# Aarhus University MSc Course Project Control of Mobile Robots (AY 2019-20)

S. L. Skovgaard
dept. of Engineering (of Aff.)
Aarhus Unitversity (of Aff.)
Aarhus, Denmark
201401682@post.au.dk

Abstract—The article will document how to develop control software for UAV's and simulate the controller using the popular Gazebo software.

## I. Introduction

The main focus of this paper will be to explain and document the control software used to control a UAV robot. The software makes use of the robot operation system ROS. The paper will explain theory, code, simulation and results with accompanying figures and plots. The main objective of this project is to get a UAV to autonomously navigate to four coordinate-points.

## II. Theory

### A. The Robot Operating System

In order to develop software for robots in a structured and easy way, the robot operating system(ROS) [1] has been used for this project. ROS is based upon the subscriber-publisher pattern where nodes is able to publish data as topics and subscribe to specific topics. In this project a node has been developed that is a subscriber and publisher which enables the node to: gather data, make calculations using this data and the publish new data. Furthermore a another node has been made solely for the purpose of gathering data and displaying this data using different plots.

### B. Parrot Bebop2 drone

In order for the UAV to be able navigate to the given coordinates, a closed-loop control algorithm has to be developed in this case a p-controller is used to control the UAV. The UAV is a Parrot Bebob2 and is controlled using the package "bebop_autonomy". By using this package it is possible to send velocity and angular velocity commands to a ROS topic and thereby control the speed and orientation of the drone.

The p-controller developed in this project works by calculating the distance from the UAV to the currently active goal and use this as the error for the controller. A constant is multiplied to the error and the product of this will be used to set the velocity of the UAV.

### C. Inertial frame and body frame

Two important aspects of UAV control software is the inertial frame and body frame.

Intertial frame: is an earth-fixed coordinates system. It is also sometimes described as a north-east-down or NED reference frame, where north is the inertial x direction, east is the inertial y direction and down is the inertial z direction [3].

Body frame: is a coordinates system fixed to the body of the vehicle with its origin at the center of mass, x-axis pointing along the fuselage reference line, z-axis pointing down perpendicular to x and y-axis. The coordinate system rotates with the vehicle.

As a task for this project the Bebop2 drone must navigate between the goal-coordinates with different fixed yaw angles. In order for the controller to work as intended a transformation of the inertial-frame coordinates system to the body frame coordinate-system is needed. This is done by rotating inertial frame three times. First around z-axis(yaw), then y-axis(pitch) of the new coordinate system and lastly around the x-axis(roll).

### D. Gazebo

Gazebo is simulation environment that works together with ROS and is therefore suitable as a simulation tool for this project. Parrot provides the Bebop2 UAV as a downloadable package for Gazebo and this package is used for this project.

## III. Python code

The code used for this project is built using the rospy package which is a Python library. Roscpp is also available if C++ is preferred.

As mentioned earlier in this article, ROS makes use of the publisher/subscribre pattern. A single node containing several classes have been made for this project. The classes are: p-controller, takeoff and plot_maker.

P-controller: The responsibility of p-controller is handle the calculations and navigation of the UAV. The error or the euclidean distance to the goal-coordinates is calculated by first calculating the distance in the x-plane then the y-plane and lastly the z-plane. This results in three different distances which all are used as the error in each axis. The

error are multiplied a constant kp and lastly the product of this is used to set the velocity in each direction, x, y and z.

Takeoff: As the name suggests this class handles the takeoff. It's a simple class that makes sure the drone has performed a complete takeoff sequence and when this is done it no longer has any responsibility.

Plot_maker: This class subscribes to the same topic as the p-controller class, /bebob/odom. It stores all values obtained from odometry and when the UAV has been to all goal-coordinates, displays a number of plots which will be discussed later in the article.

One task for this project is to navigate to the four goals with a set yaw angle, this however haven't been completed here as the author has no idea how the rotation matrices are supposed to be programmed.

## IV. Results and discussion

To confirm that the control software works as intended the UAV has been observed through the program rviz. This program makes it possible to visualize what the different sensors on the UAV sees. On figure 1 the odometry data can be seen in rviz. The red arrows indicate the heading of the robot and is plottet where ever the robot is in space. It can be seen that the UAV goes to the four goal coordinates. However there is a large amount of overshoot. The four coordinates are: (0,0,2), (0,6,2), (6,6,2) and (6,0,2).

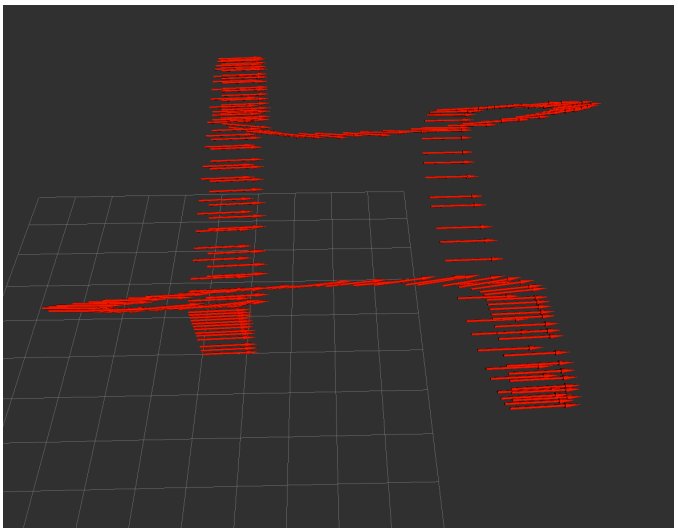

Fig. 1. Odemetry visualized through rviz

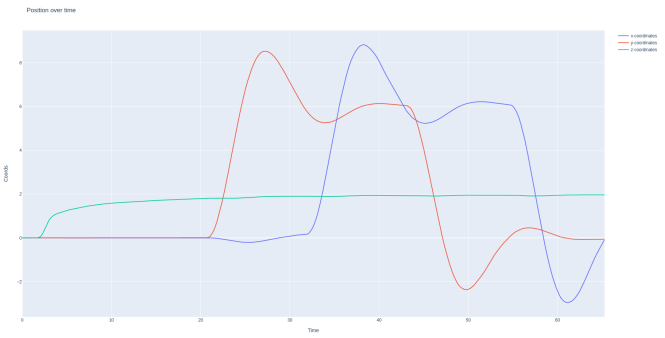


Fig. 2. x, y and z coordinates over time

The overshoot can also be seen on figure 2. Here there are three line: blue, red and green which each represents either x, y or z coordinates. The overshoot is directly controlled by the constant kp and can be lowered by lowering the value of kp, though this will also make the UAV slower, so there are pros and cons to consider when choosing kp. Here it is set to 0.1.

The error or the distance to the target over time can be seen below on figure 3. Here it can be seen that whenever the UAV reaches its goal there is a huge spike in error as the currently active goal is changed to the next goal. The overshoot is also visible in between the high narrow peaks.

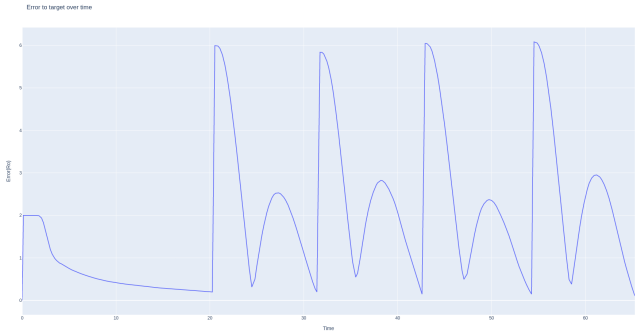


Fig. 3. Distance to target over time

As written further up in the article, the second task has not been completed.

## V. Conclusion

The main objective for this project has been to develop control software for a UAV so it would be able to navigate to four coordinate-points in 3D space. The plots shown during the Results and discussion section have clearly shown how the UAV reaches the four points. Furthermore different tools for analysing the UAV data have been used such as rviz. Task 2 haven't been completed because of time constraints and would have high priority for future work.

## References


[1] http://wiki.ros.org/, date: 3/10/2020

[2] Erdal Kaycan, "Control Of Mobile Robots, week 3: Ground Robot models", 16th of September 2020

[3] RANDAL W. BEARD and TIMOTHY W. McLAIN, SMALL UNMANNED AIRCRAFT Theory and Practice, 2012