2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing

Elia Kaufmann[1], Mathias Gehrig[1], Philipp Foehn[1],
René Ranftl[2], Alexey Dosovitskiy[2], Vladlen Koltun[2], Davide Scaramuzza[1]

***Abstract*— Autonomous micro aerial vehicles still struggle with fast and agile maneuvers, dynamic environments, imperfect sensing, and state estimation drift. Autonomous drone racing brings these challenges to the fore. Human pilots can fly a previously unseen track after a handful of practice runs. In contrast, state-of-the-art autonomous navigation algorithms require either a precise metric map of the environment or a large amount of training data collected in the track of interest. To bridge this gap, we propose an approach that can fly a new track in a previously unseen environment without a precise map or expensive data collection. Our approach represents the global track layout with coarse gate locations, which can be easily estimated from a single demonstration flight. At test time, a convolutional network predicts the poses of the closest gates along with their uncertainty. These predictions are incorporated by an extended Kalman filter to maintain optimal maximum-a-posteriori estimates of gate locations. This allows the framework to cope with misleading high-variance estimates that could stem from poor observability or lack of visible gates. Given the estimated gate poses, we use model predictive control to quickly and accurately navigate through the track. We conduct extensive experiments in the physical world, demonstrating agile and robust flight through complex and diverse previously-unseen race tracks. The presented approach was used to win the IROS 2018 Autonomous Drone Race Competition, outracing the second-placing team by a factor of two.**

SUPPLEMENTARY MATERIAL

Video: `https://youtu.be/UuQvijZcUSc`

I. INTRODUCTION

First-person view (FPV) drone racing is a fast-growing sport, in which human pilots race micro aerial vehicles (MAVs) through tracks via remote control. Drone racing provides a natural proving ground for vision-based autonomous drone navigation. This has motivated competitions such as the annual IROS Autonomous Drone Race [17] and the recently announced AlphaPilot Innovation Challenge, an autonomous drone racing competition with more than 2 million US dollars in cash prizes.

To successfully navigate a race track, a drone has to continually sense and interpret its environment. It has to be robust to cluttered and possibly dynamic track layouts. It needs precise planning and control to support the aggressive maneuvers required to traverse a track at high speed. Drone racing thus crystallizes some of the central outstanding issues in robotics. Algorithms developed for drone racing can benefit robotics in general and can contribute to areas such as autonomous transportation, delivery, and disaster relief.

Fig. 1: A quadrotor flies through an indoor track. Our approach uses optimal filtering to incorporate estimates from a deep perception system. It can race a new track after a single demonstration.

Traditional localization-based approaches for drone navigation require precomputing a precise 3D map of the environment against which the MAV is localized. Thus, while previous works demonstrated impressive results in controlled settings [18], these methods are difficult to deploy in new environments where a precise map is not available. Additionally, they fail in the presence of dynamic objects such as moving gates, have inconsistent computational overhead, and are prone to failure under appearance changes such as varying lighting.

Recent work has shown that deep networks can provide drones with robust perception capabilities and facilitate safe navigation even in dynamic environments [9], [8]. However, current deep learning approaches to autonomous drone racing require a large amount of training data collected in the same track. This stands in contrast to human pilots, who can quickly adapt to new tracks by leveraging skills acquired in the past.

In this paper, we develop a deep-learning-aided approach to autonomous drone racing capable of fast adaptation to new tracks, without the need for building precise maps or collecting large amounts of data from the track. We represent a track by coarse locations of a set of gates, which can be easily acquired in a single demonstration flight through the track. These recorded gates represent the rough global

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.

[1] Robotics and Perception Group, Dep. of Informatics, University of Zurich, Dep. of Neuroinformatics, University of Zurich and ETH Zurich.
[2] Intelligent Systems Lab, Intel

layout of the track. At test time, the local track configuration is estimated by a convolutional network that predicts the location of the closest gate together with its uncertainty, given the currently observed image. The network predictions and uncertainties are continuously incorporated using an extended Kalman filter (EKF) to derive optimal maximum-a-posteriori estimates of gate locations. This allows the framework to cope with misleading high-variance estimates that could stem from bad observability or complete absence of visible gates. Given these estimated gate locations, we use model predictive control to quickly and accurately navigate through them.

We evaluate the proposed method in simulation and on a real quadrotor flying fully autonomously. Our algorithm runs onboard on a computationally constrained platform. We show that the presented approach can race a new track after only a single demonstration, without any additional training or adaptation. Integration of the estimated gate positions is crucial to the success of the method: a purely image-based reactive approach only shows non-trivial performance in the simplest tracks. We further demonstrate that the proposed method is robust to dynamic changes in the track layout induced by moving gates.

The presented approach was used to win the IROS Autonomous Drone Race Competition, held in October 2018. An MAV controlled by the presented approach placed first in the competition, traversing the eight gates of the race track in 31.8 seconds. In comparison, the second-place entry completed the track in 61 seconds, and the third in 90.1 seconds.

II. Related Work

Traditional approaches to autonomous MAV navigation build on visual inertial odometry (VIO) [5], [1], [13], [24] or simultaneous localization and mapping (SLAM) [21], [23], which are used to provide a pose estimate of the drone relative to an internal metric map [14], [4]. While these methods can be used to perform visual teach and repeat [4], they are not concerned with trajectory generation [16], [20]. Furthermore, teach and repeat assumes a static world and accurate pose estimation: assumptions that are commonly violated in the real world.

The advent of deep learning has inspired alternative solutions to autonomous navigation that aim to overcome these limitations. These approaches typically predict actions directly from images. Output representations range from predicting discrete navigation commands (classification in action space) [11], [7], [15] to direct regression of control signals [19]. A different line of work combines network predictions with model predictive control by regressing the cost function from a single image [2].

In the context of drone racing, Kaufmann et al. [9] proposed an intermediate representation in the form of a goal direction and desired speed. The learned policy imitates an optimal trajectory [16] through the track. An advantage of this approach is that it can navigate even when no gate is in view, by exploiting track-specific context and background information. A downside, however, is the need for a large amount of labeled data collected directly in the track of interest in order to learn this contextual information. As a result, the approach is difficult to deploy in new environments.

Jung et al. [8] consider the problem of autonomous drone navigation in a previously unseen track. They use line-of-sight guidance combined with a deep-learning-based gate detector. As a consequence, the next gate to be traversed has to be in view at all times. Additionally, gates cannot be approached from an acute angle since the algorithm does not account for gate rotation. The method is thus applicable only to relatively simple environments, where the next gate is always visible.

Our approach addresses the limitations of both works [9], [8]. It operates reliably even when no gate is in sight, while eliminating the need to retrain the perception system for every new track. This enables rapid deployment in complex novel tracks.

III. Methodology

We address the problem of robust autonomous flight through a predefined, ordered set of possibly spatially perturbed gates. Our approach comprises three subsystems: perception, mapping, and combined planning and control. The perception system takes as input a single image from a forward-facing camera and estimates both the relative pose of the next gate and a corresponding uncertainty measure. The mapping system receives the output of the perception system together with the current state estimate of the quadrotor and produces filtered estimates of gate poses. The gate poses are used by the planning system to maintain a set of waypoints through the track. These waypoints are followed by a control pipeline that generates feasible receding-horizon trajectories and tracks them.

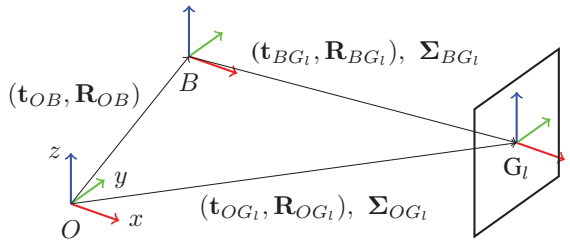


Fig. 2: Relation of odometry O, body B, and gate frame G_l.

A. Notation and Frame Convention

We denote all scalars by lowercase letters x, vectors by lowercase bold letters $\mathbf{x}$, and matrices by bold uppercase letters $\mathbf{X}$. Estimated values are written as $\hat{x}$, measured values as $\tilde{x}$.

The relevant coordinate frames are the odometry frame O, the body frame B, and the gate frames G_l, where $l \in \{1, \ldots, N_l\}$ and N_l is the number of gates. A schematic overview of the relation between coordinate frames is shown in Figure 2. The odometry frame O is the global VIO reference frame. The relation between the body frame B and the odometry frame O is given by the rotation $\mathbf{R}_{OB}$ and

Fig. 3: We collected training data for the perception system in 5 different environments. From left to right: flying room, outdoor urban environment, atrium, outdoor countryside, garage.

translation $\mathbf{t}_{OB}$. This transform is acquired through a visual inertial pose estimator. The prediction $(\tilde{\mathbf{t}}_{BG_l}, \tilde{\mathbf{R}}_{BG_l})$ is provided together with a corresponding uncorrelated covariance in polar coordinates $\tilde{\mathbf{\Sigma}}_{BG_l,pol} = \text{diag}(\tilde{\boldsymbol{\sigma}}^2_{BG_l,pol})$ of the gate's pose in the body frame. In parallel, we maintain an estimate of each gate pose $(\hat{\mathbf{t}}_{OG_l}, \hat{\mathbf{R}}_{OG_l})$ along with its covariance $\hat{\mathbf{\Sigma}}_{OG_l} = \text{cov}\left(\hat{\mathbf{t}}_{OG_l}, \hat{\mathbf{R}}_{OG_l}\right)$ in the odometry frame. This has the advantage that gate poses can be updated independently of each other.

B. Perception System

1) Architecture: The deep network takes as input a 320×240 RGB image and regresses both the mean $\tilde{\mathbf{z}}_{BG_l,pol} = [\tilde{r}, \tilde{\theta}, \tilde{\psi}, \tilde{\phi}]^\top \in \mathbb{R}^4$ and the variance $\tilde{\boldsymbol{\sigma}}^2_{BG_l,pol} \in \mathbb{R}^4$ of a multivariate normal distribution that describes the current estimate of the next gate's pose. Our choice of output distribution is motivated by the fact that we use an EKF to estimate the joint probability distribution of a gate's pose, which is known to be optimal for identical and independently distributed white noise with known covariance. The mean represents the prediction of the relative position and orientation of the gate with respect to the quadrotor in spherical coordinates. We found this to be advantageous compared to a Cartesian representation since it decouples distance estimation from the position of the gates in image coordinates. We use a single angle $\tilde{\phi}$ to describe the relative horizontal orientation of the gate, since the gravity direction is known from the IMU. Furthermore, we assume that gates are always upright and can be traversed horizontally along the normal direction. Specifically, $\tilde{\phi}$ is measured between the quadrotor's current heading and the gate's heading.

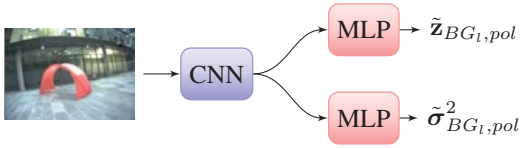


Fig. 4: Schematic illustration of the network architecture. Image features are extracted by a CNN [15] and passed to two separate MLPs to regress $\tilde{\mathbf{z}}_{BG_l,pol}$ and $\tilde{\boldsymbol{\sigma}}^2_{BG_l,pol}$, respectively.

The overall structure of the deep network is shown in Figure 4. First, the input image is processed by a Convolutional Neural Network (CNN), based on the shallow DroNet architecture [15]. The extracted features are then processed by two separate multilayer perceptrons (MLPs) that estimate the mean $\tilde{\mathbf{z}}_{BG_l,pol}$ and the variance $\tilde{\boldsymbol{\sigma}}^2_{BG_l,pol}$ of a multivariate normal distribution, respectively. A similar network architecture for mean-variance estimation was proposed in [22].

2) Training Procedure: We train the network in two stages.

In the first stage, the parameters of the CNN and $\text{MLP}_{\mathbf{z}}$, denoted by $\boldsymbol{\theta}_{\text{CNN}}$ and $\boldsymbol{\theta}_{\mathbf{z}}$, are jointly learned by minimizing a loss over groundtruth poses for images with visible gates:

$$\{\boldsymbol{\theta}^*_{\text{CNN}}, \boldsymbol{\theta}^*_{\tilde{\mathbf{z}}_i}\} = \arg\min_{\boldsymbol{\theta}_{\text{CNN}}, \boldsymbol{\theta}_{\tilde{\mathbf{z}}_i}} \sum_{i=1}^{N} ||\mathbf{y}_i - \tilde{\mathbf{z}}_i||^2_2, \tag{1}$$

where $\mathbf{y}_i$ denotes the groundtruth pose and N denotes the dataset size.

In the second stage, the training set is extended to also include images that do not show visible gates. In this stage only the parameters $\boldsymbol{\theta}_{\sigma^2}$ of the subnetwork MLP_{σ^2} are trained, while keeping the other weights fixed. We minimize the loss function proposed by [22], which amounts to the negative log-likelihood of a multivariate normal distribution with uncorrelated covariance:

$$-\log p\left(\mathbf{y} \mid \tilde{\mathbf{z}}_i, \tilde{\boldsymbol{\sigma}}^2\right) \propto \sum_{j=1}^{4} \log \tilde{\sigma}^2_j + \frac{\left(y_j - \tilde{z}_j\right)^2}{\tilde{\sigma}^2_j}. \tag{2}$$

Our use of mean-variance estimation is motivated by studies that have shown that it is a computationally efficient way to obtain uncertainty estimates [10].

3) Training Data Generation: We collect a set of images from the forward-facing camera on the drone and associate each image with the relative pose of the gate with respect to the body frame of the quadrotor. In real-world experiments, we use the quadrotor and leverage the onboard state estimation pipeline to generate training data. The platform is initialized at a known position relative to a gate and subsequently carried through the environment while collecting images and corresponding relative gate poses. To collect training data, it is not necessary to have complete tracks available. A single gate placed in different environments suffices, as the perception system only needs to estimate the relative pose with respect to the next gate at test time. Moreover, in contrast to Kaufmann et al. [9], the perception system is never trained on data from tracks and environments it is later deployed in.

Fig. 5: Our platform, equipped with an Intel UpBoard and a Qualcomm Snapdragon Flight.

C. Mapping System

The mapping system takes as input a measurement from the perception system and outputs a filtered estimate of the current track layout. By correcting the gates with the measurements from the CNN, gate displacement and accumulated VIO drift can be compensated for. The mapping part of our pipeline can be divided into two stages: measurement assignment stage and filter stage.

1) Measurement Assignment: We maintain a map of all gates $l = 1...N_l$ with states $\hat{\mathbf{x}}_{OG_l} = [\hat{\mathbf{t}}_{OG_l}, \hat{\phi}_{OG_l}]^\top$ corresponding to gate translation $\hat{\mathbf{t}}_{OG_l}$ and yaw $\hat{\phi}_{OG_l}$ with respect to the odometry frame O. The output of the perception system is used to update the pose $\hat{\mathbf{x}}_{OG_l}$ of the next gate to be passed. To assign a measurement to a gate, the measurement is transformed into the odometry frame and assigned to the closest gate. If a measurement is assigned to a gate that is not the next gate to be passed, it is discarded as an outlier. We keep track of the next gate by detecting gate traversals. The detection of a gate traversal is done by expressing the quadrotor's current position in a gate-based coordinate frame. In this frame, the condition for traversal can be expressed as

$$ {}_{G_l}\hat{\mathbf{t}}_{G_lB,x} \geq 0. \tag{3}$$

2) Extended Kalman Filter: The prediction of the network in body frame B is given by $\tilde{\mathbf{z}}_{BG,pol} = [\tilde{r}, \tilde{\theta}, \tilde{\psi}, \tilde{\phi}]^\top$ containing the spherical coordinates $[\tilde{r}, \tilde{\theta}, \tilde{\psi}]^\top$ and yaw $\tilde{\phi}$ of the gate, and the corresponding variance $\tilde{\boldsymbol{\sigma}}^2_{BG,pol}$. The transformation into the Cartesian representation $\tilde{\mathbf{z}}_{BG}$ leads to

$$\tilde{\mathbf{z}}_{BG} = \mathbf{f}(\tilde{\mathbf{z}}_{BG,pol}) = \begin{bmatrix} \tilde{r}\sin\tilde{\theta}\cos\tilde{\psi} \\ \tilde{r}\sin\tilde{\theta}\sin\tilde{\psi} \\ \tilde{r}\cos\tilde{\theta} \\ \tilde{\phi} \end{bmatrix} \tag{4}$$

$$\tilde{\boldsymbol{\Sigma}}_{BG} = \mathbf{J}_\mathbf{f}|_{\tilde{\mathbf{z}}_{pol}} \tilde{\boldsymbol{\Sigma}}_{BG,pol} \mathbf{J}_\mathbf{f}^\top|_{\tilde{\mathbf{z}}_{pol}}, \tag{5}$$

where $\mathbf{J}_{\mathbf{f},i,j} = \frac{\partial f_i}{\partial x_{pol,j}}$ is the Jacobian of the conversion function $\mathbf{f}$ and $\mathbf{J}_\mathbf{f}|_{\mathbf{z}_{pol}}$ is its evaluation at $\mathbf{z}_{pol}$. To integrate neural network predictions reliably into a map with prior knowledge of the gates, we represent each gate with its own EKF. We treat the prediction $\tilde{\mathbf{z}}_{BG}$ and $\tilde{\boldsymbol{\Sigma}}_{BG}$ at each time step as a measurement and associated variance, respectively. Similar to the state, $\tilde{\mathbf{z}}_{BG} = [\tilde{\mathbf{t}}^\top_{BG}, \tilde{\phi}_{BG}]^\top$ consists of a translation $\tilde{\mathbf{t}}_{BG}$ and rotation $\tilde{\phi}_{BG}$ around the world z-axis. Since our measurement and states have different origin frames, we can formulate the EKF measurement as follows:

$$\begin{aligned} \tilde{\mathbf{z}}_k &= \mathbf{H}_k\hat{\mathbf{x}}_k + \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k) \\ \mathbb{E}[\tilde{\mathbf{z}}_k] &= \begin{bmatrix} \mathbf{R}^{-1}_{OB,k}\ {}_O\mathbf{t}_{OG,k} - \mathbf{R}^{-1}_{OB,k}\ {}_O\mathbf{t}_{OB,k} \\ \phi_{OG,k} - \phi_{OB,k} \end{bmatrix}. \end{aligned} \tag{6}$$

Now with $\hat{\mathbf{x}}_k = [{}_O\mathbf{t}_{OG,k}, \quad \phi_{OG,k}]^\top$ we can write

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{R}^{-1}_{OB,k} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \tag{7}$$

$$\boldsymbol{\mu}_k = \begin{bmatrix} -\mathbf{R}^{-1}_{OB,k}\ {}_O\mathbf{t}_{OB,k} \\ -\phi_{OB,k} \end{bmatrix} \qquad \boldsymbol{\Sigma}_k = \tilde{\boldsymbol{\Sigma}}_{BG,k} \tag{8}$$

and, due to identity process dynamics and process covariance $\boldsymbol{\Sigma}_Q$, our prediction step becomes

$$\hat{\mathbf{x}}^*_{k+1} = \hat{\mathbf{x}}_k \qquad \hat{\mathbf{P}}^*_{k+1} = \hat{\mathbf{P}}_k + \boldsymbol{\Sigma}_Q. \tag{9}$$

The a-posteriori filter update can be summarized as follows:

$$\begin{aligned} \mathbf{K}_k &= \hat{\mathbf{P}}^*_k\mathbf{H}_k \left(\tilde{\boldsymbol{\Sigma}}_{BG,k} + \mathbf{H}_k\hat{\mathbf{P}}^*_k\mathbf{H}^\top_k\right)^{-1} \\ \hat{\mathbf{x}}_{k+1} &= \hat{\mathbf{x}}^*_k + \mathbf{K}_k(\tilde{\mathbf{z}}_k - \boldsymbol{\mu}_k - \mathbf{H}_k\hat{\mathbf{x}}^*_k) \\ \hat{\mathbf{P}}_{k+1} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\hat{\mathbf{P}}^*_k\,(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^\top + \mathbf{K}_k\tilde{\boldsymbol{\Sigma}}_{BG,k}\mathbf{K}^\top_k \end{aligned} \tag{10}$$

with $\hat{\mathbf{P}}_k$ as the estimated covariance and the superscript * indicating the a-priori predictions.

D. Planning and Control System

The planning and control stage is split into two asynchronous modules. First, low-level waypoints are generated from the estimated gate position and a desired path is generated by linearly interpolating between the low-level waypoints. Second, locally feasible control trajectories are planned and tracked using a model predictive control scheme.

1) Waypoint Generation: For each gate in our map we generate two waypoints: one lying in front of the gate relative to the current quadrotor position and one lying after the gate. Both waypoints are set with a positive and negative offset $\mathbf{p}_{wp,l\pm}$ in the x direction with respect to the gate l:

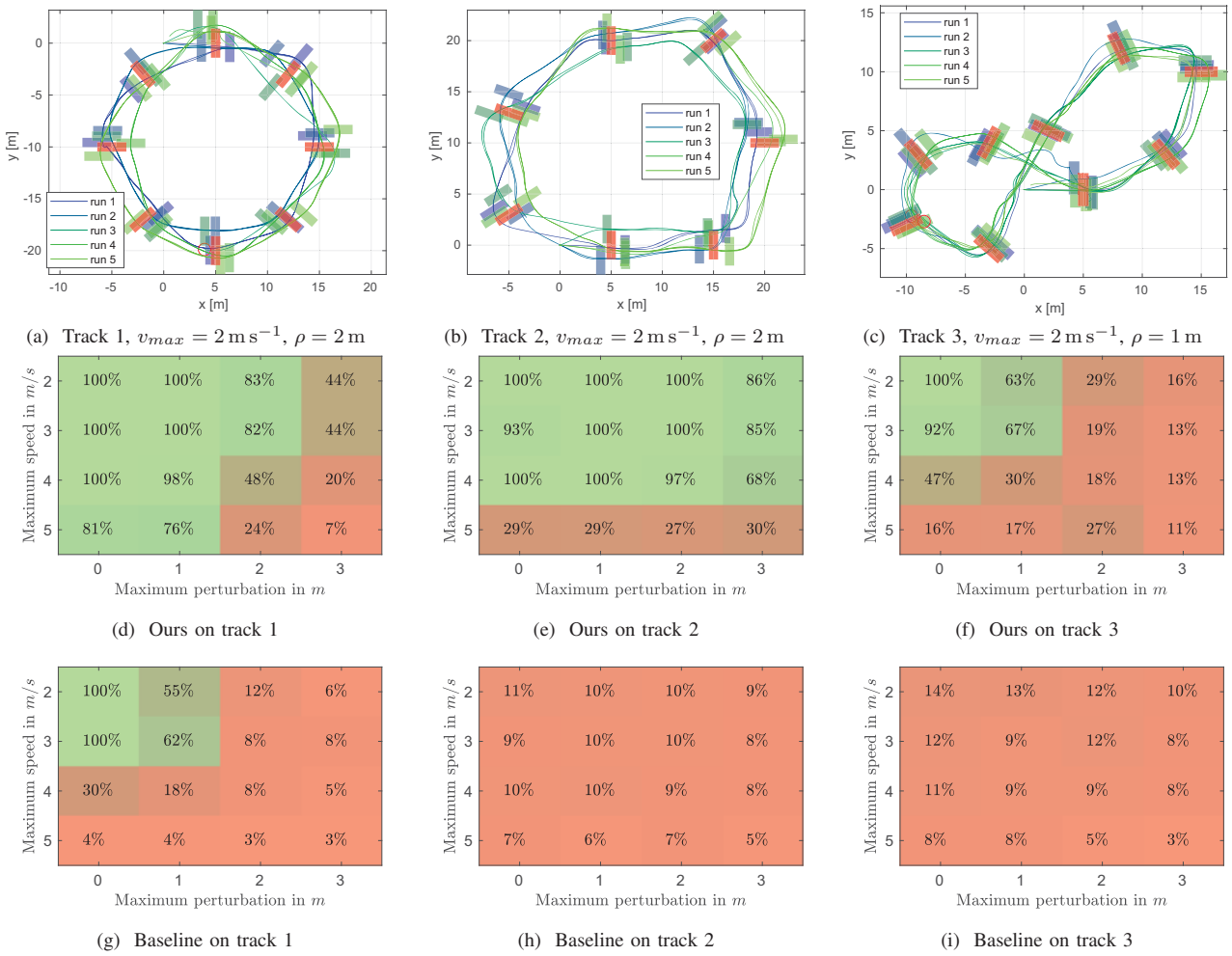
$$\mathbf{p}_{wp,l\pm} = {}_O\,\mathbf{t}_{OG_l} + \mathbf{R}_{OG_l}[\pm x_G, 0, 0]^\top, \tag{11}$$

where x_G is a user-defined constant accounting for the spatial dimension of gate l. We then linearly interpolate a path from waypoint to waypoint and use it as a reference for our controller.

2) Model Predictive Control: We formulate the control problem as a quadratic optimization problem which we solve using sequential quadratic programming as described in [3]:

$$\begin{aligned} \min_{\mathbf{u}} \int_{t_0}^{t_f} &\left(\bar{\mathbf{x}}^\top_t(t)\mathbf{Q}\bar{\mathbf{x}}_t(t) + \bar{\mathbf{u}}^\top_t(t)\mathbf{R}\bar{\mathbf{u}}_t(t)\right) dt \\ &\bar{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_r(t) \quad \bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_r(t) \\ \text{subject to} \quad &\mathbf{r}(\mathbf{x}, \mathbf{u}) = 0 \qquad \mathbf{h}(\mathbf{x}, \mathbf{u}) \leq 0. \end{aligned}$$

The states $\mathbf{x}$ and inputs $\mathbf{u}$ are weighted with positive diagonal matrices $\mathbf{Q}$ and $\mathbf{R}$ with respect to a reference $\mathbf{x}_r$ and $\mathbf{u}_r$. The equality and inequality constraints, $\mathbf{r}$ and $\mathbf{h}$ respectively,



(a) Track 1, $v_{max} = 2\,\mathrm{m\,s^{-1}}$, $\rho = 2\,\mathrm{m}$ (b) Track 2, $v_{max} = 2\,\mathrm{m\,s^{-1}}$, $\rho = 2\,\mathrm{m}$ (c) Track 3, $v_{max} = 2\,\mathrm{m\,s^{-1}}$, $\rho = 1\,\mathrm{m}$

(d) Ours on track 1 (e) Ours on track 2 (f) Ours on track 3

(g) Baseline on track 1 (h) Baseline on track 2 (i) Baseline on track 3

Fig. 6: Results of the simulation experiments. We compare the presented approach to the baseline [8] on three tracks, at different speeds and track perturbations. **(a)-(c):** Perturbed tracks and example trajectories flown by our approach. **(d)-(f):** Success rate of our method. For each data point, 5 experiments were performed with random initial gate perturbation. **(g)-(i):** Success rate of the baseline method.

are used to incorporate the vehicle dynamics and input saturations. The reference is our linearly sampled path along which the MPC finds a feasible trajectory. Note that we can run the control loop independent of the detection and mapping pipeline and reactively stabilize the vehicle along the changing waypoints.

IV. Experimental Setup

We evaluate the presented approach in simulation and on a physical system.

A. Simulation

We use RotorS [6] and Gazebo [12] for all simulation experiments. To train the perception system, we generated 45,000 training images by randomly sampling camera and gate positions and computing their relative poses. For quantitative evaluation, a 100% successful trial is defined as completing 3 consecutive laps without crashing or missing a gate. If the MAV crashes or misses a gate before completing 3 laps, the success rate is measured as a fraction of completed gates out of 3 laps: for instance, completing 1 lap counts as 33.3% success.

B. Physical System

In all real-world experiments and data collection we use an in-house MAV platform with an Intel UpBoard as the main computer running the CNN, EKF, and MPC. Additionally we use a Qualcomm Snapdragon Flight as a visual-inertial odometry unit. The platform is shown in Fig. 5. The CNN reaches an inference rate of $\sim 10\,\mathrm{Hz}$ while the MPC runs at $100\,\mathrm{Hz}$. With a take-off weight of $950\,\mathrm{g}$ the platform reaches thrust-to-weight ratio of ~ 3.

We collect training data for the perception system in five different environments, both indoors and outdoors. Example images from the environments are shown in Fig. 3. In total, we collected 32,000 images.

V. Results

Results are shown in the supplementary video at `https://youtu.be/UuQvijZcUSc`.

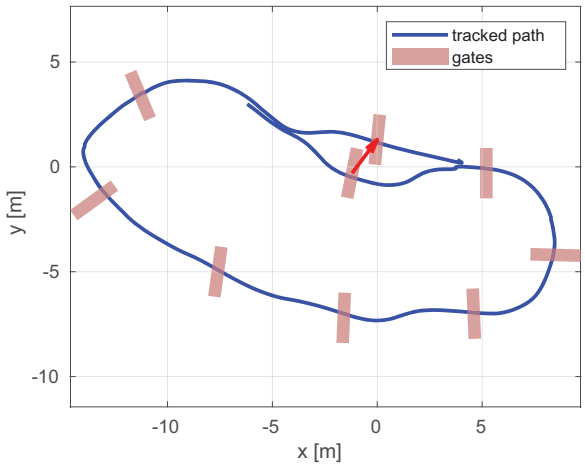


Fig. 7: Trajectory flown through multiple gates, one of which was moved as indicated by the red arrow. For visualization, only a single lap is illustrated.

A. Simulation

We first present experiments in a controlled, simulated environment. The aim of these experiments is to thoroughly evaluate the presented approach both quantitatively and qualitatively and compare it to a baseline – the method of Jung et al. [8]. The baseline was trained on the same data as our approach.

We evaluate the two methods on three tracks of increasing difficulty. Figs. 6a-c show an illustration of the three race tracks and plot the executed trajectories together with the nominal gate positions in red and the actual displaced gate positions in the corresponding track color. Our approach achieved successful runs in all environments, with speeds up to $4\,\mathrm{m\,s^{-1}}$ in the first two tracks. Additionally, gate displacement was handled robustly up to a magnitude of $2\,\mathrm{m}$ before a significant drop in performance occurred. Figs. 6d-i show the success rate of our method and the baseline on the three tracks, under varying speed and track perturbations. Our approach outperforms the baseline by a large margin in all scenarios. This is mainly because the baseline relies on the permanent visibility of the next gate. Therefore, it only manages to complete a lap in the simplest first track where the next gate can always be seen. In the more complex second and third tracks, the baseline passes at most one or two gates. In contrast, due to the integration of prior information from demonstration and approximate mapping, our approach is successful on all tracks, including the very challenging third one.

B. Physical System

To show the capabilities of our approach on a physical platform, we evaluated it on a real-world track with 8 gates and a total length of 80 meters, shown in Fig. 7. No training data for the perception system was collected in this environment. Fig. 8 summarizes the results. As in the simulation experiments, we measure the performance with

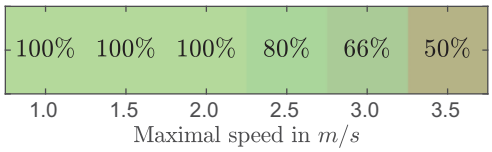


Fig. 8: Success rates of our approach in the real-world experiment. The reader is encouraged to watch the supplementary video to see the presented approach in action.

respect to the average MAV speed. As before, a success rate of 100% requires 3 completed laps without crashing or missing a gate. Our approach confidently completed 3 laps with speeds up to $2\,\mathrm{m\,s^{-1}}$ and managed to complete the track with speeds up to $3.5\,\mathrm{m\,s^{-1}}$. In contrast, the reactive baseline was not able to complete the full track even at $1.0\,\mathrm{m\,s^{-1}}$ (not shown in the figure).

An example recorded trajectory of our approach is shown in Fig. 7. Note that one of the gates was moved during the experiment, but our approach was robust to this change in the environment. Our approach could handle gate displacements of up to $3.0\,\mathrm{m}$ and complete the full track without crashing. The reader is encouraged to watch the supplementary video for more qualitative results on real tracks.

VI. Conclusion

We presented an approach to autonomous vision-based drone navigation. The approach combines learning methods and optimal filtering. In addition to predicting relative gate poses, our network also estimates the uncertainty of its predictions. This allows us to integrate the network outputs with prior information via an extended Kalman filter.

We showed successful navigation through both simulated and real-world race tracks with increased robustness and speed compared to a state-of-the-art baseline. The presented approach reliably handles gate displacements of up to $2\,\mathrm{m}$. In the physical track, we reached speeds of up to $3.5\,\mathrm{m\,s^{-1}}$, outpacing the baseline by a large margin.

Our approach is capable of flying a new track with an approximate map obtained from a single demonstration flight. This approach was used to win the IROS 2018 Autonomous Drone Race Competition, where it outraced the second-placing entry by a factor of two.

References

[1] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2015.

[2] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg. Aggressive deep driving: Combining convolutional neural networks and model predictive control. In *Conference on Robot Learning*, 2017.

[3] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. PAMPC: Perception-aware model predictive control for quadrotors. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[4] M. Fehr, T. Schneider, M. Dymczyk, J. Sturm, and R. Siegwart. Visual-inertial teach and repeat for aerial inspection. *arXiv:1803.09650*, 2018.

[5] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robot.*, 33(2), 2017.

[6] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. RotorS—A modular Gazebo MAV simulator framework. In *Robot Operating System (ROS)*. Springer, Cham, 2016.

[7] A. Giusti, J. Guzzi, D. C. Cireşan, F.-L. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. D. Caro, D. Scaramuzza, and L. M. Gambardella. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot. Autom. Lett.*, 1(2), 2016.

[8] S. Jung, S. Hwang, H. Shin, and D. H. Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robot. Autom. Lett.*, 3(3), 2018.

[9] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In *Conference on Robot Learning*, 2018.

[10] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on Neural Networks*, 22(9), 2011.

[11] D. K. Kim and T. Chen. Deep neural network for real-time autonomous indoor navigation. *arXiv:1511.04668*, 2015.

[12] N. Koenig and A. Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2014.

[13] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Robot. Research*, 34(3), 2015.

[14] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU. *IEEE Robot. Autom. Lett.*, 2(2), 2017.

[15] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robot. Autom. Lett.*, 3(2), 2018.

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

[17] H. Moon, Y. Sun, J. Baltes, and S. J. Kim. The IROS 2016 competitions. *IEEE Robot. Autom. Mag.*, 24(1), 2016.

[18] B. Morrell, R. Thakker, G. Merewether, R. G. Reid, M. Rigter, T. Tzanetos, and G. Chamitoff. Comparison of trajectory optimization algorithms for high-speed quadrotor flight near obstacles. *IEEE Robot. Autom. Lett.*, 3(4), 2018.

[19] M. Müller, V. Casser, N. Smith, D. L. Michels, and B. Ghanem. Teaching UAVs to race using UE4Sim. *arXiv:1708.05884*, 2017.

[20] M. W. Müller, M. Hehn, and R. D'Andrea. A computationally efficient motion primitive for quadrocopter trajectory generation. *IEEE Trans. Robot.*, 31(6), 2015.

[21] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.*, 33(5), 2017.

[22] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *IEEE International Conference On Neural Networks*, 1994.

[23] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robot. Autom. Lett.*, 3(3), 2018.

[24] V. Usenko, J. Engel, J. Stückler, and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.