UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria dell'Automazione e Robotica

Field and Service Robotics

Geometric tracking control of a quadrotor UAV on SE(3) with Artificial Potential Planner

Academic Year 2021/2022

Student:
Pasquale Costanzo
P38000044

Contents

Introduction

The project was developed through ROS, Gazebo and C++. Hummingbird drone model of the "rotors" package was chosen. The choice is mainly due to the fact that the "rotors" package is very complete and has full compatibility with the Noetic version of ROS.[1].

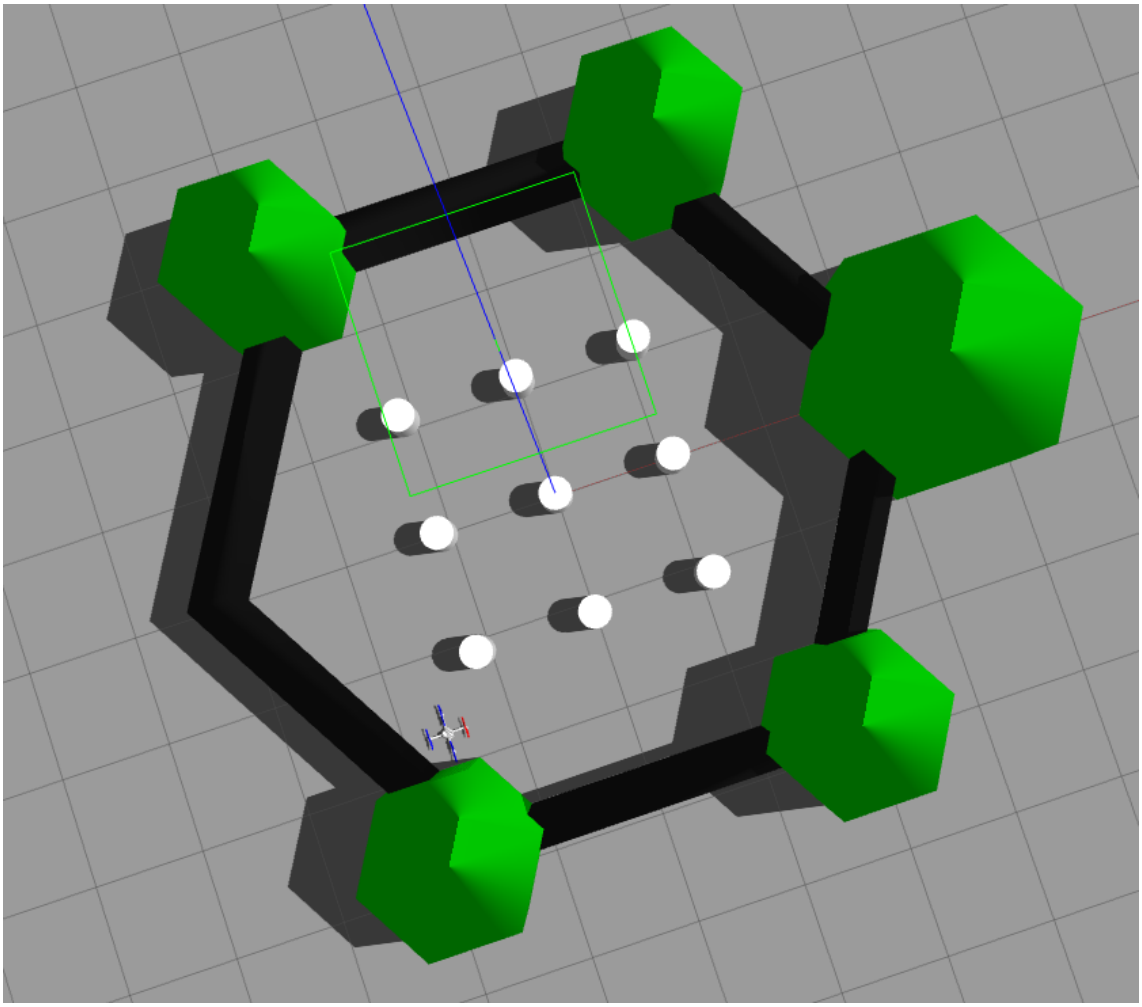

Figure 1: Scenario

The only sensor used during the simulation is imu to get the odometric information. So to avoid obstacles, a constant map has been created, since we will have no moving obstacles.

Chapter 1

Implementation

In order to be able to develop both APF planner and Geometric Tracking Controller, we need the position and speed of the drone. So to do this, an IMU has been placed that will write on the topic:

/hummingbird/odometry_sensor1/odometry

1.1 Artificial Potential Planner

We have two subscribers and one publisher. The first two are used to read the odometry and map data while the publisher is used to write the trajectory. The map was created with gmapping and turtlebot3. This is published by the map server node. The topic is */map*. To see if there are any obstacles in the map we need read the value:

- -1 Unknown;
- 0 Free;
- 1-100 Obstacles.

Unfortunately, the map created is in 2D,this implies the obvious limitations compared to a 3D map.Mainly on the z axis we have obstacles where they are present in x and y axis. This makes planning less precise but still efficient. Formula 12.11 and 12.15 in the book[3] were used to calculate the force of attraction and repulsion.

The book[3] recommends after calculating the total force, three approaches to planning . I did not use any of the three approaches . I used a simpler method where a desired position called called p_{sent} is sent:

$$p_{sent} = \frac{0.2 f_t}{\|f_t\|} + p$$

The result is functional, the drone avoids obstacles and arrive at the desired location.

p_{sent} is written on the topic *hummingbird/command/trajectory*.

1.2 Geometric Tracking Controller

The controller used is Geometric Tracking Controller[2]. We have one publisher and two subscriber. The first for publish motor speeds and the subscribers for read position and trajectory.

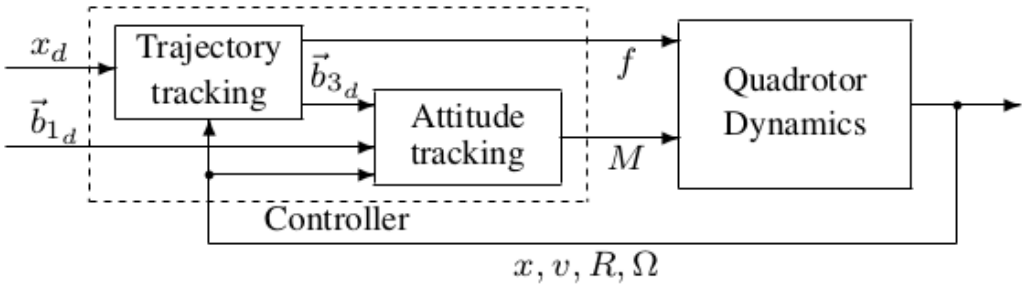

Figure 1.1: Controller Structure

The first part of this controller is calculate b_{3d}, b_{2d} and b_{1d}. But before we need :

$$Proj[b_{1d}] = \begin{bmatrix} cos(yaw_d) & sin(yaw_d) & 0 \end{bmatrix}$$

After i use equation(12) from the paper [2] and we have Rotation matrix desidered. In order to calculate e_ω we need ω_d, so i calculated $y\dot{a}w_d$ as:

$$y\dot{a}w_d = \frac{yaw_d(k) - yaw_d(k-1)}{Ts} \rightarrow \quad \omega_d = \begin{bmatrix} 0 & 0 & y\dot{a}w_d \end{bmatrix}$$

Now with equation (15) and (16) [2] we have torques and total thrust, we need to calculate the speeds of the individual motors by this relationship:

$$\begin{bmatrix} u_1^2 \\ u_2^2 \\ u_3^2 \\ u_4^2 \end{bmatrix} = \begin{bmatrix} 0 & l\tau_t & 0 & -l\tau_t \\ -l\tau_t & 0 & l\tau_t & 0 \\ \tau_t\tau_m & -\tau_t\tau_m & \tau_t\tau_m & -\tau_t\tau_m \\ \tau_t & \tau_t & \tau_t & \tau_t \end{bmatrix}^{-1} \begin{bmatrix} I_{xx} \\ I_{yy} \\ I_{zz} \\ 1 \end{bmatrix} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ f \end{bmatrix}$$

With τ_m is moment constant and τ_t is a motor constant, l length of arms and I is inertia matrix. Next we will do the square root of the absolute value of the

matrix u^2. After we can send to the topic $hummingbird/command/motor\_speed$ the rotor velocities.

Chapter 2

Simulation

For the Hummingbird drone parameters, I used the parameters in xacro file, see 2.1.

Name	Value
mass	0.716
l	0.17
I_{xx}	0.07
I_{yy}	0.07
I_{xx}	0.012
τ_t	8.54858e-06
τ_m	1.6e-2

Table 2.1: Hummingbird Parameter

While for the planner and controller I used these parameters.

Name	Value
ka	10
kr	200
r	0.3
frequency	1000hz

Table 2.2: Planner Parameter

Name	Value
kp	6
kv	4
kr	diag(0.7,0.7,0.35)
k_ω	diag(0.1,0.1,0.25)
frequency	1000hz

Table 2.3: Controller Parameter

The quadcopter start from the initial position $\begin{bmatrix}-1.6 & -1.8 & 0.1\end{bmatrix}$and arrive at the location $\begin{bmatrix}0.8 & 2.0 & 0.3\end{bmatrix}$. This choice is made because it has to overcome many obstacles to get from one point to another. We notice how the drone moves away from the obstacles and gets closer to the arrival point. The position error initially is kept high, this is due to the planner.Once you get to the end point the position error almost cancels out.

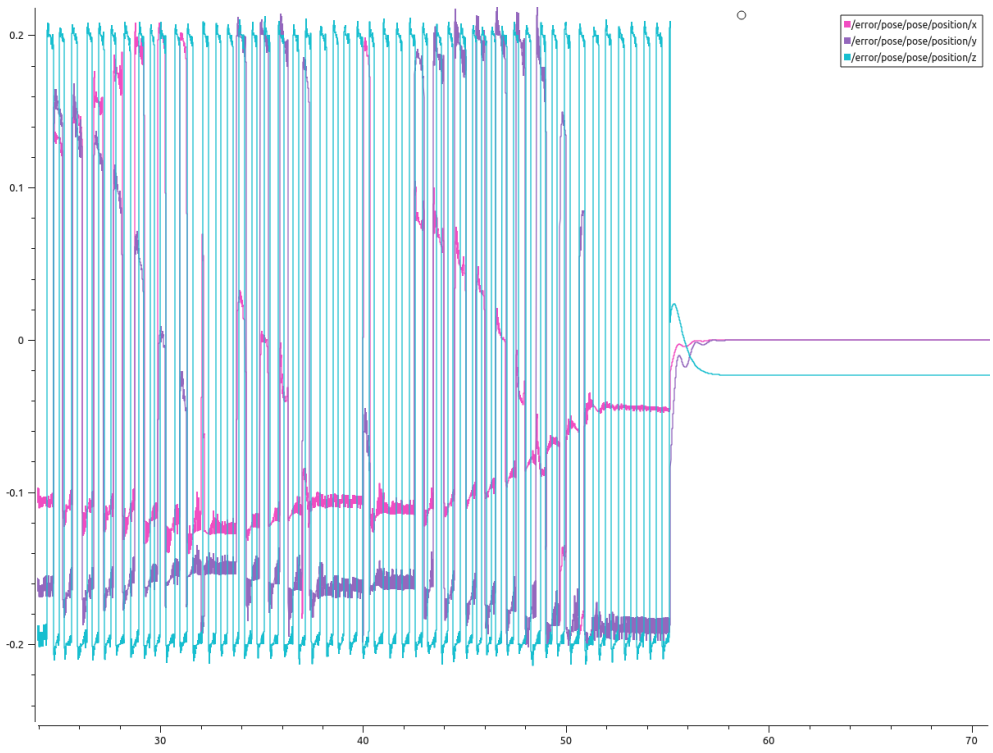

Figure 2.1: Position error

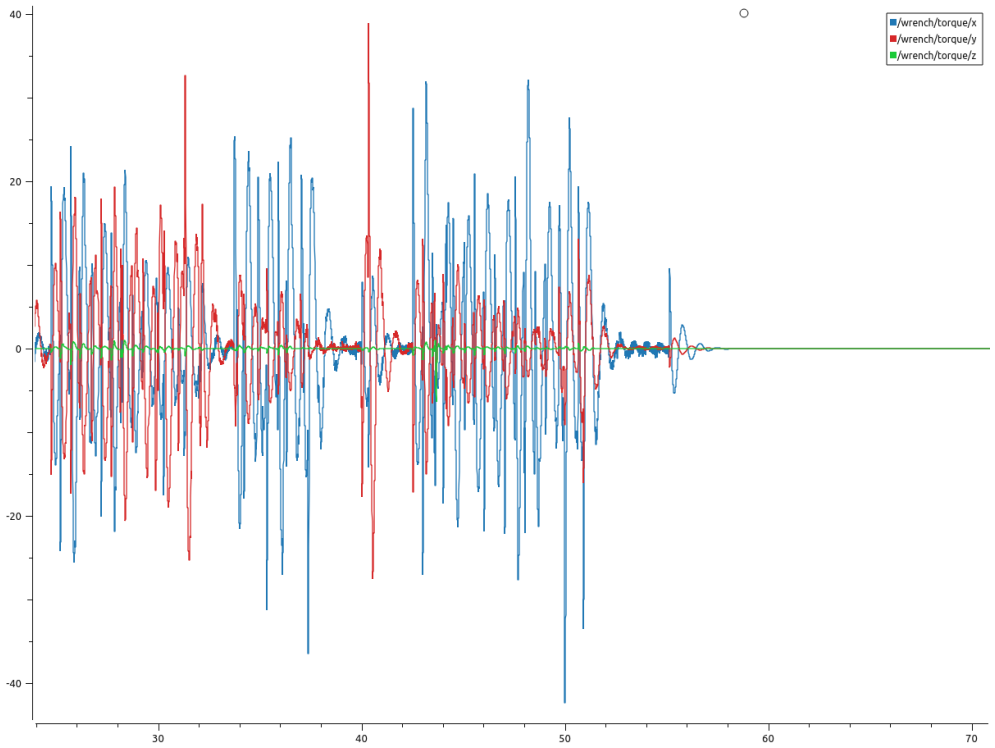

Figure 2.2: Torques

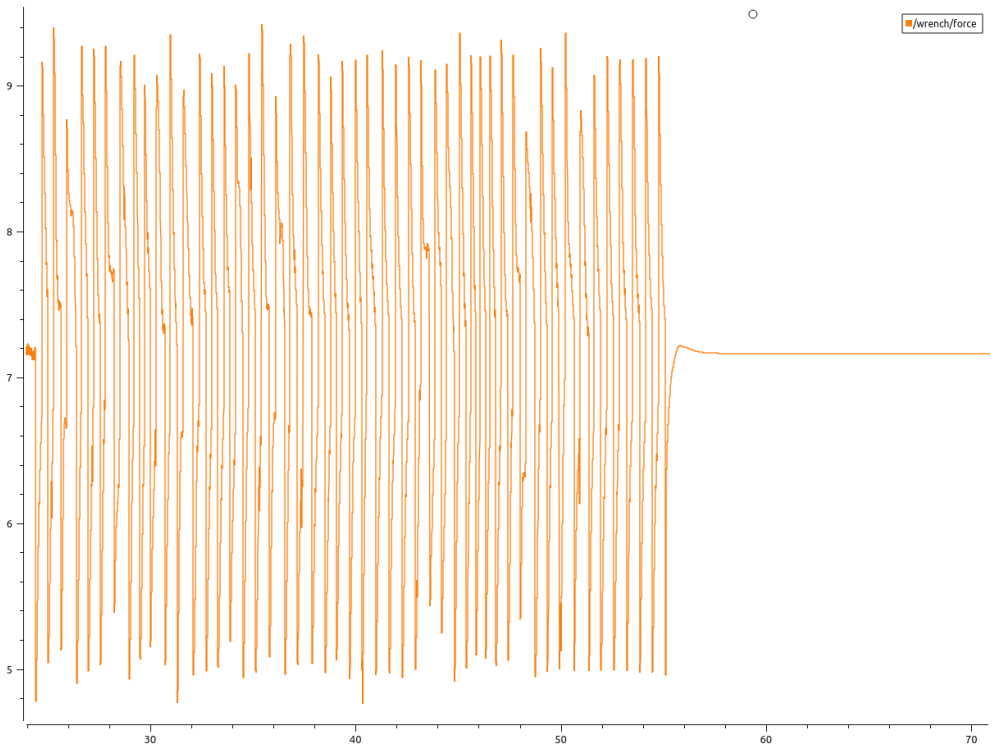

Figure 2.3: Total Thrust

Bibliography

[1] Fadri Furrer et al. "Robot Operating System (ROS)". In: *Studies Comp.Intelligence Volume Number:625* The Complete Reference (Volume 1).978-3-319-26052-5 (2016). ISBN:978-3-319-26052-5, Chapter 23.

[2] Taeyoung Lee, Melvin Leok, and N. Harris McClamroch. "Geometric tracking control of a quadrotor UAV on SE(3)". In: *49th IEEE Conference on Decision and Control (CDC)*. 2010, pp. 5420–5425. DOI: 10.1109/CDC.2010.5717652.

[3] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. 1st. Springer Publishing Company, Incorporated, 2008. ISBN: 1846286417.