GateNet: An Efficient Deep Neural Network Architecture for Gate Perception Using Fish-Eye Camera in Autonomous Drone Racing

Huy Xuan Pham, Ilker Bozcan, Andriy Sarabakha, Sami Haddadin and Erdal Kayacan

***Abstract*— Fast and robust gate perception is of great importance in autonomous drone racing. We propose a convolutional neural network-based gate detector (GateNet[1]) that concurrently detects gate's center, distance, and orientation with respect to the drone using only images from a single fish-eye RGB camera. GateNet achieves a high inference rate (up to 60 Hz) on an onboard processor (Jetson TX2). Moreover, GateNet is robust to gate pose changes and background disturbances. The proposed perception pipeline leverages a fish-eye lens with a wide field-of-view and thus can detect multiple gates in close range, allowing a longer planning horizon even in tight environments. For benchmarking, we propose a comprehensive dataset (AU-DR) that focuses on gate perception. Throughout the experiments, GateNet shows its superiority when compared to similar methods while being efficient for onboard computers in autonomous drone racing. The effectiveness of the proposed framework is tested on a fully-autonomous drone that flies on previously-unknown track with tight turns and varying gate positions and orientations in each lap.**

I. Introduction

Long-term applications of drone technology include their use in any unstructured environment [1], [2] such as search and rescue missions in underground mines [3], [4], detection and monitoring of victims trapped under collapsed buildings [5], aerial radiation detection in nuclear power plants after an accident [6], or aerial surveillance [7]–[10] . In these applications, knowledge about the environment is either not accessible or corrupted. Since flight duration of drones is limited due to battery constraints, the task must be executed as fast as possible without compromising safety. Autonomous drone racing is one of the best benchmark problems as drones must push operation to the boundaries of the performance envelope. Undoubtedly, fast and reliable gate perception plays a vital role in maneuvering through the gates without any collision.

In gate pose estimation, traditional computer vision methods tend to fail in complex background with varying lighting conditions, occlusion, and blurriness [11]–[13]. In contrast, deep learning algorithms offer more robust performance. Jung et al. [14] use a deep neural network (DNN) for gate center estimation, yet the network size is considerably large,

H. X. Pham, I. Bozcan, E. Kayacan are with the Artificial Intelligence in Robotics Laboratory (Air Lab), Department of Electrical and Computer Engineering, Aarhus University, 8000 Aarhus C, Denmark {huy, ilker, erdal} at ece.au.dk

A. Sarabakha and S. Haddadin are with the Munich School of Robotics and Machine Intelligence, Technical University of Munich (TUM), D-80797 Munich, Germany. {andriy.sarabakha, haddadin} at tum.de

[1]The code and data will be available at https://github.com/open-airlab/GateNet.git.

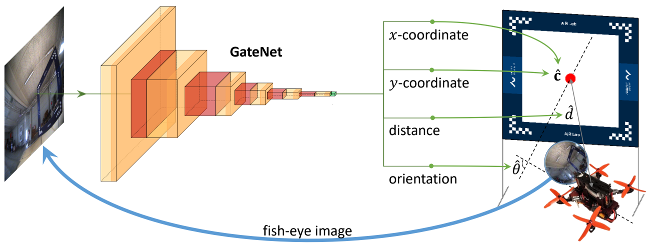

Fig. 1. An illustration of the working principles of our gate perception system. The scene images are captured by a single wide-FOV fish-eye RGB camera and fed into GateNet to estimate the gate center location, as well as distance and orientation with respect to the drone's body frame. The information then will be used to reproject the gate in 3D world frame and applied an extended Kalman Filter to achieve stable gate pose estimation.

and limited in capability due to not considering gate rotation. Kaufmann et al. [15] use two separate multilayer perceptrons to estimate the distribution of the state of a racing gate. However, it is also a large-scaled DNN achieving a low inference rate on an onboard processor. Recently, Foehn et al. [16] train a DNN to segment the four corners of a gate to correctly identify its pose. While the performance is impressive, this method and other segmentation-related methods [12], [13] usually rely on an assumption that the gate must have corners. This assumption may not be valid in case racing gates appear in various shapes, that include circular or elliptic gates. Alternative methods use images as an input to directly infer an end-to-end planning [17]–[20], or calculate a steer function [21]. While the results of end-to-end methods are promising, the decision making might become less clear, and a robot's action can be difficult to verify.

In this study, we propose a novel DNN architecture for gate perception, GateNet, that can provide an accurate estimation of a gate pose while being efficient to implement, achieving a high inference rate (up to 60Hz on an NVIDIA Jetson TX2). It is robust to gate pose changes and background disturbances. Unlike other methods that use segmentation, notably [16], the proposed perception network predicts the pose of a gate's center as we seek to maintain the generality required for a perception pipeline that can work independent of gate shape assumption. The proposed perception pipeline leverages a fish-eye camera lens with wide field-of-view (FOV) and can detect multiple gates in a tight area, enabling a longer motion planning horizon. A relatively small network architecture allows us to attain a high inference rate while requiring less training time.

The contributions of this study are: *(i)* an efficient and

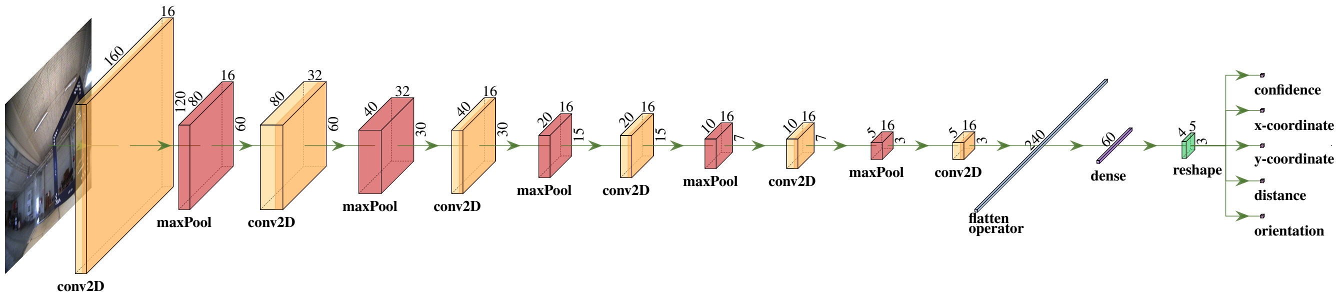

Fig. 2. Network architecture of the GateNet. After conv2D layers, batch normalization and ReLU are added consecutively. The dense layer is reshaped into a tensor which has 4 rows and 3 columns to keep the aspect ratio of input images (4:3). The depth of the output tensor is 5 since the network predicts 5 values, including (i) the confidence value, (ii) x and (iii) y offsets, and relative (iv) distance and (v) orientation, for each cell.

robust framework for multiple racing gates' pose estimation using fish-eye raw RGB images, *(ii)* a comprehensive dataset, called AU-DR, for training and benchmarking, and *(iii)* real experiments on a fully-autonomous drone that can complete previously-unknown tracks with tight turns.

The rest of this work is organized as follows. Section II summarizes our method for gate perception. Section III compares GateNet with the state-of-the-art baselines. Section IV presents real experiments to show the efficiency of the approach. Section V draws some conclusions from this work.

II. Methodology

A. GateNet

Unlike standard object detectors (e.g., [22], [23]) which predict gates' locations in the image plane, GateNet unifies predictions of (i) center, (ii) orientation, and (iii) distance of gates in a single deep neural network. The network gets an image as an input and produces an output tensor including the targeted predictions. The design and number of parameters of the network enable high inference speed that is crucial for autonomous drone racing.

1) GateNet Architecture: The network is implemented as a convolutional neural network (CNN) with a single fully-connected layer at the end (see Fig. 2). The convolutional layers extract features from an input image; and the fully-connected layer predicts the confidence values, gate centers on the image plane (the pixel values of gate centers' offsets), distance and orientation of the gates relative to the drone.

GateNet has six convolutional layers and one fully-connected layer. The first five convolutional layers are followed by batch normalization, rectified linear unit (ReLU) activation, and max-pooling with the pooling size of two. The last convolutional layer is followed by batch normalization and ReLU; but not a pooling layer. The convolutional stream reduces the shape of the feature activations to $3 \times 5 \times 16$ which is small enough to prevent any computation bottleneck for the sake of real-time inference. At the end of the convolutional stream, the extracted features are flattened to a 1D vector. Note that the flatten operator simply reshapes the 3D-tensor with the shape of $3 \times 5 \times 16$ to a 1D-vector with the size of 240. Similar to the YOLO architecture in [24], the reshaping operation allows all hidden neurons in the 3D tensor to be connected to the hidden neurons in the dense layer. Therefore, a single hidden neuron in the dense layer can be coupled to the information that exists in the whole of the last convolutional layer. We empirically choose the flatten operation over other dimensionality reduction layers (e.g., max-pooling) for the last convolutional layer since the dimension of the last convolution layer is extremely small, and dimensionality reduction can cause information loss. The flattened vector is then connected to a fully-connected (dense) layer. We do not apply any non-linearity after the fully-connected layer. The output vector is reshaped to $R \times C \times 5$, where R and C are the numbers of rows and columns of the output layer. Although reshaping of the output vector does not affect the network's forward pass during inference or backward pass during the back-propagation, it allows us to have a more intuitive encoding of the target vectors. The third dimension of the output layer has the size of five, including (i) the confidence value, (ii) the offset of a gate center in the x-axis of an image, (iii) the offset of a gate center in the y-axis of an image, (iv) the orientation of a gate relative to the drone, (v) the distance of a gate relative to the drone.

The target samples for training are prepared according to the spatial layout of gates in an input image. Firstly, an input image is divided into R rows and C columns (Fig. 3a). Then, x and y offsets are calculated for each gate by taking the differences between the gate's center and the top-left

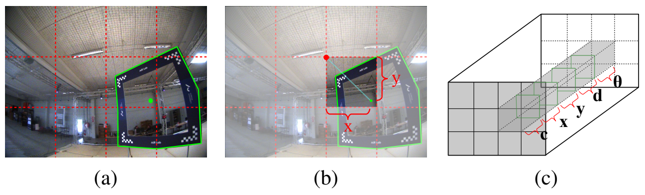

Fig. 3. Ground-truth creation process. The pixel coordinates of a gate center (the green dot in (a)) are calculated using a world-to-image transformation (a). Then, the center offsets (x and y) are calculated according to the top-left corner of the grid including the gate center (b). Each cell is assigned with a confidence value, center offsets, distance, and orientation (c). Note that since we use linear activations in the final fully connected layer, the center offsets can be negative in case a gate center falls into the outside of an image. Therefore, GateNet can predict gate centers for partially observed gates even if their gate centers are not presented in an input image.

corner of a particular grid in which the gate center appears (Fig. 3b). Note that, x and y are normalized w.r.t. the size of the grid. However, since x and y are not bounded with a non-linear activation (e.g., soft-max), the center offsets can be negative when a gate center is outside of an input image. Finally, a target sample, which is a tensor with a shape of $R \times C \times 5$, is created to store the targeted variables (Fig. 3c). The confidence value (c) is set to 1, if a gate center is presented in the corresponding grid, or is set to 0, otherwise. The distance (d) and the orientation (θ) are relative to the gate and are measured in meters and radians, respectively. We empirically choose R and C as 3 and 4, respectively, to keep the aspect ratio of the input images (4:3).

2) Loss Function: The network is subject to minimize the following multi-part objective function:

$$\begin{aligned}\mathcal{L} = &\sum_{i=0}^{R}\sum_{j=0}^{C} \lambda_{\textbf{coord}} \mathbb{1}_{ij}^{\text{obj}} \left[(x_{ij} - \hat{x}_{ij})^2 + (y_{ij} - \hat{y}_{ij})^2 \right] \\ &+ \sum_{i=0}^{R}\sum_{j=0}^{C} \mathbb{1}_{ij}^{\text{obj}} \left[\lambda_{\textbf{dist}} \left(d_{ij} - \hat{d}_{ij} \right)^2 + \lambda_{\textbf{ori}} \left(\theta_{ij} - \hat{\theta}_{ij} \right)^2 \right] \\ &+ \sum_{i=0}^{R}\sum_{j=0}^{C} \left[\lambda_{\textbf{obj}} \mathbb{1}_{ij}^{\text{obj}} + \lambda_{\textbf{noobj}} (1 - \mathbb{1}_{ij}^{\text{obj}}) \right] (c_{ij} - \hat{c}_{ij})^2 ,\end{aligned} \tag{1}$$

where $\mathbb{1}_{ij}^{\text{obj}}$ is a binary scalar denoting the presence of a gate center in the grid cell with the row index i and the column index j in a ground-truth sample. x_{ij} and $\hat{x}_{ij}$ are the ground-truth and the predicted offsets for x-axis, respectively, in the grid at (i, j). Similarly, y_{ij} and $\hat{y}_{ij}$ are the ground-truth and the predicted offsets for y-axis, respectively, in the grid at (i, j). d_{ij} and $\hat{d}_{ij}$ are the ground-truth and the predicted relative distances, respectively. θ_{ij} and $\hat{\theta}_{ij}$ are the ground-truth and the predicted relative distances, respectively, and c_{ij} and $\hat{c}_{ij}$ are the predicted and ground-truth confidence values, respectively.

Similar to the work in [24], introducing $\mathbb{1}_{ij}^{\text{obj}}$ term allows the loss function to penalize the network only if a gate center is present in a particular grid. Furthermore, the network subject is to minimize the confidence values of grid cells that do not contain any gate center. We also introduce the non-trainable parameters of $\lambda_{\textbf{coord}}$, $\lambda_{\textbf{dist}}$, $\lambda_{\textbf{ori}}$, $\lambda_{\textbf{obj}}$, $\lambda_{\textbf{noobj}}$ to put weights to the losses for a gate center, distance, orientation, objectness, and non-objectness, respectively.

B. AU-DR Dataset

We introduce a new large-scale dataset (AU-DR dataset[1]) that addresses different scene appearances in gate detection problem in the context of drone racing. AU-DR dataset contains RGB images captured by a drone camera and annotations of gates presented in the image (Fig. 4). Unlike similar studies, such as [25], we explicitly label the images according to gate layouts, including images with (i) single gate, (ii) multiple gates, (iii) occlusion, (iv) partially observable gates, (v) too far gates, and (vi) too close gates. Such a classification allows us to analyze how gate detection algorithms work for specific cases. Figure 5 shows the histograms of gates' distances and orientations in the AU-DR dataset. Moreover, we annotate the gates' edges with polygons, and provide segmentation maps for gates in our dataset. Although we do not use these annotations in our experiments, the AU-DR dataset includes these annotations for future studies.

We collect the dataset using real racing gates and a custom-made robot system (see Subsection IV-A) under different lighting conditions (e.g., sun light, artificial lights) in an indoor environment (Aarhus University Deep Tech Experimental Hub). To collect RGB images, we use a high-resolution camera with a global shutter, operated at high frequency ($30-50$Hz) to minimize motion blur. We choose a fish-eye lens with a reasonable focal length and a wide field of view (140 degrees) to improve the observability of the next gate. The ground-truth information regarding drone's pose, gates' poses are provided using a state-of-the-art motion capture system infrastructure.

The AU-DR dataset includes 30,203 samples in total. The dataset is split into three parts: 60% for training, 10% for validation and 30% for testing. Each data sample consists of an RGB image with the dimension of 720×540, a scene label, and gate annotations which include (i) the pixel coordinate of the gate center in an image frame, (ii) distance, and (iii) orientation of a gate relative to the drone.

[1]The dataset is available at `https://github.com/open-airlab/GateNet.git`.

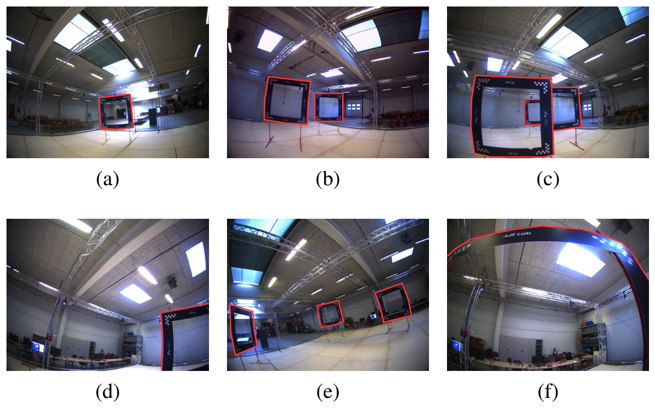

Fig. 4. Samples from the AU-DR dataset. The dataset includes different gate layout cases that appear in a drone racing scenerio: (a) single gate, (b) multiple gates, (c) occluded gates, (d) partially observable gates, (e) gates with a distant layout, and (f) a gate that is too close to the drone's camera.

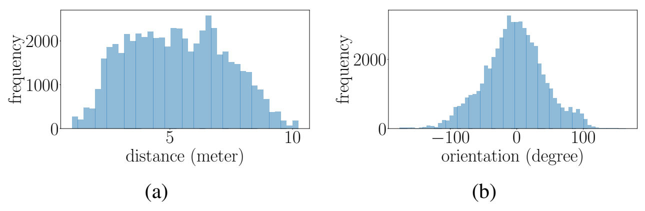

Fig. 5. Histograms of the gates' distances (a) and orientations (b) in the AU-DR dataset. The frequency is the number of samples in the dataset.

III. Evaluation and Results

We compare GateNet with ADRNet [14], DroNet [15], Darknet [24], and the method proposed in [26] for gate center detection in image plane, and gate distance and orientation prediction relative to the drone. We choose the above methods as the baselines because 1) the similarity of their methods to ours, making them comparable, 2) they include milestone results, as [14] won the IROS autonomous drone racing competition in 2016 while [15] won the competition in 2018 and 3) they are publicly available.

A. Baselines

1) ***ADRNet [14]:*** Jung et al. propose a variant of the SSD object detector [23] that is built on top of AlexNet [27]. It detects bounding box offsets of drone racing gates in a given image. In our comparison, we use ADRNet as the original to get bounding box offsets. Furthermore, we change the output layer of ADRNet with our output layer to predict the same attributes. This modified network is named ADRNet-mod.

2) ***Beauty and the Beast [15]:*** Kaufmann et al. [15] proposes a DNN composed of a CNN (specifically, DroNet [28]) for feature extraction and two separate multi-layered perceptron (MLP) to regress the parameters of a multivariate normal distribution that describes the estimation of the next gate's pose. Their MLP detect one gate (i.e., the next gate) at each time only. Therefore, it is not compatible with our dataset where the baselines are evaluated for the detection of all gates in an input image simultaneously. For the sake of compatibility, we change the original output layers with the output layer of our model. As a result, the modified network (BB-mod) is trained with the AU-DR dataset to compare with our network architecture.

3) ***Morales et al. [26]:*** Morales et al. propose a data generation method that randomly renders drone racing gates for any background and illumination conditions. They train MobileNetv2-SSDLite to detect gates in a flight area. They assign one of three categories to each gate; first gate (i.e., target gate), second target gate, and back gate, which is the gates facing backwards. They crop a detected rectangular bounding box area and feed it to a fully connected network to predict the gate distance. For the sake of compatibility, we train MobileNetv2-SSDLite with only one class (i.e., gate). We use the same fully connected network architecture proposed in [26] to predict the distance for comparison.

4) ***Darknet [24]:*** Redmon et al. [24] propose a single-shot object detector (YOLO) that works on top of a novel neural network architecture called Darknet. Moreover, a shallow implementation of YOLO (Yolov3-Tiny) is used for real-time object detection. We use Darknet as a baseline for the gate detection task. For this end, we implement a neural network (Darknet-mod) using Darknet as a backbone followed by the output layer of GateNet.

B. Evaluation Metrics

We compare GateNet with baselines in several perception tasks, including predictions of (i) gate center on image plane, (ii) gate distance and (iii) orientation relative to the UAV frame. Firstly, we compute the gate center error taking the mean absolute error between the predicted and the ground-truth gate centers on image plane. Using the notation in Eq. 1, the gate center error (E_c), the relative distance (E_d) error and the orientation errors (E_θ) can be formulated as follows:

$$\begin{aligned} E_c &= \frac{1}{N}\sum_{i=0}^{R}\sum_{j=0}^{C}(|\hat{x}_{ij}-x_{ij}|+|\hat{y}_{ij}-y_{ij}|), \\ E_d &= \frac{1}{N}\sum_{i=0}^{R}\sum_{j=0}^{C}|\hat{d}_{ij}-d_{ij}|, \\ E_\theta &= \frac{1}{N}\sum_{i=0}^{R}\sum_{j=0}^{C}|\hat{\theta}_{ij}-\theta_{ij}|, \end{aligned} \tag{2}$$

where $|.|$ is the absolute value operation, and N is the total number of test samples.

C. Training Settings

We train GateNet and the baselines with the AU-DR dataset, with resized images of the same size (160x120). The maximum limit for the number of training epochs is set to 300 yet the training is stopped when the validation error started to increase (usually around 240 epochs). We use Adam optimizer [29] with the parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is set to 0.01, and decayed linearly with 0.1 at epoch 5 and 8. We use the batch size of 32 for training. For the sake of compatibility, we use the same training parameters to train the baselines. The input images are resized to 160×120.

D. Results

Table I shows the performance of GateNet and other baselines for the gate detection task on the AU-DR test set. Thanks to having scene labels in our dataset, we evaluate the methods' performances for different gate layouts. The table also compares the number of network parameters and their inference speed on an NVIDIA Jetson TX2. As shown in Table I, GateNet has a light-weight architecture with less numbers of parameters compared to the baselines. Therefore, it is significantly faster than other methods for real-time inference on an NVIDIA Jetson TX2. The rates in Table I are calculated considering the inference itself, the pre-processing (e.g., input resizing, I/O operations) and the post-processing tasks (e.g., output decoding) which the processor must be shared during the networks' forward pass.

As can be seen in Table I, the original ADRNet is marginally better to find object centers than GateNet. This is expected since the ADRNet has a larger number of parameters that increases the network's learning capability. However, the prediction of gate distance and orientation is not available for the original ADRNet. It is also significantly slower (14 fps) than GateNet (57 fps). We modify the output layer of ADRNet (ADRNet-mod) to have the same output layer as GateNet, but again the results are similar to that of the original ADRNet.

GateNet is superior in gate distance prediction compared to other baselines (see Table I). Moreover, it has better

TABLE I

COMPARISON WITH EXISTING METHODS ON THE AU-DR TEST SET. # INDICATES THE NUMBER OF PARAMETERS IN A NETWORK. FPS REPRESENTS THE NUMBER OF FRAMES THAT CAN BE PROCESSED PER SECOND DURING INFERENCE WITH THE INPUT IMAGE SIZE OF 160×120.

			single gate			multiple gate			occlusion			partial			too close		
method	#	fps	E_c	E_d	E_θ	E_c	E_d	E_θ	E_c	E_d	E_θ	E_c	E_d	E_θ	E_c	E_d	E_θ
ADRNet [14]	2,5M	14	**0.01**	n/a	n/a	0.02	n/a	n/a	**0.01**	n/a	n/a	**0.01**	n/a	n/a	**0.01**	n/a	n/a
ADRNet-mod	2,5M	14	0.01	0.45	0.06	**0.01**	1.09	0.13	0.01	0.76	0.07	0.01	0.50	0.08	0.01	0.32	0.13
BB-mod	478K	22	0.04	0.07	0.03	0.08	0.12	0.07	0.06	0.10	0.04	0.06	0.09	0.04	0.06	0.11	0.09
Morales et al. [26]	1.1M	19	0.17	0.35	n/a	0.24	1.22	n/a	0.19	0.41	n/a	0.12	0.43	n/a	0.16	0.27	n/a
Darknet-mod	7,8M	9	0.02	0.04	**0.01**	0.04	0.08	**0.02**	0.03	0.07	0.02	0.03	0.05	0.02	0.03	0.07	0.03
GateNet	**32K**	**57**	0.02	**0.03**	0.02	0.03	**0.06**	0.03	0.02	**0.06**	**0.02**	0.02	**0.04**	**0.02**	0.02	**0.05**	**0.03**

performance for gate orientation prediction in challenging cases such as partially observable gates or gates that are too close to the camera. BB-mod and [26] have higher inference speed compared to ADRNet and Darknet-mod. However, their prediction performances are significantly worse than other baselines.

IV. EXPERIMENTS IN REAL-TIME FLIGHTS

A. Experimental Setup

For real-world experiments, a quadrotor drone (in Fig. 6) is built on a small carbon fiber frame (250mm) with a thrust-to-weight ratio of 5 using powerful motors controlled by a Pixhawk autopilot [30]. The main camera is a high-frequency Flir Blackfly RGB camera (up to 100Hz). An Intel Realsense Tracking camera T265 is used for state estimation. The overall framework runs entirely on-board on an NVIDIA Jetson TX2 computer. A racing track is created using a set of square-shaped gates having internal dimensions of $1.5\times1.5\text{m}$ but with different heights. For safety reasons, the track is placed inside a motion capture system lab, but the system is not used throughout the experiments.

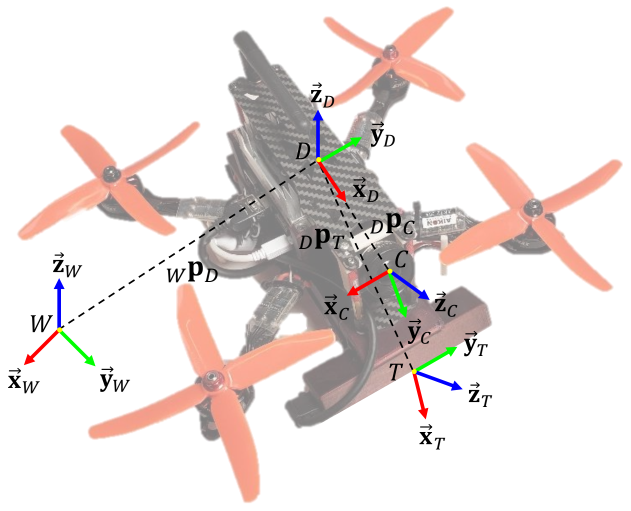

Fig. 6. The drone relies on three main components: (i) GateNet and a perception pipeline to provide gate detection and gate pose estimation, (ii) state estimation using a robust embedded solution Intel Tracker T265, (iii) a state-of-the-art control method that has high accuracy tracking performance, and a planner capable of generating an optimal minimum-snap trajectory for smooth navigation through multiple gates, and replanning under unexpected events. All calculations are handled on-board without any support from external systems. The coordinate frames W, D, C, T are the world, drone's body, camera, and Intel Tracker frames, respectively.

B. Fish-Eye Perception Pipeline and Global Mapping

We compute a global map (shown in Fig. 7) of the gates in the racing environment for high-level planning tasks. The output of GateNet can be represented as a prediction vector $\hat{\mathbf{v}} = \begin{bmatrix}\hat{c}_x & \hat{c}_y & \hat{d} & \hat{\theta}\end{bmatrix}$ of gate center $(\hat{c}_x, \hat{c}_y)$ on image plane, relative distance from the robot to the gate $\hat{d}$, and relative heading angle $\hat{\theta}$ that can be used to reconstruct the gate in the world frame by back-projection with respect to the robot's pose. As we perform our prediction on a raw image of a wide-FOV fish-eye lens, it is a non-trivial task for the back-projection operation since a closed-form solution does not exist. Therefore, we employ a gradient-search method as follows. Let $W, D, \mathcal{C}$ denote the world, drone's body, and camera, respectively. Let $\mathbf{p}_D, \mathbf{p}_G, \mathbf{p}_C$ define the drone's position, gate position, and camera position, respectively, and ${}_A\mathbf{p}_C$ is the position of C expressed in frame A. Let $\mathbf{R}_B^A$ describe the rotation matrix from frame A to frame B. Firstly, the distorted gate 3D coordinates $\hat{x}_G^h, \hat{y}_G^h$ can be back-projected from the 2D gate's center in the image plane:

$$\left[\hat{x}_G^h, \hat{y}_G^h, \cdot\right]^T = \mathbf{K}^{-1}\left[\hat{c}_x, \hat{c}_y, 1\right]^T, \tag{3}$$

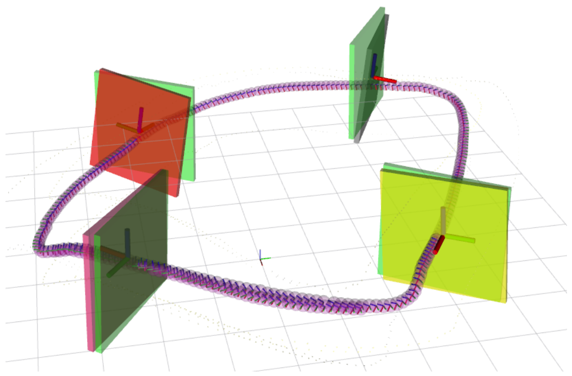

Fig. 7. Global gate map with estimated gates' poses (colored) against ground-truth information (green) during a real experiment, where a drone's trajectory can be seen (purple). [Best viewed in color].

with $\mathbf{K}$ is the camera intrinsic matrix. Assume an equidistant distortion model for the fish-eye lens surface: $\rho_d = \rho + k_1\rho^3 + k_2\rho^5 + k_3\rho^7 + k_4\rho^9$, where $\rho = \arctan(\sqrt{\left(x_G^h\right)^2 + \left(y_G^h\right)^2})$ and $(k_1, \ldots, k_4)$ are distortion coefficients. An iterative method is used to find the undistorted point starting from an initial solution $\tilde{\rho} = \tilde{\rho}_d = \sqrt{\left(\hat{x}_G^h\right)^2 + \left(\hat{y}_G^h\right)^2}$. A new solution is computed iterativelly as: $\tilde{\rho} = \tilde{\rho} + \dot{\rho}$, where:

$$\dot{\rho} = \frac{\tilde{\rho}_d - \tilde{\rho} - k_1\tilde{\rho}^3 - k_2\tilde{\rho}^5 - k_3\tilde{\rho}^7 - k_4\tilde{\rho}^9}{1 + 3k_1\rho^2 + 5k_2\rho^4 + 7k_3\rho^6 + 9k_4\rho^8}, \tag{4}$$

until it converges to the undistorted point or the maximum number of iterations (10 iterations in our case) is reached. After this step, undistorted gate 3D coordinates $\hat{\mathbf{p}}_G^u$ can be found by:

$$\hat{\mathbf{p}}_G^u = \left[\hat{x}_G^u, \hat{y}_G^u, \hat{z}_G^u\right]^T = s\left[\hat{x}_G^h, \hat{y}_G^h, 1\right]^T, \tag{5}$$

where $s = \frac{\tan\tilde{\rho}}{\tilde{\rho}_d}$. Then, the position of the gate has to be adjusted by using the estimated distance to the gate $\hat{d}$:

$${}_C\hat{\mathbf{p}}_G = \frac{\hat{d}}{||\hat{\mathbf{p}}_G^u||}\hat{\mathbf{p}}_G^u. \tag{6}$$

Finally, the position of the gate in the camera frame has to be transformed into the world frame:

$$\begin{aligned}{}_W\hat{\mathbf{p}}_G &= {}_W\mathbf{p}_D \\ &+ \left(\mathbf{R}_D^W\right)^{-1}\left(\left(\mathbf{R}_C^D\right)^{-1}\begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} {}_C\hat{\mathbf{p}}_G + {}_D\mathbf{p}_c\right).\end{aligned} \tag{7}$$

This procedure is repeated for each detected gate. Although, the computational complexity of the gradient search is $\mathcal{O}(n^2)$, where n is the dimensionality of the search space, our back-projection method usually converges after a few iterations and it is limited to 10 iterations. Therefore, it contributes a negligible amount to the computation time.

To handle the uncertainties coming from the CNN network, we employ an extended Kalman filter (EKF) for each detected gate. A simplified EKF model for measurement is as follows:

$$\begin{cases} x_k^- &= x_{k-1}^+, \\ P_k^- &= P_{k-1}^+ + Q, \\ K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1}, \\ x_k^+ &= x_k^- + K_k(z_k - h(x_k^-)), \\ P_k^+ &= (I - K_k H_k) P_k^-, \end{cases} \tag{8}$$

where the posteriori prediction and covariance matrix x_k^+ and P_k^+ are updated based on priori estimated x_k^- and P_k^-, the newest prediction from the network z_k, the calculated Kalman gain K_k. I is an identity matrix, and H_k is a Jacobian matrix of the measurement function $h(.)$, which relates x_k^- and its measurement z_k. Q and R_k are process noise and measurement noise covariance, respectively. Similar to previous studies [15], [16], coarse locations of gates on a track are required to correctly associate a detected gate to its respective EKF in the first few steps by comparing the estimated Euclidean distance error. The performance of the EKFs will be further discussed on Section IV-D.

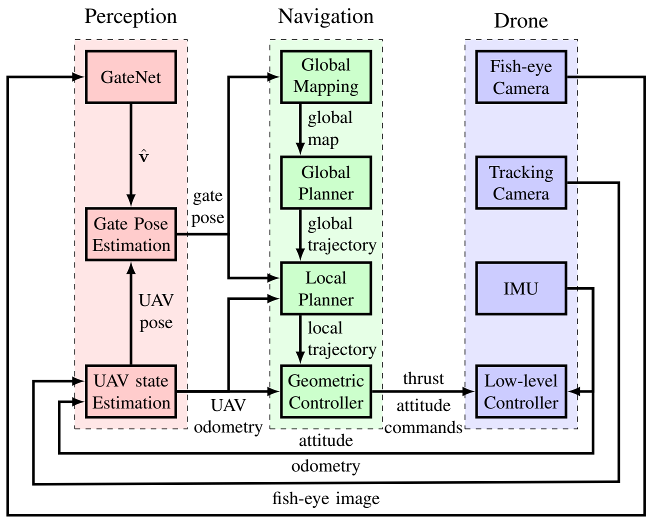

Fig. 8. Block diagram of the proposed framework with planning, perception, state estimation, and control.

C. Motion Planning and Control

We adopt a flat-state model [31] of a quadrotor that leverages the differential flatness property:

$$\begin{cases} \ddot{\mathbf{p}} &= \mathbf{R}e_3 f + g \\ \dot{\mathbf{R}} &= \mathbf{R}\omega_{[\times]}, \end{cases} \tag{9}$$

where $\ddot{\mathbf{p}}$ denotes the linear acceleration of the drone, $e_3 = [0, 0, 1]^T$ is a unit vector along the z-axis of the drone body frame and thrust force f, ω denotes its body rates, $\mathbf{R}$ is the rotational matrix given the attitude of the drone, and $[\times]$ denotes a skew-symmetric operator. A geometric controller with considering motor drag [32] is used, attaining high accuracy in tracking performance.

In motion planning for agile flights, although sampling-based algorithms [4], [33] exist thanks to their simplicity and efficiency, they often require precise dense mapping that could be slow and expensive. This work uses an optimal trajectory generation method [34], [35] to compute a minimum-snap flat-state trajectory for the drone to fly through a perceived gate's center. Since GateNet can correctly estimate multiple gates, a low-frequency global planner generates a smooth trajectory with a long-horizon of planning through all detected gates. To cope with the uncertainties of gate perception, a high-frequency receding-horizon local planner is employed to implement a portion of the global path through the next gate and add a safe path along the gate's normal vector. It can also re-plan in the events of changes in current gate estimation or a new gate detected (thus, the global plan may change). Fig. 8 details our planner architecture.

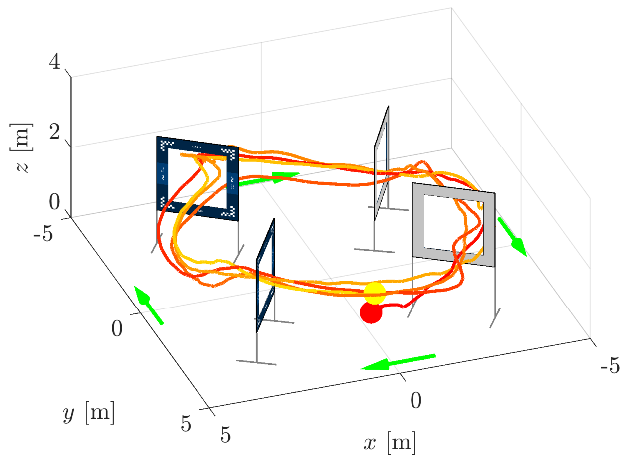

Fig. 9. Trajectory of the drone flying autonomously through the set of gates. The yellow and red spheres indicate the drone's initial and final positions, respectively; while the green arrows show the direction of the flight.

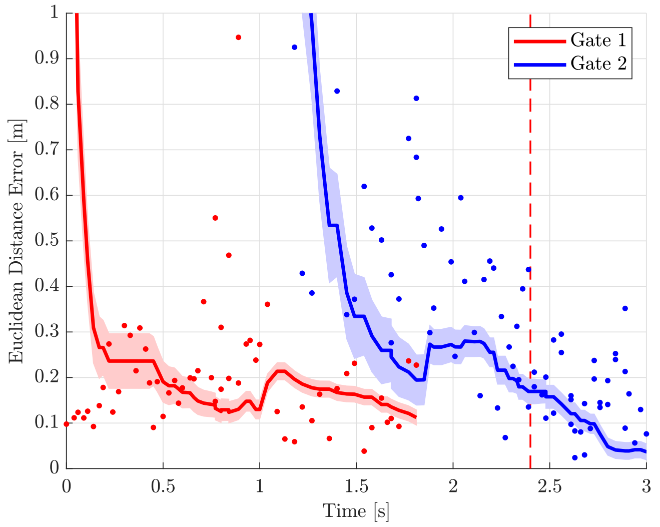

Fig. 10. A sample of EKF's effect on the accuracy and consistency of the detected gates. The dots illustrate row detections of the gates by GateNet. The solid lines show the output from EKF and the shaded regions represents the covariance matrices. The dashed vertical line depicts the instance when the gate is crossed. GateNet can detect multiple gates' poses with high accuracy. In such cases, our perception pipeline maintains a separate EKF for each individual gate. Typically before the drone passes the previous gate, the next gate's location is estimated accurately (less than 0.25m error), allowing a timely motion re-planing. The wide FOV of the fish-eye camera significantly increases the chance to see multiple gates.

D. Experimental Results

In the real-world experiments, the drone completes a race track (in Fig. 9) consisting of four gates where the gates are close to each other, with an average inter-distance of 4.5m, and have variations in both appearances and heights. Due to the tight racing track, the drone's speed is capped at $2\mathrm{m/s}$. Thanks to the high frequency and accuracy of GateNet, the EKFs converge fast enough, allowing us to obtain gate pose estimation even when the targeted gate is close. Fig. 10 shows that the EKFs improve the accuracy and consistency of the predicted model. The perception pipeline only attains a frequency of $50-55\mathrm{Hz}$ in real tests, as the computer resources are shared with trajectory-tracking controller and path planning algorithms.

To test the robustness of the gate perception pipeline throughout a race, we change some gates' positions, orientations, and also introduce disturbances (such as walking human and gate pieces) in-between each lap. Table II presents success rates in estimating a gate's pose when it varies during the flight. To disregard the effects of control and planning issues, we count it as a success if the drone perceives the gate's new location or orientation, and triggers a re-planning event. Each entry of the table represents the success ratio after eight attempts. As can be seen in Table II, GateNet can handle position and orientation variations with reasonable success rates up to 2m and $30°$, respectively. The performance of the gate perception does not decrease significantly when varying the gate location; however, it is less robust with orientation changes as performance degrades if orientation variation is bigger than $45°$. The reason is that the orientation of the gate w.r.t. the camera is negatively correlated to the area (number of pixels) of the gate in the image. In other words, the higher yaw angle of the gate is, the smaller gate will appear in the image. This can be expressed with the following mathematical relation: $\%gate = 100 - \frac{200}{\pi}|\theta|$. Additionally, the network is robust with background perturbations of human movements or gate piece disturbances (see the supplemental video).

TABLE II

SUCCESS RATES (IN PERCENTAGE) FOR GATE PASSING BASED ON EIGHT TRIALS FOR EACH CASE WHEN CHANGING A GATE'S LOCATION AND ORIENTATION.

Position \ Orientation	**0°**	**15°**	**30°**	**45°**	**60°**
0.5 m	100	100	100	75	50
1.0 m	100	100	87.5	50	12.5
1.5 m	100	100	75	50	0
2.0 m	100	87.5	75	62.5	12.5

V. Conclusion

This work proposes GateNet, a DNN capable of providing robust gate pose estimation using raw fisheye RGB images. It achieves high inference rates on a real racing drone platform. Through extensive real-time experiments, GateNet's performance is superior compared with similar studies, which makes it suitable for autonomous drone racing. We also provide the AU-DR dataset with standard racing gate appearance and labeled ground-truth information to help researchers benchmark existing perception methods, as well as train their own perception network. We demonstrate the effectiveness of GateNet in an autonomous racing scenario under tight environment setting, which can be a test-bed for improving agile flights in a more realistic conditions. In the future, we would like to improve the perception pipeline to also handle collision avoidance and other semantic object perception to improve system autonomy. Although our method is optimized for a specific indoor setting, it could be beneficial to study how the performance is transferred to different environments. The ultimate goal is to have an agile aerial robot working efficiently in search and rescue missions.

Acknowledgment

We would like to thank Deniz Bardakci, Jonas le Fevre, Rui Figueiredo, and Halil Ibrahim Ugurlu for fruitful scientific discussions. This work is supported by Aarhus University, Department of Electrical and Computer Engineering (28173).

References

[1] R. Bonatti, W. Wang, C. Ho, A. Ahuja, M. Gschwindt, E. Camci, E. Kayacan, S. Choudhury, and S. Scherer, "Autonomous aerial cinematography in unstructured environments with learned artistic decision-making," *Journal of Field Robotics*, vol. 37, no. 4, pp. 606–641, 2020.

[2] M. Gschwindt, E. Camci, R. Bonatti, W. Wang, E. Kayacan, and S. Scherer, "Can a robot become a movie director? learning artistic principles for aerial cinematography," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1107–1114.

[3] T. Dang, F. Mascarich, S. Khattak, C. Papachristos, and K. Alexis, "Graph-based path planning for autonomous robotic exploration in subterranean environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3105–3112.

[4] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion primitives-based path planning for fast and agile exploration using aerial robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 179–185.

[5] H. Nguyen, F. Mascarich, T. Dang, and K. Alexis, "Autonomous aerial robotic surveying and mapping with application to construction operations," *arXiv preprint arXiv:2005.04335*, 2020.

[6] V. Kharchenko, A. Sachenko, V. Kochan, and H. Fesenko, "Reliability and survivability models of integrated drone-based systems for post emergency monitoring of npps," in *2016 International Conference on Information and Digital Technologies (IDT)*. IEEE, 2016, pp. 127–132.

[7] I. Bozcan and E. Kayacan, "Uav-adnet: Unsupervised anomaly detection using deep neural networks for aerial surveillance," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1158–1164.

[8] I. Bozcan, J. Le Fevre Sejersen, H. Pham, and E. Kayacan, "Gridnet: Image-agnostic conditional anomaly detection for indoor surveillance," *IEEE Robotics and Automation Letters*, pp. 1–1, 2021.

[9] I. Bozcan and E. Kayacan, "Context-dependent anomaly detection for low altitude traffic surveillance," in *2021 The IEEE International Conference on Robotics and Automation (ICRA)*, 2021, p. In Print.

[10] ——, "Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8504–8510.

[11] 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.

[12] 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.

[13] L. O. Rojas-Perez and J. Martinez-Carranza, "Metric monocular slam and colour segmentation for multiple obstacle avoidance in autonomous flight," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, 2017, pp. 234–239.

[14] S. Jung, S. Hwang, H. Shin, and D. H. Shim, "Perception, guidance, and navigation for indoor autonomous drone racing using deep learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2539–2544, 2018.

[15] 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.

[16] P. Foehn, D. Brescianini, E. Kaufmann, T. Cieslewski, M. Gehrig, M. Muglikar, and D. Scaramuzza, "AlphaPilot: Autonomous Drone Racing," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[17] 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)*, 2018, pp. 0–0.

[18] E. Camci and E. Kayacan, "End-to-end motion planning of quadrotors using deep reinforcement learning," *arXiv preprint arXiv:1909.13599*, 2019.

[19] S. Zhou, M. K. Helwa, A. P. Schoellig, A. Sarabakha, and E. Kayacan, "Knowledge transfer between robots with similar dynamics for high-accuracy impromptu trajectory tracking," in *2019 18th European Control Conference (ECC)*. IEEE, 2019, pp. 1–8.

[20] E. Camci and E. Kayacan, "Learning motion primitives for planning swift maneuvers of quadrotor," *Autonomous Robots*, vol. 43, no. 7, pp. 1733–1745, 2019.

[21] 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*, vol. 36, no. 1, pp. 1–14, 2019.

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[23] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[25] 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.

[26] 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)*, 2020, pp. 1–8.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[28] 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.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2992–2997.

[31] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 5420–5425.

[32] 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.

[33] 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.

[34] 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.

[35] M. Burri, H. Oleynikova, , M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*, Sept 2015.