Exercise 1

Kinematics and control of a differential drive vehicle

Due date: March 2, 2018
Submit by Brightspace before midnight.

Course: SC42090 Robot Motion Planning and Control, TU Delft, DCSC
Contact: Linda van der Spaa, email: L.F.vanderspaa@tudelft.nl
Contact: Bruno Brito, email: Bruno.deBrito@tudelft.nl
Office hours: Wiener hall C-3-190, Thursdays 10:00 – 11:00

Acknowledgements

This exercise is derived from the identically named exercise by the Autonomous Systems Lab of ETH Zürich.

1 Introduction

The goal of this exercise is to program and implement a closed loop motion controller for a differential-drive robot. For this, two subtasks have to be solved. 1) Given desired forward and angular velocities, one should compute the corresponding wheel velocities in order to make the robot drive accordingly. 2) Develop a controller that computes velocity commands to drive the robot to a specified target position. The solutions will be developed and implemented in Matlab/Octave and tested within a simulation environment. To this end, you will have to edit the different highlighted sections in the corresponding script files. In this exercise you will apply the theory described in the book "Introduction to Autonomous Mobile Robots" Chapter 3.6.

The exercise consists of tasks that you have to do, but do not require a written answer, and questions that do require a written answer. For each task, a method of validation is provided, so you will be able to correct you solutions to the tasks yourself. The questions to be answered are indicated by the number of points a correct answer will yield between parentheses. A maximum number of 20 points can be earned by completing this exercise. **To complete this exercise, hand in your answers to these questions using at most 2 pages (A4, 11pt).**

2 Matlab Implementation

In order to complete the implementation, you will have to fill out blanks (indicated by the TODO symbol) in the provided Matlab code.

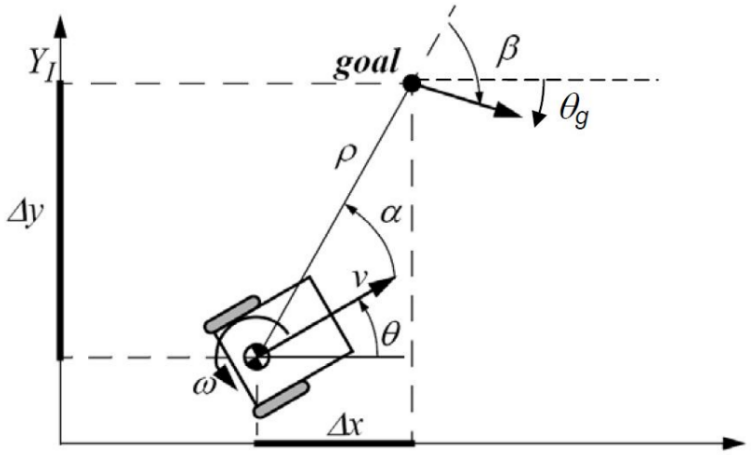

Figure 1: State feedback control onto a reference pose.

Exercise 1.1: Feed Forward Control

For a differential-drive robot, the kinematic model is described by the following equations:

$$v = \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2} \tag{1}$$

$$\omega = \frac{r\dot{\phi}_r}{2l} - \frac{r\dot{\phi}_l}{2l} \tag{2}$$

where (v, ω) represent forward and rotational velocity of the robot platform, respectively, and $(\dot{\phi}_r, \dot{\phi}_r)$ the rotational speeds of the right and left wheels. The wheel radius is given by r and l denotes half of the inter-wheel distance.

Task: Edit `calculateWheelSpeeds.m` in such a manner that it computes the rotational speeds $(\dot{\phi}_r, \dot{\phi}_r)$ based on the given velocities v and ω. Based on this feed forward controller, the robot will attempt to drive on a 0.5 m radius circular trajectory.

Validation: The feed forward controller can be evaluated by running the `test/testCircleDrive.m` script. This will directly select input velocities (v, ω) such that, if the controller is properly implemented, the robot will perform a circular trajectory of 0.5 m radius.

Exercise 1.2: Closed loop Control

In order to obtain a precise and smooth control of a differential drive robot onto a reference pose, the linear state feedback control law described in the book Introduction to Autonomous Mobile Robots that accompanies this exercise can be implemented. The control law is summarized below. Variables correspond to the ones introduced in Figure 1.

$$v = k_\rho \rho \tag{3}$$

$$\omega = k_\alpha \alpha + k_\beta \beta \tag{4}$$

Task: Implement this close-loop position controller within `calculateControlOutput.m`. Helpful Matlab commands/files are: *atan2*, and `normalizeAngle.m`. All angles are given in the positive right hand coordinate frame (counterclockwise).

Validation: Start V-REP, load scene `scene/mooc_exercises.ttt` and start the simulation. You should see a circular robot and a set of walls. Now run the script `vrep/vrepSimulation.m` within MATLAB. The robot should start driving towards the green robot ghost and should come to a stop in the same spot and the same orientation as the target. Select different target positions and orientations within the simulation environment in order to validate the proper functioning of the controller. For doing so, you can select the GhostBob Target in the scene hierarchy and change to the Object/item shift mode by clicking on the corresponding button in the toolbar. After that you can drag-and-drop the target to different positions and orientations.

(4pt) Try different values of k_α, k_β and k_ρ and explain the effects you observe. Please note that for strong stability the following conditions must hold:

$$k_\rho > 0 \tag{5}$$

$$k_\beta < 0 \tag{6}$$

$$0 < k_\alpha + \frac{5}{3}k_\beta - \frac{2}{\pi}k_\rho \tag{7}$$

Exercise 1.3: Closed-loop Control Enhanced

While the above control law exhibits a simple structure and has a strong stability condition, it also has a few shortcomings. Even though the robot might be able to drive forwards and backwards, the above control law will only output forward velocities and consequently will lead to rather cumbersome motions in some cases. Thus, a method should be derived in order to generalize the above control law for forward and backward velocities.

Looking at the simulation, one can observe that the velocity of the robot is decreasing exponentially when it draws nearer to the target pose. This leads to a very slow motion towards the end of the trajectory. A method should be derived and implemented such that the robot takes the same path as for the above control law but with a constant forward speed. This can be done by observing that the path does not change if the quotient between v and ω is the same as for the previous control law (same radius of curvature).

Task: Modify your code within `calculateControlOutput.m` in order to tackle the above shortcomings. The robot should be able to drive in both directions depending on where it is located with respect to the target pose and it should drive towards it with a constant speed. Implement your changes directly within `calculateControlOutput.m`.

Validation: Start V-REP, load scene `scene/mooc_exercises.ttt` and start the simulation. You should see a circular robot and a set of walls. Now run the script `vrep/vrepSimulation.m` within MATLAB. The robot should starts driving towards the green robot ghost and should come to a stop in the same spot and the same orientation as the target.

(5pt) Write the equation to compute v and ω, and motivate appropriately.

(4pt) Describe the tests you ran to validate your controller.

(7pt) Show the results of your tests and briefly discuss them.

To complete Exercise 1, hand in your answers to these questions using at most 2 pages (A4, 11pt).