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 ground robots 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 Husky 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 ground robot to firstly go to a certain target, secondly follow a linear line and lastly follow a exponentially increasing line.

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 both a subscriber and publisher which enables the node to: gather data, make calculations using this data and the publish new data.

B. Control Theory

In order for the robot to be able to reach it's destination target a closed-loop control algorithm has to be developed. For a ground robot the linear velocity is defined by the variable v and angular velocity is defined by the variable ω. These variables will be mentioned later in the theory section. Three different variables is needed to get a stable control algorithm: ρ, $\alpha$and β. These can be derived in a geometric manor by looking at figure 1 [2]. The following equations can be then derived [2]:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2} \tag{1}$$

$$\alpha = -\Theta + \text{atan2}(\Delta y, \Delta x) \tag{2}$$

$$\beta = -\Theta - \alpha \tag{3}$$

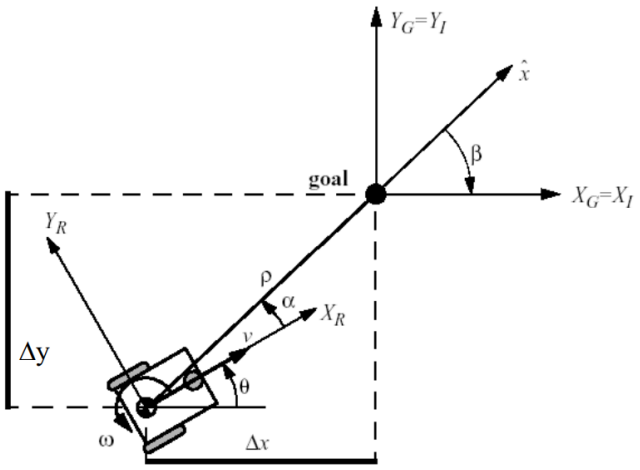

Fig. 1. Overview of ground robot and goal

It can furthermore be shown that with the following equations the control algorithm will drive the robot to the desired location.

$$v = k_\rho \rho \tag{4}$$

$$\omega = k_\alpha + k_\beta \beta \tag{5}$$

By utilizing these equations we can develop a stable closed-loop control algorithm for the ground robot if the controller coefficients also adhere to the following constraints [2]: $k_\rho > 0$, $k_\beta < 0$ and $k_\alpha - k\rho > 0$

C. Gazebo

Gazebo is simulation environment that works together with ROS and is therefore suitable as a simulation tool for this project. The ground robot used for the simulation is Clearpath's HUSKY robot. Luckily Clearpath provides the HUSKY robot 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. The Python program made makes use of a single node to act as both a publisher and subscriber. This way it's possible to subscribe to location data (x and y coordinates) from either the HUSKY robot or Gazebo, process the data and push new data to the robot. To get the most accurate position óf the node subscribes to Gazebo/model_states which gives the coordinates of the robot according to the global coordinate system in Gazebo. If this was supposed to be deployed in a real world scenario, this approach would not be feasible as the data would not exist. Odemetry from the robot would be used instead.

When the location data is pulled from the gazebo/model_states topic, x and y coordinates are extracted and used for the calculations discussed in II. To get the Theta angle quaternion coordinates are also extracted and a quaternion_to_euler function is used to get yaw (Theta) of the HUSKY robot.

When the calculations are done new velocity and angular velocity is published to the HUSKY robot. The program will then loop and do the calculations again.

IV. Results and discussion

To confirm that the control software works as intended two different plots have been made for each task. A plot that shows the desired position and the actual position of the robot versus the distance(or error) and a plot of the desired target over time. These will be presented and discussed in this section.

To get the most consistent results the robot initial position has been kept the same throughout all three different controllers. This means that Gazebo has been reset between each run and the robots initial position is (0,0).

The first plot show the most simple task. The robot must go to a stationary target, in this case (10,10). The values of k_ρ, k_α and k_β is 0.5, 0.5 and -0.05 respectively. These values has been picked based on the theory discussed in section II. By observing the plot it appears that the robot indeed does go towards the desired target. This can be seen on figure 2.

There is a small steady state error in the sense that the robot never reach the target entirely. Because the distance decreases so does the velocity meaning that the robot will slow down the closer it gets to the desired target. Eventually when time goes towards infinity it will end up on the target. For this project the error is acceptable.

The error plot shows that the error goes down over time as the robot is nearing the target destination.

The next task concerns the robot following a straight line in the form of:

$$y = 2 * x \tag{6}$$

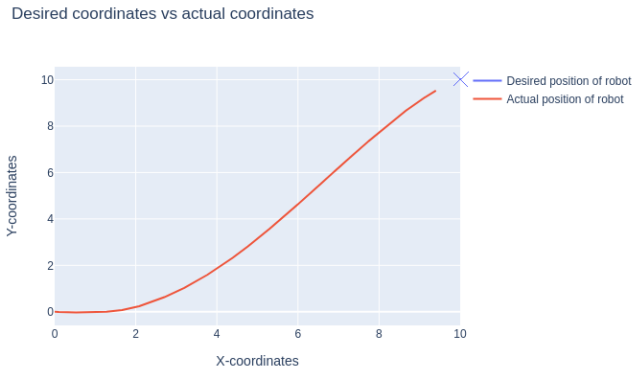

Fig. 2. Go to desired target (10,10)

The controller coefficients has been kept the same as before. This time we see some overshoot when the robot starts up. This can be seen on figure 3.

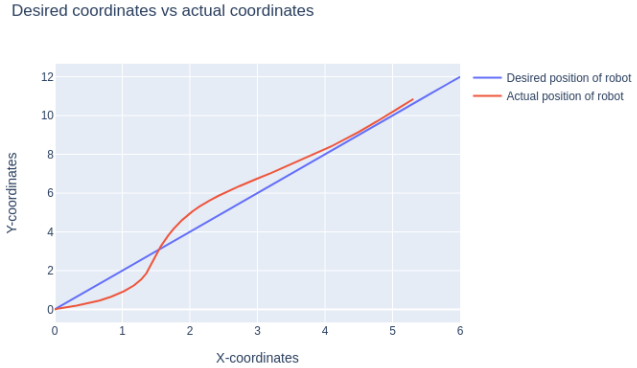

Fig. 3. First order line following position plot

The error over time can be seen on figure 4. Here it can be observed that the error is 0 at t=0. This makes sense because the starts out in (0,0). After some time the desired target moves further away from the origin and the robot has to catch up. At t=4 the robot has reached a steady state and follows the desired target with an error of around 1.3.

The last plots shows the robot following a exponentially increasing line in the form of:

$$y = 0.2 * x^2 \tag{7}$$

The controller coefficients have been kept the same as before. This time we see a small amount of overshoot again. It can also be seen that the robot is trailing behind the line (this is also visible in the error plot). This is caused by the error decreasing and therefore decreasing the velocity. The positional plot for the second order line can be seen on figure 5. The error plot is figure 6

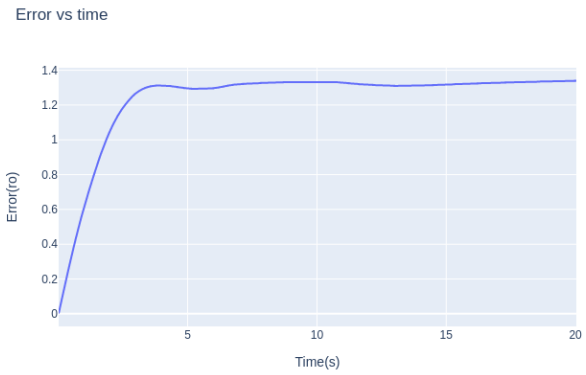

Fig. 4. First order line following error plot

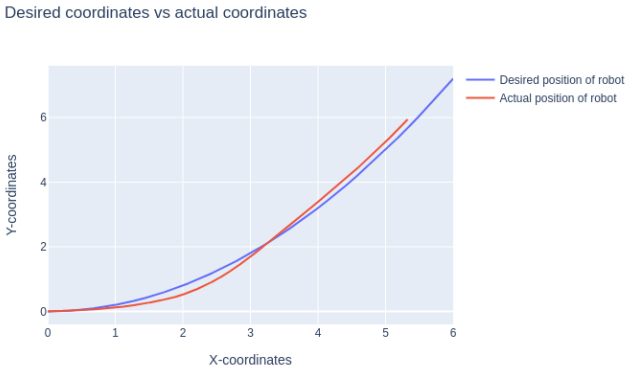

Fig. 5. Second order line following positional plot

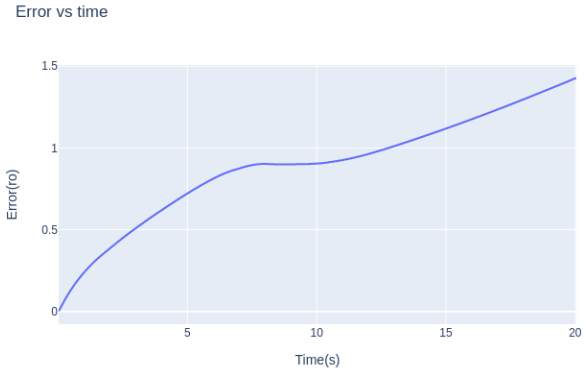

Fig. 6. Second order line following error plot

To further explore the controllers the value of k_ρ has been changed to 5 to see what happens when we deploy an unstable controller. k_α and k_β has been kept the same. The positional plot can be seen on 7. It clearly shows a great increase in overshoot to such a degree that the system is unstable. The same can be seen in the error plot on figure 8.

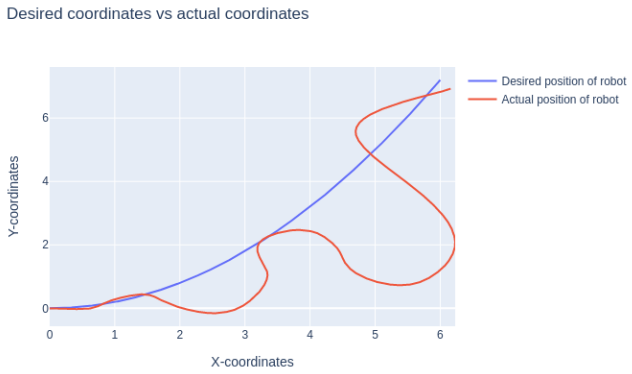

Fig. 7. Second order line following positional plot. $k_\rho = 5$

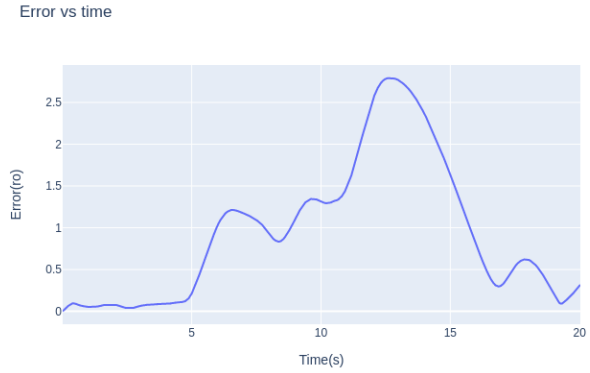

Fig. 8. Second order line following positional plot. $k_\rho = 5$

V. Conclusion

The main objective for this project has been to develop functioning control software, get an understanding of how to navigate and use Gazebo and lastly to be able to write software that makes use of the Robot Operating System. According to the results all of these objectives have been met to a satisfactory degree. The HUSKY robot has been able to go to a stationary target and also follow a moving target. It has also been shown that choosing the correct controller coefficients has a great impact on the stability of the controller. All in all this article shows to a satisfactory degree how one should design control software for a ground robot and simulate it using Gazebo.

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