, *arXiv:2208.13935*.
[162] L. Lamberti et al., "A sim-to-real deep learning-based framework for autonomous nano-drone racing," *IEEE Robot. Automat. Lett.*, vol. 9, no. 2, pp. 1899–1906, Feb. 2024.
[163] H. Yu, C. De Wagter, and G. C. de Croon, "MAVRL: Learn to fly in cluttered environments with varying speed," 2024, *arXiv:2402.08381*.
[164] M. Kulkarni and K. Alexis, "Reinforcement learning for collision-free flight exploiting deep collision encoding," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.
[165] K. Amer, M. Samy, M. Shaker, and M. ElHelw, "Deep convolutional neural network based autonomous drone navigation," in *Proc. 13th Int. Conf. Mach. Vis.*, 2021, vol. 11605, pp. 16–24.
[166] R. Madaan et al., "Airsim drone racing lab," in *Proc. NeurIPS Competition Demonstration Track*, 2020, vol. 123, pp. 177–191.
[167] U. Ates, "Long-term planning with deep reinforcement learning on autonomous drones," in *Proc. Innov. Intell. Syst. Appl. Conf.*, 2020, pp. 1–6.
[168] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for UAV attitude control," *ACM Trans. Cyber-Phys. Syst.*, vol. 3, no. 2, pp. 1–21, 2019.
[169] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4224–4230, Oct. 2019.
[170] S. Li, E. Öztürk, C. De Wagter, G. C. De Croon, and D. Izzo, "Aggressive online control of a quadrotor via deep network representations of optimality principles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 6282–6287.
[171] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: Study on landing problems," *J. Guid., Control, Dyn.*, vol. 41, no. 5, pp. 1122–1135, 2018.
[172] R. Ferede, G. de Croon, C. De Wagter, and D. Izzo, "End-to-end neural network based optimal quadcopter control," *Robot. Auton. Syst.*, vol. 172, 2024, Art. no. 104588.
[173] J. Sacks, R. Rana, K. Huang, A. Spitzer, G. Shi, and B. Boots, "Deep model predictive optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.
[174] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Proc. Robot., Sci. Syst., Virtual*, 2021, pp. 1–12.
[175] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson, "Safe reinforcement learning using black-box reachability analysis," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 10665–10672, Oct. 2022.
[176] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," *Adv. Neural Inf. Process. Syst.*, vol. 31, pp. 8299–8310, 2018.
[177] L. Pineda et al., "Theseus: A library for differentiable nonlinear optimization," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 3801–3818, 2022.
[178] C. Wang et al., "PyPose: A library for robot learning with physics-based optimization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22024–22034.
[179] S. Cheng, L. Song, M. Kim, S. Wang, and N. Hovakimyan, "DiffTune$^+$: Hyperparameter-free auto-tuning using auto-differentiation," in *Proc. 5th Annu. Learn. Dyn. Control Conf.*, 2023, vol. 211, pp. 170–183.
[180] P. Karkus, B. Ivanovic, S. Mannor, and M. Pavone, "DiffStack: A differentiable and modular control stack for autonomous vehicles," in *Proc. Conf. Robot Learn.*, 2023, pp. 2170–2180.
[181] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.
[182] G. Li, M. Mueller, V. M. Casser, N. Smith, D. Michels, and B. Ghanem, "Oil: Observational imitation learning," in *Proc. Robot., Sci. Syst.*, Freiburg im Breisgau, Germany, 2019, pp. 1–10.
[183] M. Muller, G. Li, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Learning a controller fusion network by online trajectory filtering for vision-based UAV racing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 573–581.
[184] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4CV: A photo-realistic simulator for computer vision applications," *Int. J. Comput. Vis.*, vol. 126, no. 9, pp. 902–919, 2018.
[185] M. Muller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem, "Teaching UAVs to race: End-to-end regression of agile controls in simulation," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 11–29.
[186] L. O. Rojas-Perez and J. Martinez-Carranza, "DeepPilot: A CNN for autonomous drone racing," *Sensors*, vol. 20, no. 16, 2020, Art. no. 4524.
[187] J. Fu, Y. Song, Y. Wu, F. Yu, and D. Scaramuzza, "Learning deep sensorimotor policies for vision-based autonomous drone racing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 5243–5250.
[188] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza, "Contrastive learning for enhancing robust scene transfer in vision-based agile flight," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.
[189] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg, "Daydreamer: World models for physical robot learning," in *Proc. Conf. Robot Learn.*, 2023, pp. 2226–2240.
[190] L. Smith, I. Kostrikov, and S. Levine, "Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," in *Proc. Robot., Sci. Syst. XIX*, 2023, pp. 1–9.
[191] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 1593–1599.
[192] A. Loquercio, A. Kumar, and J. Malik, "Learning visual locomotion with cross-modal supervision," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7295–7302.
[193] P. A. Ioannou and J. Sun, *Robust Adaptive Control*, Ed., Mineola, New York, USA: Dover Publications, 2012.
[194] K. J. Åström and B. Wittenmark, *Adaptive Control*, Ed., Mineola, New York, USA: Dover Publications, 2013.
[195] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and Adaptive Control*. Berlin, Germany: Springer, 2013, pp. 1–449.
[196] S. M. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, "Adaptive-control-Oriented meta-learning for nonlinear systems," in *Proc. Robot., Sci. Syst.*, 2021, pp. 1–12.
[197] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, "Learning a single near-hover position controller for vastly different quadcopters," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1263–1269.
[198] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly–A gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7512–7519.
[199] M. Kulkarni, T. J. Forgaard, and K. Alexis, "Aerial gym–Isaac gym simulator for aerial robots," 2023, *arXiv:2305.16510*.
[200] L. Martin, "Alphapilot AI drone innovation challenge," Jan. 2020. [Online]. Available: https://lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html
[201] C. de Wagter, F. Paredes-Vallés, N. Sheth, and G. C. de Croon, "Learning fast in autonomous drone racing," *Nat. Mach. Intell.*, vol. 3, 2021, Art. no. 923.
[202] E. Ackerman, "Autonomous drones challenge human champions in first 'fair' race," Jul. 2022. Accessed: May 22, 2024, [Online]. Available: https://spectrum.ieee.org/zurich-autonomous-drone-race
[203] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2114–2130, Aug. 2022.
[204] A. Loquercio, A. Saviolo, and D. Scaramuzza, "Autotune: Controller tuning for high-speed flight," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4432–4439, Apr. 2022.
[205] A. Antonini, W. Guerra, V. Murali, T. Sayre-McCord, and S. Karaman, "The blackbird dataset: A large-scale dataset for UAV perception in aggressive flight," in *Proc. Int. Symp. Exp. Robot.*, 2018, pp. 130–139.
[206] C. Pfeiffer and D. Scaramuzza, "Human-piloted drone racing: Visual processing and control," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 3467–3474, Apr. 2021.
[207] M. Bosello et al., "Race against the machine: A fully-annotated, open-design dataset of autonomous and piloted high-speed flight," *IEEE Robot. Automat. Lett.*, vol. 9, no. 4, pp. 3799–3806, Apr. 2024.
[208] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-Planner: An ESDF-free gradient-based local planner for quadrotors," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 478–485, Apr. 2021.
[209] J. Tordesillas and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 922–938, Apr. 2022.

[210] V. Kumar and N. Michael, "Opportunities and challenges with autonomous micro aerial vehicles," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1279–1291, 2012.

[211] T. Baca et al., "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," *J. Intell. Robot. Syst.*, vol. 102, no. 1, Apr. 2021, Art. no. 26.

[212] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Auton. Robots*, vol. 33, no. 1, pp. 21–39, Aug. 2012.

[213] L. Bauersfeld, L. Spannagl, G. Ducard, and C. Onder, "MPC flight control for a tilt-rotor VTOL aircraft," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 57, no. 4, pp. 2395–2409, Aug. 2021.

[214] "FPV wing racing association". [Online]. Accessed: May 22, 2024, Available: https://wrl-uk.com/

[215] E. Tal and S. Karaman, "Global incremental flight control for agile maneuvering of a tailsitter flying wing," *J. Guid., Control, Dyn.*, vol. 45, no. 12, pp. 2332–2349, 2022.

[216] E. Tal, G. Ryou, and S. Karaman, "Aerobatic trajectory generation for a VTOL fixed-wing aircraft using differential flatness," *IEEE Trans. Robot.*, vol. 39, no. 6, pp. 4805–4819, Dec. 2023.

[217] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided direct sparse odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5781–5790.

[218] Z. Teed and J. Deng, "DROID-SLAM: Deep visual slam for monocular, stereo, and RGB-D cameras," *Adv. Neural Inf. Process. Syst.*, vol. 34, pp. 16558–16569, 2021.

[219] C. Pfeiffer, S. Wengeler, A. Loquercio, and D. Scaramuzza, "Visual attention prediction improves performance of autonomous drone racing agents," *PLoS One*, vol. 17, no. 3, 2022, Art. no. e0264471.

[220] G. C. de Croon, J. J. Dupeyroux, C. De Wagter, A. Chatterjee, D. A. Olejnik, and F. Ruffier, "Accommodating unobservability to control flight attitude with optic flow," *Nature*, vol. 610, no. 7932, pp. 485–490, 2022.

[221] A. Bajcsy, A. Loquercio, A. Kumar, and J. Malik, "Learning vision-based pursuit-evasion robot policies," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024.

[222] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager, "A real-time game theoretic planner for autonomous two-player drone racing," in *Proc. Robot., Sci. Syst.*, 2018, pp. 1–9.

[223] Z. Wang, T. Taubner, and M. Schwager, "Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3D environments," *Robot. Auton. Syst.*, vol. 125, 2020, Art. no. 103410.

[224] J. Chen, K. Su, and S. Shen, "Real-time safe trajectory generation for quadrotor flight in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, 2015, pp. 1678–1685.

[225] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3681–3688.

[226] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 344–351.

[227] Y.-L. Chow, M. Pavone, B. M. Sadler, and S. Carpin, "Trading safety versus performance: Rapid deployment of robotic swarms with robust performance constraints," *J. Dyn. Syst., Meas., Control*, vol. 137, no. 3, 2015, Art. no.031005.

[228] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5883–5890.

[229] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: An SOS optimization approach," in *Proc. 13th Workshop Algorithmic Found. Robot.*, 2020, pp. 545–564.

[230] R. Luo et al., "Sample-efficient safety assurances using conformal prediction," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2022, pp. 149–169.

[231] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Auton. Robots*, vol. 39, pp. 555–571, 2015.

[232] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. IEEE 18th Eur. Control Conf.*, 2019, pp. 3420–3431.

[233] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-Jacobi reachability: A brief overview and recent advances," in *Proc. IEEE 56th Annu. Conf. Decis. Control*, 2017, pp. 2242–2253.

[234] X. Wang, K. Leung, and M. Pavone, "Infusing reachability-based safety into planning and control for multi-agent interactions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6252–6259.

[235] K. Leung et al., "On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions," *Int. J. Robot. Res.*, vol. 39, no. 10-11, pp. 1326–1345, 2020.

[236] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-predictive motion planning in unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 8552–8558.

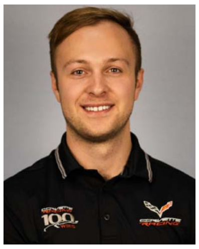
[237] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic model predictive safety certification for learning-based control," *IEEE Trans. Autom. Control*, vol. 67, no. 1, pp. 176–188, Jan. 2022.

[238] L. Bauersfeld, E. Kaufmann, and D. Scaramuzza, "User-conditioned neural control policies for mobile robotics," in *Proc. Int. Conf. Robot. Automat.*, 2023, pp. 1342–1348.

[239] T. Yu et al., "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proc. Conf. Robot Learn.*, 2019, pp. 1094–1100.

[240] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal, "Safe reinforcement learning via curriculum induction," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 12151–12162, 2020.

[241] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *J. Artif. Intell. Res.*, vol. 75, pp. 1401–1476, 2022.



Drew Hanover received the bachelor's degree in mechanical engineering from Michigan Technological University, Houghton, MI, USA, in 2018 and the master's degree in robotics from the University of Michigan, Ann Arbor, MI, in 2022.

He has worked with NASA, General Motors, and Pratt and Miller Engineering across a multitude of engineering domains. He is currently the Chief Technology Officer and the Founder of Innovire AG, Zürich, Switzerland.

Antonio Loquercio received the M.Sc. degree in robotics from ETH Zürich, Zürich, Switzerland, in 2017, and the Ph.D. degree in robotics from the University of Zürich, Zürich, in 2021.

He worked with the Berkeley Artificial Intelligence Research Lab, UC Berkeley, Berkeley, CA, USA, from 2022 to 2024. He is currently a Professor of electrical engineering and computer science with the University of Pennsylvania, Philadelphia, PA, USA.

Leonard Bauersfeld (Graduate Student Member, IEEE) received the M.Sc. degree in robotics, system and control from ETH Zürich, Zürich, Switzerland in 2020. He is currently working toward the Ph.D. degree with the Robotics and Perception Group, University of Zürich, under the supervision of Prof. D. Scaramuzza.

He works on novel approaches, combining first-principles methods with modern data-driven models to advance agile quadrotor flight. His research interests include autonomous vision-based quadrotor flight and quadrotor simulations.

Angel Romero (Graduate Student Member, IEEE) received the B.Sc. degree in electronics engineering from the University of Malaga, Málaga, Spain, in 2015, and the M.Sc. degree in robotics, systems and control from ETH Zürich, Zürich, Switzerland, in 2018. He is currently working toward the Ph.D. degree with the Robotics and Perception Group, University of Zürich, Zürich, under the supervision of Prof. D. Scaramuzza.

His research interests include finding new limits in the intersection of machine learning, optimal control, and computer vision applied to super agile autonomous quadrotor flight.

Robert Penicka received the Ph.D. degree in artificial intelligence from the Czech Technical University (CTU) in Prague, Prague, Czech Republic, in 2020.

He was a Postdoctoral Researcher with the University of Zürich between 2020 and 2022 under the supervision of Prof. D. Scaramuzza. Since 2022, he has been a Research Fellow with CTU, focusing on high-level mission planning, trajectory planning, and control for unmanned aerial vehicles. He is also currently a Postdoctoral Fellow with the Multi-Robot Systems (MRS) group, CTU. He bridged the gap between mission planning and trajectory planning, particularly in cluttered environments.

Dr. Penicka was the recipient of the Dean's Prize and second place in the Werner von Siemens Award for Industry 4.0. He was also the recipient of the Joseph Fourier Prize and the Antonin Svoboda Award for his doctoral thesis.

Yunlong Song received the M.Sc. degree in information and communication engineering from Technical University of Darmstadt, Darmstadt, Germany, in 2018. He is currently working toward the Ph.D. degree in robotics with the Robotics and Perception Group, University of Zürich, Zürich, Switzerland, under the supervision of Prof. D. Scaramuzza.

His research interests include reinforcement learning, machine learning, and robotics.

Giovanni Cioffi (Student Member, IEEE) received the M.Sc. degree in mechanical engineering from ETH Zürich, Zürich, Switzerland, in 2019. He is currently working toward the Ph.D. degree in robotics with the University of Zürich, Zürich, under the supervision of Prof. D. Scaramuzza.

His research interests include the intersection of computer vision and robotics, exploring topics such as visual(-inertial) odometry and simultaneous localization and mapping (SLAM).

Dr. Cioffi was the recipient of multiple awards in top-tier robotic conferences and journals, such as the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2023 Best Paper Award and the RA-L 2021 Best Paper Award.

Elia Kaufmann (Member, IEEE) received the B.Sc. degree in mechanical engineering in 2014, the M.Sc. degree in robotics, systems, and control from ETH Zürich, Zürich, Switzerland, in 2017, and the Ph.D. degree in informatics from the Institute for Informatics, University of Zürich, Zürich, in 2022, under the supervision of Prof. D. Scaramuzza.

His doctoral research focused on advancing the application of machine learning techniques to enhance perception and control of autonomous aerial vehicles. He is currently a Senior Autonomy Engineer with Skydio, San Mateo, California, USA.

Davide Scaramuzza (Senior Member, IEEE) received the Ph.D. degree in robotics from ETH Zürich, Zürich, Switzerland, in 2008.

He was a Postdoctoral Fellow with the University of Pennsylvania, and was a Visiting Professor with Stanford University. His research focuses on autonomous, agile microdrone navigation using standard and event-based cameras. He pioneered autonomous, vision-based navigation of drones, which inspired the navigation algorithm of the NASA Mars helicopter and many drone companies. He contributed significantly to visual-inertial state estimation, vision-based agile navigation of microdrones, and low-latency, robust perception with event cameras, which were transferred to many products, from drones to automobiles, cameras, AR/VR headsets, and mobile devices. In 2015, he cofounded Zürich-Eye, today Meta Zürich, which developed the world-leading virtual-reality headset Meta Quest. In 2020, he cofounded SUIND, which builds autonomous drones for precision agriculture. In 2022, his team demonstrated that an artificial intelligence (AI)-controlled, vision-based drone could outperform the world champions of drone racing, a result that was published in *Nature*. Many aspects of his research have been featured in the media, such as The New York Times, The Economist, and Forbes. He is a Consultant for the United Nations on disaster response, AI for good, and disarmament. He is currently also a Professor of robotics and perception with the University of Zürich, Zürich.

Dr. Scaramuzza was the recipient of many awards, including an IEEE Technical Field Award, the IEEE Robotics and Automation Society Early Career Award, a European Research Council Consolidator Grant, a Google Research Award, two NASA TechBrief Awards, and many paper awards.