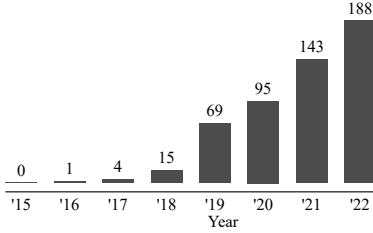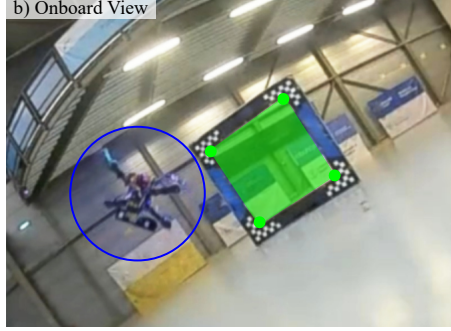# Autonomous Drone Racing: A Survey

Drew Hanover[1], Antonio Loquercio[3], Leonard Bauersfeld[1], Angel Romero[1], Robert Penicka[2],
Yunlong Song[1], Giovanni Cioffi[1], Elia Kaufmann[1] and Davide Scaramuzza[1]

Fig. 1: Drone racing is a sport rapidly gaining popularity where opponents compete on a pre-defined race track consisting of a series of gates. Autonomous drone racing research aims to build algorithms that can outperform human pilots in such competitions. a) The task of autonomous drone racing 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. 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.

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 which can operate drones in complex and uncertain environments. Additionally, flying fast enables drones to cover more ground which in turn increases productivity and further strengthens their use case. One proxy for developing algorithms used in high-speed navigation is the task of autonomous drone racing, 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 kph 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 autonomous drone racing across model-based and learning-based approaches. 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.

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

[1]D. Hanover, L. Bauersfeld, A. Romero, Y. Song, G. Cioffi, E. Kaufmann and D. Scaramuzza are with the Robotics and Perception Group, University of Zurich, Switzerland (http://rpg.ifi.uzh.ch). [2]R. Penicka is with the Multi-robot Systems Group, Czech Technical University in Prague, Czech Republic. [3]A. Loquercio is with UC Berkeley. This work was supported by the Swiss National Science Foundation (SNSF) through the National Centre of Competence in Research (NCCR) Robotics, the Czech Science Foundation (GACR) under research projects No. 23-06162M, the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 871479 (AERIAL-CORE), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

back to 3000 BC in ancient Egypt where the Pharaoh was thought to have ran 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 a human can achieve.

A. Why Drone Racing?

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. Human drone pilots need years of training to master the advanced navigation and control skills that are required to be successful in international competitions. Such skills would also be valuable to autonomous systems that must quickly and safely fly through complex environments, in applications such as disaster response, aerial delivery, and inspection of complex structures. For example, in search-and-rescue scenarios, drones must be able to rapidly navigate in complex environments in order to maximize their spatial coverage. Put more simply, drones that can fly fast, fly farther [4].

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 [5], most drones used for inspection tasks rely on

GPS navigation with the biggest limiting factor on inspection efficiency being the drones endurance and mobility. Additionally, the authors note that the most popular drones used for surveying by several US Departments of Transportation are not fully autonomous and require expert human pilots [5]. The commercial and safety advantages of highly agile drone systems is clear, however research into autonomous drone racing can also help us gain new understandings on how the visual processing and control by human pilots works, as demonstrated in [6].

Over the last five years, several projects have been launched to encourage rapid progress within the field, such as DARPA's Fast Lightweight Autonomy (FLA) [8] and European Research Council's AgileFlight [9]. With funding pools of over $1 million for each of these projects and significant commercial potential, a large incentive exists for researchers and entrepreneurs alike to explore new paradigms in agile flight research. Competitions such as the IROS'16-19 Autonomous Drone Racing series [10], NeurIPS 2019's Game of Drones [11], and the 2019 AlphaPilot Challenge [12], [13] 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.

Drone racing is a challenging benchmark which 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 algorithms to be efficient, lightweight, and provide optimal decision and control behaviors all in real-time. Additionally, we see nearly exponential growth of the number of papers in the field year over year as seen in Fig 1.

To the best of the authors' knowledge, this is the first survey on the state of the art in autonomous drone racing. 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 100kph with only a first-person view 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-dimensional 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 or losing the race, which can be decided by less than a tenth of a second.

The paper is structured as follows: First, the modeling procedure of the drone including aerodynamics, batteries, motors, cameras, and the system nonlinearities are discussed in detail in Sect. II. A classical robotics pipeline is then introduced in Sect. III, with a deep dive into literature relevant to agile flight split into Perception, Planning, and Control subsections. Afterwards, we delve into learning-based methods for Perception, Planning, and Control which rely on recent advancements from the machine learning community in Sect. IV. Then, a discussion of the development of simulation tools which can enable rapid development for agile flight applications in Sect. V. A history of drone racing competitions and the methods used for each are included in Sect. VI. Next, a summary of open source code bases, hardware platforms, and datasets for researchers is provided in Sect. VII. Finally, a forward-looking discussion on the Opportunities and Challenges for future researchers interested in autonomous drone racing in Sect. VIII.

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 autonomous drone racing models are introduced by the onboard sensors, whose characteristics need to be modeled. For example, 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 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:

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

where $\boldsymbol{p}_{WB} \in \mathbb{R}^3$ is the position of the center of mass given in the world frame, $\boldsymbol{q}_{WB} \in SO(3)$ is a quaternion defining

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
Lockheed Martin sponsored a $1M prize 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 – a significant jump over previous competitions.

2016	2018	2022
$v_{max} = 0.6$ m/s $\quad$ $v_{max} = 0.7$ m/s	$v_{max} = 2.0$ m/s $\quad$ $v_{max} = 10$ m/s	$v_{max} = 22$ m/s
2017	2020	

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.
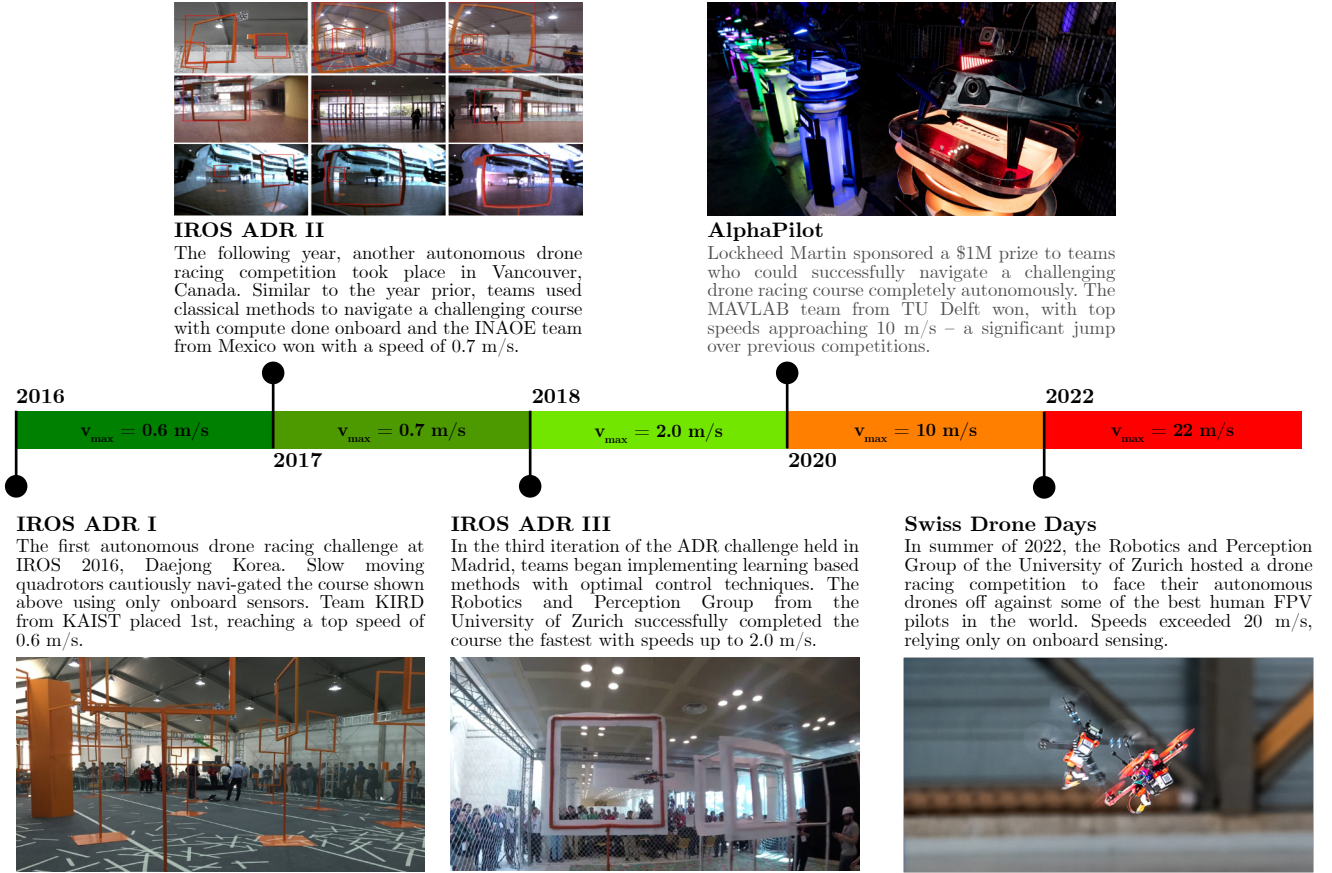
Fig. 2: History of drone racing competitions that use real drones for navigating the race track, IROS ADR II photo credit [7].

the rotation of the body frame relative to the world (vehicle attitude), $v_W \in \mathbb{R}^3$ is the velocity of the vehicle in the world frame, $\omega_B \in \mathbb{R}^3$ are the bodyrates of the vehicle, $\Omega \in \mathbb{R}^4$ are the motor speeds, $g_W = [0, 0, -9.81 \, \text{m/s}^2]^\mathsf{T}$ denotes earth's gravity, and $u \in \mathbb{R}^4$ 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 f and total torque τ 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 $q \odot f = q \cdot [0, f^\mathsf{T}]^\mathsf{T} \cdot \bar{q}$, where $\bar{q}$ 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 [14]–[18] and that the body drag is negligible. These assumptions quickly break down at the high speeds encountered in drone racing as this model neglects (a) linear rotor drag [19], [20], (b) dynamic lift [19], (c) rotor-to-rotor [21]–[23], (d) rotor-to-body [21]–[23] interactions and (e) aerodynamic body drag [20], [22].

The accuracy of the propeller model can be improved by leveraging blade-element-momentum theory, where the propeller is modeled as a rotating wing. Such first-principle approaches [24]–[28] have been shown to provide very accurate models of the wrench generated by a single propeller as they properly capture effects (a) and (b). Implemented efficiently, a Blade Element Momentum (BEM) model can be run in real-time [29] and has been successfully used to test algorithms in simulation [30], [31].

Accounting for the remaining open points (c)-(e), 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 [21]–[23], [32], [33]. 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 [29], [34]–[38]. In the early works [36], [37], the whole vehicle dynamics model was learned from data. In a similar fashion [35] uses a combination of polynomials—identified from wind-tunnel flight data—to represent the vehicle dynamics. In [29], [34], it has been shown that higher modeling accuracies can be achieved when combining a first-principle model with a data-driven component. Such combination of first-principle and data-driven models also leads to improved generalization performance, as shown in [29], which combines a BEM model with a temporal convolutional

network [39] to regress the residual wrench. With this, a high-fidelity model is obtained [29] that is still accurate at the speeds and accelerations encountered in drone racing.

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 controlled by a 'throttle' command which controls the duty cycle of a PWM (pulse-width-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 [4]. 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 [40], but for lithium-polymer batteries in drone racing this is hardly applicable because the battery discharge current often exceeds $100\,\mathrm{A}$ (e.g. 50-100 C) [41], [42]. Graybox battery models for the voltage that are based on a one-time-constant (OTC) equivalent circuit [43], [44] are much more suitable for drone racing tasks as shown in [4], 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 [4]. 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 [45] 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 bodyrates 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 [46]. The most common sensors aboard autonomous drones are monocular or stereo cameras combined with IMUs (inertial measurement units) thanks to their low cost, low weight, and mechanical robustness. Typically, the algorithms processing the visual information assume a pinhole camera [47] mounted in the center of gravity of the vehicle. In reality, the camera is usually never mounted directly at the center of gravity of the vehicle, meaning that a transformation between the camera frame and the body frame of the drone is required.

The camera and IMU must be calibrated before using their measurements. The camera intrinsic calibration uses the assumption of a pinhole model [47] and estimates the focal length, image center and distortion parameters. 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 [48] is a widespread tool to perform these calibrations.

The biggest source of measurement error of the sensors onboard a drone are 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 which dampens the mount of the camera and the IMU with respect to the vehicle frame.

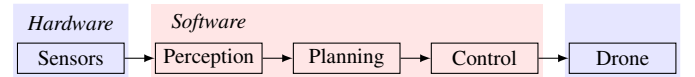## III. Classical Perception, Planning, and Control Pipeline

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

Since the inception of the field of mobile robotics, a common architecture has been primarily used to achieve autonomous navigation capabilities across a wide variety of 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. In this section, we cover recent research in each of these areas relating specifically to agile flight and autonomous drone racing. All of the 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. Most of the time, a layout or map of the race track is known apriori, and thus this section focuses exclusively on VIO methods. 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. few seconds, accumulates large drift due to scale factor errors, axis misalignment errors, and time-varying biases [49] that commonly affect off-the-self IMU measurements. The camera measurements provide rich information about the environment at a lower rate, usually around $30\,\mathrm{Hz}$, than IMU measurements. Differently from the IMU measurements, the camera measurements are affected by environmental conditions. The quality of

information that 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 [50]. 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 aerial robotic vehicles. Second, we give an overview of recent VIO algorithms that include drone dynamics in the estimation process. Third, we conclude with a discussion on the application of classical VIO methods to drone racing tasks.

1) VIO: VIO is the most common solution for state estimation of aerial vehicles [50] using only onboard sensing and computing thanks to its favorable trade-off between accuracy and computational requirements. VIO algorithms are usually composed of 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 and feature-based methods. Direct methods [51], [52] 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 [51]. On the contrary, feature-based methods [53]–[55] 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. Feature-based methods are more mature and robust than direct methods; however, the latter achieve higher reliability in low-texture environments. Hybrid methods, which combine keypoints and patches of raw pixel intensities to estimate the camera motion, also exist [56]. We refer the reader to the work in [47] for a tutorial on the VIO frontend.

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 and fixed-lag smoothing methods. Filtering methods are based on an Extended Kalman Filter (EKF). These methods propagate the state of the system using the inertial measurements and fuse the camera measurements in the update step. The pioneer filter-based VIO algorithm is the Multi-State Constraint Kalman Filter (MSCKF) originally proposed in [53]. Since then, many different versions of MSCKF have been developed [57]. Fixed-lag smoothing methods [54], [55], also referred to as sliding window estimators, solve a non-linear 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 marginalization residuals. Fixed-lag smoothing methods accumulate less linearization error than filtering methods but are more computationally demanding. We refer the reader to the work in [58] for a tutorial on the VIO backend.

Recent works [59], [60] have proposed to include event cameras [61] in VIO for flying vehicles. Event cameras do not capture images at a fixed rate but they asynchronously measure per-pixel brightness changes. The output of these cameras is a stream of events that consists of time, location,

and the sign of brightness change. Their main properties are low latency, high temporal resolution (in the order of μs), and high dynamic range (140 dB compared to 60 dB of standard cameras). Thanks to these properties, event cameras are a great complementary sensor to standard cameras. Including event data in VIO algorithms achieves increased robustness against motion blur as demonstrated in [59], [60]. UltimateSLAM [59] is a VIO algorithm that combines both standard and event cameras in a fixed-lag smoother-based VIO algorithm. In [60], a revised version of UltimeSLAM was proposed to demonstrate autonomous quadrotor flight despite rotor failure with onboard sensors.

2) Drone dynamics in VIO: The drone dynamics are used to define additional error terms in the VIO backend in [62], [63]. In [62], the authors propose VIMO which is a VIO algorithm that includes error terms on the drone transitional dynamics in a fixed-lag smoother-based backend [55]. These error terms on the drone dynamics are derived through the preintegration theory [64]. In addition to the drone state, VIMO is able to estimate the external force acting on the drone platform. The external force is modeled as a random variable distributed according to a zero-mean gaussian distribution. This choice allows the estimation of impulse-like forces acting on the drone platform.

The work in [63] extends VIMO by introducing a different model of the external force. The external force is modeled as a gaussian distribution whose mean value is equal to the difference between the commanded collective thrust and the force, up to the mass, detected by the accelerometer. This is a suitable design choice in the case when the external force affects the drone platform for long periods of time.

3) Discussions: The work in [65] presents a benchmark comparison between a number of VIO solutions on the EuRoC dataset [66]. 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 [65] 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 are not able to 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 [67]. These scenarios are the norm in drone racing.

To help the research in VIO algorithms for drone racing tasks, the work in [68] proposes 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 that are similar to the ones present in drone racing competitions.

Successful state estimation solutions for drone racing [67], [69] reduce the drift accumulated in VIO by localizing to a prior map of the track. This is a viable solution when 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. In [70], it was proposed a gate detector that uses an RGB

camera to identify the gates based on their color. All the other gate detection methods existing in the literature are based on deep learning techniques [71]. We review them in Sec. 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) [72]. This relative pose is used to constrain the VIO backend 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. Other approaches used in early drone racing competitions relied on the technique of visual servoing via stereo cameras [7], but this solution was found to be sensitive to indoor lighting changes and needed to be hand-tuned for every flight.

Recent works [73]–[75] proposed vision-based odometry algorithms that are learned end-to-end. In theory, 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 Sec. IV for a detailed review of VIO methods based on deep learning.

B. Planning

Once a state estimate $\hat{x}$ has been obtained from the perception module, the next step in the classical pipeline is to plan a feasible, time-optimal trajectory $\tau_{ref} = (x_{ref}, u_{ref})_k$, $\forall k \in 0 \ldots N$, 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.

Trajectory planning has matured over the last decade from works mostly verified in simulation to works shown in both controlled lab environments and unknown unstructured environments. Here, we categorize these methods in *polynomial and spline trajectory* planning, *optimization-based* planning, *search-based* planning, and *sampling-based* planning.

1) Polynomial and Spline Trajectories: The Polynomial and Spline methods leverage the differential flatness property [76], [77] 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 3D 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 [76], [77] minimize snap (4th order position derivative) of a trajectory. Different methods opted for minimizing jerk (3rd order position derivative) for planning a trajectory [78]. However, the trajectories that result from having jerk as the primary objective have been shown to minimize the aggressiveness of the control inputs [78], which is fundamentally different from minimizing the lap

times, where extremely aggressive trajectories are generally required. Richter et al. [79], 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 [80] proposed a polynomial-based trajectory planning method for drone racing. It jointly optimizes control effort, regularized time, and penalizes the dynamic feasibility and collisions.

Because of their numerical stability, other methods make use of B-splines for representing trajectories [81], [82] instead of high order polynomial representations that are numerically sensitive. These methods jointly optimize different objectives, simultaneously smoothness, dynamic feasibility, collision avoidance, safety [82] and vision-based target tracking [83].

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.

Yet, a time-optimal trajectory is not smooth, but rather attempts to keep the acceleration to the possible maximum at all times [84]. This means that our planned actuation needs to be able to maintain a certain value during extended periods of time, while also being able to have quick changes. Therefore polynomials cannot represent true time-optimal trajectories.

2) Optimization-based: 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 [85], simplified quadrotor models [86], [87], and full-state quadrotor models [84], [88].

Time-optimality of a trajectory could also be accomplished by using a specific path parameterization that maximizes velocity over a given path [89]. This method was shown for quadrotors in [90] for minimizing time of flight considering both translational and rotational quadrotor dynamics. However, the method only creates a velocity profile over a given path which is not further optimized.

Apart from time optimality, complying with intermediate waypoint constraints is another requirement for path planning in autonomous drone racing. 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. [91], [92]). The time allocation of the waypoints is, however, non-trivial 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 [84] introduces a complementary progress constraints (CPC) 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 above methods create

time-optimal trajectories passing through given gates, they are computationally costly and hence intractable in real-time.

3) Search-based: Search-based planning methods [93], [94] 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 increasingly computationally demanding for increasing complexity of the quadrotor model. Furthermore, the employed per-axis dynamic limits (velocity, acceleration, jerk) does not represent the true quadrotor model, which further decreases 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 multi-waypoint drone racing.

4) Sampling-based: Sampling-based methods like RRT* [95] can be used for planning trajectories for linearized quadrotor models. Several time-minimizing approaches [67], [96] use a point-mass model for high-level time-optimal trajectory planning. In [96], 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. 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 sub-optimal given a minimum time objective. In [97], the authors 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 (i.e. bang-bang) if required.

The first method for planning minimum-time trajectories in a cluttered environment for the full quadrotor model was proposed in [31]. 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, the authors of [98] 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 [99] 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.

5) 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
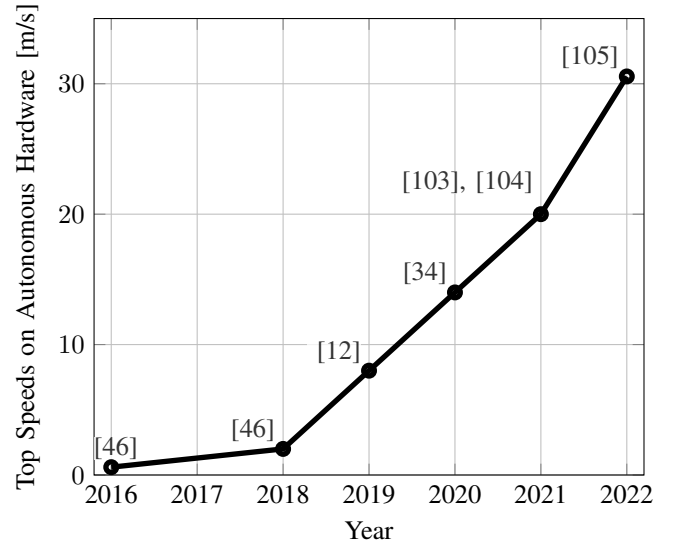

Fig. 4: Top speeds demonstrated on autonomous drones over time from both literature and competition data

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 [100]–[102].

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.

This can be tackled with more complex control approaches that do some part of the replanning online [98].

C. Control

Over the last decade, significant advancements have been made in agile multicopters control. Every year, increasing top speeds are demonstrated in the literature as shown in Figure 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 [30]. 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 com-

---

[1] https://pixhawk.org/
[2] https://github.com/betaflight/betaflight

munity. BetaFlight is the most commonly used low-level controller for agile drone flight and has been widely adopted by the First Person View (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 which 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 [106], 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 [77]. In this work, many agile maneuvers are performed onboard real drones with speeds up to 2.6 m/s.

The previous work is extended in [20], 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 lead to demonstrated flight speeds up to 4.0 m/s while reducing tracking error by 50% onboard a real drone.

The differential flatness method is further extended by in [107] by cascading an Incremental Nonlinear Dynamic Inversion (INDI) controller with the differential flatness controller described in [77] but neglects the aerodynamic model addition from [20]. The INDI controller is designed to track the angular acceleration commands $\dot{\Omega}$ from the given reference trajectory. Top speeds of nearly 13 m/s and accelerations over 2g 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 [104]. Cascaded controllers of INDI-NMPC and INDI-differential flatness are shown to track aggressive racing trajectories which achieving speeds of around 20m/s and accelerations of over 4g. 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 MPC methods applied to micro aerial vehicles can be found in [108]. 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 [109]. 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 [110], [111]. An introduction to MPPI can be found in https://autorally.github.io/. The MPPI approach can be used on agile quadrotors to navigate complex forest environments,

however, analysis was only performed in simulation [110]. Most of the successful demonstrations of MPPI come from ground robots [110], [111]. Because MPPI is a sampling based algorithm, scaling to higher-dimension state spaces like those of a quadrotor can lead to performance issues as shown in [103].

Nonlinear MPC methods are also used in [34] 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 Sec. 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 [107], authors in [103] question whether or not it is necessary to explicitly model the additional aerodynamic terms from [34] due to the added computational and modeling complexity. Instead, they propose to learn residual model dynamics online using a cascaded adaptive nonlinear model predictive control architecture. Aggressive flight approaching 20m/s and over 4g acceleration is demonstrated on real racing quadrotors. Additionally, 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 onboard a racing quadrotor [34], [103].

Contouring control methods can deal with competing optimization goals such as trajectory tracking accuracy and minimum flight times [112]. These methods minimize a cost function which makes trade-offs between these competing objectives. In [113], Nonlinear Model Predictive Contouring Control (MPCC) is applied to control small model racecars. MPCC was then extended to agile quadrotor flight in [99]. Although the velocities achieved by the MPCC controller were lower than that of [103], [104], 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 re-plan online [98] while also controlling near the limit of the flight system. Similar work uses tunneling constraints in the MPCC formulation in [114],

2) Perception Aware Model Predictive Control: When control is coupled with perception, an optimization problem that constrains or penalizes the movement of the drone to ensure that an area of interest is always kept within the field of view of the camera can be solved. 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 at all times. Coupling the perception and control problem can alleviate issues in state estimation because the racing gates are usually feature-rich.

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 MPC (PAMPC) introduced in [115]. 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 [116], where an MPPI controller is designed with a Deep Optical Flow (DOF) component which 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 [117], 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 [103]. 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 [118], CVXGEN [119], HPIPM [120], or Mosek [121]. In many drone racing papers, the optimal control problem is formulated as follows:

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

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 4th order Runge-Kutta, $RK4$. Additionally, 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:

$$\min_{\boldsymbol{u}} \sum_{k=0}^{T} \delta t \ , \qquad (3)$$
$$\text{subject to:} \quad \boldsymbol{x}_{k+1} = \boldsymbol{f}_{RK4}(\boldsymbol{x}_k, \boldsymbol{u}_k, \delta t) \ ,$$
$$\boldsymbol{x}_0 = \boldsymbol{x}_{init} \ , \quad \boldsymbol{u}_{min} \le \boldsymbol{u}_k \le \boldsymbol{u}_{max} \ ,$$

where T is the number of discrete time steps it takes to complete the race. This approach requires a time-horizon which predicts all the way until the end of the task which is intractable to optimize online.

Reinforcement learning (RL) methods [30], [101] 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 reinforcement learning 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 to develop and deploy them on hardware.

The biggest challenge for learning-based methods is collecting enough data to effectively train the neural network for the task at hand. There are currently two possibilities for data gathering. The first, mostly popular in the initial stages of learning-based robotics [69], [122]–[125] is to collect data in the real world. The data is 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 [30], [100], [126]–[128]. Both approaches have their advantages and limitations, which we will discuss in the following sections. Surveys covering existing methods for learning-based flight already exist [129], [130]. 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.

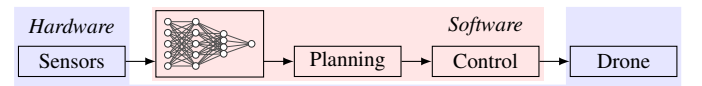### A. Learned Perception

Fig. 5: Architecture 2: 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 race track. A depiction of this architecture can be seen in

Figure 5. An overview of deep learning methods for vision-based navigation specific to the drone racing task can be found in [130].

In [124], 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 which predicts from an image both the next gate location and its uncertainty. Predictions are then fused with a visual-inertial odometry system in an Extended Kalman Filter (EKF) to predict the position of the drone on the track. Similarly in [12], 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 [12].

Oftentimes, perception networks consume precious resources onboard computationally limited drones. To minimize the network processing time, [71], [131] proposed optimized architectures for gate detection on real-world data. A similar optimization went into "GateNet" [132] 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 [133]. 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 [134].

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, [135] proposed using event cameras coupled with a sparse CNN, recurrent modules, and a You Only Look Once (YOLO) object detector to detect gates. The use of event cameras overcomes potential issues with motion blur induced by rapid movement of the drone and is a promising path forwards 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 visual-inertial odometry systems which explicitly estimate the metric state of the drone. These approaches are discussed in Sec. III. It is interesting to notice that learning-based odometry systems, such as [73]–[75] 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. A disadvantage of these methods is the high computational cost that makes them impractical for online applications. Research in end-to-end visual odometry is moving forward at a fast pace [75]. We foresee that in the near future, researchers will be able to apply these methods to the drone racing task.
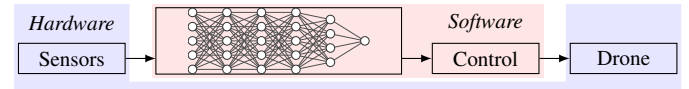
B. Learned Planning & Perception

Fig. 6: Architecture 3: Learned Planning and Perception

A tightly-coupled planning and perception stack (Figure 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 pre-training 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. In contrast to traditional methods, a locally consistent odometry system is sufficient [69], [126], [127], waving away the complexities of full-slam methods (e.g. loop-closure).

In [69], 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 re-training for each new environment. Therefore, in the follow-up work [126], data generated entirely from simulation is 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 [127], which proposes to train a neural network in simulation to map noisy sensory observations to collision-free trajectories directly.

Several other works apply a similar stacked perception and planning pipeline for other autonomous drone racing tasks [122], [123], [125], [136]. We point the interested reader to existing surveys on the role of learning in drone navigation [129].

A few works also studied the problem of planning using data-driven methods, decoupling it from the perception problem. An interesting approach demonstrated in the NeurIPS Game of Drones competition [137] used an off-the-shelf reinforcement learning algorithm in place of a classic model-based planner for drone racing [138]. More recently, a novel multimodal learning-based trajectory planning framework was introduced in [139], which can generate collision-free trajectories that avoid a dynamic obstacle while maximizing its presence in the field of view (FOV).

The big advantage of these methods is that they require less computational effort than traditional methods, possibly enabling online re-planning. 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 [30]. However, the major limitation of these methods is their sample complexity. If the training data comes from a simulator, significant simulation engineering is required to enable generalization. Conversely, if data come from the real world, generalization

is easier, but the data collection process is very slow, tedious, and expensive.
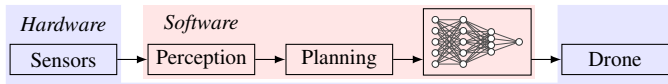
C. Learned Control

Fig. 7: Architecture 4: Learned Control

Data-driven control, like reinforcement learning, allows overcoming many limitations of prior model-based controller designs by learning effective controllers directly from experience. For example, control of a physical quadrotor using reinforcement learning was demonstrated by [140], where a neural network policy was used for waypoints tracking and vehicle recovery from harsh initialization. The neural network policy takes about $7\,\mu s$ to generate the control command given the state, while a linear MPC requires about $1000\,\mu s$. Recently, [30] demonstrated high-speed trajectory tracking using learning-based control. They additionally showed that learned policies can be made robust to sensor noise and system latency by training with simulated sensor noise and latencies. Model-free RL was also applied to low-level attitude control [141], in which a learned low-level controller trained with PPO outperformed a fully tuned PID controller on almost every metric. Similarly, [142] used model-based RL for low-level control of an a priori unknown dynamic system.

With any learning-based controller, it can be difficult to provide robustness guarantees as with traditional methods such as the Linear Quadratic Regulator (LQR). However, it is possible to make the planner and controller robust to system latencies, model uncertainties, and sensor noise by identifying them on physical drones and then adding them into the simulation environments used to gather training data [30]. While a learning-based controller may provide superior performance to classical methods, it may be the case that they cannot be used in practice due to the inability to provide an analysis of the controller's stability properties. These properties are often required in safety-critical systems such as flight controllers for aircraft. Recent works have attempted to address this using Lyapunov-stable neural network design for the control of quadrotors [143]. 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 [144].

None of the systems discussed so far deal with the challenging problem of adapting to new and uncertain environments. The field of adaptive control has studied this problem extensively [145]–[147], however, we have seen a recent push to use advancements in machine learning within the adaptive control framework. A method to learn parametric uncertainty functions is introduced in [148]. 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 [149]. We see this as a promising area of research and one that is integral for reliable performance in changing environmental conditions.
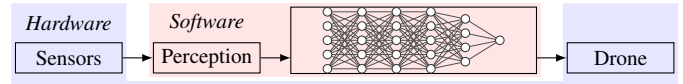
D. Learned Planning & Control

Fig. 8: Architecture 5: Learned Planning & 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 diagram of Figure 8. In autonomous drone racing, this was proposed by [101], where a neural network policy is trained with reinforcement learning to fly through a race track in simulation in near-minimum time. Major advantages of the reinforcement-learning-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 [102], deep reinforcement learning 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.

These methods inherit the classic advantage of policy learning: to achieve robustness to system latencies, model uncertainties, and sensor noise, one can identify them on physical drones and then add them into the simulation environments used to gather training data [30]. In addition, they do not require an external controller to track the plan. This eliminates the discrepancy between the planning and deployment stage, which is one of the main limitations of traditional planning methods (Sec. III-B). Some of the limitations of traditional planning still 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.
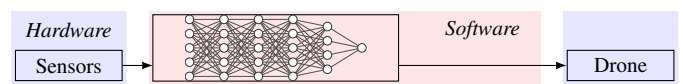
E. End-to-End Flight

Fig. 9: Architecture 7: End to End Learning

Expert pilots take raw sensory images from a first-person-view camera stream and map directly to control commands. In this section, we explore approaches emulating this holistic navigation paradigm in autonomous drones.

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 [150], [151], 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 [152]. Yet, the division into independent blocks leads to compounding errors and latencies, which negatively affect performance when flying at high speeds [127].

The second family of approaches directly maps sensor observation to commands without any modularity. This design is used by [153], 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, [154] trains an end-to-end CNN to directly predict roll, pitch, yaw, and altitude from camera images. Similarly, [155] uses 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. Given the absence of any division between perception, planning, and control, this family of approaches is potentially more robust to sensor noise and latencies. Yet, these policies are extremely data-hungry, which hinders their generalization in environments different from the training ones.

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 [153], 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 autonomous drone racing, ranging from improving the system model to end-to-end control of the vehicle. Currently, the best-performing algorithms for drone racing include a learning-based component [12], [13], 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 [30].

A trend that is recently gaining more and more popularity is training policies in simulation and deploying them in the real world [69], [126], [127]. Leveraging years of advancement in drone modeling technology (See Section II), the simulation of drone dynamics is extremely realistic and fast. Conversely, the simulation of sensor measurements (e.g. cameras, IMUs, lidars) is either inaccurate or very computationally expensive. Therefore, researchers generally aim to abstract observations using classical perception algorithms (see Section III-A) to train their model in a timely fashion and favor simulation-to-real-world transfer [127]. However, this hinders the deployment of completely end-to-end systems on real-world robots, which, as human pilots, only rely on a stream of color images. We refer the reader to Sec. VIII for the implication of this research.

While simulators may get better and faster in the near future, recent advances in real-world training [156], [157] and fine-tuning [158], [159] offer a potential alternative for zero-shot simulation to reality transfer for sensorimotor policies. 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 racing performance.

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 multi-rotors [17]. 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 multi-rotors, 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 photo-realistic simulator for the control of drones [15]. It is built on the Unreal graphics engine and features easy-to-use plugins for popular flight controllers such as PX4[3], ArduPilot[4], and others. It was used in the 2019 NeurIPS Game of Drones challenge [137]. Because of the photorealism of AirSim, it is possible to simulate the entire perception and estimation pipeline with good possibility of transfer to real-world drone systems. Additionally, AirSim comes pre-packaged with an OpenAI-Gym environment for training Reinforcement Learning algorithms. Organizations such as Bell, Airtonomy, and

---

[3]https://px4.io/
[4]https://ardupilot.org/

NASA are using AirSim to generate training data for learning-based perception models.

FlightGoggles [160] 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 rendering simulated images that correspond to the position of physical vehicles in the real world.

A recent simulator focused on Safe RL was proposed in [161]. 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.

Finally, Flightmare [16] 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 [162].

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 Figure 2. The Autonomous Drone Racing (ADR) competition was an annual competition which took place during the IROS conference between 2016 and 2019. In 2016, 11 teams competed in autonomous drone racing 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 minutes prior to the official competition to fly the course as many times as they wished. The winning team, from KAIST, made it through 10 of the 26 gates in 1 minute and 26 seconds. For comparison, a human was able to complete the entire 26-gate course in 1 minute 31 seconds. A survey summarizing the approaches used for these early competitions can be found in [163]. 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 5 teams participated in the final in-person event, with the winning team making it through 9 out of 13 gates in over 3 minutes. A summary of the winning approaches can be found in [10]. 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 prize was awarded to the winning team [164]. The competition took place first in a virtual qualifying round

which used the FlightGoggles simulation environment [160]. 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. Ultimately, professional pilot Gabriel Kocher, from the Drone Racing League, manually piloted his drone through the course in only 6 seconds. It took 11 seconds to the winner, MAVLab from TU Delft, and 15 seconds to the second-place winner, UZH-RPG from the University of Zurich, to complete the course autonomously. The two different approaches are documented in [12], [13]. Further comments are provided by the winner in [165]. Perez et al. provides an interesting overview of the types of hardware used for some of the drone racing competitions mentioned so far [46].

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 [11], [15], [137]. 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 Zurich, Switzerland, three of the world's best human pilots competed against researchers from the Robotics and Perception Group of the University of Zurich. Flight speeds exceeding 100 kph 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 Evan Ackermann discusses the multi-day event in [105].

VII. DATASETS AND OPEN SOURCE CODE

In this section, we provide an overview of the existing open source code bases as well as useful datasets for autonomous drone racing. We first discuss datasets, and then group the existing open source code bases by their use-cases in table I.

In 2018, researchers from MIT released a large scale dataset for perception during aggressive UAV flight [168]. This dataset contains over 10 hours of flight data which includes 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 Visual-Inertial Odometry (VIO) or Simultaneous Localization and Mapping (SLAM) 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 [68]. 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. Similar to the authors in [168], the authors of

TABLE I: Open Source Software and Datasets

Name and Reference	Category	Year	Link
PAMPC [115]	Controller	2018	https://github.com/uzh-rpg/rpg_mpc
Deep Drone Acrobatics [100]	Controller	2019	https://github.com/uzh-rpg/deep_drone_acrobatics
Data Driven MPC [34]	Controller	2020	https://github.com/uzh-rpg/data_driven_mpc
High MPC [166]	Controller	2022	https://github.com/uzh-rpg/high_mpc
AutoTune [167]	Controller Tuner	2022	https://github.com/uzh-rpg/mh_autotune
Blackbird [168]	Dataset	2018	https://github.com/mit-aera/Blackbird-Dataset
UZH-FPV [68]	Dataset	2019	https://fpv.ifi.uzh.ch/
NeuroBEM [29]	Dataset	2020	https://rpg.ifi.uzh.ch/NeuroBEM.html
Eye Gaze Drone Racing [6]	Dataset	2021	https://osf.io/gvdse/
Time-optimal Planning for Quadrotor Waypoint Flight [84]	Planner	2021	https://github.com/uzh-rpg/rpg_time_optimal
Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments [31]	Planner	2022	https://github.com/uzh-rpg/sb_min_time_quadrotor_planning
RotorS [17]	Simulator	2016	https://github.com/ethz-asl/rotors_simulator
AirSim [137]	Simulator	2018	https://microsoft.github.io/AirSim/
FlightGoggles [160]	Simulator	2019	https://github.com/mit-aera/FlightGoggles
Flightmare [16]	Simulator	2020	https://uzh-rpg.github.io/flightmare/
Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control [161]	Simulator	2021	https://github.com/utiasDSL/gym-pybullet-drones
Sim 2 Real Domain Randomization [126]	Sim2Real Transfer	2019	https://github.com/uzh-rpg/sim2real_drone_racing
RPG Quadrotor Control [20]	Software Stack	2017	https://github.com/uzh-rpg/rpg_quadrotor_control
Agilicious [162]	Software Stack	2022	https://github.com/uzh-rpg/agilicious
Kalibr [48]	Camera Calibration	2022	https://github.com/ethz-asl/kalibr

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.[5]

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 [6]

NeuroBEM [29] is a hybrid aerodynamic quadrotor model which combines blade-element-momentum-theory models with learned aerodynamic representations from highly aggressive maneuvers. While the model is fit to the specific quadrotor platform defined in [162], 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.

A significant amount of autonomous drone racing 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.

VIII. OPEN RESEARCH QUESTIONS AND CHALLENGES

In this section, we examine some of the biggest challenges that the field of autonomous drone racing is facing. Autonomous drone racing is a field that is growing rapidly. To quantify the rate of growth, we examined the number of papers that mentioned the key-phrase "autonomous drone racing" since 2015. The data, indicated in Fig. 10, shows exponential-like growth of the field. Thus, it is appropriate to discuss where future opportunities exist for incoming and experienced researchers alike.
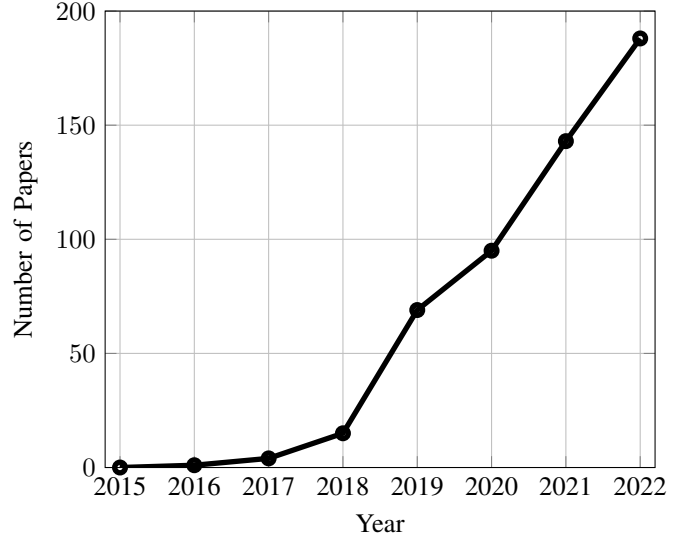
[5]https://fpv.ifi.uzh.ch/uzh/uzh-fpv-leader-board/

Fig. 10: The number of papers related to "autonomous drone racing" cataloged on Google Scholar.

A. Challenge 1: Enabling VIO for High-Speed Applications

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 visual-inertial odometry 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 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 [169]. 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 non-zero 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 [75], [170] 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 first-person-view 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 yet 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 [6], [171]. They found that expert pilots can control drones despite a 200ms latency, which is compensated by the human brain. Taking inspiration from biology, a recent work [172] 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 above 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 autonomous drone racing 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 5 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 which can propagate predictions of the opponents' states and racing lines through time are necessary to anticipate collisions or overtaking opportunities. An initial study [173] 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 2D plane. The work was further extended to 3D spaces in [174], 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: 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. Until this point, the rate of technology transfer has been slow due to challenges in flight certification and a lack of generalization between environments. However, commercial applications that leverage the full agility of the platform have much to gain. Drones that fly fast, fly farther, therefore increase the productivity of drones in every commercial sector [4]. As it stands today, drone racing algorithms can be difficult to directly transfer to a new environment due to overfitting and minimal safety guarantees. Calculating time-optimal paths that are safety critical currently takes too long for deployment in emergency scenarios. Existing works that leverage expert perception models to sense and plan around obstacles can be sensitive to changes in the environment and lead to crashes. Beyond this, we often do not have a known map ahead of time for real world applications, requiring researchers to think about how to simultaneously estimate the state of the drone while mapping the environment. Building algorithms that can continually improve from their own experience is key in enabling this transfer. While recent advances in reinforcement learning research point to the feasibility of this path [158], [159], [175], 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 [176]. A survey paper

covering safe RL methods can be found in [177]. Furthermore, a thorough review paper on continual, or life-long RL can be found in [178].

IX. Summary and Conclusions

In this survey, we provided a comprehensive overview of the task of autonomous drone racing across model-based and learning-based approaches. A history of all recent autonomous drone racing events was given, along with a list of all open-source code bases, datasets, and simulators. These resources can be used to greatly reduce the learning curve and time needed when it comes to getting started with autonomous drone racing. With these resources and a list of open challenges for the field, researchers should have the tools to push the limits.

X. Contributions

Drew Hanover initiated the idea of this paper, created the paper structure, and contributed to all sections of this paper while coordinating efforts amongst the co-authors. Antonio Loquercio contributed to the paper structure and the learning-based sections. Leonard Bauersfeld authored the Drone Modeling section and created the graphics seen throughout. Angel Romero contributed to the Classical Planning and Control sections. Giovanni Cioffi contributed to the Classical Perception and Challenges sections. Yunlong Song contributed to the Simulators and Learning-Based Planning/Control sections Robert Penicka contributed to both Classical and Learning-Based Planning sections. Elia Kaufmann contributed to the paper structure and throughout the Learning-Based sections. Davide Scaramuzza contributed to the general paper structure and revised the paper thoroughly and critically.

References

[1] T. A. Wilkinson, *Early Dynastic Egypt*. Routledge, 2002.

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

[3] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam, "Autonomous vehicles on the edge: A survey on autonomous vehicle racing," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 458–488, 2022.

[4] L. Bauersfeld and D. Scaramuzza, "Range, endurance, and optimal speed estimates for multicopters," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2953–2960, 2022.

[5] 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, p. 64, 2022.

[6] C. Pfeiffer and D. Scaramuzza, "Human-piloted drone racing: Visual processing and control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3467–3474, 2021.

[7] 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," *Journal of Field Robotics*, vol. 35, no. 1, pp. 146–166, 2018.

[8] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico *et al.*, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[9] E. R. Council, "Low-latency perception and action for agile vision-based flight." [Online]. Available: https://cordis.europa.eu/projects

[10] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intelligent Service Robotics*, vol. 12, no. 2, pp. 137–148, 2019.

[11] Microsoft, "Game of drones." [Online]. Available: https://github.com/microsoft/AirSim-NeurIPS2019-Drone-Racing

[12] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Autonomous Robots*, vol. 46, no. 1, pp. 307–320, 2022.

[13] 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 Robotics*, vol. 2, 2022.

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

[15] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[16] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 1147–1157.

[17] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot operating system (ROS)*. Springer, 2016, pp. 595–625.

[18] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *International conference on simulation, modeling, and programming for autonomous robots*. Springer, 2012, pp. 400–411.

[19] R. W. Prouty, *Helicopter performance, stability, and control*. Krieger Pub Co, 1995.

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

[21] S. Yoon, H. C. Lee, and T. H. Pulliam, *Computational Analysis of Multi-Rotor Flows*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-0812

[22] P. V. Diaz and S. Yoon, *High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-1266

[23] S. Yoon, Nasa, P. V. Diaz, D. D. Boyd, W. M. Chan, and C. R. Theodore, "Computational aerodynamic modeling of small quadcopter vehicles," 2017.

[24] R. Gill and R. D'Andrea, "Propeller thrust and drag in forward flight," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, 2017, pp. 73–79.

[25] R. Gill and R. D'Andrea, "Computationally efficient force and moment models for propellers in uav forward flight applications," *Drones*, vol. 3, no. 4, p. 77, 2019.

[26] W. Khan and M. Nahon, "Toward an accurate physics-based uav thruster model," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 4, pp. 1269–1279, 2013.

[27] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *AIAA guidance, navigation and control conference and exhibit*, 2007, p. 6461.

[28] M. Bangura and R. Mahony, "Thrust control for multirotor aerial vehicles," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 390–405, 2017.

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

[30] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[31] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, 2022.

[32] P. Ventura Diaz and S. Yoon, "High-fidelity computational aerodynamics of multi-rotor unmanned aerial vehicles," in *2018 AIAA Aerospace Sciences Meeting*, 2018, p. 1266.

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

[34] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven mpc for quadrotors," *IEEE Robotics and Automation Letters*, 2021.

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

[36] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4653–4660.

[37] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3223–3230.

[38] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," 05 2019, pp. 9784–9790.

[39] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," in *Proc. 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016, p. 125.

[40] W. Peukert, "Über die abhängigkeit der kapazität von der entladestromstärke bei bleiakkumulatoren." *Elektrotechn. Zeitschr.*, 1897.

[41] N. Galushkin, N. Yazvinskaya, and D. Galushkin, "Generalized analytical model for capacity evaluation of automotive-grade lithium batteries," *Journal of The Electrochemical Society*, vol. 162, pp. A308–A314, 01 2015.

[42] N. Galushkin, N. N. Yazvinskaya, and D. N. Galushkin, "A critical review of using the peukert equation and its generalizations for lithium-ion cells," *Journal of The Electrochemical Society*, vol. 167, no. 12, p. 120516, aug 2020.

[43] X. Zhang, W. Zhang, and G. Lei, "A review of li-ion battery equivalent circuit models," *Transactions on Electrical and Electronic Materials*, vol. 17, pp. 311–316, 12 2016.

[44] 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.

[45] 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," *Journal of Intelligent and Robotic Systems*, vol. 100, 12 2020.

[46] L. O. Rojas-Perez and J. Martinez-Carranza, "On-board processing for autonomous drone racing: An overview," *Integration*, vol. 80, pp. 46–59, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926021000559

[47] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[48] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4304–4311.

[49] Y. Yang, P. Geneva, X. Zuo, and G. Huang, "Online imu intrinsic calibration: Is it necessary?" *Proc. of Robotics: Science and Systems (RSS), Corvallis, Or*, 2020.

[50] D. Scaramuzza and Z. Zhang, "Visual-inertial odometry of aerial robots," *Encyclopedia of Robotics*, 2019.

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

[52] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, pp. 298–304.

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

[54] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[55] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[56] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

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

[58] G. Huang, "Visual-inertial navigation: A concise review," in *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2019, pp. 9572–9582.

[59] 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 Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

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

[61] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[62] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza, "Vimo: Simultaneous visual inertial model-based odometry and force estimation," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2785–2792, 2019.

[63] Z. Ding, T. Yang, K. Zhang, C. Xu, and F. Gao, "Vid-fusion: Robust visual-inertial-dynamics odometry for accurate external force estimation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 469–14 475.

[64] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[65] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 2502–2509.

[66] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[67] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "Alphapilot: Autonomous drone racing," *Robotics: Science and Systems (RSS)*, 2020. [Online]. Available: https://link.springer.com/article/10.1007/s11370-018-00271-6

[68] 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6713–6719.

[69] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 133–145. [Online]. Available: https://proceedings.mlr.press/v87/kaufmann18a.html

[70] 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," *Robotics and Autonomous Systems*, vol. 133, p. 103621, 2020.

[71] D. Zhang and D. D. Doyle, "Gate detection using deep learning," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–11.

[72] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[73] S. Wang, R. Clark, H. Wen, and N. Trigoni, "Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2043–2050.

[74] W. Wang, Y. Hu, and S. Scherer, "Tartanvo: A generalizable learning-based vo," in *Conference on Robotic Learning (CoRL)*, 2020.

[75] Z. Teed, L. Lipson, and J. Deng, "Deep patch visual odometry," *arXiv preprint arXiv:2208.04726*, 2022.

[76] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011.

[77] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Robot. Research*, 2012.

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

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

[80] Z. Han, Z. Wang, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for SE(3) planning in autonomous drone racing," *CoRR*, vol. abs/2105.10276, 2021. [Online]. Available: https://arxiv.org/abs/2105.10276

[81] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1208–1214.

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

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

[84] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, p. eabh1221, 2021.

[85] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza, "Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload," in *Robotics: Science and Systems (RSS)*, 2017.

[86] M. Hehn, R. Ritz, and R. D'Andrea, "Performance benchmarking of quadrotor systems using time-optimal control," *Autonomous Robots*, Mar. 2012.

[87] W. V. Loock, G. Pipeleers, and J. Swevers, "Time-optimal quadrotor flight," in *IEEE Eur. Control Conf. (ECC)*, 2013.

[88] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1335–1344, 2017.

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

[90] I. Spasojevic, V. Murali, and S. Karaman, "Perception-aware time optimal path parameterization for quadrotors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3213–3219.

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

[92] K. Bousson and P. F. Machado, "4d trajectory generation and tracking for waypoint-based aerial navigation," *WSEAS Transactions on Systems and Control*, no. 3, pp. 105–119, 2013.

[93] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep 2017, p. 2872–2879.

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

[95] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 5054–5061.

[96] R. Allen and M. Pavone, "A real-time framework for kinodynamic planning with application to quadrotor obstacle avoidance," in *AIAA Guidance, Navigation, and Control Conference*, 2016, p. 1374.

[97] T. Zhiling, B. Chen, R. Lan, and S. Li, "Vector field guided rrt* based on motion primitives for quadrotor kinodynamic planning," *Journal of Intelligent & Robotic Systems*, vol. 100, 12 2020.

[98] A. Romero, R. Penicka, and D. Scaramuzza, "Time-optimal online replanning for agile quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

[99] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, pp. 1–17, 2022.

[100] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[101] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[102] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[103] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 690–697, 2022.

[104] 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.*, 2022.

[105] E. Ackerman, "Autonomous drones challenge human champions in first "fair" race," Jul 2022. [Online]. Available: https://spectrum.ieee.org/zurich-autonomous-drone-race

[106] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav for extreme maneuverability," *IFAC Proceedings Volumes*, 2011.

[107] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 1203–1218, 2020.

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

[109] 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.

[110] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[111] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsiotras, and J. M. Rehg, "Autorally: An open platform for aggressive autonomous driving," *IEEE Control Systems Magazine*, vol. 39, no. 1, pp. 26–55, 2019.

[112] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 6137–6142.

[113] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, jul 2014. [Online]. Available: https://doi.org/10.1002%2Foca.2123

[114] J. Arrizabalaga and M. Ryll, "Towards time-optimal tunnel-following for quadrotors," *arXiv preprint arXiv:2110.01351*, 2021.

[115] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[116] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1207–1214, 2020.

[117] 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.

[118] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[119] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[120] 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.

[121] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, 2019.

[122] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro *et al.*, "A machine learning approach to visual perception of forest trails for mobile robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 661–667, 2015.

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

[124] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Beauty and the beast: Optimal methods meet learning for drone racing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 690–696.

[125] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3948–3955.

[126] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, 2019.

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

[128] F. Sadeghi and S. Levine, "Cad 2 rl: Real single-image flight without a single real image," in *Robotics: Science and Systems (RSS)*, 2017, pp. 48–55.

[129] T. Lee, S. Mckeever, and J. Courtney, "Flying free: A research overview of deep learning in drone navigation autonomy," in *drones*, 2021.

[130] 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*. Elsevier, 2022, pp. 371–406.

[131] 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 America Transactions*, vol. 17, no. 12, pp. 2045–2052, 2019.

[132] 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 *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4176–4183.

[133] H. X. Pham, A. Sarabakha, M. Odnoshyvkin, and E. Kayacan, "Pencil-net: Zero-shot sim-to-real transfer learning for robust gate perception in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 847–11 854, 2022.

[134] T. Morales, A. Sarabakha, and E. Kayacan, "Image generation for efficient neural network training in autonomous drone racing," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[135] K. F. Andersen, H. X. Pham, H. I. Ugurlu, and E. Kayacan, "Event-based navigation for autonomous drone racing with sparse gated recurrent network," *arXiv preprint arXiv:2204.02120*, 2022.

[136] K. Amer, M. Samy, M. Shaker, and M. ElHelw, "Deep convolutional neural network based autonomous drone navigation," in *Thirteenth International Conference on Machine Vision*, vol. 11605. SPIE, 2021, pp. 16–24.

[137] R. Madaan, N. Gyde, S. Vemprala, M. Brown, K. Nagami, T. Taubner, E. Cristofalo, D. Scaramuzza, M. Schwager, and A. Kapoor, "Airsim drone racing lab," in *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds., vol. 123. PMLR, 08–14 Dec 2020, pp. 177–191. [Online]. Available: https://proceedings.mlr.press/v123/madaan20a.html

[138] U. Ates, "Long-term planning with deep reinforcement learning on autonomous drones," in *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE, 2020, pp. 1–6.

[139] J. Tordesillas and J. How, "Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments," *arxiv, 2209.01268v1*, 07 2022.

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

[141] W. Koch, R. Mancuso, R. West, and A. Bestavros, "Reinforcement learning for uav attitude control," *ACM Transactions on Cyber-Physical Systems*, vol. 3, no. 2, pp. 1–21, 2019.

[142] 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 Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[143] H. Dai, B. Landry, L. Yang, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," *arXiv preprint arXiv:2109.14152*, 2021.

[144] M. Selim, A. Alanwar, S. Kousik, G. Gao, M. Pavone, and K. H. Johansson, "Safe reinforcement learning using black-box reachability analysis," *arXiv preprint arXiv:2204.07417*, 2022.

[145] P. A. Ioannou and J. Sun, *Robust adaptive control*. Courier Corporation, 2012.

[146] K. J. Åström and B. Wittenmark, *Adaptive control*. Courier Corporation, 2013.

[147] E. Lavretsky and K. A. Wise, "Robust adaptive control," in *Robust and adaptive control*. Springer, 2013, pp. 1 – 449.

[148] S. M. Richards, N. Azizan, J.-J. Slotine, and M. Pavone, "Adaptive-control-oriented meta-learning for nonlinear systems," *arXiv preprint arXiv:2103.04490*, 2021.

[149] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, "A zero-shot adaptive quadcopter controller." arXiv, 2022.

[150] G. Li, M. Mueller, V. M. Casser, N. Smith, D. Michels, and B. Ghanem, "Oil: Observational imitation learning," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[151] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[152] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 902–919, 2018.

[153] 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 *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[154] L. O. Rojas-Perez and J. Martinez-Carranza, "Deeppilot: A cnn for autonomous drone racing," *Sensors*, vol. 20, no. 16, p. 4524, 2020.

[155] J. Fu, Y. Song, Y. Wu, F. Yu, and D. Scaramuzza, "Learning deep sensorimotor policies for vision-based autonomous drone racing," *arXiv preprint arXiv:2210.14985*, 2022.

[156] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Day-dreamer: World models for physical robot learning." arXiv, 2022.

[157] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," 2022.

[158] 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 *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.

[159] A. Loquercio, A. Kumar, and J. Malik, "Learning Visual Locomotion with Cross-Modal Supervision," in *arXiv*, 2022.

[160] 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 *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, nov 2019. [Online]. Available: https://doi.org/10.1109%2Firos40897.2019.8968116

[161] 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 *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7512–7519.

[162] P. Foehn, E. Kaufmann, A. Romero, R. Penicka, S. Sun, L. Bauersfeld, T. Laengle, G. Cioffi, Y. Song, A. Loquercio, and D. Scaramuzza, "Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight," *Science Robotics*, vol. 7, no. 67, p. eabl6259, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abl6259

[163] H. Moon, Y. Sun, J. Baltes, and S. J. Kim, "The iros 2016 competitions [competitions]," *IEEE Robotics and Automation Magazine*, vol. 24, no. 1, pp. 20–29, 2017.

[164] L. Martin, "Alphapilot ai drone innovation challenge," Jan 2020. [Online]. Available: https://lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html

[165] C. de Wagter, F. Paredes-Vallés, N. Sheth, and G. C. de Croon, "Learning fast in autonomous drone racing," *Nat. Mach. Intell.*, vol. 3, p. 923, 2021.

[166] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Transactions on Robotics*, 2022.

[167] A. Loquercio, A. Saviolo, and D. Scaramuzza, "Autotune: Controller tuning for high-speed flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4432–4439, 2022.

[168] 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 *International Symposium on Experimental Robotics*. Springer, 2018, pp. 130–139.

[169] J. Hidalgo-Carrió, G. Gallego, and D. Scaramuzza, "Event-aided direct sparse odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5781–5790.

[170] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 558–16 569, 2021.

[171] 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.

[172] 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.

[173] 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 *Robotics: Science and Systems*, 2018.

[174] 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," *Robotics and Autonomous Systems*, vol. 125, p. 103410, 2020.

[175] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conference on Robot Learning (CoRL)*, 2019.

[176] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal, "Safe reinforcement learning via curriculum induction," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12151–12162, 2020.

[177] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[178] K. Khetarpal, M. Riemer, I. Rish, and D. Precup, "Towards continual reinforcement learning: A review and perspectives," *arXiv preprint arXiv:2012.13490*, 2020.