DIETI UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

TECHNICAL PROJECT FOR FIELD AND SERVICE ROBOTICS EXAM

Control of a VTOL UAV

Ciro Mazzocchi
M58/276

Anno Accademico 2019/2020

Contents

Chapter 1

Introduction

The aim of this project is to develop a system able to perform motion planning and control for a VTOL UAV. In particular, an AscTec Hummingbird quadcopter has been considered; however, the system can work with any other UAV, as long as its dynamic parameters are known, a 3D model is available, and a control through total propeller thrust and torques is provided.

In the following, all the project specification are summarized. Motion planning is performed via artificial potential method, where the force field is seen as a velocity vector. In order to avoid possible local minima, a virtual point-based algorithm has been implemented.

The control is achieved through a passivity-based hierarchical controller, with estimator of external wrench and unmodeled dynamics. The controller works at a fixed frequency of 1 kHz; moreover, it must be able to deal with uncertainty about the knowledge of UAV's dynamic parameters. In detail, the mass is underestimated by 10%, as well as the inertia parameters. In addition, a constant external force of 1 N is apply in each moment along one of the axis on the world frame.

It is assumed to know the exact position and orientation of the robot. The project has been developed in C++, using the ROS (Robot Operating System) framework, along with the RotorS MAV simulator [1], and the Gazebo physical engine.

Also, Plotjuggler for data processing and plotting. The project has been tested on Ubuntu 18.04 LTS, with ROS Melodic and Gazebo 9.0.

Chapter 2

Implementation

Quad-motor motion control is performed using a passivity-based control approach showed in [2]. In Figure 2.1 is reported a functional block diagram of controller.

To estimate external wrench (as wind) and unmodeled dynamics, a momentum-based estimator of first order has been developed.

Implementation of controllers is based on the *Publish/Subscriber* paradigm: each part of the controller reported in Figure 2.1 has been implemented as single node the receive and send message on following list of topic:

- */hummingbird/ground_truth/odometryNED*: on this topic is published pose and twist of drone
- */hummingbird/odometryNED/p*: on this topic is published position and speed of drone
- */hummingbird/odometryNED/eta*: on this topic is published orientation and angular speed of Euler angles
- */hummingbird/trajectory/p*: on this topic is published desired position, speed and acceleration of drone

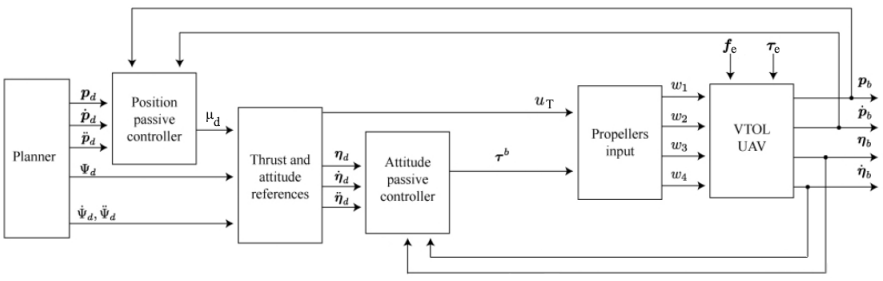

Figure 2.1: Passivity based controller

- */hummingbird/trajectory/yaw*: on this topic is published desired orientation of drone
- */hummingbird/estimator*: on this topic is published the estimation of external wrench
- */hummingbird/mu_hat*: on this topic is published the desired acceleration of drone
- */hummingbird/eta_reference*: on this topic is published the desired orientation, speed and acceleration of Euler angles
- */hummingbird/command/wrenchNED*: on this topic is published the thrust and torque to apply to drone

To pursue this aim a suitable custom message *State* has been defined with the following structure:

```
geometry\_msgs/Vector3 position
geometry\_msgs/Vector3 speed
geometry\_msgs/Vector3 acceleration
```

To guarantee a correct initialization of the estimator all nodes start 5 sample after the estimator node.

For this simulation, it is assumed that the environment has been mapped through a VI sensor, and through an object detection algorithm is possible to retrieve the position of object inside the map. For this example, the position of object are retrieved directly from *gazebo/model_states* topic.

2.1 Trajectory_node

The *trajectory_node* implements the planner block publishing p_d, $\dot{p_d}$, $\ddot{p_d}$ and yaw_d for each sample retrieved from odometry: in particular, the planned trajectory is published as *State* message and the desired yaw is published as $std\_msgs/Float64$ on */hummingbird/trajectory/p* and */hummingbird/trajectory/yaw*, respectively. The path planning problem is resolved using the artificial potential methods, where forces are seen as velocity vectors applied on the robot [3][pp. 546-551]. To retrieve the robot configuration, the Euler integration method is applied, leading to $q_{k+1} = q_k + T_s f_t(q_k)$, where T_s is the controller frequency. To retrieve the robot acceleration, instead, it has been implemented a numerical differentiator presented in [3][p. 381] and reported in Figure 2.2.

In order to tackle the local minima problem, i.e., $f_t = 0$ when $e(q) \neq 0$, a virtual obstacle-based algorithm has been implemented: when the planner detects that it is stucked in a local minima, an virtual object is placed on the drone position with a little displacement to make the local minimum a unstable point. In particular, the trapping point is placed along the orthogonal direction respect to the direction that connects the drone and the waypoint that it has to reach.

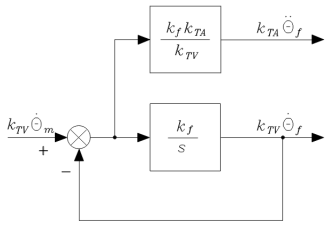

Figure 2.2: Filter implementation

Name	Type	Description
rate	double	controller frequency
rng_infl	double	Range of influence of an object
threshold	double	Minium distance from waypoint
k_att	double	attarctive force gain
k_rep	double	repulsive force gain
k_psi	double	cutoff frequency of orientation filter
pl_kf_d	double	cutoff frequency of differentiator time constant

Table 2.1: Calibration parameters of planner node.

To detect that the quad-rotor is stucked in a local minimum, a counter increments at each run of the node: if the norm of the speed vector or the norm of the position error are higher then predefined thresholds, then the timer is reset; if the timer arrives to a predefined time, this means that the quad-rotor is stucked in local minium.

The Table 2.1 contains all the configurable parameters for the planning nodes, in alphabetical order.

2.2 Outer_loop_node

The *Position passive controller* block has been implemented as a stand alone block called *outer_loop_node*: this node calculates the virtual control input as described in Equation (19) of [2]. K_p and D_p matrix are calculated as:

$$K_p = m\omega_n^2 I_3 \tag{2.1}$$

$$D_p = 2m\xi\omega_n I_3 \tag{2.2}$$

where m is mass of drone, ξ is the damping factor, ω_n is the natural frequency.

To avoid singularities in the calculus of ϕ_d and θ_d, node implements following inequalities $-0.9 \leq \mu \leq 0.9$.

The acceleration is published on */hummingbird/mu_hat* through a *geometry_msgs/Vector3* message.

The configurable parameters of this node are reported in Table 2.2.

Name	Type	Description
rate	double	Controller frequency
mass	double	Mass of drone
ol_xi	double	Damping factor
ol_wn	double	Natural frequency

Table 2.2: Calibration parameters of outer loop node.

Name	Type	Description
rate	double	Controller frequency
mass	double	Mass of drone
er_kf_d	double	Speed differentiator cutoff frequency
er_kf_d	double	Acceleration differentiator cutoff frequency

Table 2.3: Calibration parameters of eta reference node.

2.3 Eta_reference_node

The *Eta reference node* calculates desired trajectory for Euler angles using equations (21b) e (21c) of [2]. Speed and acceleration are calculated through a numerical differentiator reported in Figure 2.2 [3][p. 381]. The desired trajectory is published on */hummingbird/eta_reference* through a *quad_control/State* message. The configurable parameters of this node are reported in Table 2.3.

2.4 Inner_loop_node

The *Inner loop node* implements the *Attitude passive controller*, calculating thrust and torque using the equations (15) and (21a) of [2] at each sample. Total wrench is published on */hummingbird/command/wrenchNED* through a *geometry_msgs/Wrench* message.

K_o and D_o matrix are calculated as:

$$K_o = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \omega_n^2 \tag{2.3}$$

$$D_o = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} 2\xi\omega_n \tag{2.4}$$

where m is mass of drone, I_{ii} is the inertia matrix of i-axis, ξ is the damping factor, ω_n is the natural frequency.

The configurable parameters of this node are reported in Table 2.4.

Name	Type	Description
mass	double	Mass of drone
Ixx	double	x element of Moment of inertia
Iyy	double	y element of Moment of inertia
Izz	double	z element of Moment of inertia
il_xi	double	Damping factor
il_wn	double	Natural frequency
il_v	double	coupling parameter

Table 2.4: Calibration parameters of inner loop node.

Name	Type	Description
mass	double	Mass of drone
Ixx	double	x element of Moment of inertia
Iyy	double	y element of Moment of inertia
Izz	double	z element of Moment of inertia
we_k0	double	cutoff frequency

Table 2.5: Calibration parameters of estimator node.

2.5 Estimator_node

The *estimator_node* implements a momentum-based estimator of external wrench and unmodeled dynamics of the first order, and the estimated wrench is published on */hummingbird/estimator* as $geometry\_msgs/Wrench$.

The table 2.5 contains all the configurable parameters for the estimator node, in alphabetical order.

Chapter 3

Simulation

Simulation have been performed by using the model of an AscTec Hummingbird quad-copter with mass m and the inertia I of the vehicle are 1.2 kg and diag $(0.007, 0.007, 0.012)$ kgm^2, respectively. It can be launch through the launch file *start.launch* inside the launch folder. The controller node works with a frequency of $1kHz$, underestimating the quad-rotor parameters of a 10%. As external disturbance a force of $1N$ is applied on the quad-rotor along the x-axis.

In this simulation, the quad-rotor tracks a path formed by 4 points (defined in world NED Frame):

- $(0, 0, 0)$
- $(0, 0, -1)$
- $(10, 10, -2)$
- $(10, 10, -0.06)$

Along the path three column are placed in $(3, 4, 0)$, $(6, 3, 0)$, $(6, 6, 0)$ as represented in Figure 3.1.

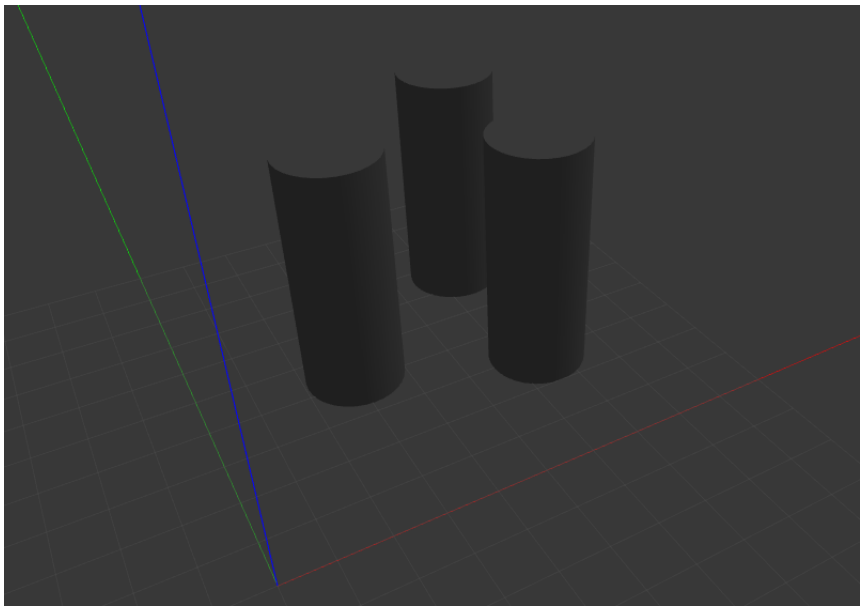

Figure 3.1: Arena used for the simulation

During the path, the quad-rotor remains stucked in a local minimum placed at the center of the three columns as showed in the Figure 3.2, but it is able to escape thanks to the virtual point-based algorithm.

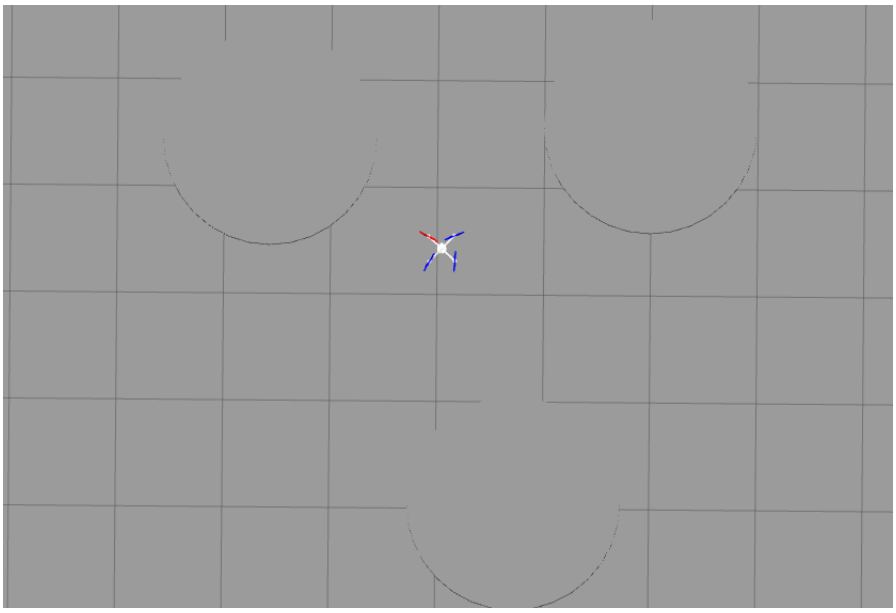

Figure 3.2: Hummingbird stuck in a local minima

The controller node and planner node parameters have been set as indicated in Table 3.1 and 3.2, respectively.

Name	**Type**
rate	1000
mass	0.062
Ixx	0.0063
Iyy	0.0063
Izz	0.0108
ol_xi	0.707
ol_wn	2
er_kf_d	3
er_kf_d	3
il_xi	0.707
il_wn	3
il_v	25
we_k0	3

Table 3.1: Parameters of controller.

Name	Type
rate	1000
rng_infl	3
threshold	0.2
k_att	0.5
k_rep	10
k_psi	0.5
pl_kf_d	10

Table 3.2: Parameters of planner node.

In Figure 3.3 and 3.4 are reported the norm of position and orientation error. Notice that the error remains contained to centimeter-scale, and the orientation controller remains contained to the 0.1 radiant.

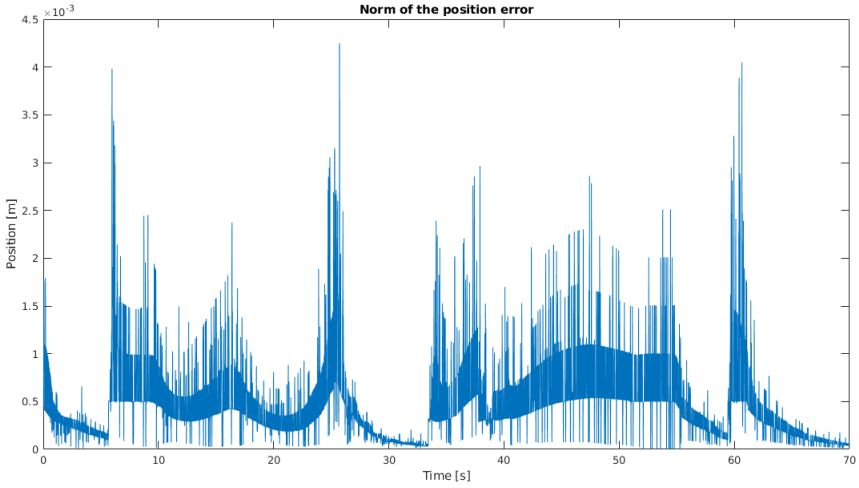

Figure 3.3: Norm of position error

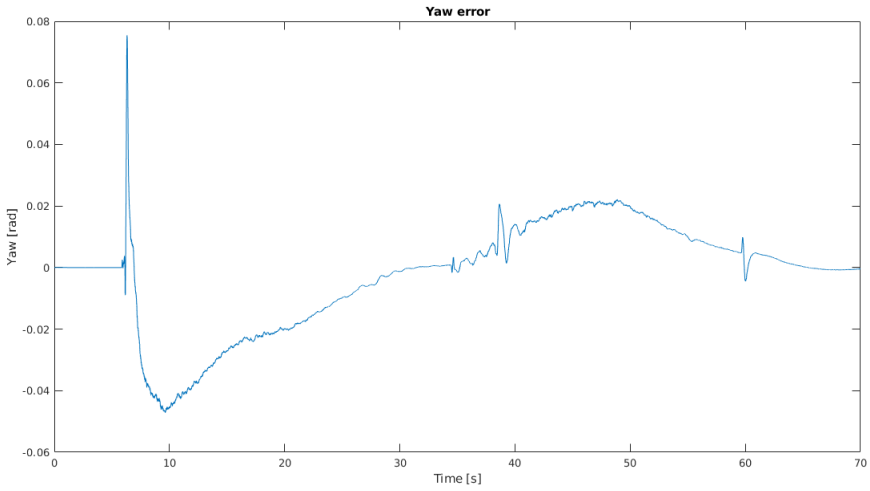

Figure 3.4: Norm of orientation error

In Figure 3.5 and 3.6 are reported the estimation of wrench. Notice that the estimator is able to compensate the uncertainties and to estimate the magnitude of wind that is applied along the x axis (in the NED frame along the y axis).

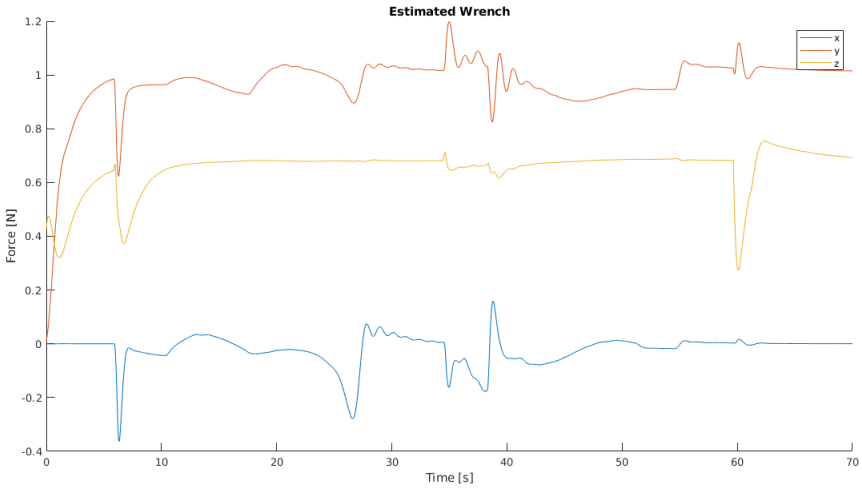

Figure 3.5: Wrench estimation - Force

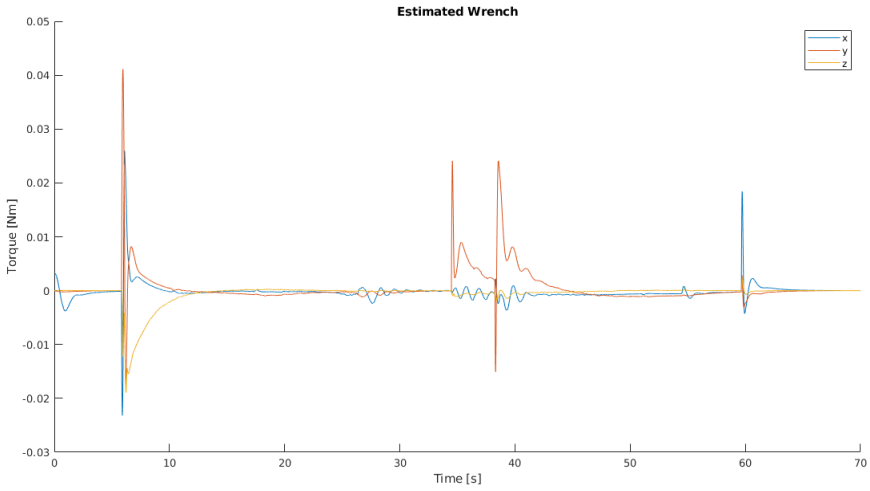

Figure 3.6: Wrench estimation - Torque

In Figure 3.7 and 3.8 are reported the thrust and torque generated by the controller.

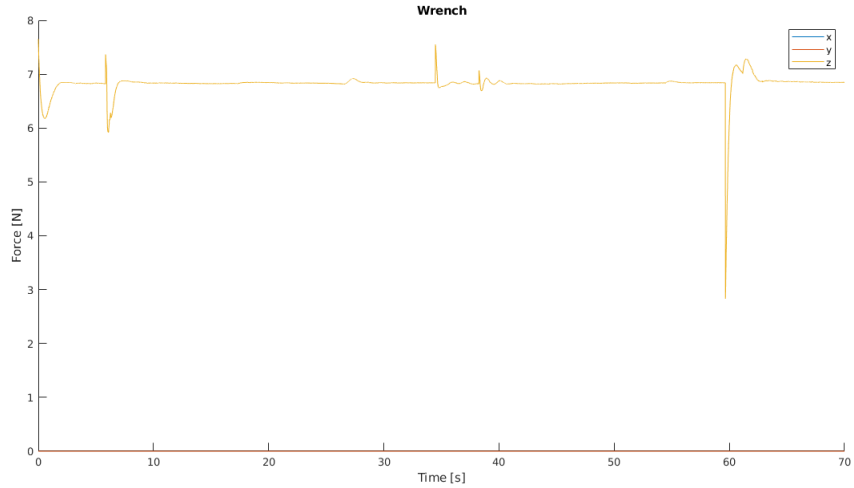

Figure 3.7: Control action - Thrust

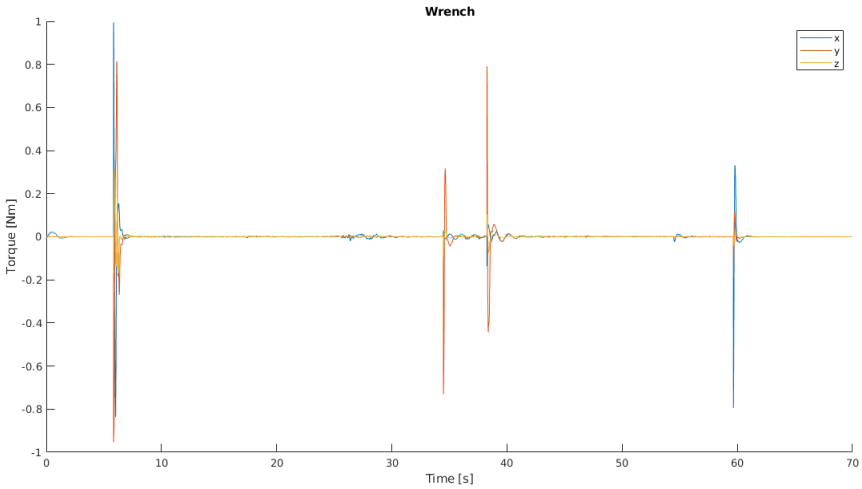

Figure 3.8: Control action - Torque

Bibliography

[1] Fadri Furrer, Michael Burri, Markus Achtelik, and Roland Siegwart. "Robot Operating System (ROS): The Complete Reference (Volume 1)". In: Springer International Publishing, 2016, pp. 595–625. ISBN: 978-3-319-26054-9.

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

[3] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics - Modelling, Planning and Control*. Springer, 2009. ISBN: 978-1-84628-642-1.

List of Figures

List of Tables