Obstacle avoidance for quadrotors using reinforcement learning and obstacle-airflow interactions

van Dam, Geart; van Kampen, Erik-jan

DOI
10.2514/6.2020-2249

Publication date
2020

Document Version
Final published version

Published in
AIAA Scitech 2020 Forum

AIAA SciTech Forum
6-10 January 2020, Orlando, FL
AIAA Scitech 2020 Forum

10.2514/6.2020-2249

Obstacle avoidance for quadrotors using reinforcement learning and obstacle-airflow interactions

G.J. van Dam* and E. van Kampen†
Delft University of Technology, Delft, 2629 HS, The Netherlands

This research investigates and proposes a new method for obstacle detection and avoidance on quadrotors, that relies solely on measurements from the accelerometer and rotor controllers. The detection of obstacles is based on the principle that the airflow around a quadrotor changes when the quadrotor is flying near a surface. A well-known example of this is the ground effect, an increase in lift force close to a ground surface. Similarly, a change in dynamics occurs when a quadrotor is flying close to a wall or ceiling. The proposed method uses a reinforcement learning controller to detect obstacles based on these measurements, and takes action to lead the quadrotor back to safety. A proof-of-concept of this method is developed by training a reinforcement learning agent to avoid obstacles beneath a descending quadrotor. This is first done in a simulated environment, where the influence of hyperparameters, the amount of noise in the state signal, and the number of training episodes are investigated. The best performing agent from simulation is evaluated during a flight experiment with the Parrot Bebop 1 drone, where it is able to prevent the quadrotor from hitting the obstacle in 80% of the episodes. Furthermore, it is shown that the same level of performance can be achieved, by learning fully from scratch, in-flight, without prior knowledge or training, during 50 real flight training episodes. An approach for extending this method to the avoidance of walls, ceilings, and smaller obstacles is discussed. Additionally, it is shown that this method can easily be extended to other quadrotors.

Nomenclature

α	Learning rate	$F_{ext,x}, F_{ext,y}, F_{ext,z}$	Estimated external forces [N]
γ	Discount rate	F_i	Thrust produced by rotor i [N]
δ_t	Temporal-Difference error	g	Standard gravitational acceleration [m/s^2]
ϵ	Exploration rate	I_{ss_t}, I_{aa_t}	Identity-indicator functions
θ	Pitch angle [rad]	I_{xx}, I_{yy}, I_{zz}	Moments of inertia [kg·m^2]
λ	Decay-rate for eligibility traces	K_b	Body lift coefficient
μ	Mean	$k_{D,x}, k_{D,y}, k_{D,z}$	Drag coefficients [N·s^2/m]
σ	Standard deviation	k_i	Rotor gain [N·s^2/rad]
$\tau_{ext,x}, \tau_{ext,y}, \tau_{ext,z}$	Estimated external torques [Nm]	l_i	Rotor torque gain [kg·m^2/rad]
ϕ	Roll angle [rad]	m	Quadrotor mass [kg]
ω_i	Rotational speed of rotor i [rad/s]	N_{episodes}	Number of episodes
a	Action	p, q, r	Body rates around x,y and z axes [rad/s]
A_{signal}	Strength of the signal	Q	Action-value function
b	Distance between opposite rotors [m]	r	Reward
c	Distance to a surface above [m]	R_{rotor}	Rotor radius [m]
d	Distance between adjacent rotors [m]	s	State
$d_{i,x}$	Distance from rotor i to the x-axis [m]	T_c	Thrust in ceiling effect [N]
$d_{i,y}$	Distance from rotor i to the y-axis [m]	T_g	Thrust in ground effect [N]
e	Ground effect model bias [N/kg]	T_∞	Thrust in free flight [N]
E_t	Eligibility traces	u, v, w	Speed in body x,y and z axes [m/s]
$F_{\text{drag},x}, F_{\text{drag},y}, F_{\text{drag},z}$	Drag forces [N]	z	Distance to a surface below [m]

*MSc Student, Control and Simulation Division, Faculty of Aerospace Engineering, geartvd@gmail.com
†Assistant Professor, Control and Simulation Division, Faculty of Aerospace Engineering, e.vankampen@tudelft.nl

I. Introduction

In recent years, Unmanned Aerial Vehicles (UAVs), and quadrotors in specific, have risen in popularity. Their low-cost, small volume and Vertical Take-Off and Landing (VTOL) capability have driven their application outside of aerial photography to many new applications, which may range from industrial inspection tasks to disaster response or package delivery applications [1].

The next step in development is expected to be increasing the level of autonomy for these quadrotors [2]. This could bring down the cost significantly and allow for larger scale deployments of these solutions. A key remaining challenge for this, similar to that for other UAV-platforms, is in-flight autonomous object detection and avoidance. Currently, solutions to this problem are usually either vision-based or range-sensor-based.

Vision-based methods rely on an onboard camera and the use of computer vision algorithms to detect and avoid obstacles. While this area of investigation and initial applications certainly look promising [3], it is also dependent on good lighting conditions. Furthermore, the implementation on quadrotors can be limited by the amount of computational resources available on such a platform.

Range-sensor solutions rely on the addition of some range sensor, like a laser rangefinder or ultrasonic sensor, for proximity measurement in order to detect obstacles. These solutions can achieve good accuracy and have been successfully implemented on quadrotors [4]. However, because of the added cost, weight and power usage of such a sensor, they can decrease flight time and limit their usability to only larger quadrotors.

This research investigates and proposes an alternative method for obstacle avoidance and detection, one that does not require the addition of any sensors but relies solely on measurements of the accelerometer and rotor controllers, both present on almost all quadrotors. This new obstacle avoidance method could be used as a stand-alone method for small quadrotors, thereby removing the need for additional sensors, thus saving cost and weight while increasing flight time. Alternatively, when being used as an addition to other obstacle avoidance solutions it can increase safety and reliability.

The detection of obstacles is based on the principle that the airflow around a quadrotor changes when the quadrotor is flying near a surface. A well-known example of this is the so-called ground effect, an increase in lift force close to a ground surface, an effect also seen in helicopters [5] and fixed-wing aircraft [6]. Similarly, a change in airflow occurs when a quadrotor is flying close to a wall [7], or ceiling [8]. The proposed method uses a Reinforcement Learning (RL) controller to detect obstacles based on these effects and take action to lead the quadrotor back to safety.

In this article, the initial development of a proof-of-concept of this low-cost method is described. This proof-of-concept is limited to the detection of large obstacles underneath a quadrotor, using the so-called ground effect. The results from this are used to assess and discuss the extension of this method to the avoidance of obstacles above and on the same level as the quadrotor.

The contribution of this research to the state of the art is twofold. This research offers a distinct addition to both fields it is part of: obstacle avoidance and reinforcement learning. First of all, this research contributes to the field of obstacle avoidance by offering an innovative method of obstacle avoidance. Furthermore, this new method is one that does not require the addition of any sensors, solely relying on the Inertial Measurement Unit (IMU) and RPM measurements already available in most UAVs.

Secondly, in the field of reinforcement learning, this research adds to the state of the art by its unconventional placement of the reinforcement learning agent within the control loop. Instead of being in full control, the RL agent is run in combination with an inner loop flight control following a predetermined flight plan. The UAV will follow the flight plan during most of the flight, but this can be overridden at any time by an intervention action of the reinforcement learning agent.

This article is structured as follows. First, a brief background on the obstacle-airflow interactions between a quadrotor and obstacles, and RL is given in section II. Then, in section III, the method in which obstacle-airflow interactions are used to detect obstacles is discussed. The reinforcement learning setup is discussed in section IV. The setup and results of the experiments carried out in simulation are discussed in section V. Similarly, the setup and results of the flight experiments are discussed in section VI. In section VII the extension of this obstacle avoidance method to walls, ceilings, and other quadrotors is discussed. The conclusions of this article are presented in section VIII, after which recommendations for future research are made in section IX.

II. Background

Background information is presented on the obstacle-airflow interactions between a quadrotor and obstacles. Furthermore, the basics of reinforcement learning, as well as the application for flight control of quadrotors and the specific reinforcement learning algorithm used for this research, are discussed.

A. Obstacle-airflow interactions between a quadrotor and obstacles

The dynamics of a quadrotor are greatly dependent on its aerodynamics, most importantly the airflow around the thrust-producing rotors. This airflow can be influenced by surfaces in proximity to the quadrotors. In the proposed method, this change in dynamics will effectively function as the source of information used by the reinforcement learning agent to determine the presence of obstacles. Therefore, it is of key importance that their effect on the quadrotor is known and can be estimated in flight. Furthermore, training in simulation will require a model of these effects.

1. Obstacles underneath the quadrotor

The influence of a horizontal surface underneath a rotor has been well researched in literature [5, 9]. Most of this research has been focused on helicopters, but in general, it can be noted that for all rotorcraft operating closely above a ground surface, the produced thrust increases [10].

To estimate this thrust increase for quadrotors, often the classical model for ground effect in helicopters is used. This analytical model is derived by Cheeseman and Bennett [5] and shown in Eq. 1. It is based on potential flow theory, under both the assumption that the helicopter is hovering and the assumption that the rotor can be modeled as a point source. The method of images is then used to derive the ratio between the thrust produced by a helicopter in ground effect (T_g) and the thrust out of ground effect (T_∞), as a function of the radius of the rotor (R_{rotor}) and the distance to the surface underneath (z).

$$\frac{T_g}{T_\infty} = \frac{1}{1 - (\frac{R_{\text{rotor}}}{4z})^2} \tag{1}$$

Validation using experimental measurements has since shown that the ground effect for a quadrotor is larger than predicted by this equation [10, 11]. Furthermore, these experiments showed that the influence of the ground effect in quadcopters was apparent up to heights of 5 times the rotor radius.

A new model, specifically for quadrotors was recently proposed by Sanchez-Cuevas et al. [10]. It was shown to represent their experimental results more closely than Eq. 1. This model accounts for the presence of multiple rotors by representing them not as one but as four sources. It assumes a quadrotor hovering above a ground surface with four co-planar rotors.

Furthermore, this new model accounts for an effect called the *fountain effect*, an additional increase in lift previously seen in tandem helicopters [12] and quadcopter experiments [11]. It can best be explained by looking at the CFD simulation of a simplified quadrotor in ground effect, as shown in Fig. 1. As expected, the wakes from each rotor spread out to the sides as they near the ground, however in the center area where the two airflows interact with each other a vortex ring appears. Due to this aerodynamic effect, an upwards force is applied to the body of the quadrotor, leading to a greater ground effect. This effect is represented in this new model by an empirical body lift coefficient [10].

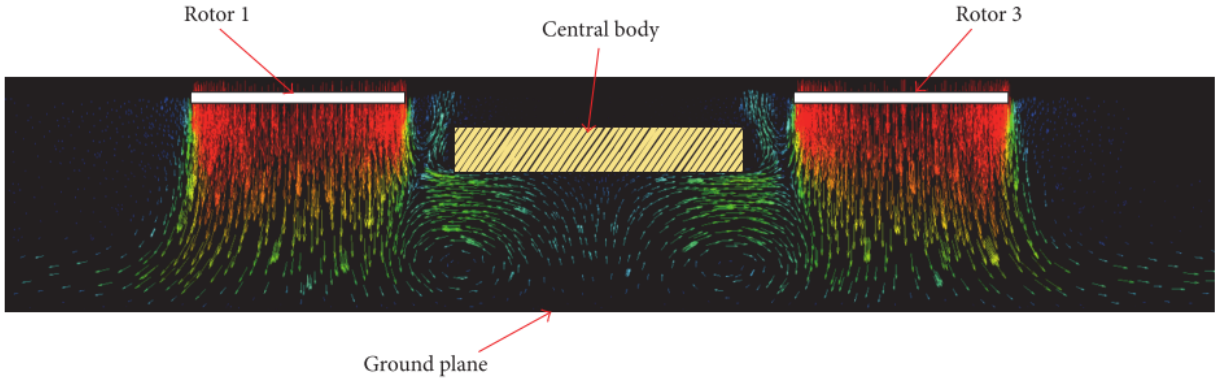


Fig. 1 CFD simulation of a simplified quadrotor model hovering close to a ground surface plane [10].

2. Obstacles above the quadrotor

Little research has been conducted on the influence of obstacles or surfaces above a quadrotor, the ceiling effect. The most relevant description and experiments are performed by Sanchez-Cuevas et al. [8]. In this research, the thrust produced by both a single rotor and a quadcopter, at varying distances to a ceiling surface, were measured and compared. These measurements were performed on a static test bench.

In Sanchez-Cuevas et al. [8], the increment in the thrust of a single rotor due to the ceiling effect is approximated by an analytical function similar to that of the ground effect shown in Eq. 1. The ratio between the thrust in (T_c) and out of the ceiling effect (T_∞) is given by equation 2, where c is the distance to the ceiling, and K_1 and K_2 are determined experimentally using ordinary least squares. A model for the increase in thrust for a complete quadcopter is not given, but it is shown that the relative increase is larger than that for a single rotor.

$$\frac{T_c}{T_\infty} = \frac{1}{1 - \frac{1}{K_1}\left(\frac{R_{\text{rotor}}}{c+K_2}\right)^2} \tag{2}$$

3. Obstacles on the same level as the quadrotor

The influence of large vertical surfaces like walls, on the same level as the quadrotor, has been described by Lee et al. [13], and Mckinnon [7]. Where the first article focuses only on reference tracking in the aerodynamic effects caused by such a wall, the second provides actual measurements of the effects caused by a wall. In that research, an Unscented Kalman Filter (UKF), based on a known model of the UAV, is used to estimate the external forces and torques near ground and wall surfaces whilst hovering.

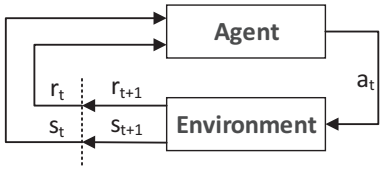
These measurements indicate a small external force away from the wall in the horizontal plane. Furthermore, a small external torque around the pitch and roll axes was noticed. Finally, a small downward external force was measured in the vertical plane. A clear increase in these three effects can be seen from a distance of 0.35 meter from the wall, about 3 times the rotor radius of the quadrotor that was used for that particular research.

B. Reinforcement learning

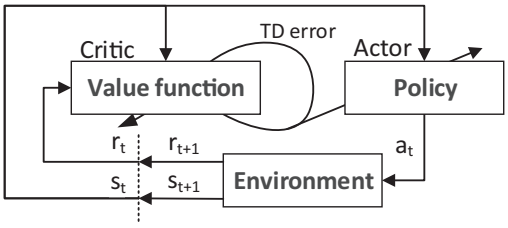
Reinforcement learning is a large and quickly developing field, containing a multitude of learning algorithms and architectures. In this research, reinforcement learning is used to find an optimal policy for the detection and avoidance of obstacles underneath a descending quadrotor. In the section below, the principles of RL and their application to flight control of multirotors is discussed. Furthermore, the method used for this research, Q-learning, is introduced.

1. Principles of reinforcement learning

Reinforcement learning is a computational approach to machine learning where an agent learns to maximize the cumulative rewards it receives when interacting with an environment. At each timestep t the agent chooses an action a_t, based on the current state s_t and its policy function, a mapping from state-space to action-space. The environment responds to this action by transitioning to state s_{t+1} and providing the agent with a numerical reward r_{t+1}, a process depicted in Fig. 2a.



(a) The agent-environment interaction in reinforcement learning.



(b) The actor-critic architecture.

Based on this reward and new state, a reinforcement learning algorithm specifies how the policy should be updated. The RL method does so with the goal of maximizing the sum of rewards received by the agent.

A special case of reinforcement learning methods are the actor-critic methods, in which the action selection and value estimation are split into two separate structures. The *actor* selects the actions and the *critic* estimates the (action-)value function and uses this to criticize the actor, as shown in Fig. 2b. This critique signal provided by the critic can be a scalar and is called the Temporal-Difference (TD) error (δ_t). It is used by the actor to adjust its policy.

2. Applying reinforcement learning to flight control of quadrotors

There are multiple examples of reinforcement learning techniques being applied to multirotors, and quadrotors in specific. Reinforcement learning agents have been used both for full control of a quadrotor [14] and for adjustment of a conventional controller [15]. Key challenges in applying reinforcement learning to flight control remain the challenge of safety[16], the challenge of robustness, the challenge of online efficiency[17] and the challenge of sample efficiency.

3. Q-learning

One often used reinforcement learning method is Q-learning. Q-learning is a model-free, off-policy, Temporal-Difference method, which has three key implications. First of all, it does not require a model of the environment but instead learns solely from interacting with the environment. Secondly, the policy it uses to select actions, its behavior policy, is not necessarily equal to the estimation policy, the policy that is being improved to approach the optimal policy. Finally, methods of the Temporal-Difference learning class makes use of bootstrapping, using previously estimated values for the action-value function $Q_t(s,a)$ for its new estimate $Q_{t+1}(s,a)$, with the goal of approaching the optimal action-value function $Q^*(s,a)$.

Q-learning is often combined with eligibility traces, another key reinforcement learning mechanism, to obtain a more general method that may learn more efficiently. One implementation of this is Watkins's Q(λ) method [18]. The update equations for this method, using replacing traces, are given in equations 3 and 4.

The choice for Watkins's Q(λ) is based on a preliminary investigation where Monte Carlo, SARSA and Q-Learning methods were tested on a simplified version of the problem at hand. Results of this investigation suggest that Watkin's Q(λ) is best suited for problems with this particular setup.

$$E_t(s,a) = \begin{cases} \min\left(\gamma\lambda E_{t-1}(s,a) + I_{ss_t} \cdot I_{aa_t}, 1\right) & \text{if } Q_{t-1}(s_t,a_t) = \max_a Q_{t-1}(s_t,a) \\ I_{ss_t} \cdot I_{aa_t} & \text{otherwise.} \end{cases} \tag{3}$$

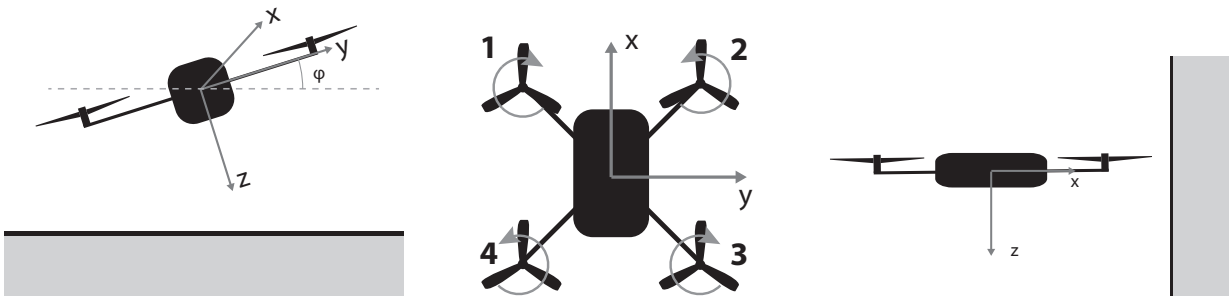
$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t E_t(s,a) \quad \text{for all s,a, with} \quad \delta_t = r_{t+1} + \gamma \max_a Q(s_{t+1},a_{t+1}) - Q(s_t,a_t) \tag{4}$$

III. Using obstacle-airflow interactions for obstacle detection

The result of the obstacle-airflow interactions on the quadrotor can be estimated by using a simple quadrotor model to estimate external forces and torques in and around all three axes. A procedure to do so is explained in detail for the external force in the vertical direction. This estimate is used to create a model for the ground effect, using measurement data gathered with the same drone that is used for the final flight experiments.

A. Frame of reference & equations of motion

For the purpose of this article, all derivations are performed within the quadrotor body frame and with the assumption of a rigid body. This right-handed coordinate frame is fixed to the quadrotor at the center of gravity, with the rotor axes pointing in the vertical z direction, the direction of thrust being negative, and the arms pointing in the x and y directions. A sketch of a quadrotor, showing this reference frame can be seen in Fig. 3.



(a) Quadrotor near a ground surface. **(b) Top view of a quadrotor.** **(c) Side view of a quadrotor near a wall.**

Fig. 3 Quadrotor body frame of reference.

The translational set of equations of motion for the quadrotor is given in Eq. 5. The gravitational acceleration is denoted as g, the mass of the quadrotor is m, the body attitude angles in pitch, roll, and yaw are given by θ, ϕ, ψ, the drag forces as F_{drag} and the thrust produced by each rotor i as F_i. Furthermore, the body speeds in forward, sideways and vertical direction are given by u, v, w, so the accelerations in the body frame are $\dot{u}, \dot{v}, \dot{w}$. Finally, p, q and r refer to the angular rates for pitch, roll, and yaw.

$$\begin{bmatrix} -mg\sin\theta + F_{\text{drag},x} \\ mg\sin\phi\cos\theta + F_{\text{drag},y} \\ mg\cos\phi\cos\theta - \sum_i F_i + F_{\text{drag},z} \end{bmatrix} = m \begin{bmatrix} \dot{u} + qw - rv \\ \dot{v} + ru - pw \\ \dot{w} + pv - qu \end{bmatrix} \tag{5}$$

Assuming symmetry about the x and y axes of the body frame, as well as negligible rate damping, the rotational set of equations of motion is given by Eq. 6. Here $\dot{p}, \dot{q}$ and $\dot{r}$, refer to the angular acceleration rates, $d_{i,x}$ to the shortest distance from rotor i to the x-axis and $d_{i,y}$ to the shortest distance to the y-axis. The moments of inertia are given as I_{xx}, I_{yy} and I_{zz}. Finally, the torque produced by each rotor is assumed to be a function of rotor speed $T(\omega_i)$.

$$\begin{bmatrix} F_1 d_{1,x} - F_2 d_{2,x} - F_3 d_{3,x} + F_4 d_{4,x} \\ F_1 d_{1,y} + F_2 d_{2,y} - F_3 d_{3,y} - F_4 d_{4,y} \\ \sum_i T(\omega_i) \end{bmatrix} = \begin{bmatrix} I_{xx}\dot{p} \\ I_{yy}\dot{q} \\ I_{zz}\dot{r} \end{bmatrix} + \begin{bmatrix} -I_{yy}qr + I_{zz}qr \\ I_{xx}pr - I_{zz}pr \\ I_{zz}pq + I_{yy}pq \end{bmatrix} \tag{6}$$

B. Estimating external forces & torques

While the equations of motion given in Eq. 5 and 6 provide a model of the quadrotor in perfect free flight conditions, without any disturbances, wind or obstacle-airflow interactions, this is not always the situation in reality. To the contrary, these obstacle-airflow interactions are exactly the subject of interest. Therefore an external force, representing the difference between reality and the simplified quadrotor model, is added to each of the translational equations of motion; $F_{ext,x}, F_{ext,y}, F_{ext,z}$. Similarly an external torque is added to each of the rotational equations of motion: $\tau_{ext,x}, \tau_{ext,y}, \tau_{ext,z}$.

Since the external forces and torques represent all unmodelled dynamics, like the influence of obstacle-airflow interactions, these forces and torques can also be used to identify these dynamics. This can be accomplished by rewriting the equations of motion and solving for the external force or torque, using the known or approximated states. An example of this will be given below for the external force in the z-direction, as this force can be caused by obstacle-airflow interactions with obstacles underneath the quadrotor. The derivation starts with the translational equation of motion in the z-direction, as shown in Eq. 7.

$$mg\cos(\phi)\cos(\theta) - \sum_i F_i + F_{\text{drag},z} + F_{ext,z} = m(\dot{w} + pv - qu) \tag{7}$$

The IMU sensor of the quadrotor consists of a 3-axis accelerometer, providing accelerations, and 3-axis gyroscope, providing angular rates. The body speeds can either be derived from an external positioning system (like GPS, or OptiTrack), integration of body accelerations, or a combination of both using sensor fusion.

The onboard accelerometer measures in the body frame, however, it does not just measure $\dot{w}$. Instead it is influenced by the gravity vector, as it measures $\hat{\dot{w}}$, with $\hat{\dot{w}} = \dot{w} - g\cos(\phi)\cos(\theta)$. So if one substitutes this in Eq. 7 and solves for $F_{ext,z}$, the following estimate for the external force in the vertical direction is derived:

$$\frac{F_{ext,z}}{m} = \hat{\dot{w}} + pv - qu + \frac{1}{m}\sum_i F_i - \frac{1}{m}F_{\text{drag},z} \tag{8}$$

Now if the following model for the thrust produced by each rotor is assumed: $F_i = k_i\omega_i^2$ [19], this can be rewritten to:

$$\frac{F_{ext,z}}{m} = \underbrace{\hat{\dot{w}} + pv - qu}_{\text{from IMU}} + \sum_i \frac{k_i}{m} \underbrace{\omega_i^2}_{\text{from motors}} - \frac{1}{m}F_{drag,z} \tag{9}$$

To estimate $\frac{k_i}{m}$ an initialization procedure is conducted when the quadrotor is hovering in free flight, without closeby obstacles or disturbances. The external force $F_{ext,z}$ and drag $F_{drag,z}$ can then assumed to be zero. The equation above can then be solved for $\frac{k_i}{m}$ if either the assumption is made that $\frac{k_i}{m}$ is equal for all four rotors, or that the force produced by each of the four rotors during this initialization procedure is equal.

$$\frac{k_i}{m} = \frac{-\hat{w} - pv + qu}{\sum_i \omega_i^2} \quad \text{assuming equal } k_i \qquad (10) \qquad\qquad \frac{k_i}{m} = \frac{-\hat{w} - pv + qu}{4\omega_i^2} \quad \text{assuming equal thrust} \qquad (11)$$

The drag force can be estimated by performing another experiment in free flight, without closeby obstacles or disturbances. However, instead of hovering the quadrotor should now move up and down. Using the previously found values for $\frac{k_i}{m}$, Eq. 9 can be solved for $F_{\text{drag},z}/m$ for each measurement point of the experiment. It can be expected that the drag is of the form $|F_{\text{drag},z}/m| = k_{D,z}w^2$, where the direction of the force is opposite to the speed [20]. A function of this form can thus be fitted to the experiment data, and solved for $k_{D,z}$ using for example linear least squares, providing a model for $F_{\text{drag},z/m}$.

$$\frac{F_{ext,z}}{m} = \underbrace{\hat{\dot{w}} + pv - qu}_{\text{from IMU}} + \sum_i \frac{k_i}{m} \underbrace{\omega_i^2}_{\text{from motors}} - F_{drag,z}/m \quad \text{, with } F_{\text{drag},z}/m = \begin{cases} -k_{D,z}w^2 & \text{if } w > 0 \\ k_{D,z}w^2 & \text{otherwise} \end{cases} \qquad (12)$$

Similarly, the following equations can be derived to estimate the external forces in x and y direction.

$$\frac{F_{ext,x}}{m} = \underbrace{\hat{\dot{u}} + qw - rv}_{\text{from IMU}} - F_{\text{drag},x}/m \quad \text{, with } F_{\text{drag},x}/m = \begin{cases} -k_{D,x}u^2 & \text{if } u > 0 \\ k_{D,x}u^2 & \text{otherwise} \end{cases} \qquad (13)$$

$$\frac{F_{ext,y}}{m} = \underbrace{\hat{\dot{v}} + ru - pw}_{\text{from IMU}} - F_{\text{drag},y}/m \quad \text{, with } F_{\text{drag},y}/m = \begin{cases} -k_{D,y}v^2 & \text{if } v > 0 \\ k_{D,y}v^2 & \text{otherwise} \end{cases} \qquad (14)$$

Following a similar process, equations can be derived to estimate the external torque around the x, y, z axes. The resulting estimators are shown in Eq. 15, 16 and 17. To get to these estimators, it is assumed that the angular rates are relatively small and qr, pr and pq can be considered negligible. Furthermore, these estimators make use of $\frac{k_i}{m}$, this is the same parameter that is used for estimating the external force in the z-direction.

The moments of inertia around the x and y-axis can be estimated by performing an experiment in free flight, where the quadrotor performs sequentially a pitch and roll changing maneuver. If the external torques are assumed to be zero, Eq. 15 and 16 can be solved for I_{xx} and I_{yy}.

$$\frac{\tau_{ext,x}}{I_{xx}} = \dot{p} - \frac{1}{I_{xx}}\left[\frac{k_1}{m}\omega_1^2 d_{1,x} - \frac{k_2}{m}\omega_2^2 d_{2,x} - \frac{k_3}{m}\omega_3^2 d_{3,x} + \frac{k_4}{m}\omega_4^2 d_{4,x}\right] \qquad (15)$$

$$\frac{\tau_{ext,y}}{I_{yy}} = \dot{q} - \frac{1}{I_{yy}}\left[\frac{k_1}{m}\omega_1^2 d_{1,y} + \frac{k_2}{m}\omega_2^2 d_{2,y} - \frac{k_3}{m}\omega_3^2 d_{3,y} - \frac{k_4}{m}\omega_4^2 d_{4,y}\right] \qquad (16)$$

To estimate the external force around the z-axis, as shown in Eq. 17 it is assumed that the torque produced by one rotor can be approximated by $T(\omega_i) \approx l_i\omega_i^2$ [19]. The parameters $\frac{l_i}{I_{zz}}$ can be estimated by letting the quadrotor perform an initialization procedure in free flight, where $\tau_{ext,z}$ is assumed to be zero. If one then either assumes equal l_i, or equal torque, Eq. 17 can be rewritten to estimate the torque parameters.

$$\frac{\tau_{ext,z}}{I_{zz}} = \dot{r} + \frac{l_1}{I_{zz}}\omega_1^2 - \frac{l_2}{I_{zz}}\omega_2^2 + \frac{l_3}{I_{zz}}\omega_3^2 - \frac{l_4}{I_{zz}}\omega_4^2 \qquad (17)$$

Finally, with respect to the accuracy of the three external torque estimates, it is important to note that angular acceleration rates are often not provided directly by an IMU. Instead, they might need to be acquired by taking the derivative of the angular rates, which introduces noticeable noise into the estimation.

C. Challenges

While the method described above provides estimates of the external forces, they are not perfect estimates. There are three main challenges for achieving accurate estimates: inaccuracies in the estimated model, noisy measurements and delays in measurements.

Inaccuracies in the estimated model can be a result of incorrect assumptions, disturbances during initialization or changing system dynamics (e.g. a difference in mass due to a change in the payload). To cope with this, especially the latter, the estimation of the rotor gains is performed on board the quadrotor at the start of every flight. This is fully automated and takes only 2 seconds. In addition to being influenced by the gravity factor, the measured body accelerations $\hat{u}, \hat{v}, \hat{w}$ are also susceptible to sensor noise and vibrations in the body frame. One common source of such vibrations are the rotors. In this research, a 4th order Butterworth filter with a cutoff frequency of 3Hz is applied to all accelerometer and gyroscope measurements in an effort to remove sensor noise and rotor-induced vibrations. This choice is based on a frequency spectrum analysis and a trade-off between remaining noise and introduced delay.

Finally, delays in the measured states can lead to inaccuracies in the estimations. One of such delays is caused by the low-pass filter discussed above, which is expected to introduce a 140ms delay [21]. Therefore, the same filter is also applied to the speed, attitude, and rotor measurements in order to keep the signals synchronized. At a descend speed of 0.3m/s this thus corresponds to a distance of 4.2cm.

D. Model for the external force in the vertical direction when flying close to the ground

In order to train reinforcement learning agents to avoid obstacles underneath the quadrotor, a training environment is created. To make this environment as realistic as possible, a model of the net vertical force caused by the ground effect is created. This model is created based on measurements gathered during three separate measurement flights with the Parrot Bebop 1 drone, the same drone that is used for the flight experiments. In these measurement flights, the quadrotor descends from 1m height, with constant speed, to 1cm above a large surface of artificial grass. At each timestep, the external force in z-direction is estimated.

The model for the external force is based on the recently proposed formula for thrust increase in ground effect provided by Sanchez-Cuevas et al. [10]. In this formula, the increase in thrust near a surface underneath the quadrotor is not only dependent on the rotor radius R_{rotor} and distance to the ground z, but also on the distance between adjacent rotors d, the distance between opposite rotors b and an empirical body lift coefficient K_b. For this research, the model is extended with a bias e and rewritten to approximate $F_{ext,z}/m$, as can be seen in Eq. 18.

$$\frac{F_{ext,z}}{m} = -g\Big[\frac{1}{1-(\frac{R_{\text{rotor}}}{4z})^2 - R_{\text{rotor}}^2(\frac{z}{\sqrt{(d^2+4z^2)^3}}) - (\frac{R_{\text{rotor}}^2}{2})(\frac{z}{\sqrt{(2d^2+4z^2)^3}}) - 2R_{\text{rotor}}^2(\frac{z}{\sqrt{(b^2+4z^2)^3}})K_b} - 1\Big] + e \tag{18}$$

Non-linear least squares, using the Trust Region Reflective (TRF) algorithm and boundaries based on the physical properties of the quadrotor, is used to fit this function to the measurement data. This results in the following parameters: $R = 0.05$m, $d = 0.181$m, $b = 0.253$m, $K_b = 0.474$, $e = -0.150$. From Fig. 4 it can be seen that this function better replicates the measurement data than the classical formula from Cheeseman and Bennett [5].

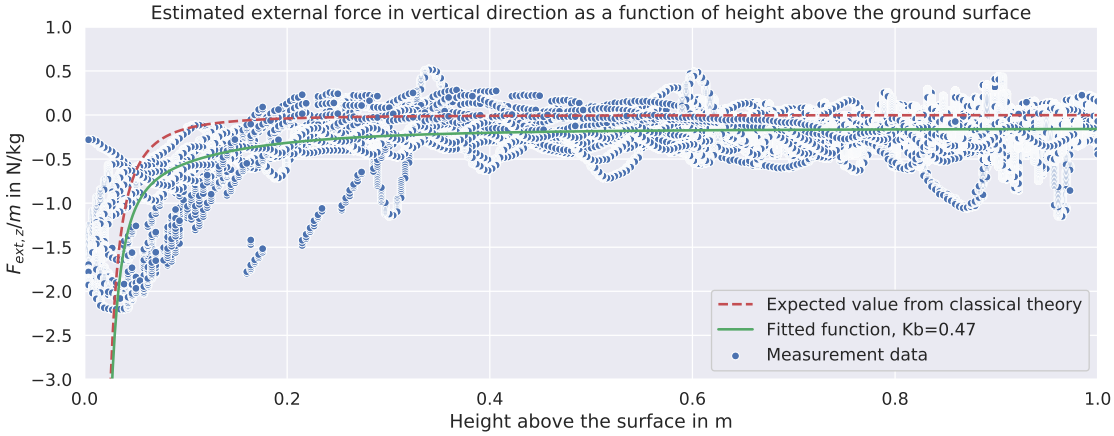


Fig. 4 Estimated external force in the vertical direction as a function of height above the ground surface.

IV. Reinforcement learning setup

In this section, the position of the reinforcement learning agent in the general control scheme are discussed. Furthermore, the states, actions, and rewards provided to the agent are introduced. Additionally, the initialization and termination conditions are highlighted. Finally, the exploration strategy, reinforcement learning algorithm, and hyperparameter are discussed.

A. General control scheme

The general control scheme of both the simulation and real flight experiment is depicted in Fig. 5. As can be seen from this figure, the RL agent is set up as an actor-critic method, where the actor is only updated after an episode has ended. The reason for this is practical, on the Parrot Bebop 1 the onboard computational capacity is limited. Therefore it is expected that the update rate of the RL agent will not be able to keep up with the frequency of the control loop, which runs at 512Hz. This high frequency is chosen to limit reaction time and give the agent the best chance of preventing collision with the obstacle underneath.

The state estimator that is indicated in the control scheme is the function that estimates $F_{ext,z}/m$. In the simulation experiments, this is estimated using the ground model discussed in section III.D. In the actual flight experiments, it is estimated based on the actual measured signals using Eq. 12.

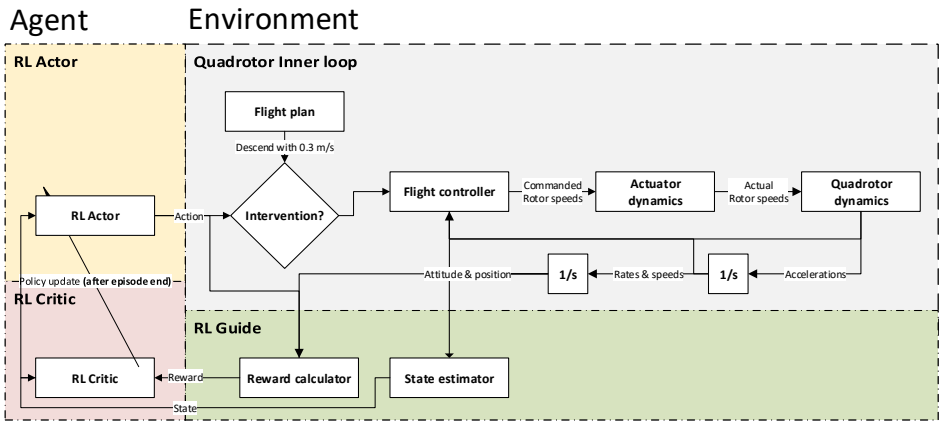


Fig. 5 High-level overview of the control scheme used in the experiments.

B. States & actions

Two states are available to the reinforcement learning agent; the current estimate of $F_{ext,z}/m$ and the action chosen in the previous timestep a_{t-1}. The estimate of external force $F_{ext,z}/m$ is discretized into 9 equally spaced bins, ranging from -0.2N/kg up to -1.00N/kg. The action in the previous timestep a_{t-1} is provided to the RL agent because the reward given by the environment depends as much on the previously chosen action as on the external force.

Based on these states, the agent then decides which action to perform. There are three actions available to the agent; No action ($a_{\text{no-action}}$), save (a_{save}) and hover (a_{hover}). When $a_{\text{no-action}}$ is chosen, the quadrotor continues on its original flight plan for one timestep, thus continuing the 0.3m/s descend. When choosing a_{save}, a short 1-second full-thrust command is sent to the quadrotor inner control loop. Finally, when choosing a_{hover}, the quadrotor inner control loop is given the command to hover for 0.5 seconds.

As mentioned above, two of the three potential actions take more than 1 timestep to execute. Hovering will take 256 timesteps and a save 512 timesteps. During this time the reinforcement learning agent is considered frozen, no new states are provided to the agent, no actions are picked by the agent and neither the action-value function, policy nor eligibility traces of the agent are updated. Any rewards that the agent might receive during this period are summed and provided to the agent for processing at the final timestep of the multistep action, together with the new state.

C. Rewards & termination

The goal of the reinforcement learning agent is to prevent the quadrotor from hitting any obstacles underneath. As such, the largest negative reward is given when the quadrotor comes to close to the obstacle below, as determined by the termination height $z_{\text{termination}}$, with height referring to the height above the obstacle. While this is referred to as a crash,

it must be noted that it is not an actual crash of the quadrotor, instead, the safety controller intervenes, preventing an actual collision, and leading the quadrotor back to safety. In all experiments, a termination height of 0.05 meter is used.

$$R(s,a) = \begin{cases} -2000 + R_a & \text{if } z \leq z_{\text{termination}} \\ R_a & \text{otherwise} \end{cases} \tag{19}$$

Furthermore, a negative action-based reward R_a is given when the agent intervenes, especially when the intervention is false. In this case, false is defined as outside of the area from which the ground effect can be measured, which was estimated to be 0.25 meter, corresponding to 4 times the rotor radius.

$$R_a(s|a = a_{\text{save}}) = \begin{cases} -500 & \text{if } z > 0.25 \\ -50 * \frac{0.25 - z}{0.25 - z_{\text{termination}}} & \text{if } z \leq 0.25 \end{cases}, \quad R_a(s|a = a_{\text{hover}}) = \begin{cases} -100 & \text{if } z > 0.25 \\ -25 & \text{if } z \leq 0.25 \end{cases} \tag{20}$$

There are three ways in which an episode can end. First of all, when the agent intervenes by performing a save, this is an episode-ending action. Secondly when the quadrotor comes to close to the surface underneath, as defined by Eq. 19. In either case, the final reward is processed by the agent and the episode ends. If neither of the two happens the episode automatically ends after 10 seconds.

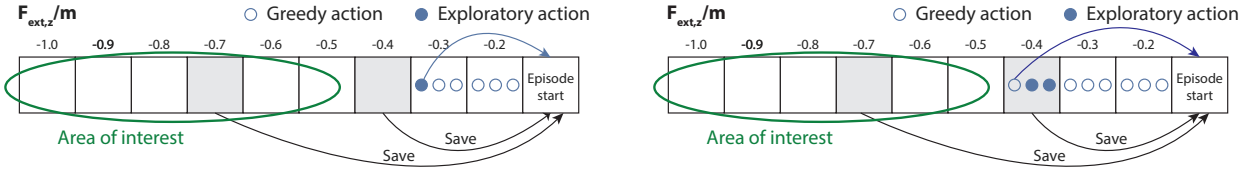
As can be concluded from these termination conditions, all episodes are finite, therefore it is not an absolute requirement to have a reward discount factor $\gamma < 1$. In the context of obstacle avoidance, it can even be argued that the negative impact of a collision in the future should not be discounted at all. Therefore, in this reinforcement learning problem, it is chosen to have $\gamma = 1.0$.

D. Exploration & initialization

Initial results with ϵ-greedy exploration strategies [22] showed that exploration was quite challenging for the RL agents, the number of state visits was highly skewed towards states with $F_{ext,z}/m$ close to zero. These states typically correspond to heights out of the ground effect. The states with more negative $F_{ext,z}/m$, that generally correspond to heights within the ground effect, were rarely visited by the agents. Further studies showed that the limited exploration of states close to the ground is inherent to the problem, for which there are two reasons.

First, the save action is episode-ending. As such, any exploration strategy which relies on random actions being picked runs the risk of ending the episode, and thus stopping further exploration, every time it does so. This is problematic when it is prone to happen early on in the episode, as it prevents the quadrotor from coming close to the ground. Therefore, the RL agent will rarely experience those states. An illustration of this can be seen in Fig. 6a.

Secondly, there are a lot of timesteps compared to the number of potential discrete states. This means that during the descent the agent will often be in the same state for many steps. This is a challenge for exploration because it requires many subsequent exploratory actions to be taken in order to reach another state. An example of this is depicted in Fig. 6b, where it must be noted that in reality it usually takes way more than three steps to progress from one state to the next. This reason relates back to the unconventional placement of the RL agent in the control scheme, as running the agent at a high frequency allows for quick interventions, but leads to this large number of timesteps per discrete state.



(a) Early ending of an episode due to exploration.

(b) Multiple consecutive exploratory actions required to go past a state where the greedy action is *save*.

Fig. 6 Two typical exploration challenges for this particular reinforcement learning problem. Note: only one dimension of the policy is shown.

To handle these issues, a combination of three different exploration strategies is implemented; exploring starts, episode-long exploratory actions, and epsilon-greedy exploration. An example of this strategy is shown in Fig. 7.

First of all, exploring starts are implemented [22]. From the perspective of the simulated or actual quadrotor, each episode always starts when hovering at 1-meter height, where it is given the command to descend with 0.3m/s. The RL

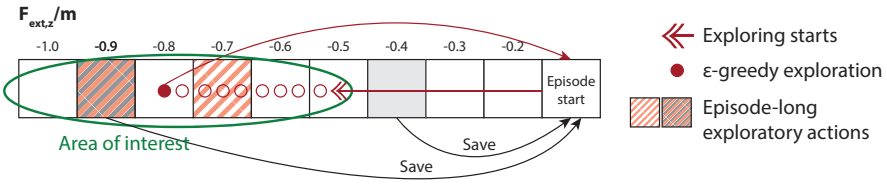


Fig. 7 Implemented threefold exploration strategy.

agent is however not initialized until height z_{expl} is reached. This exploratory starting height z_{expl} is randomly taken from the uniform distribution [0.45, 1.0] before the start of each episode. This part of the exploration strategy encourages the exploration of the states that typically occur closer to the ground.

Secondly, episode-long exploratory actions are randomly generated before the start of each episode; for each state s in the state-space there is a chance $\epsilon_{\text{episode}}$ that random action a will be taken on every visit of that state, instead of the greedy action. By doing so, this approach can mitigate the challenge of requiring multiple consecutive exploratory actions to get to another state.

Finally, the methods above are combined with ϵ-greedy exploration at each timestep. The chance of picking a random action, instead of the greedy or episode-long exploratory action for that state, is then given by ϵ_{step}.

E. Hyperparameters

There are four key hyperparameters that determine the behavior of the RL agent; the learning rate (α), exploration rate at each step (ϵ_{step}), episode-long exploration ($\epsilon_{\text{episode}}$) and decay of eligibility traces (λ). The results from the previously mentioned preliminary investigation suggest that, when using Q-learning, a high, non-decreasing, exploration rate at each step ϵ_{step}, a λ between 0.1 and 0.5, and a learning rate α that decreases to a quarter of its initial value during the first half of the episodes, produce the best-performing agents.

Based on these preliminary results, 216 different combinations of these hyperparameters were selected for further investigation. Two grid searches were carried out in the simulation environment described below. The results from the first grid search are used to determine the best set of hyperparameters for training an agent from scratch. The results from the second grid search are used to determine the set of hyperparameters that are best when a previously trained agent is placed in a new, slightly different, environment. In each grid search, 2160 agents were trained during 500 training episodes, 10 for each of the 216 different hyperparameter sets.

The sets of hyperparameters that performed the best in these two grid searches are shown in table 1. For both sets, it is found that the best performance is achieved when $\epsilon_{\text{episode}}$ linearly decreases to zero during the first half of the episodes. The learning rate decreases as well, with the learning rate in episode i given by Eq. 21.

	λ	ϵ_{step}	α_0	$\epsilon_{\text{episode}}$
Initial training	0.1	0.01	0.5	$0.5 \rightarrow 0.0$
Continued training	0.1	0.01	0.1	$0.01 \rightarrow 0.0$

Table 1 Best hyperparameter sets found for initial and continued training.

$$\text{with } \alpha_i = \alpha_0 \frac{N_{\text{episodes}}}{N_{\text{episodes}} + i} \tag{21}$$

It is important to note here that the found sets of hyperparameters are the local optimum, the best available set from the 216 analyzed sets of hyperparameters. While these 216 options were selected carefully, based on both literature [23] and a preliminary investigation, there might exist a better set of hyperparameters globally.

V. Simulation

For the first phase of the experiments, a simulation environment is created that represents the in-flight environment as much as possible. To this end, the quadrotor inner vertical loop is replicated, the quadrotor dynamics are implemented and the ground effect is simulated. This simulated environment is then used to train and evaluate multiple agents, investigate the influence of noise and the number of required training episodes.

A. Experiment setup

First, the setup of the simulation environment and the experiments conducted within this environment are presented.

1. Creating the simulation environment

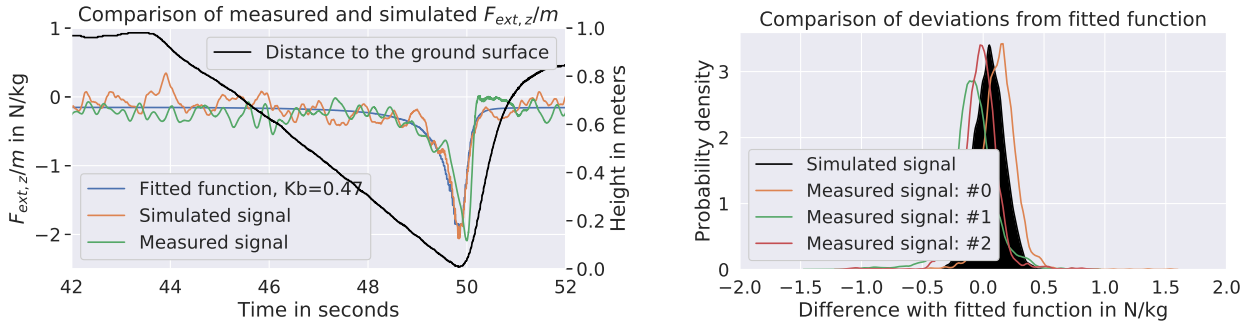
The quadrotor dynamics are implemented as described by the equations of motion given in Eq. 5 and 6. A key addition to this are the actuator dynamics. These provide the relationship between the commanded rotor speed $\omega_{cmd,i}$, from the inner control loop to the actual rotor speed ω_i. These are simulated using a 2nd order lower-pass Butterworth filter with a cut-off frequency of 15Hz. This actuator model is based on in-flight measurements from the system identification experiment described below.

The quadrotor inner vertical control loop is replicated in simulation to better reproduce the behavior of the quadrotor when it receives a command. This is especially relevant at the start of the episode when the quadrotor is hovering and receives the 0.3m/s descend command, and during interventions, when it receives a hover (a_{hover}) or save (a_{save}) command. The replication of the inner loop is based on the control scheme that is provided for the open-source flight control software [*]. The gains are determined by performing small system identification experiments with the Parrot Bebop 1 drone, where a step command is given on the vertical reference speed. The same step command is given in the simulation environment, and the gains are tuned based on the comparison.

The ground effect is simulated using the fitted function, as given in Eq. 18, and shown in Fig. 4. However, as can be seen from this figure, $F_{ext,z}/m$ is not a perfect estimator of height above the ground. As discussed in section III.C, the estimate is also influenced by inaccuracies in the estimated model, and noise or delays in measurements. As a result, the difference between the estimated external force at one timestep and the fitted model cannot be considered white noise. Instead, the $F_{ext,z}/m$ signal is quite smooth, as can be seen from Fig. 8.

In the simulation, an effort is made to replicate the stochastic deviations from the fitted function, but to keep the smoothness of the estimated signal. To do so, random normally-distributed noise is generated and filtered using a 2nd order Butterworth low-pass filter with a cutoff frequency of 1.8Hz. The mean ($\mu_{noise} = 0.055$) and standard deviation ($\sigma_{noise} = 1.392$) of this normal distribution are chosen such that the filtered noise distribution replicates the distribution seen in measurements. This can be confirmed by looking at Fig. 8, as the signals are similar, both in terms of smoothness and deviation from the fitted function.

A small difference can, however, be seen in the timing of the signal; the $F_{ext,z}/m$ signal measured in flight is delayed around 140 milliseconds. This difference can be explained by considering the 140ms delay introduced by the 4th order low-pass filter that is used to filter the in-flight measurements, as discussed in section III.C. While a low-pass filter is also used in the simulation, only the noise is filtered, not the underlying signal, as such no delay is introduced into the actual information-carrying signal, explaining the 140ms difference.



(a) $F_{ext,z}/m$ during one episode. **(b) Deviations from the fitted function.**

Fig. 8 Comparison of measured and simulated noise in the $F_{ext,z}/m$ state signal.

2. Training agents in simulation

Using the created simulation environment, and the set of hyperparameters for initial training discussed in section IV.E, 100 agents are trained in the simulation environment. Each agent is trained for 500 episodes, using the exploration strategy described in section IV.D. Their performance is evaluated during 100 fully greedy evaluation episodes. In these greedy episodes, there is no exploration and no learning. The comparison of agents is based on the average total reward

[*] `http://wiki.paparazziuav.org/wiki/Control_Loops#Vertical_loop`, as accessed on 23/01/2019

during these evaluation episodes, a measure of agent performance, and the number of episodes since the agent's policy last changed, a measure of the agent's stability. Based on this comparison the top performing agent is then selected as the *top agent*.

3. Investigating the influence of noise on agent performance

As discussed in sections III.C and V.A.1, the external force estimators are not perfect estimators of the distance to an obstacle underneath. Instead, they are stochastic signals, influenced by both inaccuracies in the estimated external force, like sensor noise, and external forces not resulting from obstacle-airflow interactions, like gusts. Furthermore, the amount of stochasticity in comparison to the underlying obstacle-airflow interaction is likely to depend on the quadrotor, obstacle characteristics and location of the obstacle with respect to the quadrotor. As such, getting an understanding of the effect of this stochasticity on the performance achievable by the RL agent is of key importance. Especially when considering the application of this object avoidance technique to other quadrotors or other types of obstacles.

Therefore, an experiment is carried out in which the noise on the $F_{ext,z}/m$ state is varied. The noise is varied from no noise, so a completely deterministic signal ($\sigma_{noise} = 0$), to 10 times as much noise as measured on the Parrot Bebop 1 drone for obstacles underneath ($\sigma_{noise} = 13.92$). The ratio between the strength of the signal (A_{signal}) and the noise (σ_{noise}) is captured by the Signal to Noise Ratio (SNR), as given in 22 [24]. Since this equation uses the strength of the signal, it also depends on the distance to the obstacle. In this case, the strength of the signal, $F_{ext,z}/m$ as caused by the ground effect, at 0.15m distance from the surface is used. This is approximately three times the rotor radius and exactly halfway between the estimated start of the ground effect area (0.25m), and the termination distance (0.05m).

$$SNR = \frac{A_{signal}^2}{\sigma_{noise}^2} \tag{22}$$

For each SNR level, 100 RL agents are trained for 500 episodes, using the hyperparameter set for initial training. Each agent is then evaluated during 100 fully greedy episodes. Both training and evaluation are carried out in the simulation environment, with the noise on the $F_{ext,z}/m$ state as defined by the respective SNR levels.

4. Investigating the number of required training episodes

In order to determine the influence of the number of training episodes on the performance of the agents, another experiment is carried out in simulation. Using the top initial hyperparameter set determined before, 100 agents are trained for each of the following number of episodes: $N_{episodes} = \{25, 50, 100, 500, 1000\}$. Evaluation is once again performed during 100 fully greedy episodes.

The results from this experiment are expected to help determine the feasibility of training an agent fully online during real flight, as the number of episodes that can be performed in real flight in a practical manner is limited.

B. Results

In the section below the results of the experiments carried out in the simulation environment are discussed.

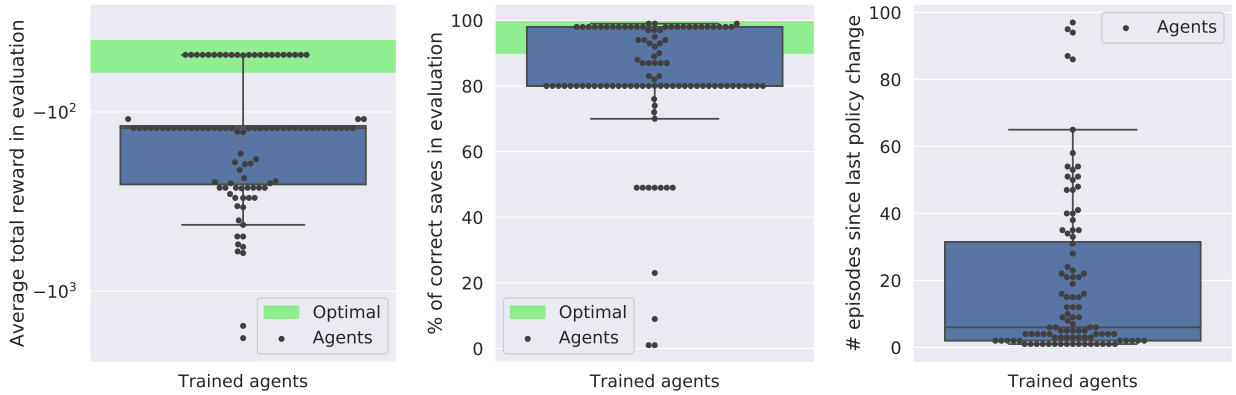
1. Training agents in simulation

The results of training 100 agents in simulation are shown in Fig. 9. They respectively show the performance of the agent, as measured by the average total reward (Fig. 9a) and the percentage of episodes resulting in a correct save (Fig. 9b). The number of episodes since the last policy change, as shown in Fig. 9c, is a metric indicating the stability of the agent.

From these figures, it can be seen that a number of agents exist with similar top performance. Further inspection shows that the similar performance of these agents is due to the fact that they have converged to a policy that is, for all practical purposes, equal. From this point forward, this policy shall be referred to as the *optimal policy*.

Further evaluation of this optimal policy is performed during 10,000 fully greedy evaluation episodes, each with uniquely random generated noise on the $F_{ext,z}/m$ signal. This results in an average total reward of -86.5 ($\sigma = 32.1$). Furthermore, 96.4% ($\sigma = 2.0$) of the episodes result in a correct save.

Of the 100 trained agents, 22% have found this optimal policy after training for 500 episodes. Given the level of performance achieved by this optimal policy, one could conclude that the RL setup works. Furthermore, 85% of all trained agents save the quadrotor successfully in $\geq$80% of the episodes.



(a) Boxplot showing the average total reward in the evaluation episodes.

(b) Boxplot of the percentage of evaluation episodes resulting in a correct save.

(c) Boxplot showing the number of episodes since the last policy change.

Fig. 9 Performance and stability metrics for the 100 agents trained in simulation.

In addition to providing insight into the performance distribution of agents trained in this environment, this experiment also set forth to select a *top agent*. This is the agent that will be used for the experiments in the flight phase. As further selection from 22 agents with the optimal policy based on performance is not evident, selection among these agents is based on the perceived stability of the agent.

The convergence of the selected *top agent* can be seen in Fig. 11. Furthermore, the final policy of this top agent can be seen in Fig. 10. For clarity, only the policy for states visited more than 10 times during the 500 training episodes are shown.

Indicated in the policy is the key intervention state (circled), the first intervention that the RL agent is likely to encounter when descending towards an obstacle underneath. Accounting for discretization, this part of the policy thus says: perform a save when: $-0.65\text{N/kg} < F_{ext,z}/m \leq -0.55\text{N/kg}$ and the previous action is $a_{no-action}$.

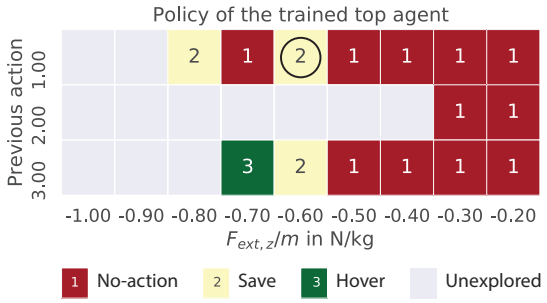


Fig. 10 Policy of the selected top agent, including the key interevention state.

From this experiment, the following conclusions can be drawn. First of all, in the simulation environment, there is one clear optimal policy, achieving high performance both in terms of average total reward (-86.5) and percentage of episodes resulting in a correct save (96.4%). Secondly, 85% of all the trained agents are able to save the quadrotor successfully in ≥80% of the episodes. Thirdly, only a small percentage of the agents (22%) converges to the optimal solution.

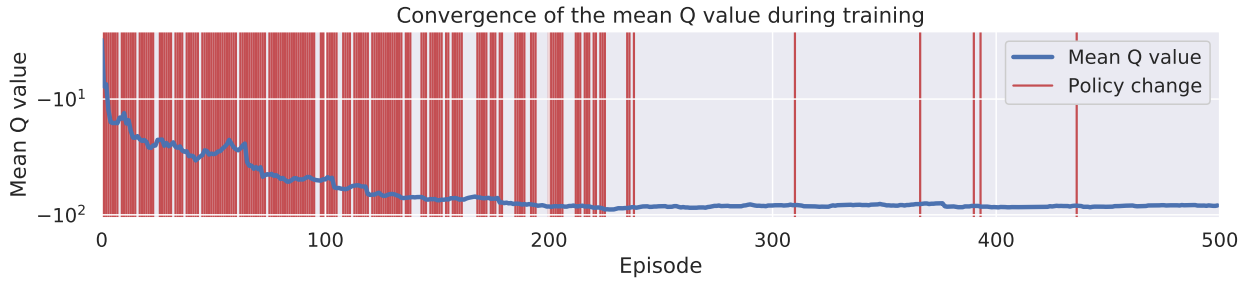


Fig. 11 Convergence of the selected top agent, as seen from the mean Q value and policy changes.

2. The influence of noise on agent performance

To investigate the effect of noise in the force and torque estimations, an experiment is conducted where 100 RL agents are trained with 26 different levels of noise added to the $F_{ext,z}/m$ state signal. The results of these experiments are shown in Fig. 12 and Fig. 13. These figures respectively show the influence of the noise on the average total reward

and the percentage of episodes resulting in a correct save. They do so as a function of the SNR level, which is taken at 0.15m distance to the obstacle underneath.

The expected rewards for three key agent behaviors are also shown in these figures. First of all, the line at -2000 indicates the expected reward for a crash. This would be the performance of an agent that always performs $a_{no-action}$. The only way in which an agent could achieve a performance worse than this would be if it performed hovering actions but still crashed every episode. Secondly, the line at -500 indicates the expected reward for a policy that always performs a save at the start of every episode, the *always-save* policy. Since the quadrotor would not yet be in the ground effect, this would always result in a false save. As such, this is the level of performance that can be achieved regardless of the noise. Finally, the line at -50 indicates the expected reward for an agent that is able to correctly save the quadrotor in every episode, but do so at the last minute, so when $z - z_{termination}$ is close to zero. Any agents with a performance better than this are thus able to save the quadrotor almost every episode and do it farther away from the obstacle.

The following observations can be made from these figures. Below -9dB, almost all agents have an average reward of -500 and 0% save rate, suggesting that they are unable to detect the presence of an obstacle underneath and thus converge to an *always-save* policy. Between -9dB and -1dB, the performance of all agents is actually worse compared to the *always-save* policy. This suggests that it is possible to detect obstacles, but that the reliability of doing so is lacking, resulting in some saves, but mostly crashes. Between -1dB and 2dB there are some top agents that are able to achieve a performance better than *always-save*. However, the stochasticity of the signal makes it difficult for the algorithm to find these better performing policies. Beyond 6dB almost all agents find a policy better than the *always-save* policy, their policies resulting in a correct save in ≥70% of the episodes.

Furthermore, the distribution of the agents' performances becomes smaller. Between a SNR of 12dB and 26dB, all trained agents are able to perform a correct save in almost every episode. Improvement is found in performing a save at larger distances to the obstacle underneath. Beyond 26dB, the performance is constant. Almost all agents are able to save the quadrotor far away from the obstacle in all episodes.

From these results, it can be concluded that using the current setup, a SNR ≥-1dB is required to outperform a trivial always-save policy and a SNR ≥2dB for most agents to do so. Beyond 6dB most agents are able to perform the obstacle avoidance task quite well, performing a correct save in ≥70% of the episodes.

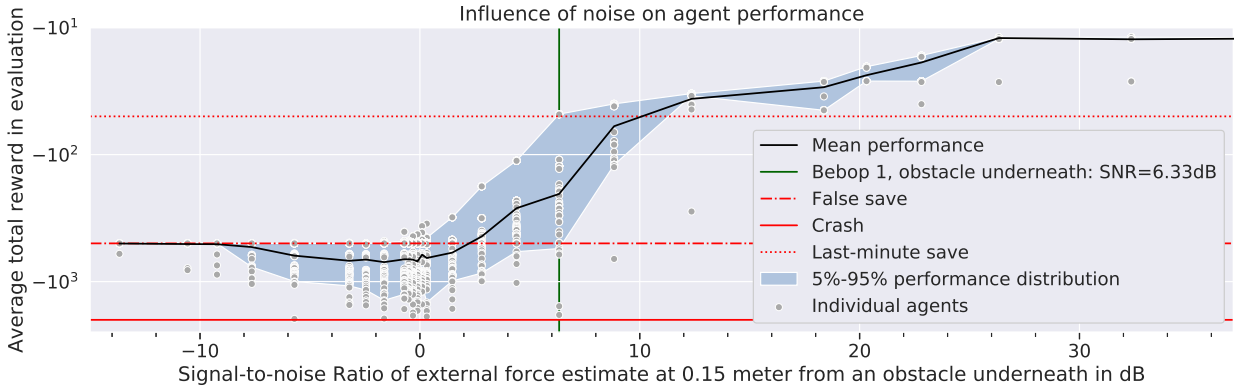


Fig. 12 Influence of noise on agent performance, as measured by the rewards.

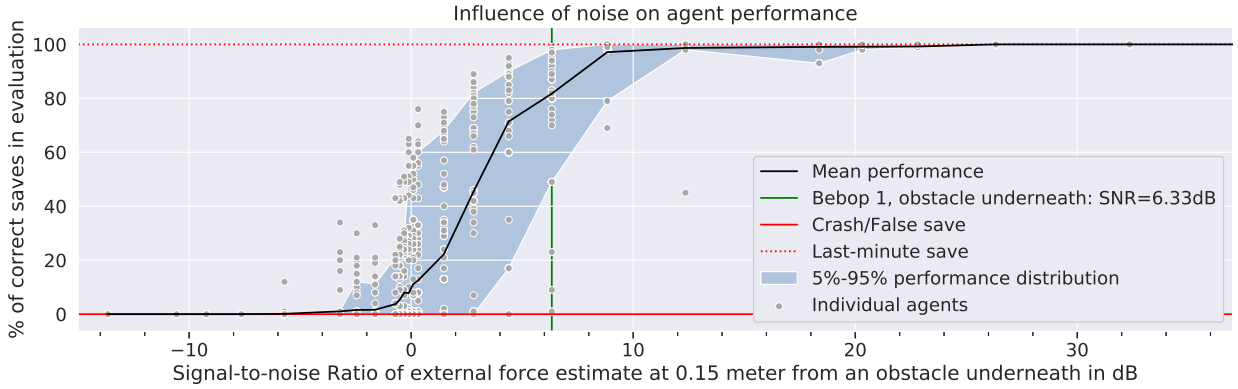


Fig. 13 Influence of noise on agent performance, as measured by the percentage of episodes resulting in a save.

3. The number of required training episodes

The results of the experiment to determine the number of required training episodes are shown in Fig. 14. Two key observations can be made. First of all, the performance of agents trained for 100 episodes is similar to those trained for 500 episodes or more. Secondly, while the mean performance of agents trained for 25 and 50 episodes is lower, in both cases there are still some agents which manage to learn the optimal policy. After training for only 50 episodes, 25% of the agents have learned the optimal policy. Considering the stochasticity involved in both the exploration and the state signal, this can be considered similar to the percentage of agents that have learned the optimal policy after training for 100 episodes (19%) and 500 episodes (22%).

It can thus be concluded that training for 100 episodes is sufficient when the purpose is to achieve the best performance for each of the trained agents. However, if one is only concerned with finding one agent with the optimal policy, it can be more efficient to train agents for only 25 or 50 episodes.

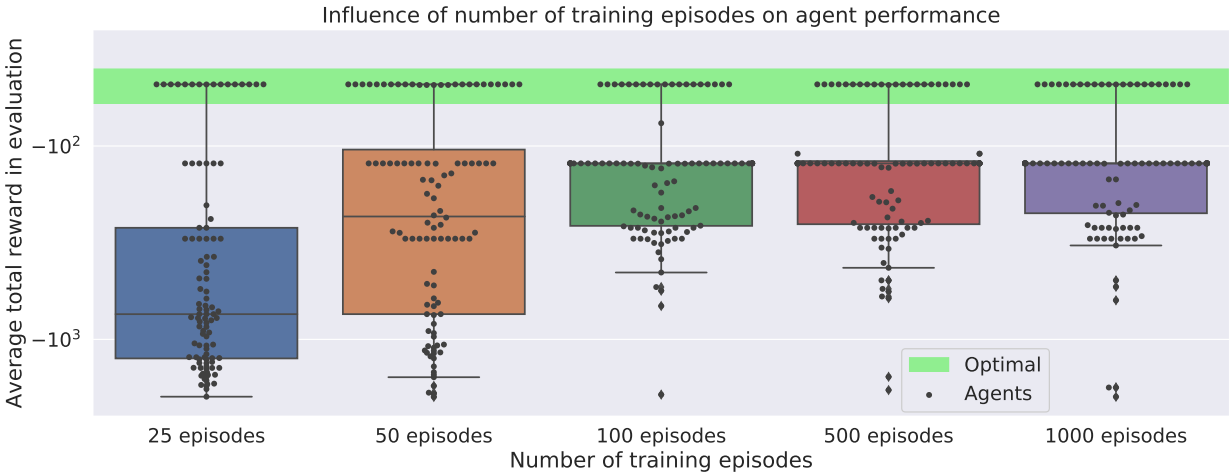


Fig. 14 Boxplot showing the influence of the number of training episodes on agent performance.

VI. Real flight

In the second phase, experiments are carried out in real flight, using a Parrot Bebop 1 quadrotor. First, the hardware and software setup that is developed for these experiments is discussed. Then the setups of three experiments that are carried out in real flight are presented; the evaluation of the top agent trained in simulation, continuing training of this top agent, and the training of an agent from scratch.

A. Experiment setup

First, the hardware and software setup, as well as the experiments carried out using this setup, are discussed.

1. Hardware and software setup

The quadrotor being used for the flight experiments is the Parrot Bebop 1 quadrotor shown in Fig. 15. This relatively inexpensive drone features 4 outrunner, brushless, motors, driving 4 rotors with a radius of 6.4cm. For the measurement of accelerations and angular rates, the Bebop relies on the MPU 6050 chip, which contains a 3-axis gyroscope and 3-axis accelerometer [26]. All flight experiments are carried out with the protection bumpers attached. With these bumpers, the drone weighs 420 grams.

Fig. 15 Parrot Bebop 1 quadrotor [25].

All experiments are carried out in the Cyberzoo of Delft University of Technology. This test area for ground robots and aerial vehicles spans 10m x 10m and is 7m high. It is equipped with the *Optitrack: Motive Tracker* optical tracking system, consisting of 24 cameras, enabling high precision positioning. This system provides the current position of the quadrotor, with 0.5cm accuracy at 120Hz, as GPS coordinates, via a wired connection to a laptop that functions as a ground station.

Onboard the quadrotor runs version 5.13 of the open source autopilot software Paparazzi. [†] The Paparazzi software suite also contains software for the ground station. The ground station is an Elitebook 8570w laptop. Communication between the quadrotor and ground station, including the position as provided by Optitrack, is done over Wi-fi.

Paparazzi is a modular platform on which additional functionality can be easily be added by creating custom modules. For the purpose of this experiment, such a custom Paparazzi module has been developed. This module is written in C and performs the following functions in-flight:

- **State estimator**: estimating $F_{ext,z}/m$ based on Eq. 12, using the filtered accelerations, motor speeds, rotational rates, body speeds, and thrust model. To enable this, the state estimator is also responsible for conducting a 2-second initialization procedure at the start of each flight. During this procedure the gains for the thrust model (k_i/m) are estimated, using Eq. 10. For the drag coefficient a constant value of $k_{D,z} = 0.271\text{N} \cdot \text{s}^2/\text{m}$ is used.
- **Reinforcement learning actor**: at every step of the episode, this actor is presented with the reward and the state (consisting of the estimated $F_{ext,z}/m$ and the previous action), based on which the actor will select one of the three actions (no-action, hover or save). It does so, based on its exploration strategy and the policy it determines at the start of each episode. When the action is to intervene (a_{hover} or a_{save}), this action is passed to the Paparazzi inner vertical control loop as a command. When the chosen action is $a_{\text{no-action}}$, the inner vertical control loop will follow the flight plan, which in all of the conducted experiments commands it to descend with 0.3m/s.
- **Safety controller**: if the quadrotor comes to close to the obstacle underneath $z < z_{\text{termination}}$, the safety controller will end the episode and send the quadrotor back to its start point of 1 meter above the obstacle. This part of the module thus has access to the height above the obstacle, as provided by the Optitrack system. This information is however not shared with the RL actor or critic, except through the rewards they receive.
- **Data logger**: storing all relevant variables for later analysis, including measured accelerations, speeds, motor speeds, estimated $F_{ext,z}/m$ and chosen actions. All these variables are written to a .csv file every timestep.

Other than the custom module, the quadrotor uses only existing Paparazzi modules models for its flight control. For stabilization in the horizontal plane, usage is made of the Incremental Nonlinear Dynamic Inversion (INDI) module [27]. For speed and positional control, the Paparazzi default inner vertical and horizontal control loops are used. It must be noted however that the *HOVER_KD* gain for the vertical loop has been increased from 100 to 600, in order to ensure closer tracking of the desired descent speed. This is required to maintain constant descent speed when nearing the obstacle underneath, as the ground effect tends to reduce the descent speed. Furthermore, the importance of thrust control with respect to the roll and pitch axes was increased from 10 : 1000 to 100 : 1000. The complete control scheme for the flight experiments can be found in appendix A.

The reinforcement learning critic was custom developed for these particular experiments and runs on the ground station laptop. It was developed in Python and uses the codebase that was originally developed for the simulation experiments. At the end of every episode, the critic retrieves the data log file of that episode from the quadrotor over FTP. For each step in this episode, Watkin's Q(λ) algorithm is then used to calculate the change in action-value function (Q), based on the state, selected action, and resulting reward. To do so, the critic needs to distinguish between greedy actions and exploratory actions. The exploratory actions being either episode-long exploratory actions or an exploratory action taken at one specific timestep by the ϵ–greedy exploration. The critic uses this distinction to reset the eligibility traces when needed. After processing every step in the episode, the result is sent to the quadrotor over the Ivy bus, a text-based (ASCII) Publish-Subscribe protocol that communicates over the local Wi-Fi network.

2. Evaluation of the top agent trained in simulation

The first experiment carried out in real flight is the evaluation of the top agent trained in simulation. The purpose of this experiment is twofold. First of all, it serves to validate that the simulation environment is a reasonably accurate representation of the real flight environment. This can then be used to argue that results from experiments carried out in simulation, like the influence of hyperparameters, noise and number of training episodes, hold for real flight as well. Secondly, it provides a first performance assessment in terms of the obstacle avoidance capability of this novel method.

In the experiment, for each of the evaluation episodes, the Bebop 1 quadrotor is commanded to descend from 1m height with $0.3m/s$. Onboard the quadrotor runs the top RL agent trained in simulation. All episodes are carried out as fully greedy evaluation episodes, so no exploring starts, no exploration and no learning by the agent, just the evaluation of the found policy. Twenty evaluation episodes are carried out, using the artificial grass surface of the TU Delft CyberZoo as the obstacle underneath.

[†] `http://wiki.paparazziuav.org/wiki/Main_Page`, as accessed on 02/04/2019

3. Continuing training in real flight

Secondly, an experiment is carried out where training of the top agent from the simulation experiment is continued in real flight. This is done during 100 training episodes, using the same exploration strategy as used in the simulations, and the hyperparameters found to be best for helping an agent adjust to a slightly different environment. After training, the agent is once again evaluated during 20 fully greedy evaluation episodes. In all flights, the artificial grass surface functions as the obstacle underneath.

4. Training from scratch in real flight

Finally, using the knowledge gained during previous experiments, an experiment is conducted where multiple RL agents are trained fully from scratch, during a real flight. Training is conducted during 50 episodes, using the same exploration strategy as used in the simulations, and the hyperparameters found to be best for the initial training of agents. Evaluation is performed during 20 fully greedy evaluation episodes. In these flights the artificial grass surface functions as the obstacle underneath, however, if this yields promising results, an additional evaluation is performed using an alternative obstacle, a 75cm x 53cm x 17.5cm box placed underneath the quadrotor.

This experiment serves to determine whether a RL agent can be trained fully in flight to avoid obstacles underneath it, using the obstacle-airflow interactions caused by this obstacle. If this is indeed possible, it shows the potential for the extension of this method to other quadrotors, or other types of obstacles, without requiring simulation environments or models of the obstacle-airflow interaction.

B. Results

Using the hardware and software setup described in section VI.A.1, several experiments are carried out in real flight, using the Parrot Bebop 1 quadrotor. First, the top agent from the simulation phase is evaluated in flight. Secondly, an experiment is conducted in which training of the previously found top agent is continued in flight. Finally, an experiment is conducted where multiple agents are trained during real flight, without any prior knowledge of the environment.

1. Evaluation of the agent trained in simulation

Of the 20 in-flight evaluation episodes, the top RL agent trained in simulation is able to perform a correct save in 80%. As can be seen from Fig. 16, 5% of episodes result in a false save and 15% in a crash. This results in an average reward of -350. When comparing this to the performance of this agent in simulation; with an average reward of -86.5 (σ = 32.1), and 96.4% (σ = 2.0) of episodes ending in a correct save; it is clear that the performance in real flight is significantly worse. Based on these results, it is estimated to be around 20% worse.

An explanation for this difference in performance could be the 140ms delay in the $F_{ext,z}/m$ signal, introduced by the low-pass Butterworth filter, discussed in section III.C. As this delay is present in real flight, but not in the simulation, obstacles are expected to be detected slightly later in real flight, potentially leading to worse performance.

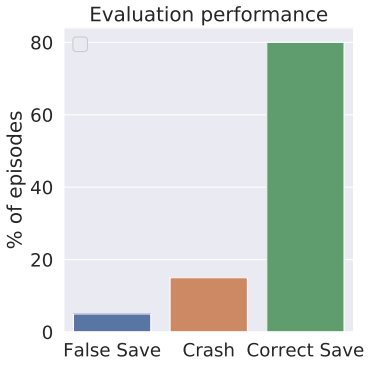


Fig. 16 Evaluation results of the agent trained in simulation.

While the results above provide the first assessment of the obstacle avoidance capability of this new method, there is a second goal of the experiment; validating that the simulation environment is an accurate representation of the real flight environment. There are two key observations that are especially relevant to this. First of all, the observation that the agent is able to perform well in the real flight environment, while only having being trained in the simulation environment. This suggests that a good performance in simulation corresponds to a good performance in real flight. Secondly, the exact level of performance achieved by this policy in the simulation environment is different from the performance in real flight.

While the first observation speaks to validate the simulation model, the second raises the question of whether the optimal policy found in simulation is also the optimal policy in real flight. To answer this question, and further validate the simulation environment, this is investigated in the next two flight experiments. This is done by seeing if an even better policy can be found by continuing training of the top agent in real flight, or training a new agent from scratch in real flight.

2. Continuing training in real flight

To check if the performance of the top agent can be further improved, training is continued in flight for 100 episodes, using the previously determined set of hyperparameters deemed best for adjusting to a slightly different environment. Due to the stochasticity involved, both in the exploration and in the $F_{ext,z}/m$ signal, this experiment is conducted five times.

In all five training runs it can be seen that the agent experimented with different policies. After the 100 training episodes, 4 of the 5 agents have converged to the same policy they started with, the top policy from the simulations. The other agent has ended up with the policy shown in Fig. 17. Accounting for discretization this policy says: perform a save when: $-0.75\text{N/kg} < F_{ext,z}/m \leq -0.65\text{N/kg}$ and the previous action was $a_{\text{no-action}}$. Compared to the optimal policy found in simulation, the RL agent thus intervenes later, at a more negative $F_{ext,z}/m$.

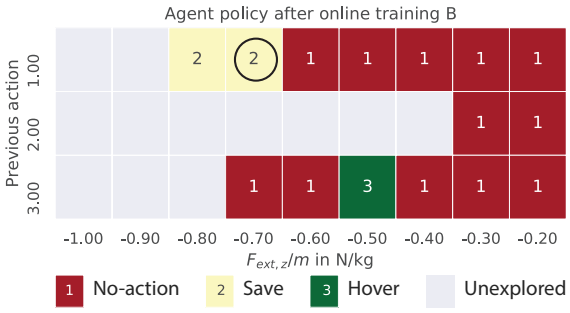


Fig. 17 Alternative policy found after continuing training online.

To test whether this policy is better than the initial policy, it is evaluated during 20 fully greedy episodes. The performance is clearly worse. Only 5% of the episodes results in a correct save, 95% result in a crash. This also becomes clear from the average total reward, which is -1902.

This result thus supports the hypothesis that the optimal policy found in simulation, is the optimal policy in real flight as well. Furthermore, this implies that the performance shown in Fig. 16, performing a correct save in 80% of the episodes, is the best achievable performance within the current setup.

3. Learning from scratch in real flight

Since the simulation experiments show that even after training for only 50 episodes, some RL agents have found the optimal policy, an attempt can be made to train a RL agent fully from scratch, during a real flight. Of the 5 agents trained in real flight, one has converged to a policy with the top performance, as can be seen from the evaluation results shown in Fig. 18. During the 20 evaluation episodes, this agent correctly performed a save 16 times (80%), crashed 2 times (10%) an performed a false save 2 times (10%).

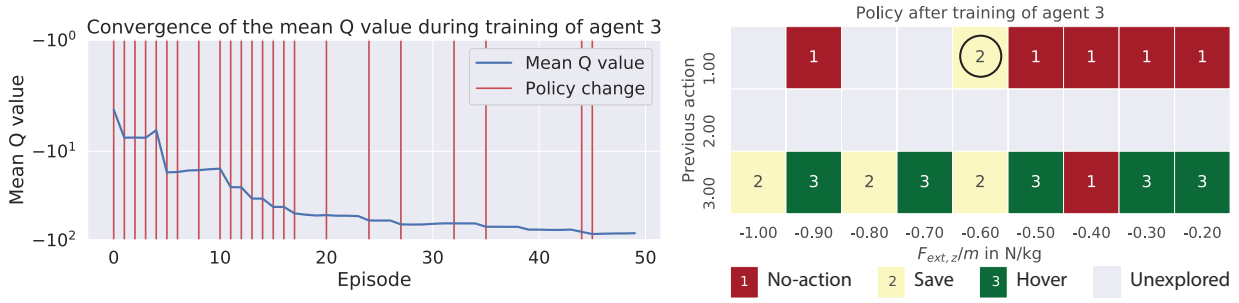
Fig. 18 Evaluation performance of the 5 agents trained from scratch in flight.

Further inspection of this particular agent shows that its policy, as shown in Fig. 19b, is very similar to the optimal policy found in simulation. Both have the same key intervention, a save when $-0.65\text{N/kg} < F_{ext,z}/m \leq -0.55\text{N/kg}$ and the previous action is $a_{\text{no-action}}$. It can thus be argued that for practical purposes this policy is equal to the optimal one found in simulation. As such, this speaks for validation of the simulation environment.

This agent is also evaluated using a 75cm x 53cm x 17.5cm box as the obstacle underneath, instead of the artificial grass surface. During the 20 fully greedy evaluation episodes, this results in the agent performing a correct save 15 times (75%) and a crash 5 times (5%).

From a practical perspective, it takes 15 minutes to train a single agent and evaluation takes another 7 minutes. This includes the flight time for the 50 episodes, the initialization procedures and the replacement of batteries (3x). If enough batteries are available, training 5 agents sequentially could thus be conducted in less than 2 hours. This is important when considering the extension of this method to other quadrotors, or other types of obstacles.

Overall, it can thus be concluded from this experiment that it is possible to successfully train an agent fully in flight, in only 50 episodes to detect and avoid obstacles underneath a descending quadrotor. Upon approaching an obstacle, a detect-and-avoid accuracy of at least 80% can be achieved by a reinforcement learning agent with the optimal policy.



(a) Convergence of the third agent trained from scratch. **(b) Policy of the third agent trained from scratch.**

Fig. 19 Convergence and policy of the third agent trained from scratch.

There is however no guarantee that every agent that is trained will always converge to this optimal policy. Results from simulation and real flight suggest that only 20-25% of agents will converge to the optimal policy. Therefore, it is recommended to train multiple agents, evaluate them and select the best.

VII. Extension to other types of obstacles and other quadrotors

While the results discussed above look promising, they of course only demonstrate the obstacle avoidance capability for one specific drone, the Parrot Bebop 1, and only for large obstacles underneath the quadrotor. To assess the potential of this method as a general method of obstacle avoidance, the extension to other quadrotors and other types of obstacles must be considered.

A. Extension to other obstacles

When considering the extension of this method to other obstacles, one could consider extending it to detect large obstacles above, like a ceiling, obstacles on the same level, like a wall, or to smaller or otherwise different obstacles. The feasibility of using this method to detect walls and ceiling surfaces will depend on the strength of the effects caused by these surfaces and the amount of noise and other disturbances present in the estimated forces and torques. For the Parrot Bebop 1, an initial estimate of the SNR of the effects caused by the ceiling and wall is made as part of this research.

A large surface above the quadrotor, e.g. a ceiling, is known to cause an external force in the vertical direction [8]. The effect can be estimated using the measurement data from Sanchez-Cuevas et al. [8]. Adjusting for the different rotor radius, a first estimate of the external force caused by the ceiling would be $F_{ext,z}/m \approx -0.32$N/kg. If similar noise as seen in the presence of obstacles underneath is then assumed, this results in a SNR of 8.63dB. Since this is larger than the SNR seen for obstacles underneath, it implies that these obstacles are easier to detect. One must, however, note that the force, in this case, is pulling the quadrotor towards the surface above, in contrast to pushing the quadrotor away from a surface underneath. As such, the intervention action might be more difficult or require a sooner intervention, thereby perhaps reducing the performance.

To estimate the feasibility of detecting large vertical surfaces on the same level, like walls, an experiment is conducted using the Parrot Bebop 1 drone. In this experiment, a 1m wide, 2.05m high screen is placed inside the CyberZoo. The quadrotor then hovers at a height of 1.5 meter at varying distances from the wall, ranging from 1-meter to up to 1-centimeter distance. The measurement data is used to construct plots similar to Fig. 4, showing the estimated forces and torques versus the distance to the wall.

The clearest influence of the wall can be seen in the torque around the y-axis, the effect is however relatively small compared to the noise. An initial analysis estimates the SNR to be -4.15dB. For such a SNR it is not expected that agents will able to find a policy that results in correct saves, as can be seen from Fig. 20.

It must, however, be noted however that this is only an initial analysis. It is expected that by analyzing the effects in more detail, using better sensors, or using more than 1 external force or torque as a state, the performance might be further improved. Furthermore, the performance could also be improved by using some of the other recommendations discussed in section IX.

The extension to smaller or otherwise different types of obstacles is something that might be done incrementally. As mentioned in the results, the RL agent trained in this research is already able to detect a large box. By continuing training with smaller obstacles underneath, perhaps in combination with refining the used discretization, the detection limits can be found and improved.

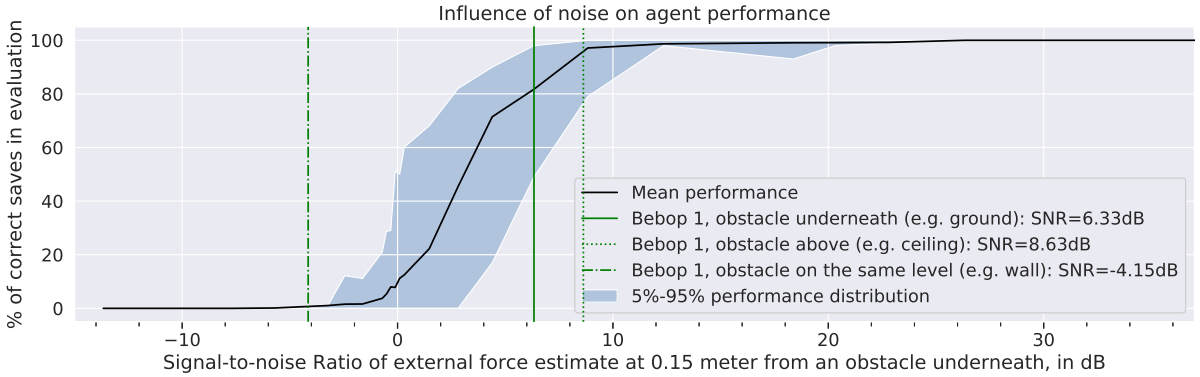


Fig. 20 Estimated Signal-to-Noise ratios and resulting performance, for the detection of walls, ceiling and ground surfaces.

Overall, the following conclusions can be drawn with respect to the extension of this method to other obstacles. The extension of this obstacle avoidance method to the detection of surfaces above the quadrotor, e.g. ceilings, is expected to achieve a similar or even better performance as for surfaces underneath. The extension to surfaces on the same level as the quadrotor, e.g. walls, is expected to require some significant improvements to the SNR of the estimated forces and torques, or to the usage thereof, before a similar performance can be reached. The extension to smaller or otherwise different types of obstacles requires further research, and possibly a more dense discretization of the states.

B. Extension to other quadrotors

Finally, the extension to other quadrotors. The whole method has been set up such that it requires little prior knowledge of the quadrotor. For example, no assumptions are made about the mass of the quadrotor and the moments of inertia are calculated in flight. Furthermore, the rotor gains k_i/m are automatically estimated during initialization procedures. The following steps are suggested to implement this obstacle avoidance method on another quadrotor:

1) Perform an experiment where the quadrotor first moves up and down without any nearby obstacles and then descends to 5cm above a ground surface three times. Use the gathered measurement data to estimate the drag coefficient as discussed in section III.B, and if need be, adjust the discretization bounds for the estimated external forces and/or torques.
2) Train multiple agents in flight, using a large horizontal surface as the obstacle underneath. Suggested is to train at least 5 agents for 50 episodes, using the hyperparameters discussed in section IV.E.
3) Evaluate all agents during a number of evaluation episodes, at least 20 is suggested, and pick the best agent for implementation.

While following these steps should provide a good basis, there is no guarantee on the performance of the optimal obstacle avoidance policy. This will especially depend on the ratio between noise and obstacle-airflow interaction effects within the state signal, the SNR, as discussed in section V.B.2.

To demonstrate the extension of this method to other quadrotors, an experiment is performed using the Parrot Bebop 2 drone. This successor to the Bebop 1 has a larger frame, larger rotors (R_{rotor} = 7.5cm) and new motors [28]. Furthermore, in the experiment the Bebop 2 is flown without the bumpers, thereby potentially altering the airflow, and thus the obstacle-airflow interactions. Other than this, the setup is as described in section VI.A.4. Two RL agents are then trained to avoid obstacles underneath this quadrotor. After 50 training episodes, one of the agents has already learned a policy with similar performance as was achieved on the Parrot Bebop 1 drone. In the 40 fully greedy evaluation episodes it is able to perform a correct save in 80% of the episodes, 2.5% resulted in a crash and 17.5% in a false save. It can thus be concluded that the obstacle avoidance method can be extended to other quadrotors in only a few steps.

VIII. Conclusion

In this research, a first step in the development of a novel obstacle avoidance method for quadrotors is taken. A reinforcement learning agent is successfully trained to detect and avoid obstacles underneath a descending quadrotor.

To accomplish this, a simple quadrotor model is introduced and used to estimate external forces and torques around all three axes. Measurement flights with a Parrot Bebop 1 quadrotor are performed in order to model one of these

estimators, the external force in the vertical direction ($F_{ext,z}/m$) as a function of distance to the ground. These models are used to create a simulation environment in which RL agents can be trained. This simulation environment represents the actual flight environment as much as possible, replicating not only the quadrotor equations of motion but also its inner loop flight control. Furthermore, the states ($F_{ext,z}$ and the previous action), actions ($a_{\text{no-action}}$, a_{hover} and a_{save}), and rewards of the environment are defined. In this environment, several experiments are then performed in simulation. These are performed using the Watkin's Q(λ) reinforcement learning algorithm, a custom threefold exploration strategy, and hyperparameters determined during an extensive grid search.

From these results, it can be concluded that the estimated external force in the vertical direction is a good indicator for the presence of large surfaces underneath. Furthermore, a reinforcement learning agent can be trained to use this estimated external force to detect and avoid obstacles underneath. However, due to the stochastic nature of the problem, not all agents will find the optimal policy. Multiple agents should thus be trained to ensure an optimal solution is found.

The best agent trained in simulation is evaluated in real flight, during an experiment with the Parrot Bebop 1, running the Paparazzi open-source flight software, inside the Delft University of Technology CyberZoo. The results show that the agent is able to save the quadrotor from hitting the obstacle underneath in 80% of the episodes. An attempt is made to further improve this performance by continuing training online, but this yields no further improvement, suggesting that the optimal policy found in simulation is also the optimal policy in real flight.

Finally, it is shown that it is possible to train an agent fully from scratch during a real flight. This is accomplished by training 5 agents during 50 episodes each, without prior knowledge or training in simulation. Of these 5 agents, one found the optimal policy, confirming the conclusion that multiple agents should be trained in order to ensure that the optimal policy is found. Since this did not require any simulation beforehand, it suggests that a RL agent can be trained to avoid other types of obstacles or obstacles on another quadrotor in real flight in a similar fashion.

For the extension of this method to other quadrotors, a procedure is presented. Furthermore, this extension is demonstrated using the Parrot Bebop 2 drone. Showing that similar performance can be achieved on another quadrotor in only a few short steps, without requiring any specific quadrotor model or simulation.

Furthermore, an approach for extending this method to the avoidance of walls, ceilings, and smaller obstacles is discussed and the expected performance when doing so on the Parrot Bebop 1 is assessed. For surfaces above the quadrotor, e.g. ceilings, the method is expected to achieve a similar or even better performance as for surfaces underneath. The extension to surfaces on the same level as the quadrotor, e.g. walls, is expected to require some improvements to the SNR of the estimated forces and torques, or to the usage thereof, before a similar performance can be achieved.

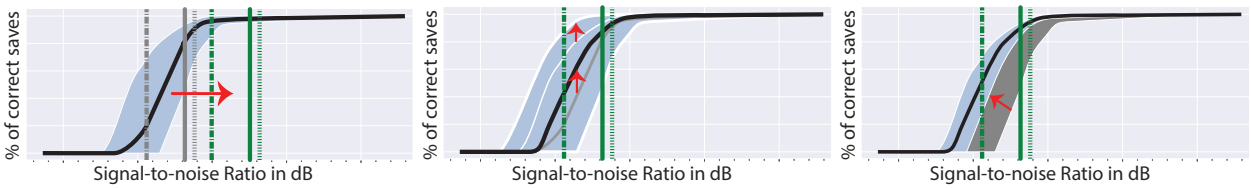
Overall, it can be concluded that it is possible to use reinforcement learning and obstacle-airflow interactions for the detection and avoidance of large obstacles underneath a Parrot Bebop 1 quadrotor. Furthermore, it is expected that this method can be extended to other quadrotors, as well as to large obstacles or surfaces above the quadrotor.

IX. Recommendations for future research

While initial success for this new obstacle avoidance method is shown, there are many points on which the method can be improved. First of all, this method can be extended to other quadrotors, or other types of obstacles, e.g. walls, ceilings, or smaller obstacles, as discussed in section VII.

Secondly, the estimations of the external forces and torques might be improved, thereby increasing the signal-to-noise ratios, the effect of which is shown in Fig. 21a. This could be done by improving the estimator, reducing noise or correcting for other disturbances. Improving the estimator could be accomplished by using more accurate models for the produced thrust, drag or quadrotor dynamics. Reducing the noise in the estimator could be achieved by using better sensors, improved filtering, or by reducing latencies in the underlying measurements. Finally, there can be other effects, like wind, turbulence or the airflow of other aerial vehicles, causing external forces and torques on the quadrotor. Methods might be developed by which they can be identified and corrected. In the case of wind and turbulence, a correction might reduce false positives or false negatives, improving the performance of the obstacle avoidance method. In the case of other aerial vehicles, identification might extend the applicability to dynamic obstacle avoidance.

Finally, the way the estimated external forces and torques are being used might be improved. This could lead to a better optimal policy, the expected effect of which is shown in Fig. 21b, a better distribution of performance among trained agents, as shown in Fig. 21c or a combination of the two. Two ways in which this might be accomplished are; the combination of multiple estimators and improvements to the reinforcement learning setup. For obstacle-airflow interactions expected to result in more than one external force or torque, e.g. those caused by a wall, providing multiple estimators as a state to the RL agent could improve performance. Another way the proposed method might be improved is by improving the reinforcement learning setup. Of the many ways in which this could potentially be accomplished,



(a) Improvements resulting in a better signal-to-noise ratio. **(b) Improvements resulting in a better optimal policy.** **(c) Improvements resulting in a better performance distribution.**

Fig. 21 Potential influence of proposed improvements on agent performance.

two interesting potential improvements can already be recommended for future research. First, the representation of the action-value might be improved; either by increasing the discretization density, or using another function approximator such as a Support Vector Machine (SVM) or a neural network. Secondly, the reward structure of the current setup might be improved, for example by providing a positive reward for a correct save, with a larger distance to the obstacle resulting in a larger positive reward. Initial experiments suggest that this can increase the percentage of the agents finding the optimal policy in 50 episodes, from 20-25% to 40%.

To conclude, future research could both improve upon the current obstacle avoidance capabilities of this low-cost method, and extend the method to other quadrotors and types of obstacles. Thereby increasing the potential of this method as a primary obstacle avoidance method for small quadrotors, or as a secondary obstacle detection method for larger quadrotors.

References

[1] DroneDeploy, "Commercial Drone Industry Trends," Tech. Rep. March, DroneDeploy, 2017.

[2] Zhang, T., Li, Q., Zhang, C.-S., Liang, H.-W., Li, P., Wang, T.-M., Li, S., Zhu, Y.-L., and Wu, C., "Current trends in the development of intelligent unmanned autonomous systems," *Front Inform Technol Electron Eng*, Vol. 18, No. 1, 2017, pp. 68–85. doi:10.1631/FITEE.1601650, URL `http://dx.doi.org/10.1631/FITEE.1601650`.

[3] Alberto, F. B.-f., and Gabriel, O., "Visual Navigation for Mobile Robots : A Survey," *Journal of intelligent and robotic systems*, Vol. 53, No. 3, 2008, pp. 263–296. doi:10.1007/s10846-008-9235-4.

[4] Stowers, J., Hayes, M., and Bainbridge-Smith, A., "Altitude control of a quadrotor helicopter using depth map from Microsoft Kinect sensor," *2011 IEEE International Conference on Mechatronics*, 2011, pp. 358–362. doi:10.1109/ICMECH.2011.5971311.

[5] Cheeseman, I. C., and Bennett, W. E., "The Effect of the Ground on a Helicopter Rotor in Forward Flight," *Aeronautical Research Council Reports and Memoranda*, Vol. 3021, 1955, p. 12. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.226.6371`.

[6] Rozhdestvensky, K. V., "Wing-in-ground effect vehicles," *Progress in Aerospace Sciences*, Vol. 42, No. 3, 2006, pp. 211–283. doi:10.1016/j.paerosci.2006.10.001.

[7] Mckinnon, C. D., "Data Driven , Force Based Interaction for Quadrotors by Data Driven , Force Based Interaction for Quadrotors," Ph.D. thesis, University of Toronto (Canada), 2015.

[8] Sanchez-Cuevas, P. J., Heredia, G., and Ollero, A., "Multirotor UAS for bridge inspection by contact using the ceiling effect," *2017 International Conference on Unmanned Aircraft Systems, ICUAS 2017*, 2017, pp. 767–774. doi:10.1109/ICUAS.2017.7991412.

[9] Rossow, V. J., "Effect of Ground and/or Ceiling Planes on Thrust of Rotors in Hover," *Nasa Technical Memorandum*, Vol. 86754, 1985.

[10] Sanchez-Cuevas, P., Heredia, G., and Ollero, A., "Characterization of the aerodynamic ground effect and its influence in multirotor control," *International Journal of Aerospace Engineering*, Vol. 2017, 2017. doi:10.1155/2017/1823056.

[11] Sharf, I., Nahon, M., Harmat, A., Khan, W., Michini, M., Speal, N., Trentini, M., Tsadok, T., and Wang, T., "Ground effect experiments and model validation with Draganflyer X8 rotorcraft," *2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014*, 2014, pp. 1158–1166. doi:10.1109/ICUAS.2014.6842370.

[12] Griffiths, D. A., "A study of dual-rotor interference and ground effect using a free-vortex wake model," *American Helicopter Society 58th Annual Forum, Montreal, Canada, June 11-13*, 2002.

[13] Lee, D., Awan, A., Kim, S., and Kim, H. J., "Adaptive Control for a VTOL UAV Operating Near a Wall," *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4835. doi:10.2514/6.2012-4835, URL http://arc.aiaa.org/doi/10.2514/6.2012-4835.

[14] Hwangbo, J., Sa, I., Siegwart, R., and Hutter, M., "Control of a Quadrotor with Reinforcement Learning," *IEEE Robotics and Automation Letters*, Vol. 2, No. 4, 2017, pp. 1–8. doi:10.1109/LRA.2017.2720851, URL http://arxiv.org/abs/1707.05110.

[15] Junell, J., Mannucci, T., Zhou, Y., and Kampen, E., "Self-tuning Gains of a Quadrotor using a Simple Model for Policy Gradient Reinforcement Learning," *AIAA Guidance, Navigation, and Control Conference*, 2016, pp. 1–16. doi:10.2514/6.2016-1387, URL http://dx.doi.org/10.2514/6.2016-1387.

[16] Mannucci, T., van Kampen, E., de Visser, C., and Chu, Q., "Safe exploration algorithms for reinforcement learning controllers," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 4, 2018, pp. 1069–1081.

[17] Zhou, Y., van Kampen, E., and Chu, Q., "Incremental Model Based Heuristic Dynamic Programming for Nonlinear Adaptive Flight Control," *Imavs*, 2016.

[18] Watkins, C., and Holloway, R., "Learning From Delayed Rewards," , 1989. URL https://www.researchgate.net/publication/33784417.

[19] García Carrillo, L. R., Dzul López, A. E., Lozano, R., and Pégard, C., "Modeling the Quad-Rotor Mini-Rotorcraft," *Quad Rotorcraft Control*, Springer, London, 2013, pp. 23–34. doi:10.1007/978-1-4471-4399-4, URL http://link.springer.com/10.1007/978-1-4471-4399-4_2.

[20] Salih, A. L., Moghavvemi, M., Mohamed, H. A. F., and Gaeid, K. S., "Flight PID controller design for a UAV quadrotor," *Scientific Research and Essays*, Vol. 5, No. 23, 2010, pp. 3660–3667. URL http://www.academicjournals.org/SRE.

[21] Manal, K., and Rose, W., "A general solution for the time delay introduced by a low-pass Butterworth digital filter: An application to musculoskeletal modeling," *Journal of Biomechanics*, Vol. 40, 2007, pp. 678–681. doi:10.1016/j.jbiomech.2006.02.001, URL www.elsevier.com/locate/jbiomech.

[22] Sutton, R. S., and Barto, A. G., *Reinforcement Learning : An Introduction*, 2015th ed., MIT press, 2015.

[23] Sutton, R. S., Precup, D., and Singh, S., "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, Vol. 112, No. 1, 1999, pp. 181–211.

[24] Kourepenis, A., Borenstein, J., Connelly, J., Elliott, R., Ward, P., and Weinberg, M., "Performance of MEMS inertial sensors," *IEEE 1998 Position Location and Navigation Symposium (Cat. No.98CH36153)*, IEEE, 1998, pp. 1–8. doi:10.1109/PLANS.1998.669861, URL http://ieeexplore.ieee.org/document/669861/.

[25] RChelicoptershop.nl, "Parrot Bebop Drone red," , 2019. URL https://www.rchelicoptershop.nl/parrot-bebop-drone/parrot-bebob-drone-red.

[26] Paparazzi UAV, "Bebop," , 2019. URL https://wiki.paparazziuav.org/wiki/Bebop.

[27] Smeur, E. J. J., Chu, Q., and De Croon, G. C. H. E., "Adaptive Incremental Nonlinear Dynamic Inversion for Attitude Control of Micro Air Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 3, 2015, pp. 450–461. doi:10.2514/1.G001490, URL http://arc.aiaa.org.

[28] Parrot, "Bebop 2 vs Bebop Drone comparison - Parrot Blog," , 2016. URL http://blog.parrot.com/2016/01/12/comparison-bebop-2-vs-bebop-drone/.

A. Appendix: Setup flight experiments

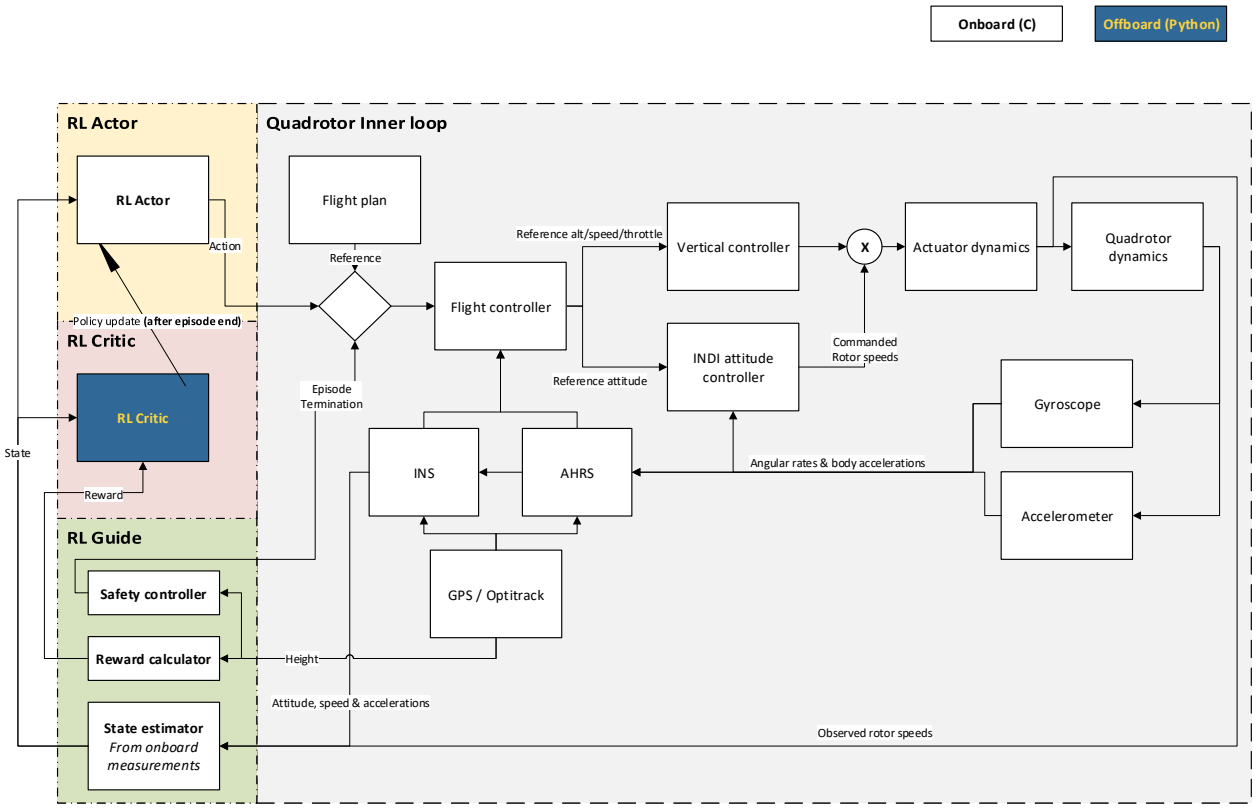


Fig. 22 Extended control scheme for the flight experiments