Exercise 2

Kinematics and control of a differential drive vehicle

1. **Introduction**

The goal of this exercise is to program and implement a closed-loop motion controller for a differential-drive robot. For this, three 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)With the result of the previous task, you should develop a teleoperation of the robot via keyboard. 3) 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.

2. **Feed Forward Control**

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

$$v = \frac{r\varphi_r}{2} + \frac{r\varphi_l}{2} \quad (1)$$

$$\omega = \frac{r\varphi_r}{2l} - \frac{r\varphi_l}{2l} \quad (2)$$

where (v,ω) represent forward and rotational velocity of the robot platform, respectively, and (φr, ˙ φl) the spinning speed of the right and left wheels. The wheel radius is given by r and l denotes half of the inter-wheel distance.

Task: Please edit **code/commom/vrep/calculateWheelSpeeds.m** in such a manner that it computes the spinning speeds (φr, φl) based on the given velocities v and ω. Based on this feed forward controller, the robot will attempt to drive on a 0.5m radius circular trajectory.

Validation: The feed forward controller can be evaluated by running the **/code/common/vrep/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 0.5m radius.

3. **Teleoperation**

On this task, the student must implement a program

(**/code/common/vrep/teleoperation.m**) for teleoperation via keyboard. At least five commands are necessary:

3.1. Print a linear forward speed (positive)

3.2. Print a linear backward speed (negative)

3.3. Print a rotational speed counter-clockwise (positive)

3.4. Print a rotational speed clockwise (negative)

3.5. Immediately stop the robot

Commands are executed upon pressing the key associated with the command. For smooth and precise operation, the program should gradually vary the speeds as the corresponding key is pressed in sequence. The scene that contains the simulation is **scene/Exercise2.ttt.** In your code, the variable **realRobot** can assume two values: 0 for simulation testing and 1 for testing with the real robot.

4. 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 task can be implemented (see Chapter 3.6, p.91ff). The control law is summarized below. Variables correspond to the ones introduced in Figure 1.

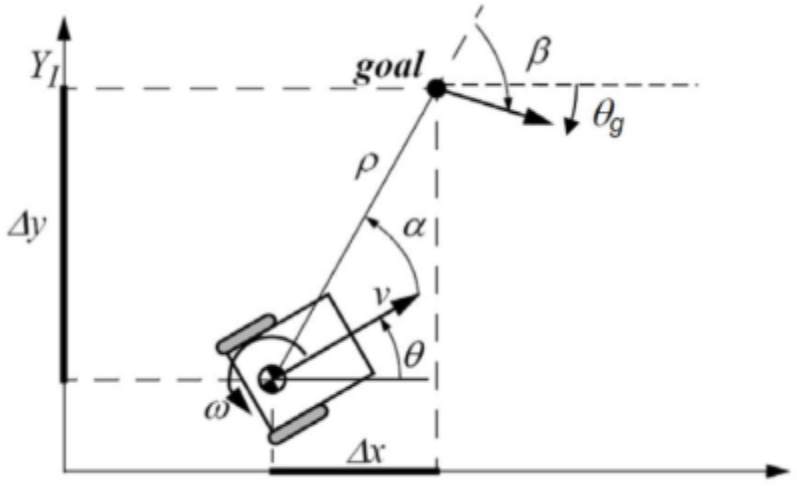

Figure 1: State feedback control onto a reference pose

$$v = k_\rho \rho \quad (3)$$

$$\omega = k_\alpha \alpha + k_\beta \beta \quad (4)$$

Task: Please implement this close-loop position controller within **code/commom/vrep/calculateControlOutput.m**. Helpful Matlab commands/files are: **atan2**, and **normalizeAngle.m**. All angles are given in the positive right-hand coordinate frame (counter clockwise).

Validation: Start V-REP, load scene **scene/Exercise2.ttt** and start the simulation. You should see a circular robot and a set of walls. Now run the script **vrep/controller.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. 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 *GhostPioneer_p3dxTarget* 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. Please note that for strong stability the following conditions must hold (see Chapter 3.6, p.91ff):

$$k_\rho > 0, \quad (5)$$

$$k_\beta < 0, \quad (6)$$

$$0 < k_\alpha + \tfrac{5}{3} k_\beta - \tfrac{2}{\pi} k_\rho \quad (7)$$

5. 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 performs the same trajectory as for the above control law but with a constant forward speed. This can be done by observing that the trajectory does not change if the quotient between v and ω is kept constant and thus the output of the control law can be scaled.

Task: Please improve 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/Exercise2.ttt** and start the simulation. You should see a circular robot and a set of walls. Now run the script vrep/controller.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.