UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

**RL & FSR TECHNICAL PROJECT – PROJECT 3**

# CONTROL SYSTEM FOR AN AERIAL ROBOT

Proff.
Jonathan Cacace
Fabio Ruggiero

Candidato
Pasquale Marra
P38/002

Anno Accademico 2021/2022

# Contents

# Introduction

The goal of this project is to develop a control system for an aerial robot employed in a navigation task. We will start with the low level control, which is used to bring the quadrotor to a desired pose, then we'll focus on the planner, that generates the desired pose on the basis of a list of way points and the surrounding enviroment.

Firstly, we give a brief introduction about the adopted quadrotor and its analytical model; then, we will describe the low level control that takes as input the desired trajectory for the quadrotor and generates the proper set of velocities to impose to the propellers. Thanks to the presence of an estimator, the controller can be employed also in the case of external disturbances and unmodelled dynamics.

Once the quadrotor is able to move, we can talk about the planner. It consists in an online version of the artificial potential method: it takes as input the laser scan measurements and it modifies the trajectory accordingly. The planner is also able to cope with two kind of obstacles: solid obstacles, that must be avoided by flying around them, and empty obstacles, that must be passed through. The latter are recognized by means of a 2D camera.

The software runs on ROS noetic, RViz is used for the visualization and Gazebo for the simulation.

# Chapter 1

# Aerial robot description and modelling

The aerial robot chosen for this project is shown in Fig.1.1.

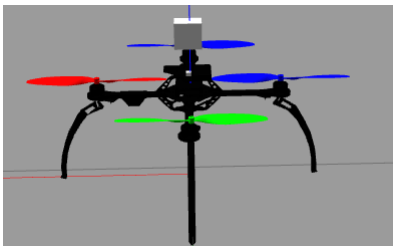

Figure 1.1: Hummingbird™quadrotors

It is a hummingbird™drone, that is a quadrotor by Skyrocket. It is an unmanned aerial vehicle (UAV) endowed with four propellers that produce lift and torque about its center of rotation, letting us to move the drone in the air by means of a predefined path, as we'll see later. In order to localise the drone and to interact with the enviroment, the hummingbird is endowed with the following sensors:

- IMU and GPS: used for the odometry and the localization of the drone in the enviroment;
- 2D camera: to retrieve visual information from the enviroment;
- 2D laser scan: used to evaluate the distance of the drone from the obstacles;

## 1.1 Model

Let's define a world-fixed inertia frame $\Sigma_i$ and a body-fixed reference frame $\Sigma_b$ placed at the UAV's center of mass. We can express the absolute position of the UAV with respect to $\Sigma_i$ by means of the vector $\mathbf{p}_b = [x \quad y \quad z]^T$ and the rotation of $\Sigma_b$ wrt $\Sigma_i$ through the matrix:

$$\mathbf{R}_b(\boldsymbol{\eta}_b) = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \tag{1.1}$$

where $\boldsymbol{\eta}_b = [\phi \quad \theta \quad \psi]^T$ represents the roll-pitch-yaw Euler angles vector. Then, using the Newton-Euler formulation we can retrieve the RPY quadrotor dynamic model:

$$\begin{cases} m\ddot{\mathbf{p}}_b = m\mathbf{g} - u\mathbf{R}_b(\boldsymbol{\eta}_b)\mathbf{i}_3 + \mathbf{f}_u \\ \mathbf{M}(\boldsymbol{\eta}_b)\ddot{\boldsymbol{\eta}}_b = -\mathbf{C}(\boldsymbol{\eta}_b, \dot{\boldsymbol{\eta}}_b)\dot{\boldsymbol{\eta}}_b + \mathbf{Q}(\boldsymbol{\eta}_b)^T\boldsymbol{\tau}_b^b + \boldsymbol{\tau}_u \end{cases} \tag{1.2}$$

Please refer to [1] for the meaning of the symbols used in the above expressions and the introduced assumptions. For control purposes, it is useful to draw attention to the following relations:

$$\begin{cases} u = c_T(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ \tau_\phi = lc_T(\omega_2^2 - \omega_4^2) \\ \tau_\theta = lc_T(\omega_1^2 - \omega_3^2) \\ \tau_\psi = c_Q(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \\ \boldsymbol{\tau}_b^b = [\tau_\phi \quad \tau_\theta \quad \tau_\psi]^T \end{cases} \tag{1.3}$$

where $l$ represents the arm length of the drone, $c_T$ and $c_Q$ are the thrust and drag factors, $\omega_i \quad i = 1, 2, 3, 4$ are the propellers velocity.
The problems that we are going to cope with are: firstly to determine the inputs $u$ and $\boldsymbol{\tau}_b^b$, and consequently the propellers velocity, to track a trajectory; secondly to implement a planner that modify the trajectory accordingly with the enviroment.

# Chapter 2

# Packages and software

The code developed for the project ([3]) runs on ROS Noetic and uses the Gazebo enviroment for the simulation ( that interact with the ROS structure by means of wrappers) and Rviz for the visualization of the robot model and the sensors data. The elaboration of the data acquired during the simulation, through a rosbag, has been carried out thanks to Matlab.
In the following, we are going to introduce the implemented packages and the motivations behind the adopted choices.

## 2.1 my_scenario

The package my_scenario deals with spawning the drone model and launching the enviroment in which it will fly; in addition to this, the compilation of this package will generate the shared library: libmy_force_plugin.so. Including this plugin in the UAV model, it'll be applyed on the UAV a force of 1 N directed along the x direction of the world frame.
The scenario is represented in Fig.2.1, the main features are:

- take-off and landing platform: the former is green and the latter is red;
- solid obstacles: there are two models, that differs in their radius: 0.5 m and 1 m (Fig.2.2);
- empty obstacles: they consists in a window frame and they differ in their height. On the bottom left corner there is an Aruco marker, that is used by the planner to detect and cope with this type of obstacle (Fig.2.3);
- walls: they are used to delimit the space in which the UAV will fly.

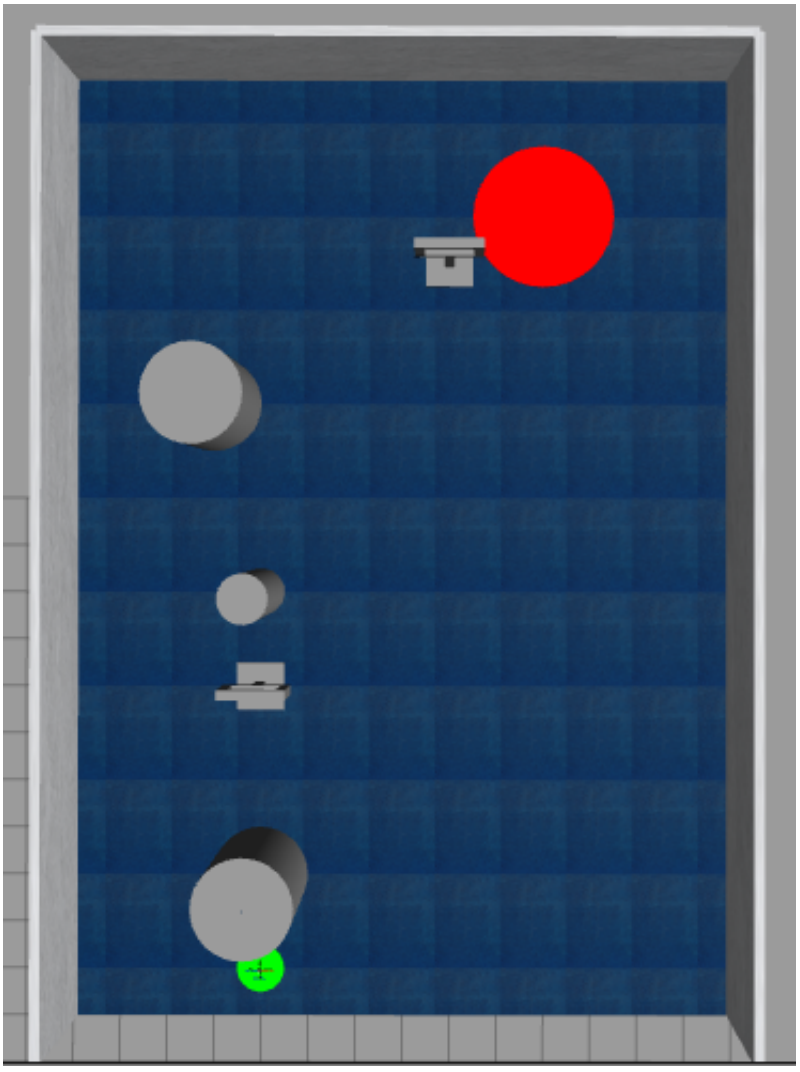

Figure 2.1: Enviroment

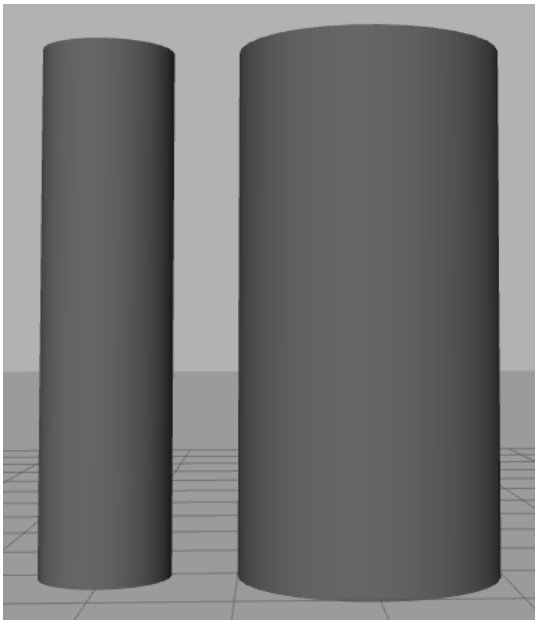

Figure 2.2: Solid Obstacles

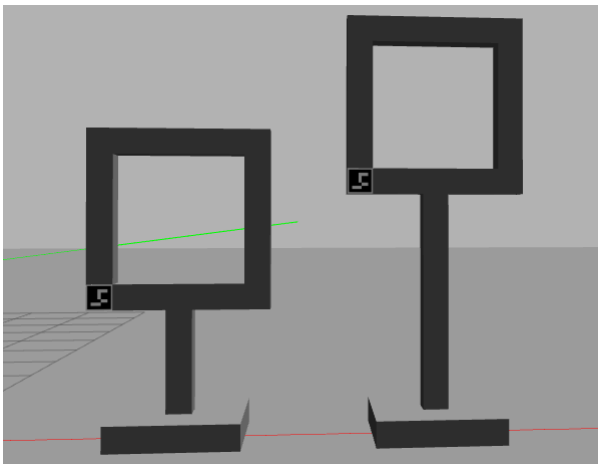

Figure 2.3: Empty Obstacles

## 2.2 custom_msg_pkg

This package contains the definition of a customized message for the communication between the low level control and the planner, whose structure is:

| int32 | ctrl_action |
|---|---|
| geometry_msgs/Vector3 | pos |
| geometry_msgs/Vector3 | vel |
| geometry_msgs/Vector3 | acc |
| float64 | yaw |
| float64 | dyaw |
| float64 | ddyaw |

The ctrl_action field is used by the planner to punctuate the different phases of the task execution; the other fields are used to communicate the desired trajectory for the linear part and for the yaw.

## 2.3 mav_control

This package implements the passivity-based control in [1]:

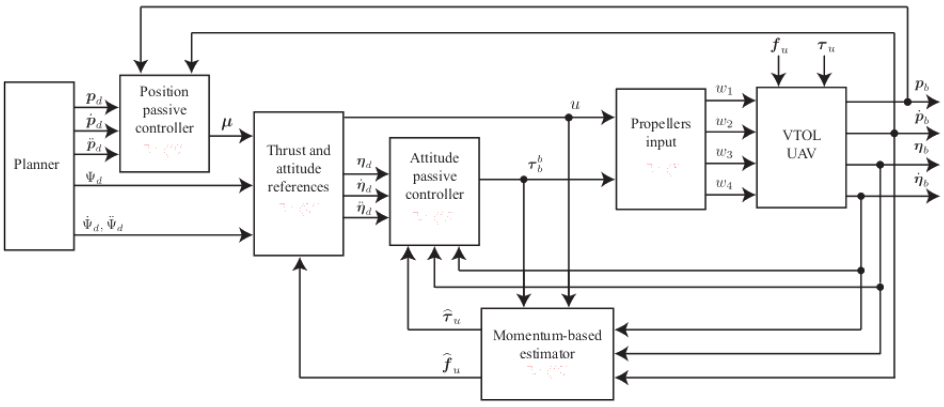


Figure 2.4: Control scheme

The planner is not part of the low level control and we'll be presented in the next section. We can distinguish two loops: the inner loop for the attitude control and the outer loop for the position control. There is also a momentum-based estimator: it is not necessary for the control, but it is usefull to improve the performance of the overall scheme when they are considered unmodelled dynamics and external disturbances.

This same structure is coded in a class called $MAVCTRL$ and for each block there is a member function that executes the corresponding task. There are also two other member functions used in the initial phases to test the correctness of the allocation matrix.

The communication between the planner and the controller happens on the topic */planner_des* and it is managed through a subscriber to this topic (the trajectory published on this topic is referred to the world NED frame); similarly, the communication between the controller and the UAV is carried out by a publisher on the topic */hummingbird/cmd/motor_vel*.

Along with the source code, in the package we can find a *conf* directory in wich there are the yaml files used to load the parameters for the controller. In particular:

- apc_gains.yaml: for the attitude passive control gains;
- ppc_gains.yaml: for the position passive control gains;
- mbe_gains.yaml: for the momentum-based estimator gains;
- uav_param.yaml: for the UAV parameters;

As requested, the UAV mass considered by the controller is 10% lighter than the actual one. The gains have been tuned by assigning a desired position one meter away from the current one and adjusting their value consequently:

| | |
|---|---|
| apc_gains | $\nu = 0.1$; $K_o = diag(0.4, 0.4, 0.8)$; $D_o = diag(0.1, 0.1, 0.4)$ |
| ppc_gains | $K_p = 3.3I$; $K_d = 6.5I$ |
| mbe_gains | $K_o = diag(10, 10, 10, 10, 10, 10)$; $D_o = diag(10, 10, 10, 100, 100, 100)$ |

## 2.4 mav_planner

The goal of the package *mav_planner* is to generate a desired trajectory for the UAV, on the base of a predefined set of waypoints, taking into account for the enviroment: it consists into the implementation of an online version of the Artificial Potential method [2], in which the force field is seen as an acceleration vector. In addition, the empty obstacle management is fulfilled thanks to the *aruco_ros* library. Also in this case, there is a class called $MAVPLN$ whose member functions make the needed calculation. The list of waypoints is managed like a FIFO container in order to simplify the insertion and remotion of new waypoints in case of need.

### 2.4.1 Implementation of the Artificial Potential method

Recall that the artificial potential method requires the evaluation of the attractive force, the repulsive force and the total potential. These calculations are carried out by three member functions that take into account the current position of the UAV, the current destination and the position of the obstacles: only the solid obstacles and the walls are considered as true obstacles, while the empty obstacles are treated in a different way, as we'll see later. About the solid obstacles, let's consider the case in which an obstacle is detected from the laser measurements: if the distance from the obstacle is less than a fixed threshold, then the relative measures are used to evaluate a certain number of points in which repulsive charges are "placed" .

It is known that this planning method presents a drawback: the local minima problem. In order to face this problem when it happens, there is a function that takes action when the total force is less than 0.02 and the potential is positive: it evaluates the distance between the UAV and the current destination and between the UAV and the nearest obstacle: if the former is smaller than the latter, then the repulsive force is neglected because there is no risk of collision; in the other case, a new waypoint is randomly generated taking into account the distance of the nearest obstacle.

### 2.4.2 Empty obstacle management

The empty obstacles are designed like window because, once they are detected, the UAV must pass through them. The detection is possible since the UAV is endowed with a camera, whose images are elaborated thanks to the *aruco_ros* library, and the empty obstacles present an AR-marker on the bottom left corner: a callback is called every time an AR-marker is detected. The empty obstacle management is designed so as to follow a sequence of steps, and the transition among them is entrusted to a set of boolean flags:

- *_emp_obs_distant*: it become true when the obstacle is detected and the next waypoint to reach is farther, with respect to the UAV, than the obstacle itself. A new waypoint is added to the list and it is located in front of the obstacle. In addition, from now on, the angular portion of the laser measures occupied by the obstacle is neglected, so as not to have a repulsive force generated by the empty obstacle itself;
- *_emp_obs_near*: when the obstacle is sufficiently near, the added waypoint is removed and a new waypoint is added, based on more accurate measures. Then, the next phase starts;
- *_emp_obs_approaching*: this phase is true as long as the UAV doesn't reach the front waypoint. Then a new way point is placed right behind the obstacle;
- *_emp_obs_crossing*: this is the last phase, during which the UAV crosses the obstacle. Once the waypoint is reached, the UAV returns to the original list of waypoints.

### 2.4.3 Take-off and landing

The take-off and the landing phases don't depend on the artificial potential method, in fact the planner simply gives as reference a desired constant pose and check for the UAV to arrive where indicated.

# Chapter 3

# Simulations

In the following, we are going to see the simulation results: the plots have been generated by means of Matlab and using the data recorded by a rosbag during the simulation. Project requirements:

- The mass in the controller is understimated by 10 %;
- An external force of 1N is continuously acting along the x-direction of the world inertia frame.

## 3.1 On the necessity of the estimator

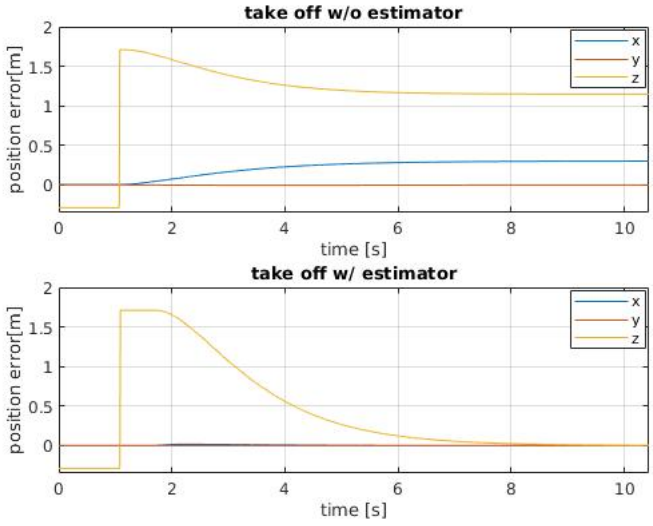


Figure 3.1: Take-off error without estimator (top) and with estimator (bottom)

In Fig.3.1 it is shown the position error during the take-off when the estimator is off and when it is on: the initial position is $[0 \quad 0 \quad 0]^T$ and the desired one is $[0 \quad 0 \quad -2]^T$ (NED frame). When the estimator isn't activated, there is a significant error along the vertical direction, due to the uncertainty on the mass, and an error along the x-direction, due to the external disturbance; in the other case, it is clear how the presence of the estimator improves the behaviour of the UAV, in fact we can notice that there is just a little bump along the x-direction during the initial istants of the take off and we can appreciate a zero steady-state error along all the directions.

For this reason, the task execution requires the presence of the estimator to cope with these problems and it will be considered active during the simulation.

## 3.2 Task execution

The main task consists into reaching a sequence of predefined waypoints (NED frame):

- p1: $[0.0 \quad -4.0 \quad -1.0]^T$;
- p2: $[0.0 \quad -9.5 \quad -1.0]^T$;
- p3: $[4.0 \quad -13.0 \quad -1.0]^T$;
- p4: $[6.0 \quad -16.0 \quad -1.5]^T$;

this list is dynamically modified by adding new waypoints when the UAV encounters an empty obstacle. Notice that there is no value for the yaw, because we have considered a zero constant referece for it, so as to simplify the empty obstacle management. The path traveled by the UAV to reach each of these waypoints is generated and modified, accordingly to the enviroment, by the planner. The resulting trajectory components are in in Fig.3.2 and the corresponding errors between the desired and the current pose are in Fig.3.3.

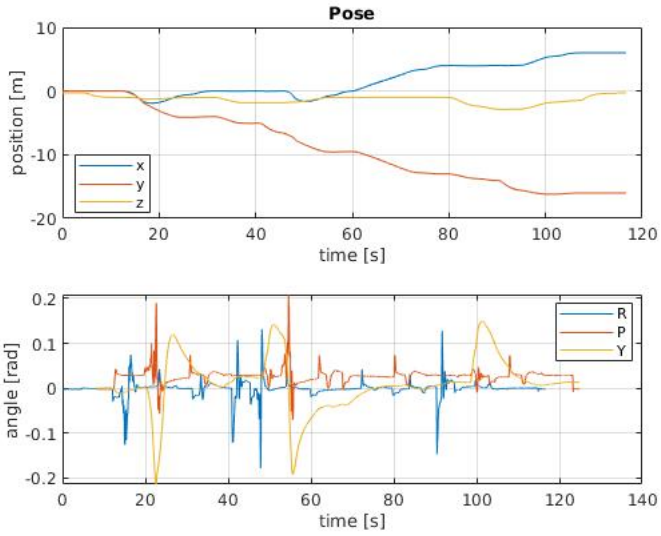


Figure 3.2: Trajectory of the UAV

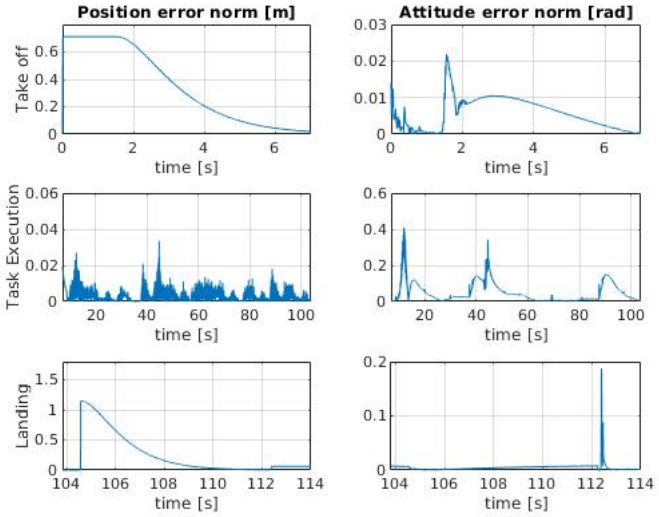


Figure 3.3: Trajectory errors of the UAV

The time evolution of the total force and the total potential is shown in Fig3.4.

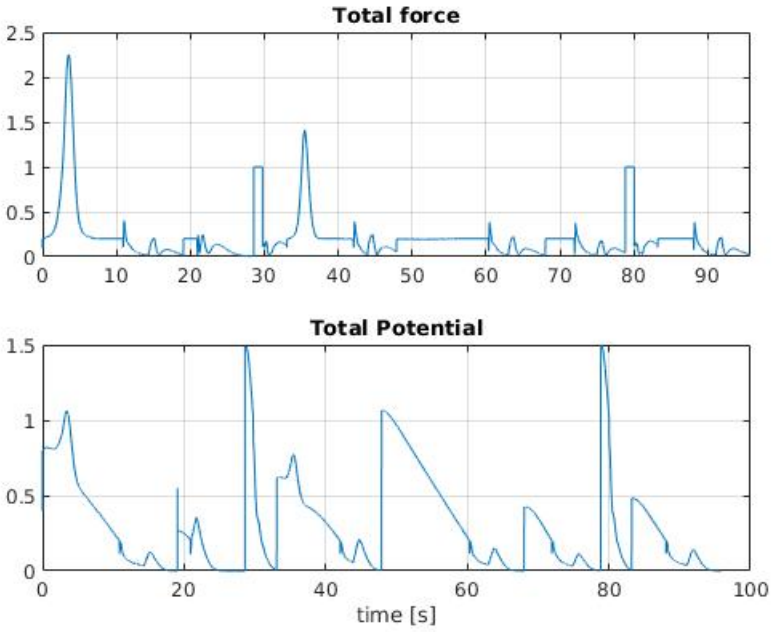


Figure 3.4: From the top: norm of the total force and total potential

Notice that the potential drops as the UAV approaches a waypoint. Another thing to point out is that because of the high forces generated during the motion, it has been added a saturation on the velocity of the UAV: the reason is that the desired velocity is the integral of the total force and even if there is no more repulsive force, which appears to be the main cause of huge total force, the desired velocity takes time to reduce its value and in the meanwhile the drone goes far away before reaching the goal position. Thanks to the saturation, the UAV has a fast recovery.

The estimated forces and torques during the motion are:

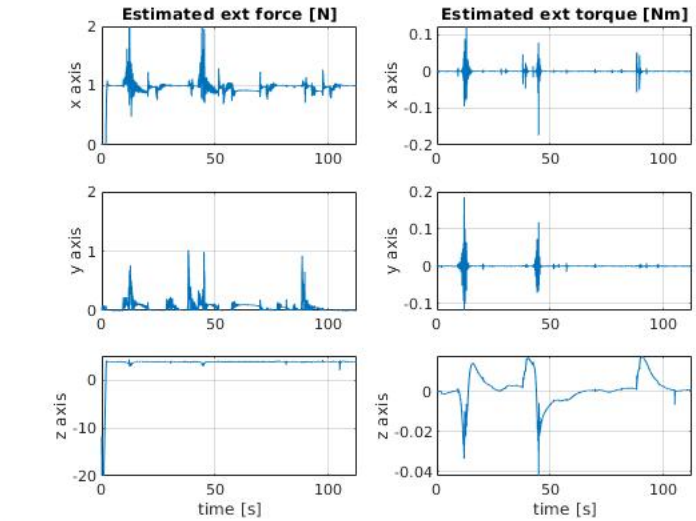


Figure 3.5: Estimated force and torque for each direction

If we consider their average, the estimated values are in accordance with wath we'd expect.

Finally, we consider the commanded motor velocities:

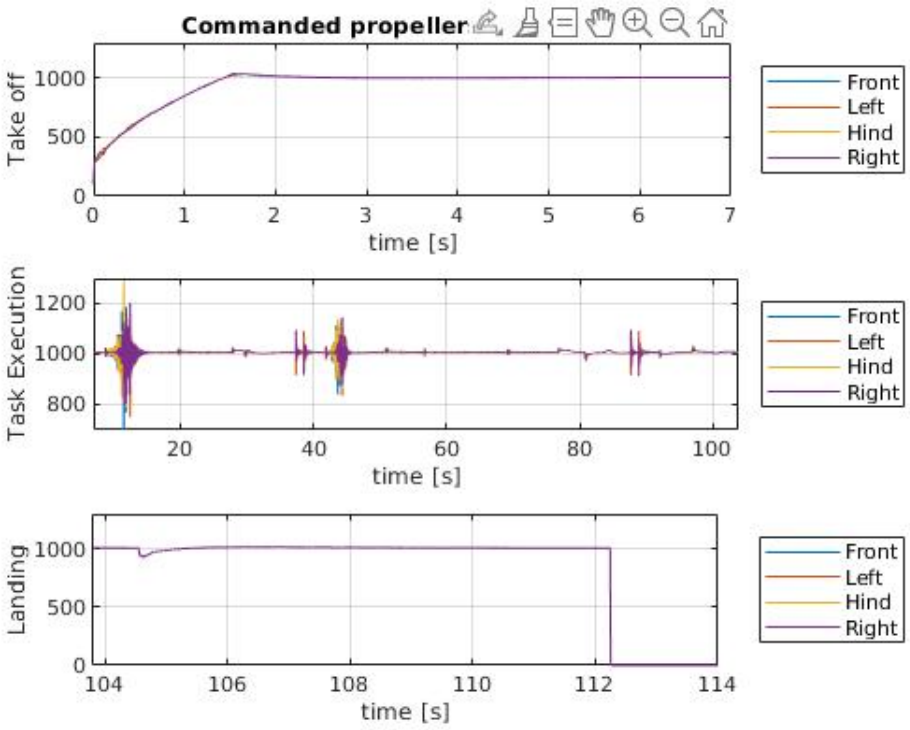


Figure 3.6: Velocity command generated by the planner

Notice that during the take off and the landing, the velocities are basically the same for all the propellers because it is just a vertical motion. During the landing, when the UAV is sufficiently near to the ground, the propellers are turned off and the UAV falls on the platform.

# Chapter 4

# Final considerations and future works

With this control the UAV is able to reach a set of predefined waypoints and to cope with different obstacles. We've seen that the estimator is not essential for the control, but its employment is suggested in order to improve the overall performances. Talking about the planner, some improvements can be done, for example about the way the empty obstacles are managed: in this project it is been neglected the relative rotation between the UAV and the empty obstacle, in fact it is assumed that there is no relative rotation between them. If this is not the case, the UAV may fail in crossing the obstacle because of a wrong pose estimation. To take into account also for this situation, it can be used the information about the pose from the *aruco_ros* library.

# Bibliography

[1] Ruggiero Fabio, Cacace Jonathan, Sadeghian Hamid, Lippiello Vincenzo. (2015). Passivity-based control of VToL UAVs with a momentum-based estimator of external wrench and unmodeled dynamics. Robotics and Autonomous Systems. 72. 139-151. 10.1016/j.robot.2015.05.006.

[2] *Robotics: Modelling , Planning and Control* - B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo – Springer, 2009

[3] *https* : *//github.com/PasqualeMarra/RL_FSR_project.git*