Homework 3- Robot sensors and simple path planning.

Carlos Hansen Mendoza

October 21, 2019

1 Introduction

The purpose of this homework is that after controlling a ground robot to understand the difference in controlling a drone through a basic controller (P-controller) in two situations, first, following a series of waypoints and second follow the same waypoints while maintaining an specific orientation in each section of the trajectory.

2 Task 01

For these task the drone was simulated flying with an arbitrary yaw angle, with the only intention to reach the waypoints (0,6,2), (6,6,2), (6,0,2), (0,0,2).

During the experiment the X,Y, Z poses of the robot and its distance to the target waypoint was saved into a *.csv file for later plotting with Matlab.

The controller used for this task is located in the file: `task01.py`

For this task the actual trajectory of the robot can be seen in Figure 1.

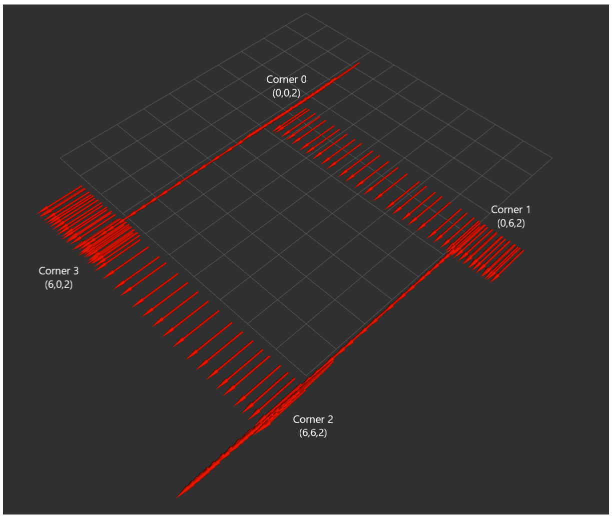

Figure 1: Poses of the robot captured using the RVIZ tool with a PD controller of $kp_x = kp_y = 0.05$ and a $kp_z = 0.2$

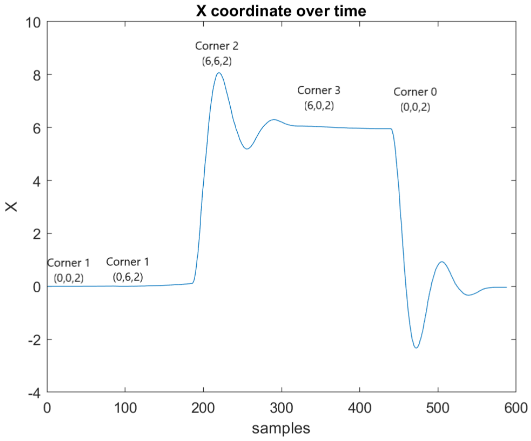

Figure 2: Plot of the X coordinate over time with a PD controller of $kp_x = 0.05$

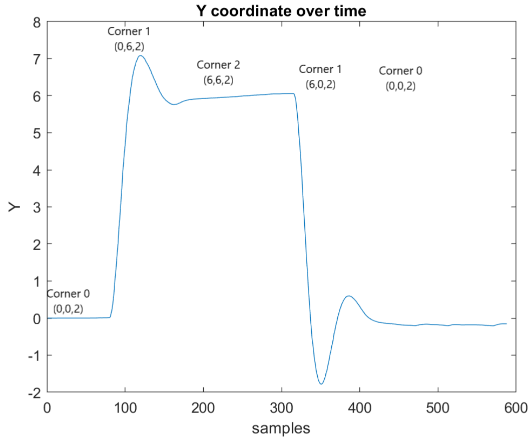

Figure 3: Plot of the Y coordinate over time with a PD controller of $kp_y = 0.05$

One point that is clearly noticeable in the trajectory Fig 1 and in each of the plot of the X-axis (Fig 2) and Y-axis (Fig 3) is the overshot of the controller because the movement is along one axis at the time this is clearly noticeable in all of the plots. And in terms of a value of overshoot in each attempt its clearly measurable in Fig 4

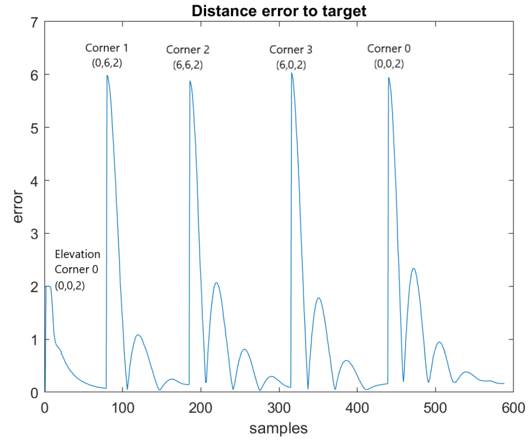

Figure 4: Plot of the Distance error to target over time with a PD controller of $kp_x = kp_y = 0.05$ and a $kp_z = 0.2$

In the case of the Z-axis (Fig 5) the controller slowly tries to reach the target point during the whole simulation.

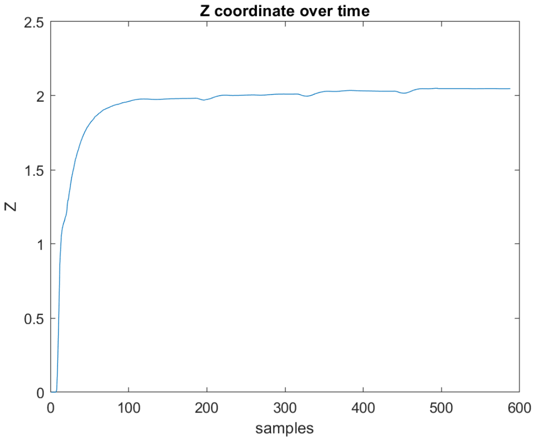

Figure 5: Plot of the Z coordinate over time with a PD controller of $kp_z = 0.2$

3 Task 02

For these task the drone was simulated flying with an specific yaw angle, with the intention to reach the waypoints (0,6,2), (6,6,2), (6,0,2), (0,0,2) with such yaw angle.

The desired yaw angled can be seen in the table 1

During the experiment the X,Y, Z poses of the robot and its distance to the target waypoint was saved into a *.csv file for later plotting with Matlab.

The controller used for this task is located in the file: `task02.py`

For this task the actual trajectory of the robot can be seen in Figure 6 and as difference of the one in the Task 1 as the drone is rotating the overshoot do not happen only in the direction of movement.

The overshoot mention can be noticed also in the plot of the X-axis (Fig 7) and Y-axis (Fig 8).

In the case of the Z-axis (Fig 9) the controller slowly tries to reach the target point during the whole simulation.

Related to the orientation for each trajectory the controller do a pretty good job in achieving the desired orientation as it can be seen in Figure 10.

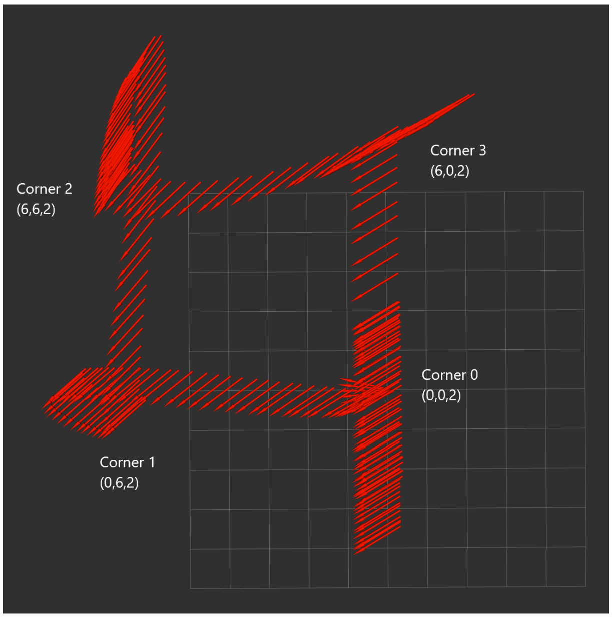

Figure 6: Poses of the robot captured using the RVIZ tool with a PD controller of $kp_x = kp_y = 0.05$ and a $kp_z = 0.2$ trying to maintain a certain orientation with a PD controller of $kp_\psi = 0.2$

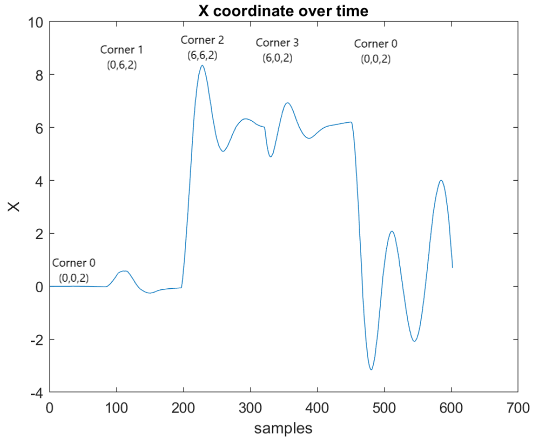

Figure 7: Plot of the X coordinate over time with a PD controller of $kp_x = 0.05$

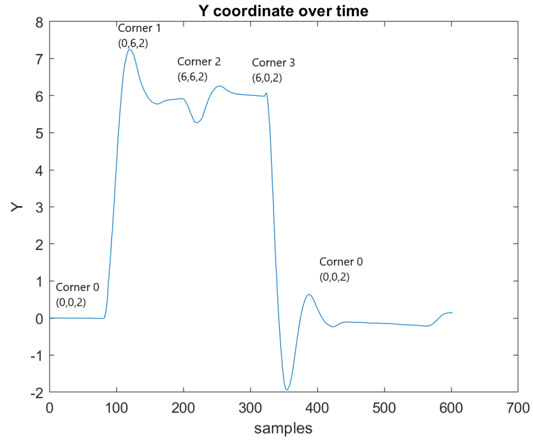

Figure 8: Plot of the Y coordinate over time with a PD controller of $kp_x = 0.05$

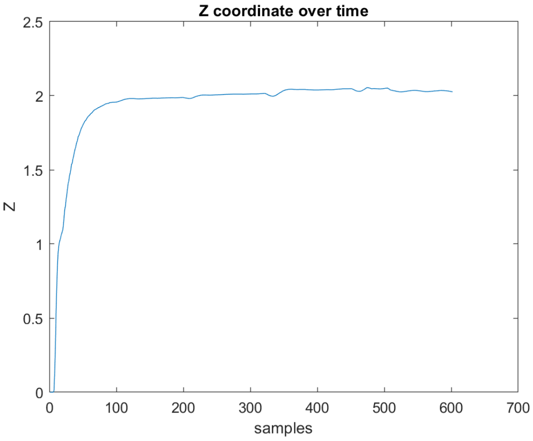

Figure 9: Plot of the Z coordinate over time with a PD controller of $kp_z = 0.2$

Initial Corner	End Corner	ψ
(0,0,2)	(0,6,2)	30 deg
(0,6,2)	(6,6,2)	45 deg
(6,6,2)	(6,0,2)	60 deg
(6,0,2)	(0,0,2)	30 deg

Table 1: Desired angles

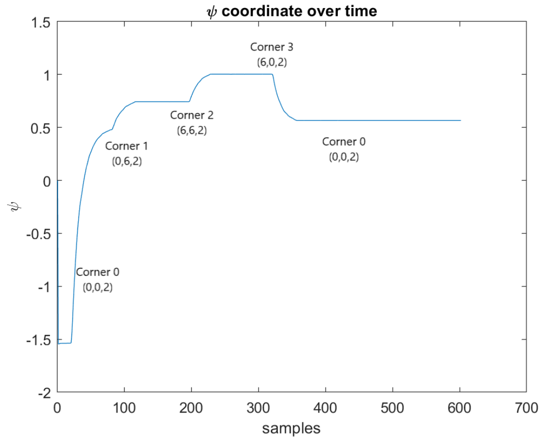

Figure 10: Orientation of the robot in radians whiile. The Proportional constant for this controller was $kp_\psi = 0.2$

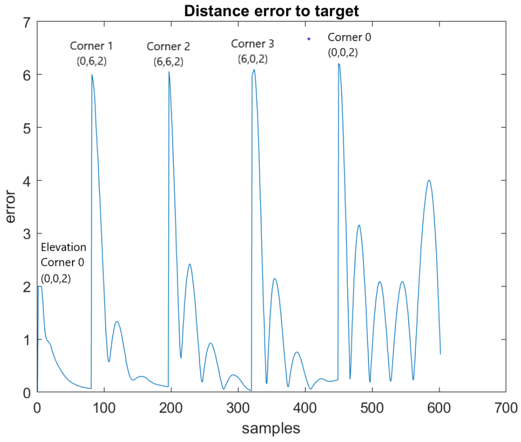

Figure 11: Plot of the Distance error to target over time with a PD controller of $kp_x = kp_y = 0.05$, $kp_z = 0.2$ and $kp_\psi = 0.2$

The error to the desired waypoint can be seen in Figure 11

An error that I could not solve lies in the controller of which point should the drone go, once the drone has given a complete cycle the motion controller diverges causing the drone overshooting every time more and more.