# Catch Me, If You Can! A Mediated Perception Approach Towards Fully Autonomous Drone Racing

**Florian Ölsner** FLORIAN.OELSNER@SPLEENLAB.AI
*Spleenlab.ai, Germany*

**Stefan Milz** STEFAN.MILZ@SPLEENLAB.AI
*Spleenlab.ai, Germany*
*Ilmenau University of Technology, SECSY, Germany*

## Abstract

Automated flight, e.g. first person view drone racing is a challenging task involving many sub-problems like monocular object detection, 3D pose estimation, mapping, optimal path planning and collision avoidance. Treating this problem, we propose an intuitive solution for the *NeurIPS (2019) Game of Drones* competition, especially the perception focused tier. We formulate a modular system composed of three layers: machine learning based perception, mapping and planning. Fundamental is a robust gate detection for target guidance accompanied with a monocular depth estimation for collision avoidance. The estimated targets are used to create and update the 3D gate positions within a map. Rule based trajectory planning is finally used for optimal flying. Our approach runs in real-time on a state of the art GPU and is able to robustly navigate through different simulated race tracks under challenging conditions, e.g. high speeds, confusing gate positioning and irregular shapes. Our approach ranks on the 3rd place on the final leader board. In this paper we present our system design in detail and provide additional experimental results.

**Keywords:** drone racing, autonomous flight, object detection, depth estimation, mapping

## 1. Introduction

First person view drone racing is a modern sport that recently attracted a lot of interest in the AI research community (Kaufmann et al., 2019; Loquercio et al., 2019; Jung et al., 2018a,b). Besides the competitive thrill it poses many practical problems for an autonomous unmanned aerial vehicle (UAV) including:

- Robust perception based on a monocular video stream
- Optimal trajectory planning with many degrees of freedom (DoF)
- Collision avoidance with static and dynamic objects
- Stable control of a non-linear system at high speeds

Current fully autonomous solutions are still far behind human performance. To accelerate the development and research in this area, several competitions were introduced in the past years, e.g. *IROS Drone Racing*[1] or *AlphaPilot*[2]. The most recent *NeurIPS 2019 Game*

1. https://rise.skku.edu/2019-drone-racing
2. https://www.herox.com/alphapilot/community

*of Drones (GoD)*[3] is the first fully simulation based challenge. Competitors simply need a PC equipped with a decent GPU to participate which makes access to the challenge much easier. Simulation avoids the risk of damaging any equipment during development and offers additional information like object poses and their kinematics. Insights gained from simulation can be transferred to real world applications and even cross-modal training is possible (Bonatti et al., 2019). The simulation environment also offers the possibility to test real hardware in the loop (HIL), e.g. flight controller or machine learning chips which is an important component in the development cycle of applied robotics and another step forward democratizing automated aircrafts.

This paper outlines a solution to the perception focused part of GoD (tier 2). We propose a modular perception based approach that employs state-of-the-art convolutional neural networks (CNNs) for pathway detection and collision avoidance in combination with a simple rule-based mapping and policy algorithm. In the following we will describe the task and the system architecture in detail. In addition to the official final race where our approach ranked 3rd we also provide qualitative and quantitative results from experiments on the training grounds. The evaluation demonstrates that our approach reliably solves the given task and and is capable to finish in a competitive time. Nevertheless, the results also indicate several remaining problems with room for improvement for future competitions to come.

## 2. Related Work

Autonomous navigation of small scale UAVs using on-board sensors only is a young field of research. Early works rely on GPS and inertial measurement unit (IMU) sensors only (Hoffmann et al., 2004), use laser range finders (He et al., 2008; Grzonka et al., 2009) or monocular RGB cameras (Engel et al., 2012, 2014), optionally with an additional depth sensor (Bylow et al., 2013). These techniques employ classical approaches in path planning and computer vision based on visual-inertial odometry or simultaneous localization and mapping (SLAM). They work well when operating in static, feature-rich environments with a good initialization but are less suited when a higher semantic understanding of the surroundings is required. More recent methods use deep neural networks to address this problem. There are two main paradigms in this area.

**The behavior reflex approach** tries to learn a direct mapping from the input domain to the action space. Decisions are made end-to-end using the raw sensor input. Most prominent approaches are imitation learning (Blukis et al., 2018; Bonatti et al., 2019) and deep reinforcement learning (RL) (Walker et al., 2019; Shin et al., 2019). Imitation learning requires a vast amount of training data to generalize well and it is naturally limited by the performance of the demonstrator. RL is more powerful but also (even in simulation) very expensive in terms of training time and hardware requirements.

**The mediated perception approach** semantically reasons about the scene and determines the flying decision based on it. Those systems follow a modular design that decouples the path planning and control from the initial perception step. The approach is popular in drone racing and collision avoidance (Kaufmann et al., 2019; Jung et al., 2018a,b). It enables a more task specific network training which is feasible under limited availability of data,

3. https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing

time and computing power. Furthermore those sub-systems can be tested and validated in a modular way. By attaching redundant paths e.g. passive collision avoidance modules, a higher level of safety can be achieved. For this reason, the solution proposed in this paper follows a similar approach as the winners of the *IROS 2018 Drone Race* (Kaufmann et al., 2019), who aggregate detections obtained from a CNN using an extended Kalman filter in combination with model predictive control. While their system focused on save traversal of a race track in a real indoor environment at comparatively low speed, we target drone racing in a simulated outdoor environment at high velocities under challenging conditions. Our system uses active perception for target guidance (object detection) and passive perception for collision avoidance (monocular depth estimation).

## 3. Challenge

Figure 1: The *AirSim* environment: scenes from the final (left) and qualifier (right) tracks.

The GoD competition uses simulated race tracks in AirSim (Shah et al., 2018) which are marked by consistently textured rectangular gates. They are embedded into realistically looking environments like shown in Figure 1. The general goal is to fly through all gates in the correct order without any collisions. High level control commands and accurate global estimates for pose and kinematics of the drone are available within the environment. That leaves two main tasks for the competition: perception and motion planning. This paper solely addresses tier 2 of the competition which focuses mainly on perception and collision avoidance with static objects without competing drones. As input, only single RGB images of size $(256 \times 144)$ with small field of view are available for the front camera. Noisy gate positions in the correct order are given in advance. To summarize, an autonomous racing system on this tier needs to cope with the following challenges:

- Robust detection and distinction of gates with varying shapes and viewpoints in different environments and lighting conditions
- Inference of 3D gate positions without sensory depth information
- Optimal path planning that enables high speeds, avoids collisions and ensures observability of the target gate
- Dealing with non-linear drag and drift effects at higher speeds
- Providing detections at high frame rates with low latency

## 4. System Design

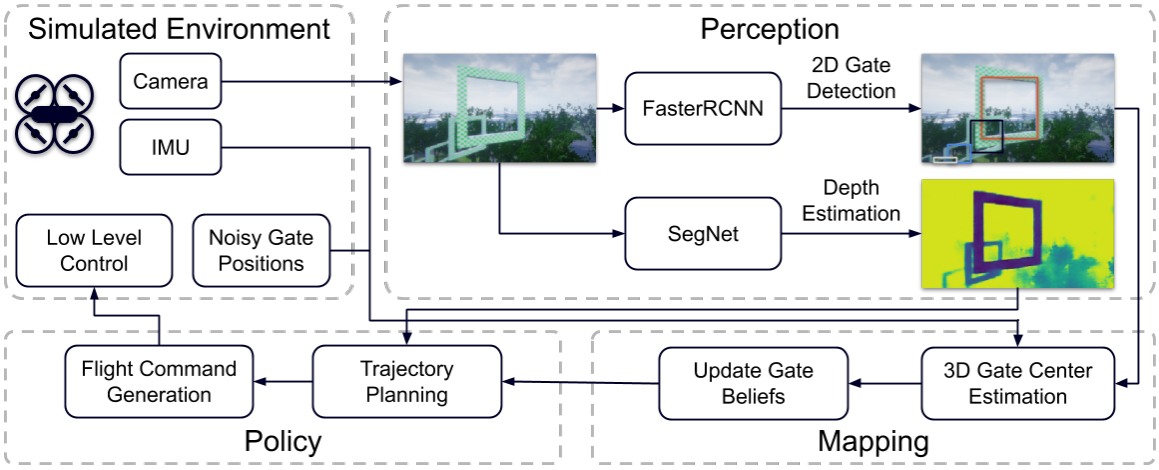



Figure 2: System Architecture

The system is designed in a fly-by-detection scheme. An overview is depicted in Figure 2. It can be subdivided into three main parts: perception, mapping and policy.

**Perception** This pipeline step processes the images obtained from the on-board camera in order to locate the next gates and potential objects that need to be avoided. Two standard CNNs are utilized here. First, for the **active perception**, a FasterRCNN (Ren et al., 2015) with a MobileNetV2 backbone (Sandler et al., 2018) is responsible for detecting up to five 2D bounding boxes around the visible gates. Secondly, for the **passive perception**, an independent SegNet (Badrinarayanan et al., 2015) estimates a dense monocular depth map over the whole image. The architectures are specifically chosen because of their good accuracy-latency trade-off which is important, because the perception step is the most computationally expensive.

**Mapping** In theory a reliable detection of the next gate's bounding box in 2D is sufficient to determine the direction of flight given the known intrinsic and extrinsic camera parameters. However, we found that maintaining a sparse 3D map has several important benefits:

- **Robustness to noise**: By averaging over time the influence of false detections is reduced and outliers are easier to spot.
- **Gate pass detection**: Close to a gate its frame moves out of view. The drone needs to know when the gate is actually passed and to safely adjust the flight direction towards the next gate without the risk of hitting or missing the current one.
- **Speed planning**: To traverse the race track as fast as possible high velocities are important. In case of sharp curves or high lateral positional differences it is not safe to fly too fast. Knowing about 3D gate locations in advance allows for optimal and foresighted speed planning.

The map is represented as an ordered set of pairs $\mathcal{M} = \{(\boldsymbol{p}_i, \mathcal{O}_i) \mid \boldsymbol{p}_i \in \mathbb{R}^3, i = 1 \dots N\}$, where $\boldsymbol{p}_i$ is the belief position of gate $i$ in world coordinates, $\mathcal{O}_i$ are the associated observations of that gate and $N$ is the total number of gates. The set of observations is defined as $\mathcal{O}_i = \{\boldsymbol{c}_{i_k}, \boldsymbol{T}_{i_k} \mid \boldsymbol{c}_{i_k} \in \mathbb{R}^2, \boldsymbol{T}_{i_k} \in \mathbb{SE}(3), k = 1 \dots K_i\}$ where $\boldsymbol{c}_{i_k}$ denotes the 2D position of the detected gate center and $\boldsymbol{T}_{i_k}$ is the associated camera pose for which accurate estimates are available from the simulated IMU. $\mathcal{M}$ is initialized using the given noisy gate positions from the simulator. When new detections are available the current gate beliefs are projected onto the respective image planes by the known function $\boldsymbol{\pi}$ using the projection operator $\otimes$:

$$\hat{\boldsymbol{c}}_{i_k} = \boldsymbol{\pi}(\boldsymbol{T}_{i_k} \otimes \boldsymbol{p_i}) \tag{1}$$

All detected bounding box centers are accordingly matched, filtered and saved. In the next step the affected gate beliefs are updated. For each observation a 3D location in world coordinates is estimated as the point on the projection line running through the bounding box center with minimum distance to the current gate belief:

$$\begin{aligned} \hat{\boldsymbol{p}}_{i_k} &= \boldsymbol{T}_{i_k}^{-1} \otimes s \cdot \boldsymbol{\pi}^{-1}(\boldsymbol{c}_{i_k}) \\ s &= \text{argmin}_s \, \|\hat{\boldsymbol{p}}_{i_k} - \boldsymbol{p}_i\|_2 \end{aligned} \tag{2}$$

The updated gate belief is then computed as the weighted average:

$$\boldsymbol{p}_i = \frac{1}{K_i} \sum_{k=1}^{K_i} w_k \hat{\boldsymbol{p}}_{i_k} \tag{3}$$

The exact weight distribution is a tunable hyperparameter. We found that giving new observations higher weights works best, because they are usually more accurate and reliable. In our final setting we used $w_k = k^2 / \sum_{j=1}^{K_i} j^2$.

**Policy** For the policy there are three main objectives:

- **Time**: Complete the whole track as fast as possible, i.e. pass all gates in the correct order while flying with a high average velocity.
- **Collisions**: Avoid collisions with solid objects like gates or the ground, because that adds a time penalty and additionally slows down the drone.
- **Observability**: Make sure that the next gate is in the field of view in order to allow for good detections and map updates which is crucial to accomplish the first two tasks.

Because the gate positions are computed and refined during the race, the trajectory planning also needs to be defined in an adaptive online fashion. To keep it simple and fast the trajectory is only locally defined in form of a straight line to a waypoint with a constant velocity and a yaw angle. The procedure is detailed further in Algorithm 1. For the movement the drone is allowed to use all three translational DoF. After each map update the local trajectory is recalculated. When the drone is close enough to the waypoint the current gate is counted as passed. Before it proceeds to the next gate the current movement direction is maintained for a short period of time to avoid collisions with the current gate caused by abrupt changes in direction. Overall that results in a piecewise linear path but in practice

the trajectory is more smooth because the commands are passed asynchronously and the movements are constrained by simulated physics. During each iteration the generated velocity vector is verified by the monocular depth prediction. If objects come critically close, the velocity is reset to zero. In practise this interupt occurs rarely.

**Algorithm 1:** LocalTrajectory planning

**Input:** Map $\mathcal{M}$, Depth image $\boldsymbol{D}$, Gate index $i$
**Output:** Velocity vector $\boldsymbol{v}$, yaw angle $\gamma$
$\boldsymbol{p}_i, \mathcal{O}_i \leftarrow \mathcal{M}[i]$
$\boldsymbol{p}_{i+1}, \mathcal{O}_{i+1} \leftarrow \mathcal{M}[i+1]$
$\boldsymbol{T} \leftarrow \text{getPoseIMU}()$
$\boldsymbol{p}_{cam} \leftarrow \boldsymbol{T} \otimes \boldsymbol{p}_{way}$
$\theta \leftarrow \text{calcAzimuth}(\boldsymbol{p}_{cam})$
**if** $\theta < \theta_{max}$ **then** $\gamma \leftarrow \theta$
**else** $\gamma \leftarrow 0$
$v \leftarrow \text{computeSpeed}(v_{base}, \boldsymbol{p}_i, \boldsymbol{p}_{i+1}, \mathcal{O}_i, \boldsymbol{T})$
$\boldsymbol{d} \leftarrow \text{rotate}(p_{cam}, T^{-1})$
$c \leftarrow \text{checkCollision}(\boldsymbol{D}) \quad c \in \{0, 1\}$
$\boldsymbol{v} \leftarrow \frac{v}{\|d\|_2} \boldsymbol{d} \cdot c$

## 5. Training

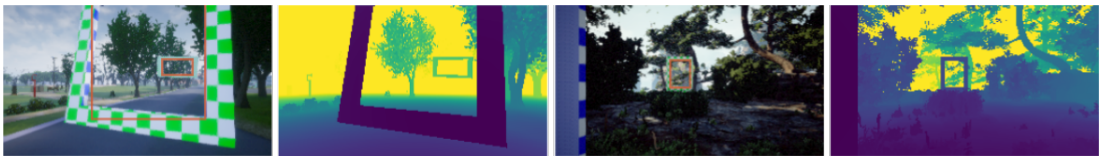


Figure 3: Dataset example with ground truth bounding boxes (left) and depth map (right).

Both neural networks are trained on a self-created dataset extracted from the training environments that were provided by the challenge organizers. In these special environments the simulation allows to obtain and set the true gate poses, to change their size and proportions and to teleport the drone to any location. On top of that, depth maps are available for the captured images. Leveraging these capabilities a dataset of 20.000 samples is generated. Each sample $i$ contains an RGB image from the on-board camera $\boldsymbol{I}_i \in \mathbb{R}^{3\times h\times w}$, a depth map $\boldsymbol{D}_i \in \mathbb{R}^{h\times w}$ and bounding boxes $\boldsymbol{B}_i \in \mathbb{R}^{k\times 4}$ for all $k$ visible gates in that image. An exemplary pair is visualized in Figure 3.

In order to train accurate models that are able to generalize well to unseen tracks the dataset needs to contain a large variety of different viewpoints, gate distances, backgrounds, lighting conditions and gate appearances. To accomplish this each sample is generated using the steps in Algorithm 2. To increase the diversity even further data was extracted from multiple different simulator maps. Both models were trained following the procedures described in (Ren et al., 2015) and (Badrinarayanan et al., 2015). For the FasterRCNN object classes the relative order of the visible gates was used (i.e. class 1 for the next gate, class 2 for the subsequent one and so on). Training was conducted on a *Nvidia RTX 2080ti* using the ADAM optimizer (Kingma and Ba, 2015) and took for both models less than a

**Algorithm 2:** Generation of a single training sample

**Input:** Gates $\mathcal{G}$
**Output:** $\boldsymbol{I}, \boldsymbol{D}, \boldsymbol{B}$
$\alpha \leftarrow$ sampleAngleUniform()
$\boldsymbol{d} \leftarrow$ sampleVecUniform()
$\mathcal{G} \leftarrow$ rotateAndShiftTrack$(\mathcal{G}, \alpha, \boldsymbol{d})$
$i \leftarrow$ sampleGateIdxUniform()
$\boldsymbol{T} \leftarrow$ samplePoseBetweenGates$(\mathcal{G}[i], \mathcal{G}[i+1])$
teleportDrone$(\boldsymbol{T})$
**for** $i \leftarrow 0$ **to** *len*$(\mathcal{G})$ **do**
  $\alpha_i \leftarrow$ sampleAngleNormal()
  $\boldsymbol{d}_i \leftarrow$ sampleVecNormal()
  $\mathcal{G}[i] \leftarrow$ rotateAndShiftGate$(\mathcal{G}[i], \alpha_i, \boldsymbol{d}_i)$
  $\mathcal{G}[i] \leftarrow$ distortSizeAndShape$(\mathcal{G}[i])$
**end**
$\boldsymbol{I}, \boldsymbol{D}, \boldsymbol{B} \leftarrow$ captureData$(\mathcal{G})$
**if** *gateIsinView*$(\mathcal{G}[i])$ **then goto** Start

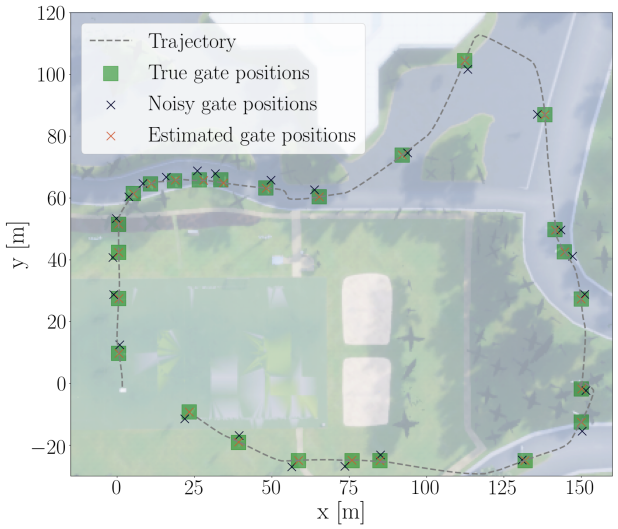
couple of hours until convergence and using a minimum of 200 epochs. The dataset was split into a training and a validation set in a 9:1 ratio.

## 6. Results

Figure 4: Qualitative results on the test tracks: (a, b, e, f) predicted bounding boxes on input image, (c, d, g, h) estimated depth maps with corresponding input image, (top row) good examples, (bottom row) failure cases.

After training the gate detector achieves an average precision (AP) of 0.93 and an average recall (AR) of 0.94 within an considered intersection over union (IoU) range of $0.5 \ldots 0.95$ on the validation set. Depths are estimated with a mean root square error (MRSE) of 2.45 in a range of $0 \ldots 25$m. Both scores are compliant to the state-of-the-art. Some qualitative results from the test tracks are shown in Figure 4. Despite occasional mistakes the gate detector is able to localize the closest gates even when they are partially occluded or viewed from a flat angle. We tested the performance of the whole system on one of the training environments. Figure 5($a$)subfigure depicts the race track from above with the estimated gate positions from the final map $\mathcal{M}$. The average error is 0.38 m which is sufficient for the

drone to complete the whole track without missing any gates in 7/10 runs at moderate speed. The success rate strongly depends on the base speed and the initial noisy gate positions. At higher velocities fewer detections are available for each gate leading to less accurate predictions. For the test tracks policy hyperparameters were manually tuned towards a higher average speed which resulted in a drastically reduced success rate. The scores[4] for the best runs are listed in Table 5($b$)subfigure. Our approach ranked 3rd in both rounds right after the 2nd place.




($a$) Policy evaluation on training track.

| Team | $r$ [%] | $t$ [s] | $v_{max}$ [$\frac{m}{s}$] | $\bar{v}$ [$\frac{m}{s}$] |
|---|---|---|---|---|
| Qualification Round | | | | |
| USRG | 100 | 58.6 | 19.0 | 4.8 |
| Sangyun | 100 | 70.6 | 17.8 | 4.1 |
| **Spleenlab** | **100** | **76.3** | **18.8** | **4.3** |
| Soren | 93 | 59.1 | 13.0 | 4.9 |
| Kukks | 50 | 208.5 | 13.1 | 3.9 |
| Final Round | | | | |
| Sangyun | 100 | 56.9 | 19.3 | 9.8 |
| USRG | 100 | 81.2 | 17.5 | 7.1 |
| **Spleenlab** | **100** | **83.1** | **15.2** | **7.1** |
| Dédale | 62 | 252.0 | 8.1 | 2.4 |

($b$) GoD ranking results.

Figure 5: Experimental results. Figure (a) shows one of the training tracks with estimated gate positions from above. Table (b) lists the final competition leader board, with $r$ as the percentage of passed gates, $t$ as time and the maximum and average velocities $v_{max}$ and $\bar{v}$.

## 7. Conclusion

We present a real-time capable and competitive solution for simulated autonomous drone racing that is composed of three main pipeline steps: perception, mapping and policy. The final competition scores and the experimental results have shown that the system is able to detect and locate the 3D gate positions accurately and reliably. Furthermore, a high frame rate and careful hyper-parameter tuning was required to accomplish the final race performance. However, there is still room for improvement and we plan to investigate further on a more accurate 3D gate position estimation as well as their orientations for a more optimal path planning. Other interesting research directions include the incorporation of RL to solve parts of the problem or adding more redundant perception paths to increase robustness.

4. https://microsoft.github.io/AirSim-NeurIPS2019-Drone-Racing/leaderboard_final.html

## References

Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

Valts Blukis, Nataly Brukhim, Andrew Bennett, Ross A. Knepper, and Yoav Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. In *Robotics: Science and Systems*, volume abs/1806.00047, 2018.

Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. Learning controls using cross-modal representations: Bridging simulation and reality for drone racing. *arXiv preprint arXiv:1909.06993*, 2019.

Erik Bylow, Jürgen Sturm, Christian Kerl, Fredrik Kahl, and Daniel Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems*, volume 2, page 2, 2013.

Jakob Engel, Jürgen Sturm, and Daniel Cremers. Camera-based navigation of a low-cost quadrocopter. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2815–2821. IEEE, 2012.

Jakob Engel, Jürgen Sturm, and Daniel Cremers. Scale-aware navigation of a low-cost quadrocopter with a monocular camera. *Robotics and Autonomous Systems*, 62(11): 1646–1656, 2014.

Slawomir Grzonka, Giorgio Grisetti, and Wolfram Burgard. Towards a navigation system for autonomous indoor flying. In *2009 IEEE International Conference on Robotics and Automation*, pages 2878–2883. IEEE, 2009.

Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *2008 IEEE International Conference on Robotics and Automation*, pages 1814–1820. IEEE, 2008.

Gabe Hoffmann, Dev Gorur Rajnarayan, Steven L Waslander, David Dostal, Jung Soon Jang, and Claire J Tomlin. The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, volume 2, pages 12–E. IEEE, 2004.

Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, 35(1):146–166, 2018a.

Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018b.

Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning

for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego*, 2015.

Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 2019.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

Sang-Yun Shin, Yong-Won Kang, and Yong-Guk Kim. Automatic drone navigation in realistic 3d landscapes using deep reinforcement learning. In *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 1072–1077. IEEE, 2019.

Ory Walker, Fernando Vanegas, Felipe Gonzalez, and Sven Koenig. A deep reinforcement learning framework for uav navigation in indoor environments. In *2019 IEEE Aerospace Conference*, pages 1–14. IEEE, 2019.