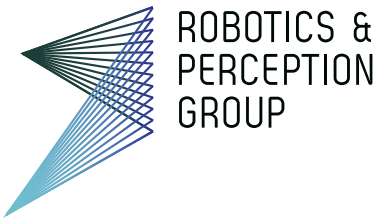
University of Zurich[UZH]
Department of Informatics

University of Zurich[UZH]
ETH zürich
Institute of Neuroinformatics



David Oort Alonso

Collaborative Aerial Transportation Using Deep Reinforcement Learning

Semester Thesis

Robotics and Perception Group
University of Zurich

Supervision

Yunlong Song
Ángel Romero
Dr. Sihao Sun
Prof. Dr. Davide Scaramuzza

February 6, 2022

Contents

Abstract **iii**

Nomenclature **v**

1 Introduction **1**
1.1 Related Work 2

2 Method **3**
2.1 Dynamic Model 4
2.1.1 Equations of Motion 5
2.2 Learning Architecture 8
2.2.1 Learning to Hover 8
2.2.2 Learning to Race 9
2.2.3 Policy Training 10

3 Experiments **11**
3.1 Learning to hover 11
3.2 Learning to race 12

4 Discussion **16**
4.1 Conclusion 16
4.2 Future Work 17

Abstract

In this work, we present the first successful application of Deep Reinforcement Learning to the Collaborative Aerial Transportation task. A centralized policy controlling an arbitrary number of quadrotors attached to a payload is trained using a state-of-the-art dynamic model to perform tasks of increasing difficulty. These tasks are trained and evaluated in simulation and range from hovering with a single quad, to racing a payload attached to three quadrotors through a sequence of gates in a complex track.

Nomenclature

Notation

x_i	position of quadrotor i in the inertial frame
R_i	orientation of quadrotor i with respect to the inertial frame
Ω_i	angular velocity of quadrotor i
q_i	normalized direction of link i
ω_i	angular velocity of link i
l_i	length of link i
ρ_i	offset from payload CoG to attachment point of link i, in the payload frame
x_0	position of the payload in the inertial frame
R_0	orientation of the payload with respect to the inertial frame
J_0	inertia matrix of the payload
m_0	mass of the payload
J_i	inertia matrix of quadrotor i
m_i	mass of quadrotor i
dt	simulation time step
n	number of quadrotors in the collaborative system

Acronyms and Abbreviations

RPG	Robotics and Perception Group
DoF	Degree of Freedom
API	Application Programming Interface
CoG	Center of Gravity
MAV	Micro Aerial Vehicle
RL	Reinforcement Learning
PPO	Proximal Policy Optimization
UAV	Unmanned Aerial Vehicle
MCS	Motion Capture System

Chapter 1

Introduction

Flying robots are increasingly being used for transportation. Delivery of goods, search and rescue and construction are some of the applications where aerial robots can increase the speed and efficiency of operations. The use of a single flying robot for such applications can pose limitations in the time-efficiency, reliability, power-consumption, payload capabilities and size constraints for executing such tasks.

All of these limitations can be addressed with a team of aerial robots. For the case of a payload-carrying team of aerial robots, which is the focus of this work, it is obvious to see that such a system provides more accurate control of the payload pose, removes single points of failure and can be sized for a wide range of payload dimensions and masses without requiring different UAVs.

Deploying a team of aerial robots for collaborative tasks introduces challenges in localization, control and coordination between agents. In the case of a team of aerial robots carrying a payload, this requires *physical* interaction between the agents involved, thereby coupling the dynamics of the agents and leading to aerodynamic and other types of interference.

This work aims to tackle the challenge of collaborative aerial transportation using deep reinforcement learning techniques, something which has not been achieved before. Specifically, it aims to take a first step in solving this class of problems by training a centralized deep reinforcement policy controlling all agents, which are embodied by quadrotors and attached to the payload via rigid links.

The outline of this report is as follows. In the remainder of this chapter, we summarize previous work done in the field of collaborative aerial transportation using non-learning-based methods as well as work in the field of Reinforcement Learning that provides the foundation for our work. In Chapter 2, we outline our approach in creating the quadrotor-payload environment as well as the deep reinforcement learning architecture used to train a centralized policy on a range of tasks of increasing difficulty. In Chapter 3 we present the results of our approach, so far limited to simulation. Lastly, in Chapter 4 we summarize the contributions of this work as well as some of its limitations and how to tackle

them in future efforts in this direction.

1.1 Related Work

The problem of collaborative aerial transport is not new; slung load dynamics have been modelled for decades in the context of helicopter flight and solutions to this problem in the context of MAVs have been presented as early as 2012 in [13] and [11].

In recent years, several approaches have been proposed to achieve collaborative aerial transportation. The first broad distinction that can be made about these approaches is whether they control a slung load system [9, 20, 10, 7] or a system where the load is rigidly attached to the MAVs [18, 17, 11]. Despite the more compact resulting system when attaching the MAVs rigidly to the payload, doing so increases the system inertia, and couples the DoF of the system. In slung load systems, a passive connection is used to suspend the payload from the MAVs with the use of rigid links or non-rigid cables. Such a connection requires minimal design for the load and vehicles and is generally low cost. In addition, the DoF of the payload are easier to decouple from the DoF of the MAVs, increasing the space of possible configurations and thereby improving the maneuverability of the system. The second broad distinction that can be made is whether the control law used to control the systems is centralized [9, 20, 13] or decentralized [20, 10, 7, 18, 17, 11]. A centralized approach can lead to more optimal control and is easier to design but it requires communication between all MAVs and a central coordinator. Decentralized approaches typically employ a leader-follower scheme [7, 18, 17], which has the disadvantage of relying on a single master agent for navigation and thereby is subject to a single point of failure. To address this, [10] presents a fully distributed approach where there is no hierarchy between agents, and instead all agents communicate with each other.

All of these approaches have shown limited agility and speeds during payload-carrying tasks, operating in near hover condition, except for [20, 9] which show speeds of up to ≈ 2m/s but only in a simulated environment, and [13] whose approach is based on computing an offline, open-loop optimal trajectory and tracking it online on a physical platform, and is limited to a specific ball-throwing task.

Reinforcement Learning has been applied to the problem of aerial transport in [6, 1] and more recently, Deep Reinforcement Learning has been shown to enable near-time-optimal racing of a single quadrotor through a track of gates [16]. This makes Deep Reinforcement Learning a promising approach to address some of the agility limitations of previous work in collaborative aerial transport. Indeed, the goal of this paper is to be the first stepping stone towards agile collaborative aerial transport using Deep Reinforcement Learning.

Chapter 2

Method

This chapter presents both the approach to modelling the problem as well as the approach to solving it, thereby arriving at the results presented in 3.

An overview of our approach is shown in Figure 2.1. After selecting an existing dynamic model, we implemented it in MATLAB to quickly debug and verify its stability and soundness. Next, we ported this MATLAB model to the Flightmare simulator [15] by creating a new OpenAI gym-style [3] environment for Reinforcement Learning. Using this environment, we trained a centralized policy on a set of increasing tasks using PPO.

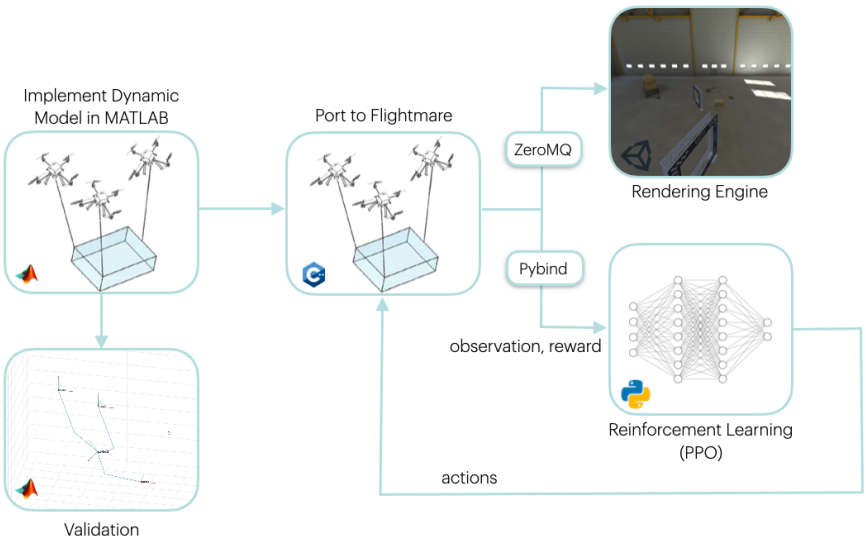


Figure 2.1: Overview of the approach of this project

2.1 Dynamic Model

Several dynamic models of a slung load system have been developed over the years. A common method is a hybrid model where the cables attaching the payload and MAVs can be taut and slack [2, 19, 4]. Another approach has been to model the cable by a serial connection of rigid links where the payload is the last link. However, this method is limited to a point-mass payload and a single quadrotor. In [5], a method of modelling the link as a revolute and prismatic joint is presented, but in the context of a single MAV. Lastly, a popular approach in recent work [9, 20, 10] has been to model the cables as rigid links which provides a kinematic relationship between the payload and the MAVs. This last approach is chosen for this project because it can easily scale to an arbitrary number of MAVs, with arbitrary points of attachment to the payload and is more straightforward to implement than a hybrid model. As will be discussed in Chapter 4, this model is not suited for high-speed aggressive flight, as in such regimes the taut assumption of the cable is violated and unmodelled aerodynamics play a significant role. That said, this model is sufficient to showcase the potential of our approach in simulation.

Consider n quadrotor UAVs that are connected to a payload, that is modeled as a rigid body, via massless links, as depicted in Figure 2.2. Our convention is to denote any variables related to the payload with a subscript of 0 and any variables related to quadrotor and link i with the subscript i. We choose an inertial reference frame $\{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$ and bodyfixed frames $\left\{\vec{b}_{j_1}, \vec{b}_{j_2}, \vec{b}_{j_3}\right\}$ for $0 \leq j \leq n$ as follows. For the inertial frame, the third axis $\vec{e}_3$ points upwards and the other axes are chosen to form an orthonormal frame. The origin of the j-th body-fixed frame is located at the center of mass of the payload for $j = 0$ and at the mass center the quadrotor for $1 \leq j \leq n$. The third body-fixed axis $\vec{b}_{i_3}$ is normal to the plane defined by the centers of rotors, and it points upwards along the direction of the thrust vector.

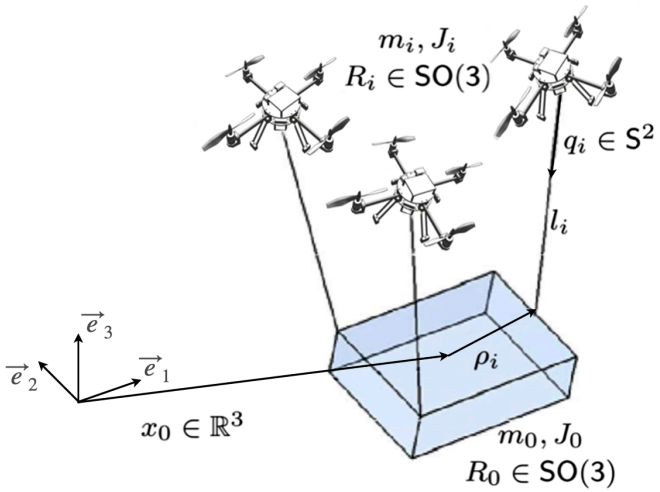


Figure 2.2: Implemented dynamic model diagram

The CoG position of the payload is denoted by $x_0 \in \mathbb{R}^3$, and its attitude is given by $R_0 \in \mathrm{SO}(3)$. We define $\rho_i \in \mathbb{R}^3$ as the point on the payload where link i is attached, expressed in the zeroth (payload) body-fixed frame. The other end of the link is attached to the CoG of the i-th quadrotor. The direction of the link from the CoG of the i-th quadrotor toward the payload is defined by the unit-vector $q_i \in \mathrm{S}^2$, where $\mathrm{S}^2 = \left\{q \in \mathbb{R}^3 \middle| \|q\| = 1\right\}$, and the length of the i-th link is denoted by $l_i \in \mathbb{R}$.

Let $x_i \in \mathbb{R}^3$ be the location of the mass center of the i-th quadrotor with respect to the inertial frame. As the link is assumed to be rigid, we have $x_i = x_0 + R_0\rho_i - l_i q_i$. The attitude of the i-th quadrotor is defined by $R_i \in \mathrm{SO}(3)$, which represents the linear transformation of the representation of a vector from the i-th body-fixed frame to the inertial frame.

The mass and the inertia matrix of the payload are denoted by $m_0 \in \mathbb{R}$ and $J_0 \in \mathbb{R}^{3\times3}$, respectively. The mass and the inertia matrix of the i-th quadrotor are denoted by $m_i \in \mathbb{R}$ and $J_i \in \mathbb{R}^{3\times3}$, respectively. The i-th quadrotor can generates a thrust $f_i R_i e_3 \in \mathbb{R}^3$ with respect to the inertial frame, where $f_i \in \mathbb{R}$ is the total thrust magnitude and $e_3 = [0, 0, 1]^T \in \mathbb{R}^3$. It also generates a moment $M_i \in \mathbb{R}^3$ with respect to its body-fixed frame. The control input of this system corresponds to $\{f_i, M_i\}_{1\le i\le n}$. Note that, as shown in Figure 2.3, the input to the dynamic model was modified to collective thrust and body rates for each quadrotor in the learning environment.

2.1.1 Equations of Motion

The kinematic equations for the payload, quadrotors, and links are given by

$$\dot{q}_i = \omega_i \times q_i = \hat{\omega}_i q_i \qquad \dot{R}_0 = R_0\hat{\Omega}_0, \quad \dot{R}_i = R_i\hat{\Omega}_i \tag{2.1}$$

where $\omega_i \in \mathbb{R}^3$ is the angular velocity of the i-th link, satisfying $q_i \cdot \omega_i = 0$, and Ω_0 and $\Omega_i \in \mathbb{R}$ are the angular velocities of the payload and the i-th quadrotor expressed with respect to its body-fixed frame, respectively. The hat map $\hat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ is defined by the condition that $\hat{x}y = x \times y$ for all $x, y \in \mathbb{R}^3$, and the inverse of the hat map is denoted by the vee map $\vee : \mathfrak{so}(3) \to \mathbb{R}^3$. The velocity of the i-th quadrotor is given by $\dot{x}_i = \dot{x}_0 + \dot{R}_0\rho_i - l_i\dot{q}_i$.

We equations of motion are derived according to Lagrangian mechanics in [9] and are given by the following:

$$M_q\left(\ddot{x}_0 - ge_3\right) - \sum_{i=1}^{n} m_i q_i q_i^T R_0\hat{\rho}_i\dot{\Omega}_0 = \sum_{i=1}^{n} u_i^{\parallel} - m_i l_i \left\|\omega_i\right\|^2 q_i - m_i q_i q_i^T R_0\hat{\Omega}_0^2\rho_i \tag{2.2}$$

$$\begin{aligned}\left(J_0 - \sum_{i=1}^{n} m_i\hat{\rho}_i R_0^T q_i q_i^T R_0\hat{\rho}_i\right)\dot{\Omega}_0 + \sum_{i=1}^{n} m_i\hat{\rho}_i R_0^T q_i q_i^T\left(\ddot{x}_0 - ge_3\right) \\ +\hat{\Omega}_0 J_0\Omega_0 = \sum_{i=1}^{n}\hat{\rho}_i R_0^T\left(u_i^{\parallel} - m_i l_i\left\|\omega_i\right\|^2 q_i - m_i q_i q_i^T R_0\hat{\Omega}_0^2\rho_i\right)\end{aligned} \tag{2.3}$$

$$\dot{\omega}_i = \frac{1}{l_i}\hat{q}_i\left(\ddot{x}_0 - ge_3 - R_0\hat{\rho}_i\dot{\Omega}_0 + R_0\hat{\Omega}_0^2\rho_i\right) - \frac{1}{m_i l_i}\hat{q}_i u_i^{\perp} \tag{2.4}$$

$$\mathrm{J}_i\dot{\Omega}_i + \Omega_i \times J_i\Omega_i = M_i \tag{2.5}$$

where $M_q = m_y I + \sum_{i=1}^{n} m_i q_i q_i^T \in \mathbb{R}^{3\times 3}$, which is symmetric, positive-definite for any q_i. The vector $u_i \in \mathbb{R}^3$ represents the control force at the i-th quadrotor, i.e., $u_i = f_i R_i e_3$. The vectors $u_i^{\|}$and $u_i^{\perp} \in \mathbb{R}^3$ denote the orthogonal projection of u_i along q_i and the orthogonal projection of u_i to the plane normal to q_i, respectively, i.e.,

$$u_i^{\|} = \left(I + \hat{q}_i^2\right) u_i = \left(q_i \cdot u_i\right) q_i = q_i q_i^T u_i \tag{2.6}$$

$$u_i^{\perp} = -\hat{q}_i^2 u_i = -q_i \times \left(q_i \times u_i\right) = \left(I - q_i q_i^T\right) u_i \tag{2.7}$$

Therefore, $u_i = u_i^{\|} + u_i^{\perp}$.

Alternatively, the (coupled) equations of motion can be expressed in matrix form as follows:

$$\begin{bmatrix} m_T & \sum_{i=1}^{n} -m_i R_0\hat{\rho}_i & m_1 l_1\hat{q}_1 & \cdots & m_n l_n\hat{q}_n \\ \sum_{i=1}^{n} m_i\hat{\rho}_i R_0^T & \bar{J}_0 & m_1 l_1\hat{\rho}_1 R_0^T\hat{q}_1 & \cdots & m_n l_n\hat{\rho}_n R_0^T\hat{q}_n \\ -m_1 l_1\hat{q}_1 & m_1 l_1\hat{q}_1 R_0\hat{\rho}_1 & m_1 l_1^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ -m_n l_n\hat{q}_n & m_n l_n\hat{q}_n R_0\hat{\rho}_n & 0 & \ldots & m_n l_n^2 \end{bmatrix} \begin{bmatrix} \ddot{x}_0 \\ \dot{\Omega}_0 \\ \dot{\omega}_1 \\ \vdots \\ \dot{\omega}_n \end{bmatrix} =$$

$$\begin{bmatrix} -\sum_{i=1}^{n}\left\{m_i R_0\hat{\Omega}_0^2\rho_i + m_i l_i\left\|\omega_i\right\|^2 q_i\right\} + m_T ge_3 + \sum_{i=1}^{n} u_i \\ -\hat{\Omega}_0\bar{J}_0\Omega_0 - \sum_{i=1}^{n} m_i l_i\hat{\rho}_i R_0^T\left\|\omega_i\right\|^2 q_i + \sum_{i=1}^{n}\hat{\rho}_i R_0^T\left(u_i + m_i ge_3\right) \\ m_1 l_1\hat{q}_1 R_0\hat{\Omega}_0^2\rho_1 - l_1\hat{q}_1\left(u_1 + m_1 ge_3\right) \\ \vdots \\ m_n l_n\hat{q}_n R_0\hat{\Omega}_0^2\rho_n - l_n\hat{q}_i\left(u_n + m_n ge_3\right) \end{bmatrix}$$

where $m_T = m_0 + \sum_{i=1}^{n} m_i \in \mathbb{R}^3$ and $\bar{J}_0 = J_0 - \sum_{i=1}^{n} m_i\hat{\rho}_i^2 \in \mathbb{R}^{3\times 3}$. This can be rewritten more compactly as:

$$\dot{x}_{0q} = M_{0q}^{-1} f_{0q}(x, u) \tag{2.8}$$

where $x = [x_0^T, v_0^T, r_0^T, \Omega_0^T, q_i^T, \omega_i^T, r_i^T, \Omega_i^T]^T$, $x_{0q} = [v_0^T, \Omega_0^T, \omega_i^T]^T$ and $u = [u_i]$ for $i = 1, ..., n$. In the above, r_0^T, r_i^T correspond to the quaternion representation of R_0 and R_i respectively.

The states are integrated using forward Euler using the following:

$$
\begin{aligned}
\dot{x}_0 &:= \dot{x}_0 + \ddot{x}_0 \mathrm{dt} \\
x_0 &:= x_0 + \dot{x}_0 \mathrm{dt} \\
r_i &:= \frac{dt}{2} \Lambda(\Omega_i) r_i && i = 0, ..., n \\
\Omega_i &:= \Omega_i + \dot{\Omega}_0 \mathrm{dt} && i = 0, ..., n \\
\omega_i &:= \omega_i + \dot{\omega}_0 \mathrm{dt} && i = 1, ..., n \\
q_i &:= q_i + \dot{q}_i \mathrm{dt} && \text{s.t.} \|q_i\| = 1
\end{aligned}
$$

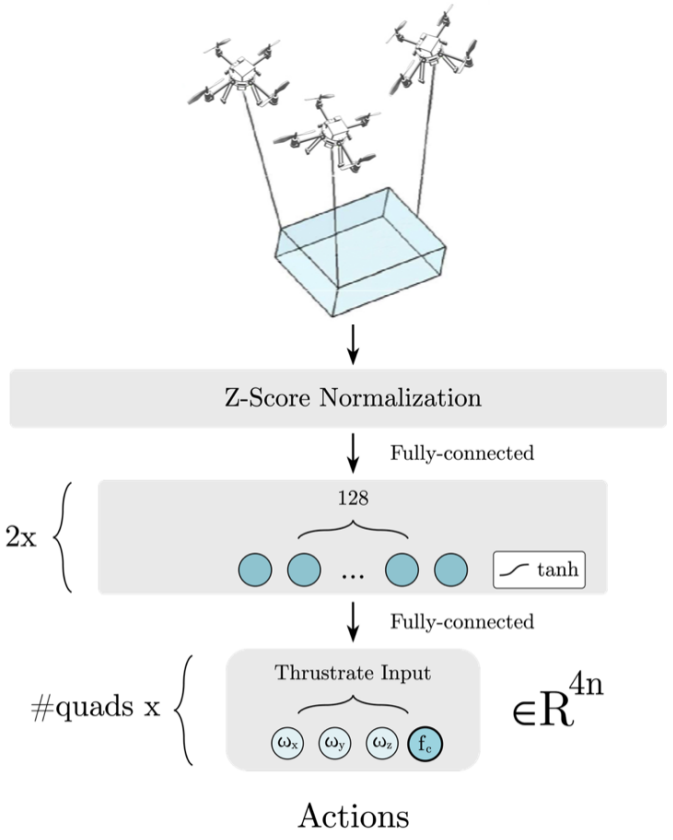


Figure 2.3: Overview of the learning architecture used to train a centralized policy

2.2 Learning Architecture

As mentioned in the introduction of this chapter, we implemented the above dynamic model as an environment in the Flightmare [15] simulator, which includes an API for reinforcement learning which can train hundreds of agents in parallel.

An overview of the learning architecture is shown in 2.3. This architecture remains the same across all tasks. The agent is represented by a policy neural network parametrized by parameters θ and is trained to map a task-dependent observation vector from the environment into a vector of actions $a_t = [a_t^{1T}, ..., a_t^{nT}]^T \in \mathbb{R}^{4n}$ where $a_t^i = [\Omega_x^i, \Omega_y^i, \Omega_z^i, f_i]^T \in \mathbb{R}^4$ (body-rate and thrust control). The feedback provided to the agent is a task-dependent reward signal.

2.2.1 Learning to Hover

We begin by training our agent on a simple task: to carry the payload from a randomly initialized point on the map to a goal position and to hover in that goal position. We begin in an environment with one quadrotor attached to the payload, $n = 1$ and progressively train it on configurations with more quadrotors attached to the payload which become more complex to control in a coordinated manner and have increasing observation and action dimensions for the policy to map between.

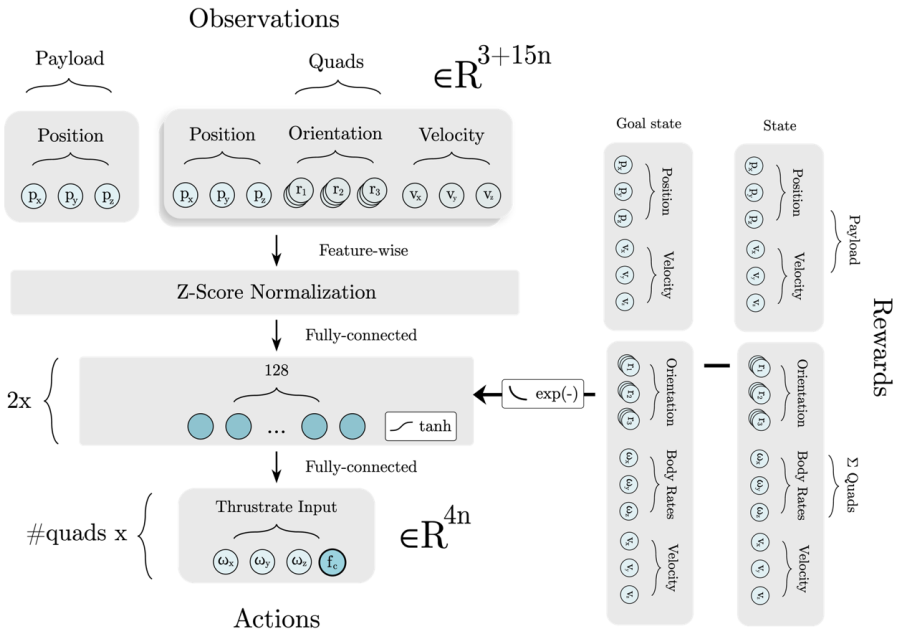


Figure 2.4: Overview of the learning architecture for the hover task.

A diagram of the hover task learning architecture is shown in Figure 2.4. Note that the core learning module remains unchanged, only the observations and rewards to the module change. The observation vector consists of the payload

position, as well as the position orientation and linear velocity of each quadrotor in the environment. Throughout this work, the quadrotor orientation is described using rotation matrices as was done in previous work on quadrotor control using RL [8]. The reward is computed by first taking the norm of the difference between components of the system state (payload position and linear velocity, quadrotor orientation in euler angles, bodyrate and velocity) and their respective goal state, taking the negative exponent of each of the resulting norms and finally adding all of these reward components. The goal state is to get the payload position to a prescribed position and to keep the rest of state components as close as possible to 0.

In addition to the constant reward signal above, sparse penalties are given to the agent in the event of a crash, which are proportional to the speed of impact of the payload with the walls or ground.

2.2.2 Learning to Race

Learning to fly to a waypoint and to hover is a good warm up task to validate that the environment has been implemented correctly and that the learning algorithm is able to correctly interpret observations and rewards, but it does not address the agility limitations found in previous work. To showcase the potential of RL in the control of multiple MAVs attached to a payload, a racing task similar to the one presented in [16] to generate single agent near-time-optimal trajectories. In this task, the payload is to be flown through a series of gates in a complex track in the shortest time possible, without colliding with the gates or the ground.

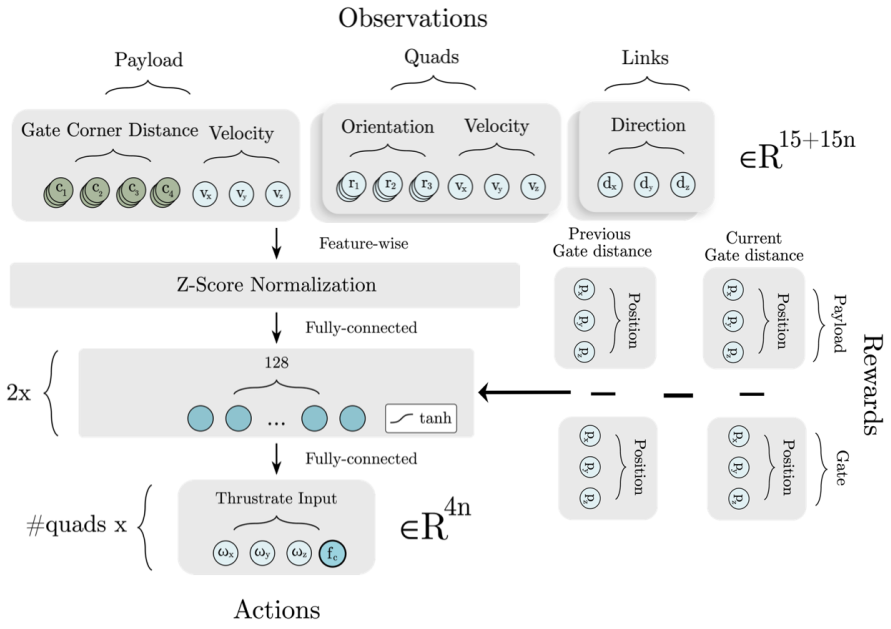


Figure 2.5: Overview of the learning architecture for the racing task.

In Figure 2.5 a diagram of the high-level architecture used to train the agent for

this task is shown. Note that, again, the core learning module has not changed. To switch between a simple hover task and a complex racing task, it is only necessary to change the observation vector and reward signal.

For this task, the observation vector consists of the relative position between the payload and each corner of the next gate to be passed, the payload linear velocity, as well as the orientation and linear velocity of each quadrotor and the direction of every link. The reward has less components than in the hover task and simply consists of a gate progress reward which measures the difference between the distance of the payload to the gate center in the previous time step and in the current time step.

In addition to the constant reward signal above, sparse penalties are given to the agent in the event of a crash, which are proportional to the speed of impact of the payload with the walls, ground or gates. In addition, sparse rewards are given in the event of completing a lap.

2.2.3 Policy Training

We train our stochastic policy using the Proximal Policy Optimization (PPO) algorithm [14], an on-policy, policy gradient method that is well-known for its ease of implementation and good performance on continuous state and action space problems. Our implementation of PPO is based on [12] and [16] and has been modified to learn in a high dimensional observation and action space. We use a few techniques during training to prevent overfitting and to increase training speed:

1. *Parallel Sampling Scheme*: by training our agent in simulation, we are ablet to parallelize the training process by performing rollouts in 100 environments simultaneously which significantly speeds up the data collection process in every epoch.

2. *Distributed Initialization Strategy*: similar to [16], in the racing task, we initialize the system in previously saved valid states in the track segment where the agent last crashed. If this is not done, the first parts of the track will be explored much more than the last and trickiest parts of the track.

3. *Randomization Curriculum*: When the environment is reset, we apply a disturbance to the initial state, the amplitude of which depends on the training progress. We do this to encourage more exploration of the vast state space and make the policy robust to small disturbances which can make it deviate from its nominal trajectory.

Chapter 3

Experiments

We perform our experiments in the Flightmare [15] simulated environment, using the dynamics presented in section 2.1 and the observations and reward discussed in section 2.2. Experiments are conducted attaching a varying number of quadrotors with the same properties to a payload with constant properties. Quadrotor and payload masses and inertias are the same on all experiments: $J_0 = J_i = \text{diag}([0.0025,\ 0.0021,\ 0.0043])$ kg m^2, $m_0 = 0.5$ kg, $m_i = 0.752$ kg. The maximum thrust of each of the four motors on every quadrotor is 8.299 N, and the maximum bodyrate that can be commanded is $\omega_{max} = [8, 8, 4]$ rad/s. In addition, through all these experiments, the links are attached to the CoG of the payload ($\rho_i = 0$).

We train the agents in a vectorized fashion, doing 100 rollouts per epoch and 10 epochs per iteration. The discount factor $\gamma = 0.99$ and the number of steps per rollout is adjusted per experiment to make sure that the time horizon is long enough for the agent to be rewarded or penalized for finishing a lap or crashing, respectively. It is important to restate that the agent is only penalized for payload crashes with the environment, not for any quadrotor collisions.

3.1 Learning to hover

The first task that our agent attempts to learn is to hover with a single quadrotor attached to a payload. The goal position where the payload should be brought is $p_{goal} = [0, 0, 5]$. In Figure 3.2 it is visible that the agent successfully achieves this task. Interestingly, the command input plots on the bottom left show quite high frequency behaviour. The reason for this is that the coefficient for the body rate norm is too low so the agent does not get penalized directly for it, and since the rotational dynamics of the quad are decoupled from the rest of the system, it does not affect the other states as can be seen in the rest of the plots.

The next step in difficulty is to perform the same task but by controlling 3 quadrotors instead of 1. Figure 3.3 shows the results achieved on this task and Figure 3.1 shows a visualization of the hover state the quadrotors end in. Intuitively, One would think that the quadrotors would just stack on top of each

Figure 3.1: Rendering of the hovering task with 3 quadrotors in Unity

other, all producing the same thrust, but here again, upon closer inspection it can be seen that the thrust magnitude, as well as the link direction or the quadrotor position is not linked to a reward so there is no reason why one quadrotor would take most of the load while the other quadrotors carry mostly their own weight, which is what is happening in this experiment.

3.2 Learning to race

After obtaining promising results in the hovering task, the agent was challenged to a much more complex task. In this case, due to the presence of gates in the environment, the observation dimension grows to $\mathbb{R}^{15+15n}$ as shown in Figure 2.5, and reward shaping (through progress rewards) is crucial for the agent to understand that the task is to race the payload through complex track in the shortest time possible. A policy is trained on progressively harder tracks, starting with a circular track, then a figure 8 track and finally a Split S track which has a layout as shown in Figure 3.4.

We begin by training an agent controlling a single quadrotor and obtain the results shown in Figure 3.4. As can be seen from the top speeds, from the thrust profile of the quadrotor (almost constant maximum thrust) and from the smooth trajectory of the payload through the gates, the agent is pushing the limits of the performance of the system and completing a lap in a time of 6.8s.

Finally, the the hardest task, we train an agent to control 3 quadrotors such that the payload they collaboratively carry passes through all gates in the shortest

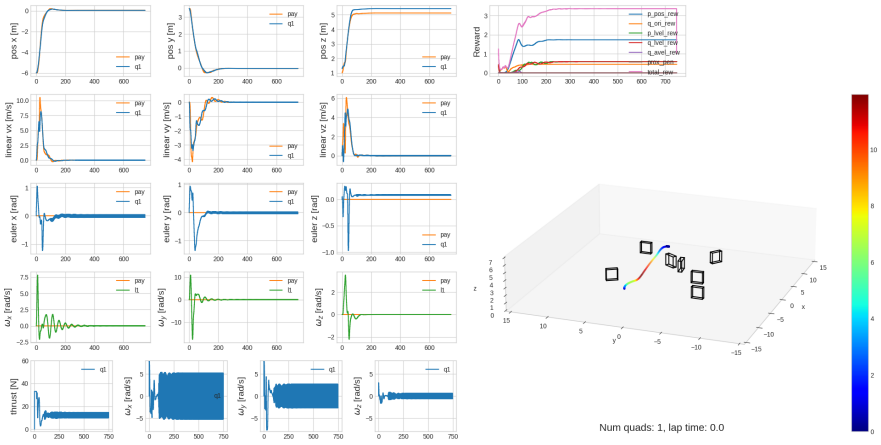


Figure 3.2: Evaluation of a policy trained for 700 iterations on the hovering task with a single quadrotor. The length of the cable is $l_1 = 0.3$m. The 3D plot on the right shows the trajectory of the payload.

time possible so that it maximizes the progress reward while avoiding collision penalties. The results of evaluating the policy after 2000 iterations are shown in Figure 3.5. A rendered visualization of the configuration in Unity is shown in 3.6. Once again it can be inferred from the 3D plot, and the thrust profile that the platform is being pushed to its speed limits, achieving a superior lap time of 5.84s.

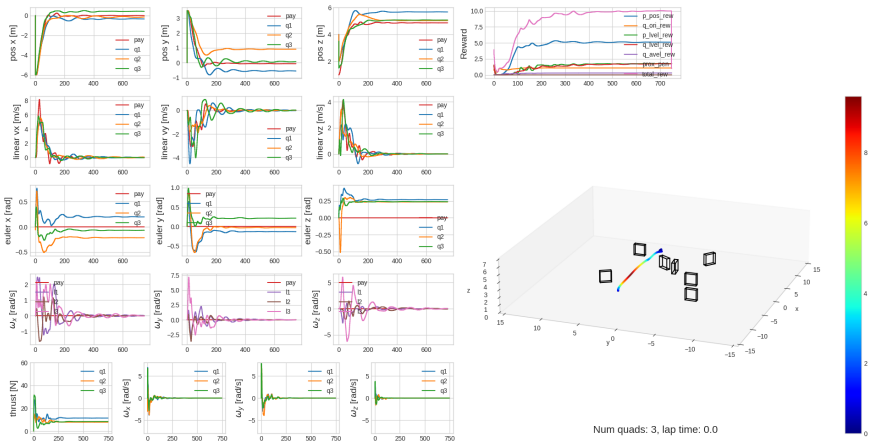


Figure 3.3: Evaluation of a policy trained for 500 iterations on the hovering task with a 3 quadrotors. The length of the links are $l_1 = 1$m, $l_2 = 1$m, $l_3 = 0.5$m. The observations $s_t \in \mathbb{R}^{48}$ and the actions $a_t \in \mathbb{R}^{12}$. The 3D plot on the right shows the trajectory of the payload.

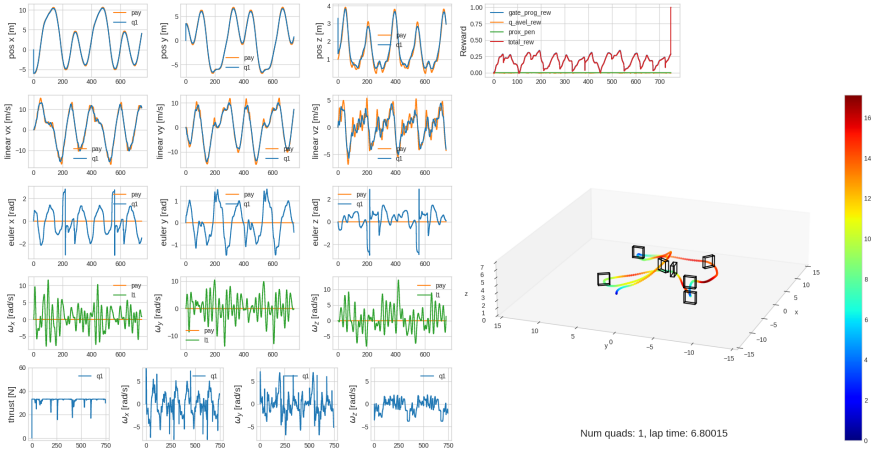


Figure 3.4: Evaluation of a policy trained for 900 iterations on the racing task on the *Split S* with a single quadrotor. The length of the cable is $l_1 = 0.3$m. The 3D plot on the right shows the trajectory of the payload.

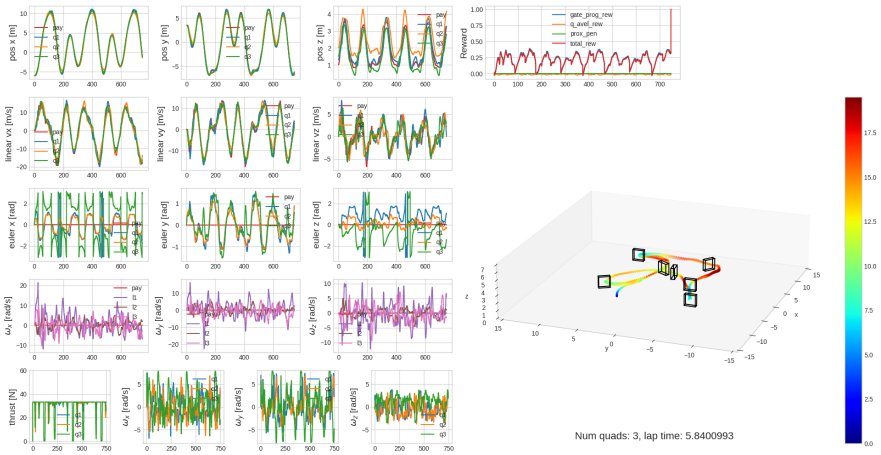


Figure 3.5: Evaluation of a policy trained for 2000 iterations on the racing task on the *Split S* track with a 3 quadrotors. The length of the cables are $l_1 = 0.3$m, $l_2 = 1$m, $l_3 = 0.5$m. The observations $s_t \in \mathbb{R}^{60}$ and the actions $a_t \in \mathbb{R}^{12}$. The 3D plot on the right shows the trajectory of the payload.

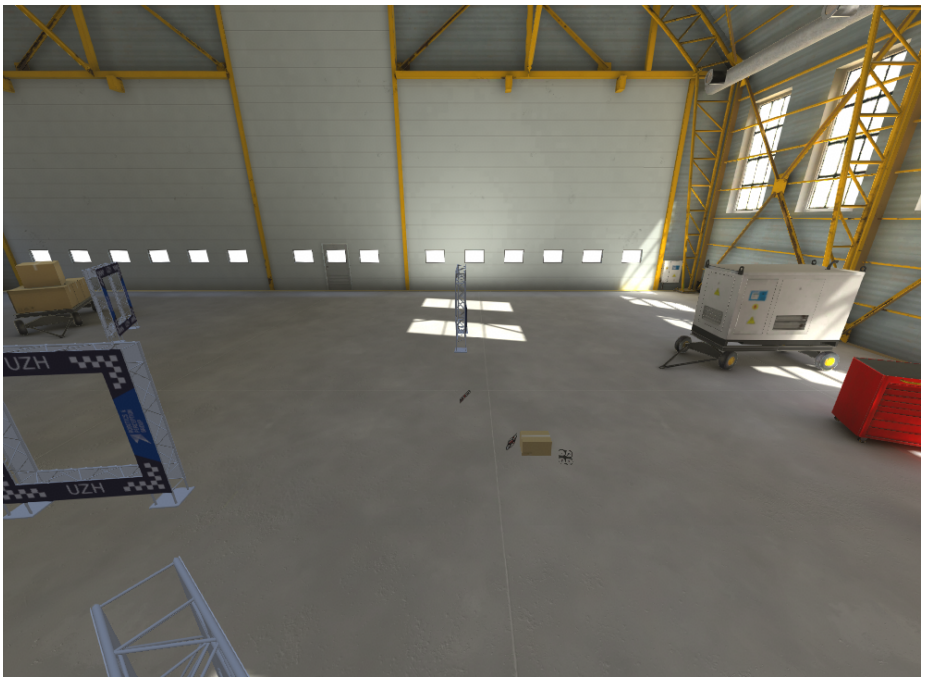


Figure 3.6: Rendering of the racing task for 3 quadrotors in Unity.

Chapter 4

Discussion

4.1 Conclusion

In this work, we have presented a novel approach of tackling the collaborative aerial transportation problem. By leveraging a state-of-the-art Deep RL architecture developed for single agent racing and an existing dynamic model of a slung load system, we have successfully trained a centralized policy to perform tasks ranging from waypoint following and hovering to racing though a complex track. This is a step towards agile collaborative flight and shows that Deep RL has potential in this field. We have also built a new RL environment from scratch in the Flightmare simulator, which easily allows the user to train a system with an arbitrary number of quadrotors to perform payload-carrying tasks.

As a first step towards collaborative aerial transportation using Deep RL, our results so far do come with a handful of limitations.

First of all, the presented solution consists of a centralized policy which requires communication of some sort. In the best case, all quadrotors can share their state estimates with each other and run the centralized policy onboard and only execute the action pertaining to their quadrotor ID. An alternative is to use a MCS that broadcasts the global observations to all states or to a central unit that computes the next action of each quadrotor and broadcasts it. This would clearly add latency to the system, which is something that has not been modelled so the current policy does not know how to handle it.

Secondly, the dynamic model presented is not well suited for high speed and agressive flight. It does not include aerodynamic drag or a high fidelity quadrotor model that takes into account the low-level controller that runs onboard physical drones. In addition, no limits are present on the orientation of the drones or the payload allowing degenerate configurations which in a real system would be impossible to reach (as an example, think about the configuration where the quadrotor points straight down in the direction of the link its attached to).

Thirdly, the trained policy is not robust against changes in dynamical proper-

ties such as link lengths or masses and inertias, because these have not been randomized during training. For the racing task, it is also not robust to track changes.

Finally, all observations have been provided to the agent without the addition of noise and include states which would be hard to estimate without a MCS.

4.2 Future Work

The logical next step is to train a policy that can be deployed on a real-world system.

This would require addressing most of the limitations discussed above and more generally, improving the realism of the training environment:

- Adding support for non-rigid links. In aggressive flight regimes, real cables will be compressed and become slack in which case the kinematic relationship between the payload and quadrotor no longer holds.
- Adding an aerodynamic drag model to the payload and to the quadrotors.
- Incorporating the high-fidelity quadrotor model used in [16] into the slung payload dynamic model, as well as the low-level controller that runs onboard the physical quadrotors.
- Diversify and randomize the environment during training. This includes dynamical properties as well as gate positions.
- Add noise to the observations and try to reduce their dimension to as low as possible, and preferrably to a set of easily measurable states.
- Incorporate actuation delay to better replicate real world conditions. This can be done using a buffer.
- Add a safety reward so that there is a safety margin in how close the quadrotors get to each other and how close the payload gets to the edges of the gate.
- Ultimately, decentralizing the learning architecture using a multi-agent policy, would allow to deploy this system without requiring communication and each agent would operate using only onboard observations.

Bibliography

[1] Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine. Model-based meta-reinforcement learning for flight with suspended payloads. *IEEE Robot. Autom. Lett.*, abs/2004.11345, 2020.

[2] Morten Bisgaard, Jan Bendtsen, and Anders la Cour-Harbo. Modeling of generic slung load system. *J. Guidance, Control, and Dynamics*, 32:573–585, 03 2009.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[4] Cédric de Crousaz, Farbod Farshidian, Michael Neunert, and Jonas Buchli. Unified motion control for dynamic quadrotor maneuvers demonstrated on slung load and rotor failure tasks. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2223–2229, 2015.

[5] Davide Falanga, Philipp Foehn, Davide Scaramuzza, Naveen Kuppuswamy, and Russ Tedrake. Fast trajectory optimization for agile quadrotor maneuvers with a cable-suspended payload. 07 2017.

[6] Aleksandra Faust, Ivana Palunko, Patricio Cruz, Rafael Fierro, and Lydia Tapia. Learning swing-free trajectories for uavs with a suspended load. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4902–4909, 2013.

[7] Michael Gassner, Titus Cieslewski, and Davide Scaramuzza. Dynamic collaboration without communication: Vision-based cable-suspended load transport with two quadrotors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5196–5202, 2017.

[8] Jemin Hwangbo, Inkyu Sa, Roland Siegwart, and Marco Hutter. Control of a quadrotor with reinforcement learning. *IEEE Robot. Autom. Lett.*, PP:1–1, 06 2017.

[9] Taeyoung Lee, Koushil Sreenath, and Vijay Kumar. Geometric control of cooperating multiple quadrotor uavs with a suspended payload. In *IEEE Conf. Decision Control (CDC)*, pages 5510–5515, 2013.

[10] Guanrui Li, Rundong Ge, and Giuseppe Loianno. Cooperative transportation of cable suspended payloads with mavs using monocular vision and inertial sensing. *IEEE Robot. Autom. Lett.*, 6(3):5316–5323, 2021.

[11] Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar. *Cooperative Grasping and Transport Using Multiple Quadrotors*, pages 545–558. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[12] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[13] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative quadrocopter ball throwing and catching. In *J. Intell. Robot. Syst.*, pages 4972–4978, 2012.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[15] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. *Conf. on Robotics Learning (CoRL)*, abs/2009.00563, 2020.

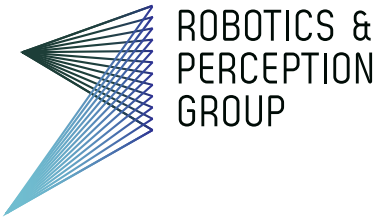
[16] Yunlong Song, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. Autonomous drone racing with deep reinforcement learning. *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, abs/2103.08624, 2021.

[17] Andrea Tagliabue, Mina Kamel, Roland Siegwart, and Juan Nieto. Robust collaborative object transportation using multiple mavs. *The International Journal of Robotics Research*, 38(9):1020–1044, 2019.

[18] Andrea Tagliabue, Mina Kamel, Sebastian Verling, Roland Siegwart, and Juan Nieto. Collaborative transportation using mavs via passive force control. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 5766–5773, 2017.

[19] Sarah Tang and Vijay Kumar. Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2216–2222, 2015.

[20] Jad Wehbeh, Shatil Rahman, and Inna Sharf. Distributed model predictive control for uavs collaborative payload transport. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 11666–11672, 2020.



Title of work:

Collaborative Aerial Transportation Using Deep Reinforcement Learning

Thesis type and date:

Semester Thesis, February 6, 2022

Supervision:

Yunlong Song
Ángel Romero
Dr. Sihao Sun
Prof. Dr. Davide Scaramuzza

Student:

Name:	David Oort Alonso
E-mail:	oodavid@student.ethz.ch
Legi-Nr.:	18-909-176

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, 6. 2. 2022: