Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Deep Drone Acrobatics

Elia Kaufmann*‡, Antonio Loquercio*‡, René Ranftl†, Matthias Müller†, Vladlen Koltun†, Davide Scaramuzza‡

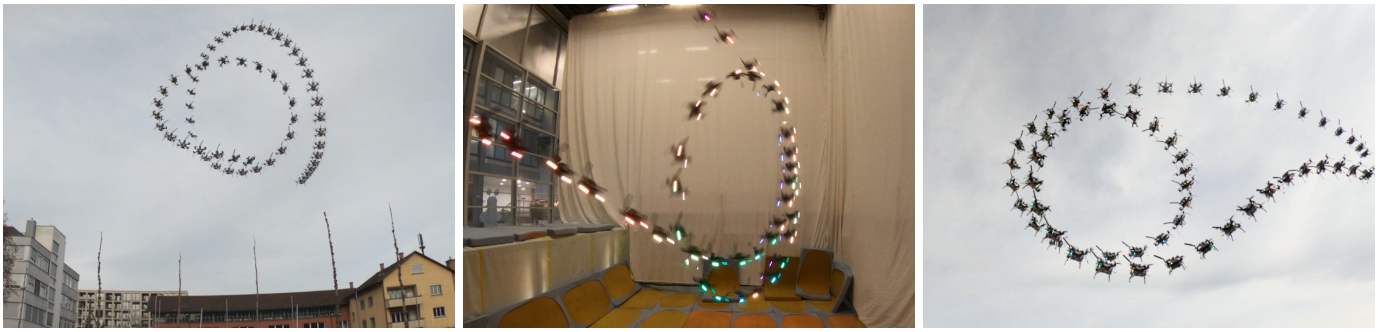

Fig. 1: A quadrotor performs a Barrel Roll (left), a Power Loop (middle), and a Matty Flip (right). We safely train acrobatic controllers in simulation and deploy them with no fine-tuning (*zero-shot transfer*) on physical quadrotors. The approach uses only onboard sensing and computation. No external motion tracking was used.

***Abstract*—Performing acrobatic maneuvers with quadrotors is extremely challenging. Acrobatic flight requires high thrust and extreme angular accelerations that push the platform to its physical limits. Professional drone pilots often measure their level of mastery by flying such maneuvers in competitions. In this paper, we propose to learn a sensorimotor policy that enables an autonomous quadrotor to fly extreme acrobatic maneuvers with only onboard sensing and computation. We train the policy entirely in simulation by leveraging demonstrations from an optimal controller that has access to privileged information. We use appropriate abstractions of the visual input to enable transfer to a real quadrotor. We show that the resulting policy can be directly deployed in the physical world without any fine-tuning on real data. Our methodology has several favorable properties: it does not require a human expert to provide demonstrations, it cannot harm the physical system during training, and it can be used to learn maneuvers that are challenging even for the best human pilots. Our approach enables a physical quadrotor to fly maneuvers such as the Power Loop, the Barrel Roll, and the Matty Flip, during which it incurs accelerations of up to 3g.**

Supplementary Material

A video demonstrating acrobatic maneuvers is available at https://youtu.be/2N_wKXQ6MXA. Code can be found at https://github.com/uzh-rpg/deep_drone_acrobatics.

I. Introduction

Acrobatic flight with quadrotors is extremely challenging. Human drone pilots require many years of practice to safely master maneuvers such as power loops and barrel rolls[1]. Existing autonomous systems that perform agile maneuvers require external sensing and/or external computation [23, 1, 3]. For aerial vehicles that rely only on onboard sensing and computation, the high accelerations that are required for acrobatic maneuvers together with the unforgiving requirements on the control stack raise fundamental questions in both perception and control. Therefore, they provide a natural benchmark to compare the capabilities of autonomous systems against trained human pilots.

Acrobatic maneuvers represent a challenge for the actuators, the sensors, and all physical components of a quadrotor. While hardware limitations can be resolved using expert-level equipment that allows for extreme accelerations, the major limiting factor to enable agile flight is reliable state estimation. Vision-based state estimation systems either provide significantly reduced accuracy or completely fail at high accelerations due to effects such as motion blur, large displacements, and the difficulty of robustly tracking features over long time frames [4]. Additionally, the harsh requirements of fast and precise control at high speeds make it difficult to tune controllers on the real platform, since even tiny mistakes can result in catastrophic outcomes for the platform.

The difficulty of agile autonomous flight led previous work to mostly focus on specific aspects of the problem. One line of research focused on the control problem, assuming near-perfect state estimation from external sensors [23, 1, 15, 3]. While these works showed impressive examples of agile flight, they focused purely on control. The issues of reliable perception and state estimation during agile maneuvers were cleverly circumvented by instrumenting the environment with sensors (such as Vicon and OptiTrack) that provide near-perfect state estimation to the platform at all times. Recent works addressed the control and perception aspects in an integrated way via techniques like perception-guided trajectory optimization [9, 10, 33] or training end-to-end visuomotor agents [34]. However, acrobatic performance of high-acceleration maneuvers with only onboard sensing and computation has not yet been achieved.

In this paper, we show for the first time that a vision-based autonomous quadrotor with only onboard sensing and computation is capable of autonomously performing agile maneuvers with accelerations of up to 3g, as shown in Fig. 1.

*Equal contribution.
‡Robotics and Perception Group, University of Zurich and ETH Zurich.
†Intelligent Systems Lab, Intel.
[1]https://www.youtube.com/watch?v=T1vzjPa5260

This contribution is enabled by a novel simulation-to-reality transfer strategy, which is based on abstraction of both visual and inertial measurements. We demonstrate both formally and empirically that the presented abstraction strategy decreases the simulation-to-reality gap with respect to a naive use of sensory inputs. Equipped with this strategy, we train an end-to-end sensimotor controller to fly acrobatic maneuvers *exclusively* in simulation. Learning agile maneuvers entirely in simulation has several advantages: (i) Maneuvers can be simply specified by reference trajectories in simulation and do not require expensive demonstrations by a human pilot, (ii) training is safe and does not pose any physical risk to the quadrotor, and (iii) the approach can scale to a large number of diverse maneuvers, including ones that can only be performed by the very best human pilots.

Our sensorimotor policy is represented by a neural network that combines information from different input modalities to directly regress thrust and body rates. To cope with different output frequencies of the onboard sensors, we design an asynchronous network that operates independently of the sensor frequencies. This network is trained in simulation to imitate demonstrations from an optimal controller that has access to privileged state information.

We apply the presented approach to learning autonomous execution of three acrobatic maneuvers that are challenging even for expert human pilots: the Power Loop, the Barrel Roll, and the Matty Flip. Through controlled experiments in simulation and on a real quadrotor, we show that the presented approach leads to robust and accurate policies that are able to reliably perform the maneuvers with only onboard sensing and computation.

II. Related Work

Acrobatic maneuvers comprehensively challenge perception and control stacks. The agility that is required to perform acrobatic maneuvers requires carefully tuned controllers together with accurate state estimation. Compounding the challenge, the large angular rates and high speeds that arise during the execution of a maneuver induce strong motion blur in vision sensors and thus compromise the quality of state estimation.

The complexity of the problem has led early works to only focus on the control aspect while disregarding the question of reliable perception. Lupashin et al. [23] proposed iterative learning of control strategies to enable platforms to perform multiple flips. Mellinger et al. [25] used a similar strategy to autonomously fly quadrotors through a tilted window [25]. By switching between two controller settings, Chen et al. [6] also demonstrated multi-flip maneuvers. Abbeel et al. [1] learned to perform a series of acrobatic maneuvers with autonomous helicopters. Their algorithm leverages expert pilot demonstrations to learn task-specific controllers. While these works proved the ability of flying machines to perform agile maneuvers, they did not consider the perception problem. Indeed, they all assume that near-perfect state estimation is available during the maneuver, which in practice requires instrumenting the environment with dedicated sensors.

Aggressive flight with only onboard sensing and computation is an open problem. The first attempts in this direction were made by Shen et al. [33], who demonstrated agile vision-based flight. The work was limited to low-acceleration trajectories, therefore only accounting for part of the control and perception problems encountered at high speed. More recently, Loianno et al. [20] and Falanga et al. [10] demonstrated aggressive flight through narrow gaps with only onboard sensing. Even though these maneuvers are agile, they are very short and cannot be repeated without re-initializing the estimation pipeline. Using perception-guided optimization, Falanga et al. [9] and Lee et al. [17] proposed a model-predictive control framework to plan aggressive trajectories while minimizing motion blur. However, such control strategies are too conservative to fly acrobatic maneuvers, which always induce motion blur.

Abolishing the classic division between perception and control, a recent line of work proposes to train visuomotor policies directly from data. Similarly to our approach, Zhang et al. [34] trained a neural network from demonstrations provided by an MPC controller. While the latter has access to the full state of the platform and knowledge of obstacle positions, the network only observes laser range finder readings and inertial measurements. Similarly, Li et al. [18] proposed an imitation learning approach for training visuomotor agents for the task of quadrotor flight. The main limitation of these methods is in their sample complexity: large amounts of demonstrations are required to fly even straight-line trajectories. As a consequence, these methods were only validated in simulation or were constrained to slow hover-to-hover trajectories.

Our approach employs *abstraction* of sensory input [27] to reduce the problem's sample complexity and enable *zero-shot sim-to-real transfer*. While prior work has demonstrated the possibility of controlling real-world quadrotors with zero-shot sim-to-real transfer [21, 32], our approach is the first to learn an end-to-end sensorimotor mapping – from sensor measurements to low-level controls – that can perform high-speed and high-acceleration acrobatic maneuvers on a real physical system.

III. Overview

In order to perform acrobatic maneuvers with a quadrotor, we train a sensorimotor controller to predict low-level actions from a history of onboard sensor measurements and a user-defined reference trajectory. An observation $\boldsymbol{o}[k] \in \mathbb{O}$ at time $k \in [0, \dots, T]$ consists of a camera image $\mathcal{I}[k]$ and an inertial measurement $\phi[k]$. Since the camera and IMU typically operate at different frequencies, the visual and inertial observations are updated at different rates. The controller's output is an action $\boldsymbol{u}[k] = [c, \boldsymbol{\omega}^\top]^\top \in \mathbb{U}$ that consists of continuous mass-normalized collective thrust c and bodyrates $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^\top$ that are defined in the quadrotor body frame.

The controller is trained via *privileged learning* [5]. Specifically, the policy is trained on demonstrations that are provided by a privileged expert: an optimal controller that has access to privileged information that is not available to the sensorimotor student, such as the full ground-truth state of the platform $\boldsymbol{s}[k] \in \mathbb{S}$. The privileged expert is based on a classic

optimization-based planning and control pipeline that tracks a reference trajectory from the state $\boldsymbol{s}[k]$ using MPC [9].

We collect training demonstrations from the privileged expert in simulation. Training in simulation enables synthesis and use of unlimited training data for any desired trajectory, without putting the physical platform in danger. This includes maneuvers that stretch the abilities of even expert human pilots. To facilitate zero-shot simulation-to-reality transfer, the sensorimotor student does not directly access raw sensor input such as color images. Rather, the sensorimotor controller acts on an *abstraction* of the input, in the form of feature points extracted via classic computer vision. Such abstraction supports sample-efficient training, generalization, and simulation-to-reality transfer [27, 35].

The trained sensorimotor student does not rely on any privileged information and can be deployed directly on the physical platform. We deploy the trained controller to perform acrobatic maneuvers in the physical world, with no adaptation required.

The next section presents each aspect of our method in detail.

IV. Method

We define the task of flying acrobatic maneuvers with a quadrotor as a discrete-time, continuous-valued optimization problem. Our task is to find an end-to-end control policy $\pi\colon \mathbb{O} \to \mathbb{U}$, defined by a neural network, which minimizes the following finite-horizon objective:

$$\min_{\pi} J(\pi) = \mathbb{E}_{\rho(\pi)}\Big[\sum_{k=0}^{k=T} \mathcal{C}(\boldsymbol{\tau}_r[k], \boldsymbol{s}[k])\Big], \tag{1}$$

where $\mathcal{C}$ is a quadratic cost depending on a reference trajectory $\boldsymbol{\tau}_r[k]$ and the quadrotor state $\boldsymbol{s}[k]$, and $\rho(\pi)$ is the distribution of possible trajectories $\{(\boldsymbol{s}[0], \boldsymbol{o}[0], \boldsymbol{u}[0]), \dots, (\boldsymbol{s}[T], \boldsymbol{o}[T], \boldsymbol{u}[T])\}$ induced by the policy π.

We define the quadrotor state $\boldsymbol{s}[k] = [\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\omega}]$ as the platform position $\boldsymbol{p}$, its orientation quaternion $\boldsymbol{q}$, and their derivatives. Note that the agent π does not directly observe the state $\boldsymbol{s}[k]$. We further define the reference trajectory $\boldsymbol{\tau}_r[k]$ as a time sequence of reference states which describe the desired trajectory. We formulate the cost $\mathcal{C}$ as

$$\mathcal{C}(\boldsymbol{\tau}_r[k], \boldsymbol{s}[k]) = \boldsymbol{x}[k]^\top \mathcal{L} \boldsymbol{x}[k], \tag{2}$$

where $\boldsymbol{x}[k] = \boldsymbol{\tau}_r[k] - \boldsymbol{s}[k]$ denotes the difference between the state of the platform and the corresponding reference at time k, and $\mathcal{L}$ is a positive-semidefinite cost matrix.

A. Reference Trajectories

Both the privileged expert and the learned policy assume access to a reference trajectory $\boldsymbol{\tau}_r[k]$ that specifies an acrobatic maneuver. To ensure that such reference is dynamically feasible, it has to satisfy constraints that are imposed by the physical limits and the underactuated nature of the quadrotor platform. Neglecting aerodynamic drag and motor dynamics, the dynamics of the quadrotor can be modelled as

$$\begin{aligned} \dot{\boldsymbol{p}}_{\text{WB}} &= \boldsymbol{v}_{\text{WB}} \\ \dot{\boldsymbol{v}}_{\text{WB}} &= {}_{\text{W}}\boldsymbol{g} + \boldsymbol{q}_{\text{WB}} \odot \boldsymbol{c}_{\text{B}} \\ \dot{\boldsymbol{q}}_{\text{WB}} &= \frac{1}{2}\Lambda\left(\boldsymbol{\omega}_{\text{B}}\right) \cdot \boldsymbol{q}_{\text{WB}} \\ \dot{\boldsymbol{\omega}}_{\text{B}} &= \boldsymbol{J}^{-1} \cdot \left(\boldsymbol{\eta} - \boldsymbol{\omega}_{\text{B}} \times \boldsymbol{J} \cdot \boldsymbol{\omega}_{\text{B}}\right) , \end{aligned} \tag{3}$$

where $\boldsymbol{p}_{\text{WB}}$, $\boldsymbol{v}_{\text{WB}}$, $\boldsymbol{q}_{\text{WB}}$ denote the position, linear velocity, and orientation of the platform body frame with respect to the world frame. The gravity vector ${}_{\text{W}}\boldsymbol{g}$ is expressed in the world frame and $\boldsymbol{q}_{\text{WB}} \odot \boldsymbol{c}_{\text{B}}$ denotes the rotation of the mass-normalized thrust vector $\boldsymbol{c}_{\text{B}} = (0, 0, c)^\top$ by quaternion $\boldsymbol{q}_{\text{WB}}$. The time derivative of a quaternion $\boldsymbol{q} = (q_w, q_x, q_y, q_z)^\top$ is given by $\dot{\boldsymbol{q}} = \frac{1}{2}\Lambda(\boldsymbol{\omega}) \cdot \boldsymbol{q}$ and $\Lambda(\boldsymbol{\omega})$ is a skew-symmetric matrix of the vector $(0, \boldsymbol{\omega}^\top)^\top = (0, \omega_x, \omega_y, \omega_z)^\top$. The diagonal matrix $\boldsymbol{J} = \text{diag}(J_{xx}, J_{yy}, J_{zz})$ denotes the quadrotor inertia, and $\boldsymbol{\eta} \in \mathbb{R}^3$ are the torques acting on the body due to the motor thrusts.

Instead of directly planning in the full state space, we plan reference trajectories in the space of *flat outputs* $\boldsymbol{z} = [x, y, z, \psi]^\top$ proposed in [24], where x, y, z denote the position of the quadrotor and ψ is the yaw angle. It can be shown that any smooth trajectory in the space of flat outputs can be tracked by the underactuated platform (assuming reasonably bounded derivatives).

The core part of each acrobatic maneuver is a circular motion primitive with constant tangential velocity $\boldsymbol{v}_l$. The orientation of the quadrotor is constrained by the thrust vector the platform needs to produce. Consequently, the desired platform orientation is undefined when there is no thrust. To ensure a well-defined reference trajectory through the whole circular maneuver, we constrain the norm of the tangential velocity $\boldsymbol{v}_l$ to be larger by a margin ε than the critical tangential velocity that would lead to free fall at the top of the maneuver:

$$\|\boldsymbol{v}_l\| > \varepsilon\sqrt{rg}, \tag{4}$$

where r denotes the radius of the loop, $g = 9.81\,\text{m}\,\text{s}^{-2}$, and $\varepsilon = 1.1$.

While the circular motion primitives form the core part of the agile maneuvers, we use constrained polynomial trajectories to enter, transition between, and exit the maneuvers. A polynomial trajectory is described by four polynomial functions specifying the independent evolution of the components of $\boldsymbol{z}$ over time:

$$z_i(t) = \sum_{j=0}^{j=P_i} a_{ij} \cdot t^j \quad \text{for} \quad i \in \{0, 1, 2, 3\} \,. \tag{5}$$

We use polynomials of order $P_i = 7$ for the position components ($i = \{0, 1, 2\}$) of the flat outputs and $P_i = 2$ for the yaw component ($i = 3$). By enforcing continuity for both start ($t = 0$) and end ($t = t_m$) of the trajectory down to the 3rd derivative of position, the trajectory is fully constrained. We minimize the execution time t_m of the polynomial trajectories, while constraining the maximum speed, thrust, and body rates throughout the maneuver.

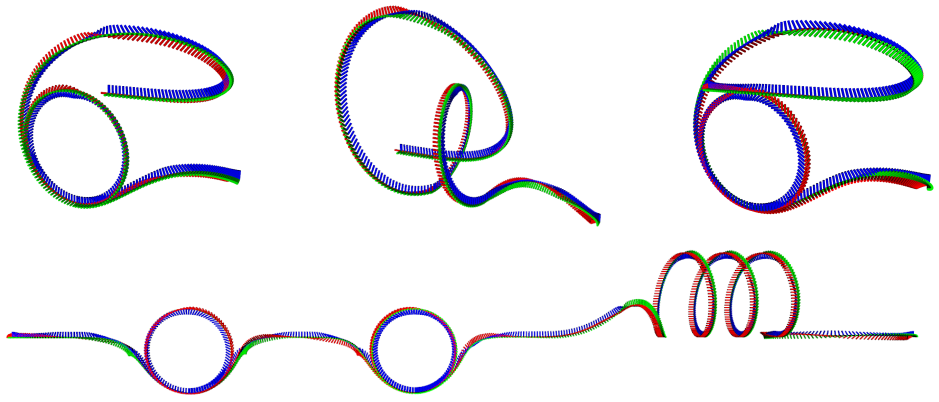

Fig. 3: Reference trajectories for acrobatic maneuvers. Top row, from left to right: Power Loop, Barrel Roll, and Matty Flip. Bottom row: Combo.

Finally, the trajectories are concatenated to the full reference trajectory, which is then converted back to the full state-space representation $\boldsymbol{\tau}_r(t)$ [24]. Subsequent sampling with a frequency of $50\,\mathrm{Hz}$ results in the discrete-time representation $\boldsymbol{\tau}_r[k]$ of the reference trajectory. Some example trajectories are illustrated in Figure 3.

B. Privileged Expert

Our privileged expert π^* consists of an MPC [9] that generates collective thrust and body rates via an optimization-based scheme. The controller operates on the simplified dynamical model of a quadrotor proposed in [26]:

$$\begin{aligned} \dot{\boldsymbol{p}}_{\mathrm{WB}} &= \boldsymbol{v}_{\mathrm{WB}} \\ \dot{\boldsymbol{v}}_{\mathrm{WB}} &= {}_{\mathrm{W}}\boldsymbol{g} + \boldsymbol{q}_{\mathrm{WB}} \odot \boldsymbol{c}_{\mathrm{B}} \\ \dot{\boldsymbol{q}}_{\mathrm{WB}} &= \frac{1}{2}\Lambda\left(\boldsymbol{\omega}_B\right) \cdot \boldsymbol{q}_{\mathrm{WB}} \end{aligned} \tag{6}$$

In contrast to the model (3), the simplified model neglects the dynamics of the angular rates. The MPC repeatedly optimizes the open-loop control problem over a receding horizon of N time steps and applies the first control command from the optimized sequence. Specifically, the action computed by the MPC is the first element of the solution to the following optimization problem:

$$\begin{aligned} \pi^* = \min_{\boldsymbol{u}} \Bigg\{ & \boldsymbol{x}[N]^\top \mathcal{Q} \boldsymbol{x}[N] \\ & + \sum_{k=1}^{N-1} \left(\boldsymbol{x}[k]^\top \mathcal{Q} \boldsymbol{x}[k] + \boldsymbol{u}[k]^\top \mathcal{R} \boldsymbol{u}[k] \right) \Bigg\} \\ \text{s.t.} \quad & \boldsymbol{r}(\boldsymbol{x}, \boldsymbol{u}) = 0 \\ & \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}) \leq 0, \end{aligned} \tag{7}$$

where $\boldsymbol{x}[k] = \boldsymbol{\tau}_r[k] - \boldsymbol{s}[k]$ denotes the difference between the state of the platform at time k and the corresponding reference $\boldsymbol{\tau}_r[k]$, $\boldsymbol{r}(\boldsymbol{x}, \boldsymbol{u})$ are equality constraints imposed by the system dynamics (6), and $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u})$ are optional bounds on inputs and states. $\mathcal{Q}, \mathcal{R}$ are positive-semidefinite cost matrices.

C. Learning

The sensorimotor controller is trained by imitating demonstrations provided by the privileged expert. While the expert has access to privileged information in the form of ground-truth state estimates, the sensorimotor controller does not access any privileged information and can be directly deployed in the physical world [5].

A lemma by Pan et al. [28] formally defines an upper bound between the expert and the student performance as

$$\begin{aligned} J(\pi) - J(\pi^*) &\leq C_{\pi^*} \mathbb{E}_{\rho(\pi)} \left[DW(\pi, \pi^*) \right] \\ &\leq C_{\pi^*} \mathbb{E}_{\rho(\pi)} \mathbb{E}_{\boldsymbol{u}^* \sim \pi^*} \mathbb{E}_{\boldsymbol{u} \sim \pi} [\|\boldsymbol{u}^* - \boldsymbol{u}\|], \end{aligned} \tag{8}$$

where $DW(\cdot, \cdot)$ is the Wasserstein metric [13] and C_{π^*} is a constant depending on the smoothness of expert actions. Finding an agent π with the same performance as the privileged controller π^* boils down to minimizing the discrepancy in actions between the two policies on the expected agent trajectories $\rho(\pi)$.

The aforementioned discrepancy can be minimized by an iterative supervised learning process known as DAGGER [31]. This process iteratively collects data by letting the student control the platform, annotating the collected observations with the experts' actions, and updating the student controller based on the supervised learning problem

$$\pi = \min_{\hat{\pi}} \mathbb{E}_{\boldsymbol{s}[k] \sim \rho(\pi)} [\|\boldsymbol{u}^*(\boldsymbol{s}[k]) - \hat{\pi}(\boldsymbol{o}[k])\|], \tag{9}$$

where $\boldsymbol{u}^*(\boldsymbol{s}[k])$ is the expert action and $\boldsymbol{o}[k]$ is the observation vector in the state $\boldsymbol{s}[k]$. Running this process for $O(N)$ iterations yields a policy π with performance $J(\pi) \leq J(\pi^*) + O(N)$ [31].

Naive application of this algorithm to the problem of agile flight in the physical world presents two major challenges: how can the expert access the ground-truth state $\boldsymbol{s}[k]$ and how can we protect the platform from damage when the partially trained student π is in control? To circumvent these challenges, we train *exclusively* in simulation. This significantly simplifies the training procedure, but presents a new hurdle: how do we minimize the difference between the sensory input received by the controller in simulation and reality?

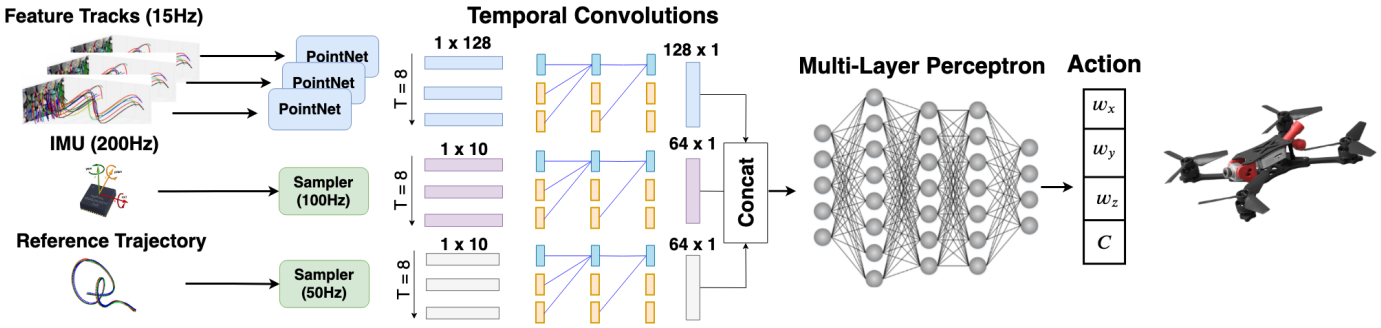

Fig. 4: Network architecture. The network receives a history of feature tracks, IMU measurements, and reference trajectories as input. Each input modality is processed using temporal convolutions and updated at different input rates. The resulting intermediate representations are processed by a multi-layer perceptron at a fixed output rate to produce collective thrust and body rate commands.

Our approach to bridging the gap between simulation and reality is to leverage *abstraction* [27]. Rather than operating on raw sensory input, our sensorimotor controller operates on an intermediate representation produced by a perception module [35]. This intermediate representation is more consistent across simulation and reality than raw visual input.

We now formally show that training a network on abstractions of sensory input reduces the gap between simulation and reality. Let $M(\boldsymbol{z} \mid \boldsymbol{s}), L(\boldsymbol{z} \mid \boldsymbol{s}) \colon \mathbb{S} \to \mathbb{O}$ denote the observation models in the real world and in simulation, respectively. Such models describe how an on-board sensor measurement $\boldsymbol{z}$ senses a state $\boldsymbol{s}$. We further define $\pi_r = \mathbb{E}_{\boldsymbol{o}_r \sim M(\boldsymbol{s})}[\pi(\boldsymbol{o}_r[k])]$ and $\pi_s = \mathbb{E}_{\boldsymbol{o}_s \sim L(\boldsymbol{s})}[\pi(\boldsymbol{o}_s[k])]$ as the realizations of the policy π in the real world and in simulation. The following lemma shows that, disregarding actuation differences, the distance between the observation models upper-bounds the gap in performance in simulation and reality.

Lemma 1: For a Lipschitz-continuous policy π the simulation-to-reality gap $J(\pi_r) - J(\pi_s)$ is upper-bounded by

$$J(\pi_r) - J(\pi_s) \leq C_{\pi_s} K \mathbb{E}_{\rho(\pi_r)}\big[DW(M, L)\big], \tag{10}$$

where K denotes the Lipschitz constant.

Proof: The lemma follows directly from (8) and the fact that

$$\begin{aligned} DW(\pi_r, \pi_s) &= \inf_{\gamma \in \Pi(\boldsymbol{o}_r, \boldsymbol{o}_s)} \mathbb{E}_{(\boldsymbol{o}_r, \boldsymbol{o}_s)}[d_p(\pi_r, \pi_s)] \\ &\leq K \inf_{\gamma \in \Pi(\boldsymbol{o}_r, \boldsymbol{o}_s)} \mathbb{E}_{(\boldsymbol{o}_r, \boldsymbol{o}_s)}[d_o(\boldsymbol{o}_r, \boldsymbol{o}_s)] \\ &= K \cdot DW(M, L), \end{aligned}$$

where d_o and d_p are distances in observation and action space, respectively. ■

We now consider the effect of abstraction of the input observations. Let f be a mapping of the observations such that

$$DW(f(M), f(L)) \leq DW(M, L). \tag{11}$$

The mapping f is task-dependent and is generally designed – with domain knowledge – to contain sufficient information to solve the task while being invariant to nuisance factors. In our case, we use feature tracks as an abstraction of camera frames. The feature tracks are provided by a visual-inertial odometry (VIO) system. In contrast to camera frames, feature tracks primarily depend on scene geometry, rather than surface appearance. We also make inertial measurements independent of environmental conditions, such as temperature and pressure, by integration and de-biasing. As such, our input representations fulfill the requirements of Eq. (11).

As the following lemma shows, training on such representations reduces the gap between task performance in simulation and the real world.

Lemma 2: A policy that acts on an abstract representation of the observations $\pi_f \colon f(\mathbb{O}) \to \mathbb{U}$ has a lower simulation-to-reality gap than a policy $\pi_o \colon \mathbb{O} \to \mathbb{U}$ that acts on raw observations.

Proof: The lemma follows directly from (10) and (11). ■

D. Sensorimotor Controller

In contrast to the expert policy π^*, the student policy π is only provided with onboard sensor measurements from the forward-facing camera and the IMU. There are three main challenges for the controller to tackle: (i) it should keep track of its state based on the provided inputs, akin to a visual-inertial odometry system [11, 8], (ii) it should be invariant to environments and domains, so as to not require retraining for each scene, and (iii) it should process sensor readings that are provided at different frequencies.

We represent the policy as a neural network that fulfills all of the above requirements. The network consists of three input branches that process visual input, inertial measurements, and the reference trajectory, followed by a multi-layer perceptron that produces actions. The architecture is illustrated in Fig. 4. Similarly to visual-inertial odometry systems [7, 8, 11], we provide the network with a representation of the platform state by supplying it with a history of length $L = 8$ of visual and inertial information.

To ensure that the learned policies are scene- and domain-independent, we provide the network with appropriate abstractions of the inputs instead of directly feeding raw inputs. We design these abstractions to contain sufficient information to solve the task while being invariant to environmental factors that are hard to simulate accurately and are thus unlikely to transfer from simulation to reality.

The distribution of raw IMU measurements depends on the exact sensor as well as environmental factors such as pressure and temperature. Instead of using the raw measurements as input to the policy, we preprocess the IMU signal by applying bias subtraction and gravity alignment. Modern visual-inertial odometry systems perform a similar pre-integration of the inertial data in their optimization backend [29]. The resulting inertial signal contains the estimated platform velocity, orientation, and angular rate.

We use the history of filtered inertial measurements, sampled at $100\,\mathrm{Hz}$, and process them using temporal convolutions [2]. Specifically, the inertial branch consists of a temporal convolutional layer with 128 filters, followed by three temporal convolutions with 64 filters each. A final fully-connected layer maps the signal to a 128-dimensional representation.

Another input branch processes a history of reference velocities, orientations, and angular rates. It has the same structure as the inertial branch. New reference states are added to the history at a rate of $50\,\mathrm{Hz}$.

For the visual signal, we use *feature tracks*, i.e. the motion of salient keypoints in the image plane, as an abstraction of the input. Feature tracks depend on the scene structure, ego-motion, and image gradients, but not on absolute pixel intensities. At the same time, the information contained in the feature tracks is sufficient to infer the ego-motion of the platform up to an unknown scale. Information about the scale can be recovered from the inertial measurements. We leverage the computationally efficient feature extraction and tracking frontend of VINS-Mono [29] to generate feature tracks. The frontend extracts Harris corners [14] and tracks them using the Lucas-Kanade method [22]. We perform geometric verification and exclude correspondences with a distance of more than 2 pixels from the epipolar line. We represent each feature track by a 5-dimensional vector that consists of the keypoint position, its displacement with respect to the previous keyframe (both on the rectified image plane), and the number of keyframes that the feature has been tracked (a measure of keypoint quality).

To facilitate efficient batch training, we randomly sample 40 keypoints per keyframe. The features are processed by a reduced version of the PointNet architecture proposed in [30] before we generate a fixed-size representation at each timestep. Specifically, we reduce the number of hidden layers from 6 to 4, with $32, 64, 128, 128$ filters, respectively, in order to reduce latency. The output of this subnetwork is reduced to a 128-dimensional vector by global average pooling over the feature tracks. The history of resulting hidden representations is then processed by a temporal convolutional network that has the same structure as the inertial and reference branches.

Finally, the outputs of the individual branches are concatenated and processed by a synchronous multi-layer perceptron with three hidden layers of size $128, 64, 32$. The final outputs are the body rates and collective thrust that are used to control the platform.

We account for the different input frequencies by allowing each of the input branches to operate asynchronously. Each branch operates independently from the others by generating an output only when a new input from the sensor arrives. The multi-layer perceptron uses the latest outputs from the asynchronous branches and operates at $100\,\mathrm{Hz}$. It outputs control commands at approximately the same rate due to its minimal computational overhead.

Fig. 5: Example images from simulation (left) and the real test environment (right).

E. Implementation Details

We use the Gazebo simulator to train our policies. Gazebo can model the physics of quadrotors with high fidelity using the RotorS extension [12]. We simulate the AscTec Hummingbird multirotor, which is equipped with a forward-facing fisheye camera. The platform is instantiated in a cubical simulated flying space with a side length of 70 meters. An example image is shown in Fig. 5 (left).

For the real-world experiments we use a custom quadrotor that weighs $1.15\,\mathrm{kg}$ and has a thrust-to-weight ratio of 4:1. We use a Jetson TX2 for neural network inference. Images and inertial measurements are provided by an Intel RealSense T265 camera.

We use an off-policy learning approach. We execute the trained policy, collect rollouts, and add them to a dataset. After 30 new rollouts are added, we train for 40 epochs on the entire dataset. This collect-rollouts-and-train procedure is repeated 5 times: there are 150 rollouts in the dataset by the end. We use the Adam optimizer [16] with a learning rate of $3e-4$. We always use the latest available model for collecting rollouts. We execute a student action only if the difference to the expert action is smaller than a threshold $t = 1.0$ to avoid crashes in the early stages of training. We double the threshold t every 30 rollouts. We perform a random action with a probability $p = 30\%$ at every stage of the training to increase the coverage of the state space. To facilitate transfer from simulation to reality, we randomize the IMU biases and the thrust-to-weight ratio of the platform by up to 10% of their nominal value in every iteration. We do not perform any randomization of the geometry and appearance of the scene during data collection.

V. Experiments

We design our evaluation procedure to address the following questions. Is the presented sensorimotor controller advantageous to a standard decomposition of state estimation and control? What is the role of input abstraction in facilitating transfer from simulation to reality? Finally, we validate our design choices with ablation studies.

A. Experimental Setup

We learn sensorimotor policies for three acrobatic maneuvers that are popular among professional drone pilots as well as a

Maneuver	Input			Power Loop		Barrel Roll		Matty Flip		Combo	
	Ref	IMU	FT	Error (↓)	Success (↑)	Error (↓)	Success (↑)	Error (↓)	Success (↑)	Error (↓)	Success (↑)
VIO-MPC	✓	✓	✓	43 ± 14	**100%**	79 ± 43	**100%**	92 ± 41	**100%**	164 ± 51	70%
Ours (Only Ref)	✓			250 ± 50	20%	485 ± 112	85%	340 ± 120	15%	∞	0 %
Ours (No IMU)	✓		✓	210 ± 100	30%	543 ± 95	85%	380 ± 100	20%	∞	0 %
Ours (No FT)	✓	✓		28 ± 8	**100%**	64 ± 24	95%	67 ± 29	**100%**	134 ± 113	85%
Ours	✓	✓	✓	$\mathbf{24 \pm 5}$	**100%**	$\mathbf{58 \pm 9}$	**100%**	$\mathbf{53 \pm 15}$	**100%**	$\mathbf{128 \pm 57}$	**95%**

TABLE I: Comparison of different variants of our approach with the baseline (VIO-MPC) in terms of the average tracking error in centimeters and the success rate. Results were averaged over 20 runs. Agents without access to IMU data perform poorly. An agent that has access only to IMU measurements has a significantly lower tracking error than the baseline. Adding feature tracks further improves tracking performance and success rate, especially for longer and more complicated maneuvers.

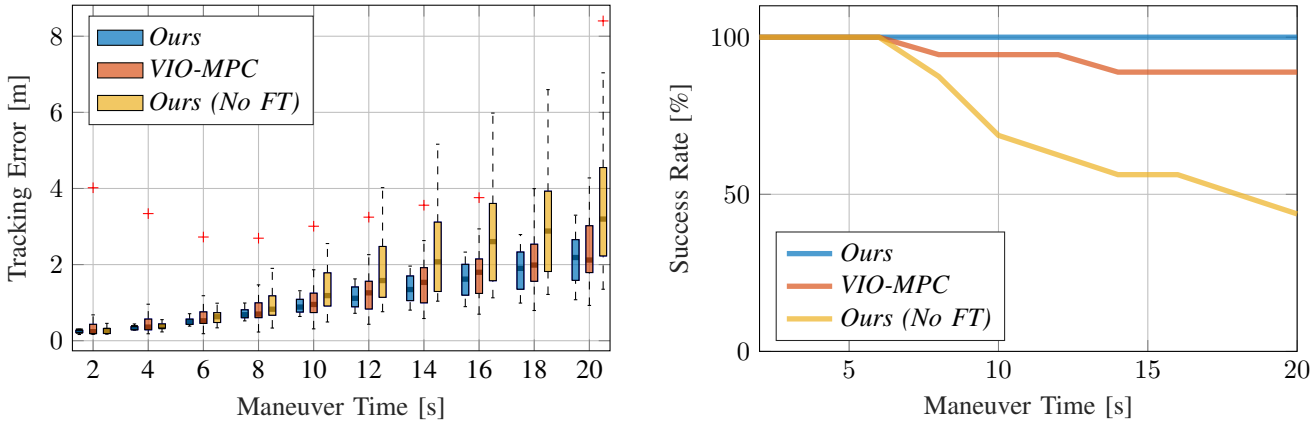

Fig. 6: Tracking error (left) and success rate (right) over time when a maneuver is executed repeatedly in simulation. The controllers were trained to complete the maneuver for six seconds and generalize well to longer sequences. Our learned controller, which leverages both IMU and visual data, provides consistently good performance without a single failure.

policy that consists of a sequence of multiple maneuvers.

1) Power Loop: Accelerate over a distance of $4\,\mathrm{m}$ to a speed of $4.5\,\mathrm{m\,s^{-1}}$ and enter a loop maneuver with a radius of $r = 1.5\,\mathrm{m}$.
2) Barrel Roll: Accelerate over a distance of $3\,\mathrm{m}$ to a speed of $4.5\,\mathrm{m\,s^{-1}}$ and enter a roll maneuver with a radius of $r = 1.5\,\mathrm{m}$.
3) Matty Flip: Accelerate over a distance of $4\,\mathrm{m}$ to a speed of $4.5\,\mathrm{m\,s^{-1}}$ while yawing $180°$ and enter a backwards loop maneuver with a radius of $r = 1.5\,\mathrm{m}$.
4) Combo: This sequence starts with a triple Barrel Roll, followed by a double Power Loop, and ends with a Matty Flip. The full maneuver is executed without stopping between maneuvers.

The maneuvers are listed by increasing difficulty. The trajectories of these maneuvers are illustrated in Fig. 3. They contain high accelerations and fast angular velocities around the body axes of the platform. All maneuvers start and end in the hover condition.

For comparison, we construct a strong baseline by combining visual-inertial odometry [29] and model predictive control [9]. Our baseline receives the same inputs as the learned controllers: inertial measurements, camera images, and a reference trajectory.

We define two metrics to compare different approaches. We measure the average root mean square error in meters of the reference position with respect to the true position of the platform during the execution of the maneuver. Note that we can measure this error only for simulation experiments, as it requires exact state estimation. We thus define a second metric, the average success rate for completing a maneuver. In simulation, we define success as not crashing the platform into any obstacles during the maneuver. For the real-world experiments, we consider a maneuver successful if the safety pilot did not have to intervene during the execution and the maneuver was executed correctly.

B. Experiments in Simulation

We first evaluate the performance for individual maneuvers in simulation. The results are summarized in Table I. The learned sensorimotor controller that has access to both visual and inertial data (*Ours*) is consistently the best across all maneuvers. This policy exhibits a lower tracking error by up to 45% in comparison to the strong *VIO-MPC* baseline. The baseline can complete the simpler maneuvers with perfect success rate, but generally has higher tracking error due to drift in state estimation. The gap between the baseline and our controller widens for longer and more difficult sequences.

Table I also highlights the relative importance of the input modalities. Policies that only receive the reference trajectories but no sensory input (*Ours (Only Ref)*) – effectively operating open-loop – perform poorly across all maneuvers. Policies

Input	Train		Test 1		Test 2	
	Error (↓)	Success (↑)	Error (↓)	Success (↑)	Error (↓)	Success (↑)
Image	90 ± 32	80%	∞	0%	∞	0%
Ours	$\mathbf{53 \pm 15}$	**100%**	$\mathbf{58 \pm 18}$	**100%**	$\mathbf{61 \pm 11}$	**100%**

TABLE II: Sim-to-sim transfer for different visual input modalities. Policies that directly rely on images as input do not transfer to scenes with novel appearance (Test 1, Test 2). Feature tracks enable reliable transfer. Results are averaged over 10 runs.

Maneuver	Power Loop	Barrel Roll	Matty Flip
Ours (No FT)	100%	90%	100%
Ours	100%	100%	100%

TABLE III: Success rate across 10 runs on the physical platform.

that have access to visual input but not to inertial data *(Ours (No IMU))* perform similarly poorly since they do not have sufficient information to determine the absolute scale of the ego-motion. On the other hand, policies that only rely on inertial data for sensing *(Ours (No FT))* are able to safely fly most maneuvers. Even though such controllers only have access to inertial data, they exhibit significantly lower tracking error than the *VIO-MPC* baseline. However, the longer the maneuver, the larger the drift accumulated by purely-inertial *(Ours (No FT))* controllers. When both inertial and visual data is incorporated *(Ours)*, drift is reduced and accuracy improves. For the longest sequence *(Combo)*, the abstracted visual input raises the success rate by 10 percentage points.

Fig. 6 analyzes the evolution of tracking errors and success rates of different methods over time. For this experiment, we trained a policy to repeatedly fly barrel rolls for four seconds. We evaluate robustness and generalization of the learned policy by flying the maneuver for up to 20 seconds at test time. The results again show that (abstracted) visual input reduces drift and increases robustness. The controller that has access to both visual and inertial data *(Ours)* is able to perform barrel rolls for 20 seconds without a single failure.

To validate the importance of input abstraction, we compare our approach to a network that uses raw camera images instead of feature tracks as visual input. This network substitutes the PointNet in the input branch with a 5-layer convolutional network that directly operates on image pixels, but retains the same structure otherwise. We train this network on the Matty Flip and evaluate its robustness to changes in the background images. The results are summarized in Table II. In the training environment, the image-based network has a success rate of only 80%, with a 58% higher tracking error than the controller that receives an abstraction of the visual input in the form of feature tracks *(Ours)*. We attribute this to the higher sample complexity of learning from raw pixels [35]. Even more dramatically, the image-based controller fails completely when tested with previously unseen background images (*Test 1*, *Test 2*). (For backgrounds, we use randomly sampled images from the COCO dataset [19].) In contrast, our approach maintains a 100% success rate in these conditions.

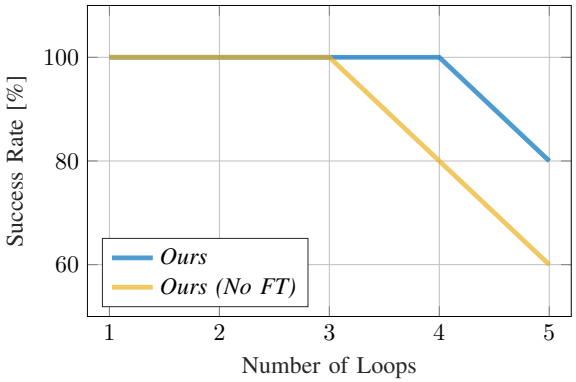

Fig. 7: Number of successful back-to-back Power Loops on the physical quadrotor before the human expert pilot had to intervene. Results are averaged over 5 runs.

C. Deployment in the Physical World

We now perform direct simulation-to-reality transfer of the learned controllers. We use exactly the same sensorimotor controllers that were learned in simulation and quantitatively evaluated in Table I to fly a physical quadrotor, with no fine-tuning. Despite the differences in the appearance of simulation and reality (see Fig. 5), the abstraction scheme we use facilitates successful deployment of simulation-trained controllers in the physical world. The controllers are shown in action in the supplementary video.

We further evaluate the learned controllers with a series of quantitative experiments on the physical platform. The success rates of different maneuvers are shown in Table III. Our controllers can fly all maneuvers with no intervention. An additional experiment is presented in Fig. 7, where a controller that was trained for a single loop was tested on repeated execution of the maneuver with no breaks. The results indicate that using all input modalities, including the abstracted visual input in the form of feature tracks, enhances robustness.

VI. Conclusion

Our approach is the first to enable an autonomous flying machine to perform a wide range of acrobatics maneuvers that are highly challenging even for expert human pilots. The approach relies solely on onboard sensing and computation, and leverages sensorimotor policies that are trained entirely in simulation. We have shown that designing appropriate abstractions of the input facilitates direct transfer of the policies from simulation to physical reality. The presented methodology is not limited to autonomous flight and can enable progress in other areas of robotics.

Acknowledgement

This work was supported by the Intel Network on Intelligent Systems, the National Centre of Competence in Research Robotics (NCCR) through the Swiss National Science Foundation, and the SNSF-ERC Starting Grant.

References

[1] Pieter Abbeel, Adam Coates, and Andrew Y Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv:1803.01271* (2018).

[3] Adam Bry, Charles Richter, Abraham Bachrach, and Nicholas Roy. "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments". In: *The International Journal of Robotics Research* 34.7 (2015), pp. 969–1002.

[4] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian D. Reid, and John J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[5] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. "Learning by Cheating". In: *Conference on Robot Learning (CoRL)*. 2019.

[6] Ying Chen and Néstor O Pérez-Arancibia. "Controller Synthesis and Performance Optimization for Aerobatic Quadrotor Flight". In: *IEEE Transactions on Control Systems Technology* (2019), pp. 1–16.

[7] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. "VINet: Visual-inertial odometry as a sequence-to-sequence learning problem". In: *AAAI Conference on Artificial Intelligence*. 2017.

[8] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct sparse odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* 40.3 (2018), pp. 611–625.

[9] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. "PAMPC: Perception-aware model predictive control for quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[10] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[11] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry". In: *IEEE Transactions on Robotics* 3.1 (2017), pp. 1–21.

[12] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. "RotorS–A modular Gazebo MAV simulator framework". In: *Robot Operating System (ROS)*. Springer, 2016, pp. 595–625.

[13] Alison L Gibbs and Francis Edward Su. "On choosing and bounding probability metrics". In: *International Statistical Review* 70.3 (2002), pp. 419–435.

[14] Christopher G. Harris and Mike Stephens. "A Combined Corner and Edge Detector". In: *Proceedings of the Alvey Vision Conference*. 1988.

[15] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. "Control of a quadrotor with reinforcement learning". In: *IEEE Robotics and Automation Letters* 2.4 (2017), pp. 2096–2103.

[16] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. 2015.

[17] Keuntaek Lee, Jason Gibson, and Evangelos A Theodorou. "Aggressive Perception-Aware Navigation using Deep Optical Flow Dynamics and PixelMPC". In: *arXiv:2001.02307* (2020).

[18] Shuo Li, Ekin Ozturk, Christophe De Wagter, Guido de Croon, and Dario Izzo. "Aggressive Online Control of a Quadrotor via Deep Network Representations of Optimality Principles". In: *arXiv:1912.07067* (2019).

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision (ECCV)*. 2014.

[20] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU". In: *IEEE Robotics and Automation Letters* 2.2 (2016), pp. 404–411.

[21] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. "Deep Drone Racing: From Simulation to Reality with Domain Randomization". In: *IEEE Transactions on Robotics* 36.1 (2020), pp. 1–14.

[22] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence*. 1981.

[23] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. "A simple learning strategy for high-speed quadrocopter multi-flips". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 1642–1648.

[24] Daniel Mellinger and Vijay Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 2520–2525.

[25] Daniel Mellinger, Nathan Michael, and Vijay Kumar. "Trajectory generation and control for precise aggressive maneuvers with quadrotors". In: *The International Journal of Robotics Research* 31.5 (2012), pp. 664–674.

[26] Mark W Mueller, Markus Hehn, and Raffaello D'Andrea. "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2013, pp. 3480–3486.

[27] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladen Koltun. "Driving Policy Transfer via Modularity and Abstraction". In: *Conference on Robot Learning (CoRL)*. 2018.

[28] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. "Agile Autonomous Driving Using End-to-End Deep Imitation Learning". In: *Robotics: Science and Systems (RSS)*. 2018.

[29] Tong Qin, Peiliang Li, and Shaojie Shen. "VINS-Mono: A robust and versatile monocular visual-inertial state estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[30] René Ranftl and Vladlen Koltun. "Deep fundamental matrix estimation". In: *European Conference on Computer Vision (ECCV)*. 2018.

[31] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[32] Fereshteh Sadeghi and Sergey Levine. "CAD2RL: Real Single-Image Flight Without a Single Real Image". In: *Robotics: Science and Systems (RSS)*. 2017.

[33] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. "Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor". In: *Robotics: Science and Systems (RSS)*. 2013.

[34] Tianhao Zhang, Gregory Kahn, Sergey Levine, and Pieter Abbeel. "Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 528–535.

[35] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. "Does computer vision matter for action?" In: *Science Robotics* 4.30 (2019).