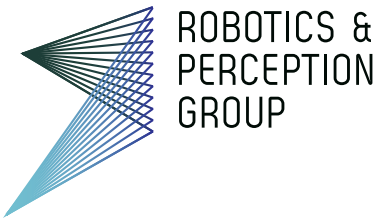
University of Zurich[UZH]
Department of Informatics

University of Zurich[UZH]
ETH zürich
Institute of Neuroinformatics



Cafer Mertcan Akcay

Reinforcement Learning for Offboard Control of a Racing Drone

Semester Thesis

Robotics and Perception Group
University of Zurich

Supervision

Dr. Christian Pfeiffer
Angel Romero
Yunlong Song
Prof. Dr. Davide Scaramuzza

Apr 2022

Contents

Abstract

Autonomous drone racing relies on algorithms running on computers to control the drones. Currently, on-board computers are used to run these algorithms. However, off-board controlled drones can be lighter and reach much higher speeds. Therefore, we integrated an off-the-shelf, low-latency RC-link product to our flight stack to control a lightweight racing drone by an off-board computer. Commands from our flight stack are transmitted from computer to drone via this RC-link, as well as battery voltage telemetry from drone to computer. We measured the latency in the system and confirmed that the off-board system has a very similar latency to the on-board system. We then employed two methods to push the off-board controlled drone to its physical limits. The first method is to track a time-optimal reference trajectory that is computed for this drone and the second method is to train a deep reinforcement learning policy in simulation to deploy in real world. We flew the off-board controlled drone in real word in a racing track using these two methods. We were able to push the drone to 70% of its actual limit in terms of thrust-to-weight ratio with the first method and its full limit with the second method. In both methods, by using off-board control, we were able to reach lap times that were not possible with the on-board controlled drone.

Nomenclature

Acronyms and Abbreviations

CPC	Complementary Progress Constraints
FC	Flight Controller
GPS	Global Positioning System
IMU	Inertial Measurement Unit
MPC	Model Predictive Control
RC	Radio Controller
RF	Radio Frequency
RL	Reinforcement Learning
RPG	Robotics and Perception Group
TWR	Thrust-to-Weight Ratio
VIO	Visual Inertial Odometry

Chapter 1

Introduction

The ultimate goal of autonomous drone racing is to push drones to their physical limits while successfully following race tracks. This is achieved by real-time software stacks which estimate the state of the drone (position, rotation, velocity etc.) with the help of on-board sensory information or external motion capture systems and give the necessary command to move the drone to the desired state. Currently, these algorithms run on on-board computers because it is required to have reliable and low-latency communication between the high-level computer, which gives yaw, pitch, roll and throttle commands and the low-level flight controller (FC), which drives each motor based on these signals. This is particularly important because as latency increases, it becomes harder for the flight stack to control the drone and make necessary maneuvers to follow the race track as fast as possible. However, on-board computers have the disadvantage of adding extra weight as they are mounted on drones, and as a result, decreasing the thrust-to-weight ratio (TWR), which is a measure directly proportional to acceleration, hence the speed. On the other hand, enabling off-board control for autonomous drones make it possible to have lighter drones with higher TWRs that can reach higher speeds. Thus, in this project, we develop a communication interface between a drone and an off-board computer using an off-the-shelf radio controller (RC)-link product consisting of a transmitter and a receiver which is used by professional drone racers. It provides a reliable and low-latency communication which are essential qualities for the agile control of a drone. Then, we test the communication interface by flying the off-board controlled drone in a racing track to push the drone to its physical limits with two approaches to see how much we can push with each. The first approach is to generate a time optimal reference trajectory and to follow this trajectory by model predictive control (MPC). The second approach is training the drone in simulation by deep reinforcement learning (RL) to make it learn a policy which can fly the drone in the race track it is trained for. In Chapter 2, the methodology of the communication interface and the methods that are used to fly the off-board controlled drone will be explained in detail and in Chapter 3, the results of the experiments are presented. Finally in Chapter 4, the advantages, limitations, and future directions of this work are discussed.

1.1 Related Work

Autonomous drone racing is an extension of the more general problem of time-optimal multi-waypoint flight where each gate in racing track is a waypoint and the drone needs to pass all of them as fast as possible. The literature in time-optimal multi-waypoint flight can be divided into model-based approach and learning-based approach.

1.1.1 Model-Based Approach

State-of-the-art model-based works approach this problem in two steps: generating a time-optimal trajectory [3, 8] and following this trajectory by control algorithms. MPC is an example control algorithm that can be employed to track a pre-defined trajectory [1, 2]. In [3], complementary progress constraints (CPC) is proposed, where a time-optimal trajectory is computed by solving an optimization problem by discretizing the trajectory. The optimization problem is constrained by quadrotor dynamics and waypoints. Then, this time-optimal reference trajectory is tracked by MPC to achieve agile flights in real world in the racing track.

1.1.2 Learning-Based Approach

Deep RL proves to be successful in robotics domain and it has been used to control robots autonomously, such as robotic arms [5], legged robots [7], and quadrotors [6]. In [11], deep RL is used to fly a quadrotor in a racing track to achieve a performance close to platform limits. It is achieved by training an agent in simulation and rewarding the progress between the waypoints (i.e gates) such that the agent tries to finish the track as fast as possible. Then, the trained policy is deployed in real world in a racing arena.

Chapter 2

Method

2.1 Quadrotor Specifications

The specifications of on-board and off-board controlled drones that are referred in this work are given in Table 2.1. They can produce the same maximum thrust and they have the same frame and hardware components. The only difference is that the on-board drone (Figure 3.1) carries a computer on top while the off-board drone (Figure 3.2) does not. Using off-board control allows us to remove the on-board computer and results in a 50% increase in TWR and smaller moment of inertia coefficients.

Drone	Max Thrust (N)	Mass (kg)	TWR	diag(J) (gm^2)
on-board	34	0.752	4.6	[2.5, 2.1, 4.3]
off-board	34	0.500	6.9	[2.3, 1.7, 3.7]

Table 2.1: Specifications of the on-board and off-board controlled drones.

2.2 Communication Interface

The communication between the off-board computer and the drone is two-sided and handled by a Team BlackSheep (TBS) Crossfire Micro TX v2 transmitter and a TBS Crossfire Nano RX receiver [1]. Normally, the transmitter is connected to an RC as an external transmitter to send the commands from the pilot to the receiver, which is connected to the drone's FC. In our case, RC is replaced by the computer and the pilot is replaced by our flight stack Agilicious [4]. Agilicious is a software framework that can handle autonomous and agile flight tasks and it can be used by model-based and neural-network-based controllers. The pipeline of our interface can be seen in Figure 2.1.

[1] https://www.team-blacksheep.com

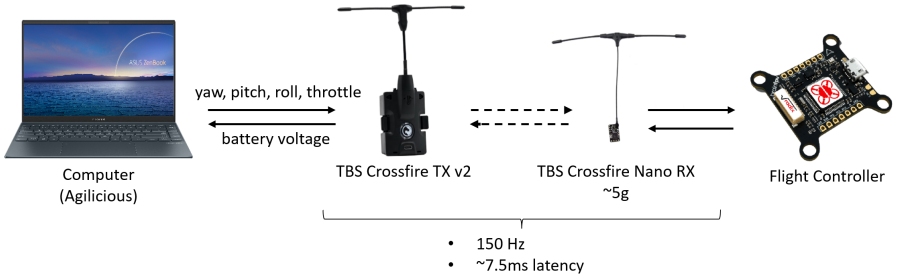


Figure 2.1: Communication pipeline.

TBS products use a proprietary serial communication protocol, called CRSF, to handle the communication between its products. Thanks to the support we get from our collaborator TBS, we are able to use CRSF protocol and construct the frames for our purposes by packing the control commands generated by Agilicious (yaw, pitch, roll, throttle) in the required format. Similarly, we decode the battery voltage telemetry coming from the FC, which is required for running our control algorithms. We can successfully transmit the commands from Agilicious to FC and decode the battery voltage telemetry coming from FC to Agilicious.

The off-board computer communicates with the transmitter by serial communication over a USB to serial converter. The transmitter sends the commands to the receiver which has an update frequency of 150Hz. Average latency at this step from the moment a command leaves the computer until it reaches the FC is approximately 7.5ms. The receiver also communicates with the FC by serial communication and finally, FC gives corresponding signals to motor controllers. Downlink from the FC to the computer follows the same way in the opposite direction.

It is essential to have a reliable link between the off-board computer and the drone with a low latency. Therefore, we measure the latency in the pipeline by using a load cell. We find the overall latency in the system by measuring the elapsed time from the moment the command leaves the computer until the moment it reaches the motors of the drone. We periodically send high and low thrust from Agilicious, 1 second each, and log the time we send each command. We also record the measurement in the load cell. Then, we fit exponentials to the load cell data using motor parameters to find the delay until we observe high and low thrust in motors for 125 high to low or low to high transitions. The result of the current off-board system (Betaflight + TBS) is given in Table 2.2 and illustrated in Figure 2.2 together with other communication setups used before with the Agilicious. Betaflight + Laird is also an off-board communication setup, while the other two are on-board communication setups. We reduce the latency by 50% comparing to the other off-board setup, Betaflight + Laird, and we have on average only 1ms more latency compared to the on-board setup with Betaflight FC. Thus, we can use off-board communication without compromising from the latency.

Method	Min(ms)	Q1(ms)	Median(ms)	Q3(ms)	Max(ms)
Betaflight + TBS	22	33	39	44	51
Betaflight + Laird	67	68	76	81	87
Betaflight on-board	35	36	38	44	49
agiNuttx	31	33	36	37	38

Table 2.2: Latency comparison of our off-board setup with other setups used by Agilicious. Our latency statistics are acquired from 125 latency measurements. Statistics of other setups are acquired from [4]

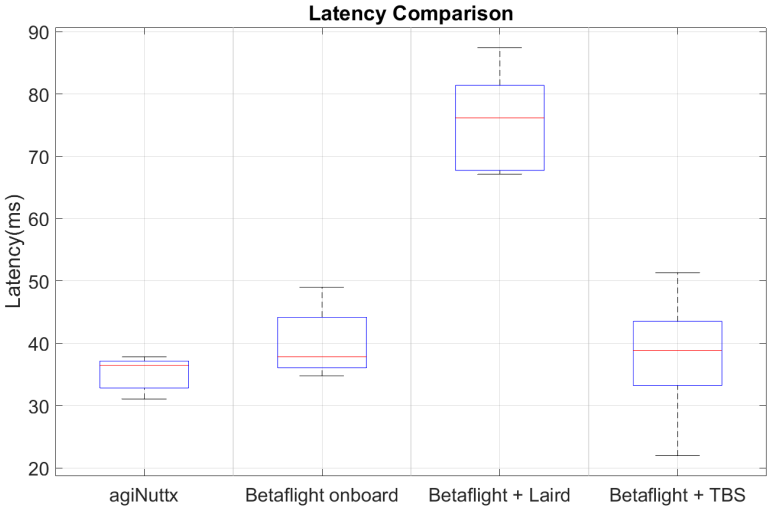


Figure 2.2: Latency comparison.

2.3 Split-S Track

We test our off-board controlled platform in Split-S racing track (Figure 2.3). It is a 3D race track and it consists of 7 gates, each having a passing area of 1.5x1.5m square. The track dimensions are roughly 14x14m and the height of the highest gate center is 3.5m.

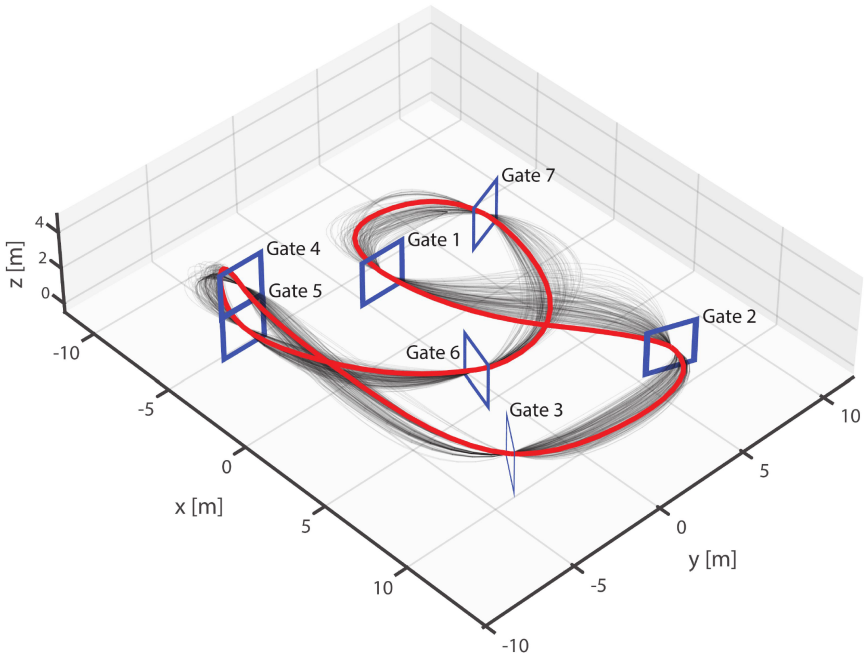


Figure 2.3: Race track with an example trajectory.

2.4 CPC + MPC

To test the communication pipeline, we compute reference trajectories using CPC [3]. CPC aims to find a time-optimal trajectory given a set of waypoints by solving an optimization problem constrained by quadrotor dynamics. The center points of the gates in the race track (Figure2.3) are given as waypoints to find a time optimal trajectory. The reference trajectory is then tracked by using an MPC algorithm. We compute multiple reference trajectories, each with a different TWR value to see how much we can push the drone to its physical limits with CPC+MPC approach.

2.5 Deep RL

The off-board communication protocol is also tested by training a deep RL policy in simulation to fly the drone in real world inside a tracking arena. The simulation environment is Flightmare [10], which includes the race track layout (Figure2.3) we use for real world flights and provides a deep RL API. The policy is trained using the methodology of [11] for the off-board controlled drone for multiple TWR values to achieve the fastest flight in the track possible with the RL approach. The methodology of [11] can be described as follows. The observation space of the RL problem consists of the drone's state (velocity, acceleration, orientation, body rates) and the current state at the track (next gate's position and orientation). The action space is rotor thrusts. The reward function has two components: progress reward and safety reward. If the drone

moves closer to the next gate between two time steps, it receives a reward proportional to the amount of progress towards the next gate. In order to encourage the drone to pass closer to the gate center, a negative safety reward is given to punish positions farther from the gate center while passing the gate. The agent is trained using proximal policy optimization (PPO) [9] in a vectorized environment, which allows training multiple agents at the same time and accelerates training.

Chapter 3

Experiments

3.1 Experiment Setup

The experiments reported in this section are conducted with the off-board controlled drone (Figure 3.2) in the racing track in Figure 2.3. Simulated experiments are conducted in the simulation environment of Agilicious and the real world experiments are conducted inside a large-scale tracking arena using position tracking by infrared cameras and infrared markers. The experiments that had been done with the on-board controlled drone (Figure 3.1) in previous works are also reported for comparison in this section. The specifications of the drones are listed in Table 2.1.

Figure 3.1: On-board controlled drone.

Figure 3.2: Off-board controlled drone.

3.2 CPC + MPC

We compute CPC trajectories with multiple TWRs to see how much we can push the limits with this method and we use MPC to follow these trajectories. The reference trajectories computed with a TWR of 4.5 and 4.9 are successfully followed in the racing track in the arena while the reference trajectory with a TWR of 5.3 cannot be followed with this approach and results in a crash. The lap times are given in Table 3.1 together with the success rates. The table shows that we reach the limit at TWR of 4.9 on off-board controlled platform with CPC + MPC approach.

TWR	Lap time (s)		Success rate (laps)
	Reference	Real world	
4.5	5.1	5.1	3/3
4.9	4.8	4.8	3/3
5.3	4.7	-	0/3

Table 3.1: Lap times using CPC+MPC approach for three TWR values.

In Figure 3.3, the reference and the real world trajectories are plotted with gates for TWR=4.9. It can be seen that the drone passes through the gates while following the reference trajectory. If we further increase the TWR, the drone cannot follow the reference anymore and it crashes or cannot pass through the gates. An example failure case is given in Figure 3.4 where the drone fails to follow the trajectory with a large margin.

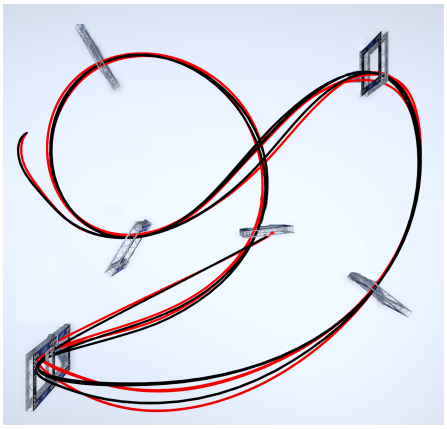


Figure 3.3: Reference trajectory (black) and trajectory flown in real world (red) with TWR=4.9.

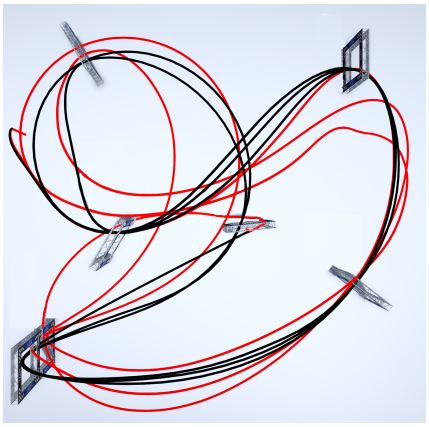


Figure 3.4: Reference trajectory (black) and trajectory flown in real world (red) with TWR=6.1.

With the on-board controlled drone, the highest TWR that could be flown by this approach was 3.3 and the corresponding lap time was 6.12s [3]. We are able to fly a TWR of 4.9, which is 48% more compared to the on-board controlled drone and the drone is able to finish the track in 21% shorter duration using off-board control compared to the on-board controlled drone. The on-board controlled drone has a limit TWR of 4.6 (Table 2.1) and it could fly a maximum TWR of 3.3, which is at the 70% of the drone's limit. The off-board controlled drone has a limit TWR of 6.9 (Table 2.1) and it can fly a maximum TWR of 4.9, which is also at the 70% of the drone's limit. Thus, we can argue that CPC + MPC method can push the quadrotors up to 70% of their limits. By using off-board control, we increase the limit of the quadrotor which also increases the maximum performance we can get by using CPC + MPC.

3.3 Deep RL

We train RL policies with several TWRs to push the off-board controlled drone to its limits. The lap times are reported in Table 3.2 for the simulation and the real world experiments. The RL policy with a TWR of 6.9, which is the actual limit of the drone, can pass all the gates in 4.22s. We also experiment with a policy trained with a higher TWR than the actual limit to observe the effects of deploying a policy trained for a TWR higher than actual value. It leads to a worse performance and proves that the optimum lap time is acquired at the actual TWR of the quadrotor. It can be interpreted that we get higher performance when the simulation that we train the policy in overlaps more with the real world. The table also suggests that the lap times are lower than the simulation except the highest TWR which is larger than the actual limit. This difference might be due to mismatches between the physics models in the simulation and the real world, and the quadrotor parameters.

TWR	Minimum Lap Time (s)		Success rate (laps)
	Simulation	Real World	
5.7	4.80	4.69	3/3
6.5	4.48	4.33	3/3
6.9	4.30	4.22	6/6
7.3	4.16	4.32	3/3

Table 3.2: Lap times in simulation and real world experiments with multiple TWRs using RL policies.

In Table 3.3, the gate passing errors are reported for TWR=6.9 in simulation and real world, which is the Euclidean distance between the center of the gate and the point where the drone passes the gate. It is observed that the mean gate passing error in real world is similar to the one in the simulation.

Gate Passing Error (cm)					
Simulation			Real World		
Min	Max	Mean	Min	Max	Mean
13	33	23	4.2	56	27

Table 3.3: Gate passing errors in simulation and real world experiment for TWR=6.9, from 21 gate passings.

In Table 3.4, the velocity and acceleration statistics are reported for TWR=6.9 from 3 laps. We can reach about 100 km/h with the offboard-controlled setup.

Linear Velocity (km/h)			
Simulation		Real World	
Mean	Max	Mean	Max
62.3	90.4	65.9	98.3

Linear Acceleration (m/s^2)			
Simulation		Real World	
Mean	Max	Mean	Max
57.0	76.3	55.2	70.6

Table 3.4: Linear velocity and acceleration statistics in simulation and in real world experiments for TWR=6.9 from 3 laps.

The trajectories flown in the simulation and in the real world are given in Figure 3.5 with the gates. It can be seen that the drone can successfully pass through the gates even though it deviates from the trajectory flown in simulation. This shows that learning a policy makes the drone robust against deviations from the simulation while flying in real world.

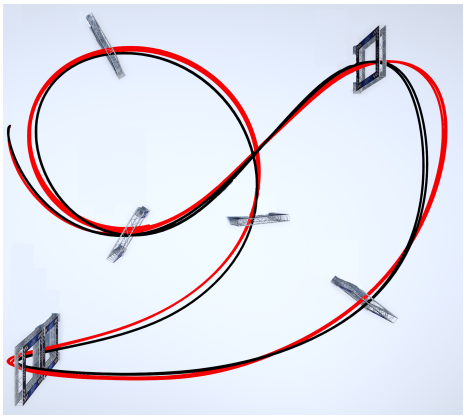


Figure 3.5: Simulation trajectory (black) and trajectory flown in real world (red) with TWR=6.9.

With the on-board controlled drone, the maximum TWR that could be flown with this approach was 4.6 and the fastest lap time was 5.35s. By using the off-board controlled drone, we can fly with a maximum TWR of 6.9 and achieve a 21% less lap time. The limit of on-board controlled drone is a TWR of 4.6 and the limit of the off-board controlled drone is a TWR of 6.9 2.1. This approach could fly both drones in their maximum limit. By increasing the physical limit of the drone with off-board control, we managed to achieve much faster lap times.

Chapter 4

Discussion

4.1 Advantages/Limitations

Our approach has several advantages over on-board control of drones for autonomous racing. The immediate advantage is reducing the weight on the drone due to the on-board computer, thus having a faster drone with higher TWR. Moreover, more computational power is available on off-board computers compared to on-board computers. However, off-board communication is no longer feasible in some applications outside of drone racing, such as when the drone needs to function in remote places where it is outside the range of the transmitter or the radio signal is blocked by some RF noise or obstacle.

4.2 Conclusion

In this work, we proposed a low-latency communication interface for off-board control of racing drones and showed that it is as good as on-board communication in terms of latency. Furthermore, we displayed its merit by flying in a challenging race track with two approaches, CPC + MPC and deep RL, and reaching speeds which were not possible with the on-board controlled drone.

Our work also shows that we can fly quadrotors in their full capacity by using deep RL approach [11] while we can fly only up to 70% of the actual limit by following a CPC reference trajectory by MPC [3]. It is mostly due to the innate differences between these two approaches. RL policy is trained in simulation where the drone experiences many states in the racing track and learns what to do in these states. Therefore, the policy is successful at controlling the drone despite the deviations from the simulation environment. However, the approach of following a time optimal CPC trajectory by MPC tries to follow a reference trajectory and it cannot adapt its behavior well to the deviations from the reference along the way. Therefore, it becomes harder to track the reference as the reference speed increases and it becomes impossible after some limit.

4.3 Future Work

An interesting direction for this work would be to employ more telemetry data such as IMU, camera images, and GPS. The RC-link allows a large bandwidth to be communicated and these different modalities could increase the performance of state estimation and make it possible to use visual-inertial odometry (VIO) with off-board control. Another future direction would be building autonomous micro drones. Since the drones do not need to carry on-board computers, they can be built much smaller. There are some scenarios where autonomous micro drones can be useful such as search and rescue operations. Finally, the proposed off-board communication could be used to control a flock of drones from a single computer, which is more convenient than placing a computer on each drone in the flock.

Bibliography

[1] Moses Bangura and Robert Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[2] M. Diehl, H.G. Bock, H. Diedam, and P.-B. Wieber. Fast direct multiple shooting algorithms for optimal robot control. In *Lecture Notes in Control and Information Sciences*, pages 65–93. Springer Berlin Heidelberg.

[3] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56), 2021.

[4] Philipp Foehn, Angel Romero, Davide Scaramuzza, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 2022.

[5] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017.

[6] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robotics and Automation Letters*, 2(4):2096–2103, October 2017.

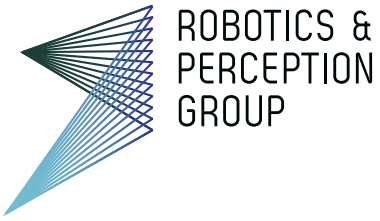
[7] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), October 2020.

[8] Gilhyun Ryou, Ezra Tal, and Sertac Karaman. Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers. *The International Journal of Robotics Research*, 40(12-14):1352–1369, 2021.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[10] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator, 2020.

[11] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. *CoRR*, abs/2103.08624, 2021.



Title of work:

Reinforcement Learning for Offboard Control of a Racing Drone

Thesis type and date:

Semester Thesis, Apr 2022

Supervision:

Dr. Christian Pfeiffer
Angel Romero
Yunlong Song
Prof. Dr. Davide Scaramuzza

Student:

Name:	Cafer Mertcan Akcay
E-mail:	cakcay@student.ethz.ch
Legi-Nr.:	21-946-603

Statement regarding plagiarism:

By signing this statement, I affirm that I have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.lehre.uzh.ch/plagiate/20110314_LK_Plagiarism.pdf`

Zurich, 26. 4. 2022: _____