Lab class: Autonomous robotics
Exercise sheets

Institut für Neuroinformatik

January 29, 2020

Student information

____

Name

____

ID number (German: Matrikelnummer)

Grade overview

Problem	Tutor signature
A	
B	
1.1	
1.2	
1.3	
2.1	
2.2	

A Basic movement commands

Problem: Let the robot drive from the starting position to the target on a printout of the first environment (Figure 1). The trajectory of the robot has to remain within the dashed area and may not touch any obstacles. (Hint: Set wheel speeds, pause, repeat.)

Educational objectives of the problem:

- Understanding basics of Matlab
- Getting to know basic robot control

B Kinematics

Problem: We now have an environment without obstacles, where the starting and end position can be varied (Figure 2). Write a program that brings the robot from an arbitrary starting position to an arbitrary end position. The program should get the coordinates (e.g., in millimeters) of the starting position and end position as well as the initial orientation (e.g., in degrees) of the robot as parameters. The coordinates should be expressed relative to the global (allocentric) coordinate frame as defined in the printout. The final orientation of the robot does not matter. We have marked some exemplary positions $P1, \ldots, P4$ in the environment that you may use, but other coordinates also have to work as starting and end positions. Try all combinations of starting positions and end positions with a number of initial orientations to make sure there are no errors in your code.[1]
(Hint: Turn first, drive later.)

Educational objectives of the problem:

- Understanding the trigonometry for determining the direction of the target
- Understanding how to rotate the robot by a given angle
- Understanding how to make the robot drive a given distance

[1] You will continue to use this code on subsequent problems so make sure that it is free of errors.

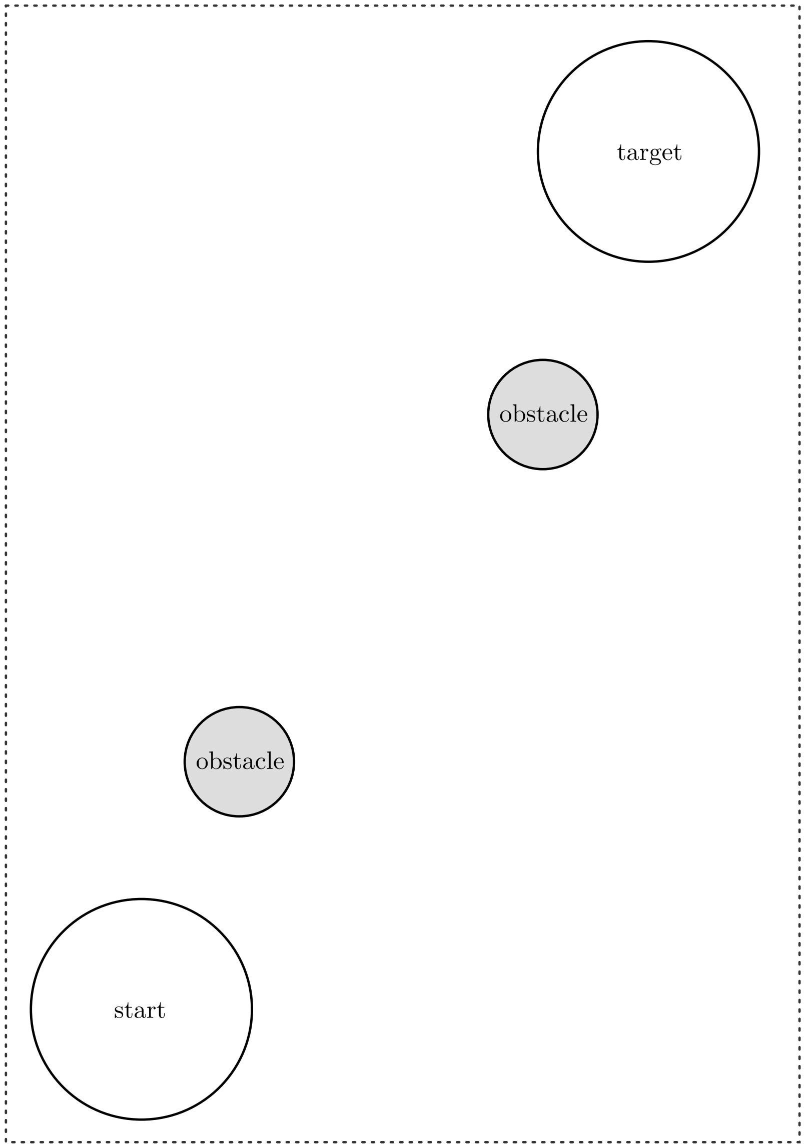

Figure 1: The environment for the first problem.

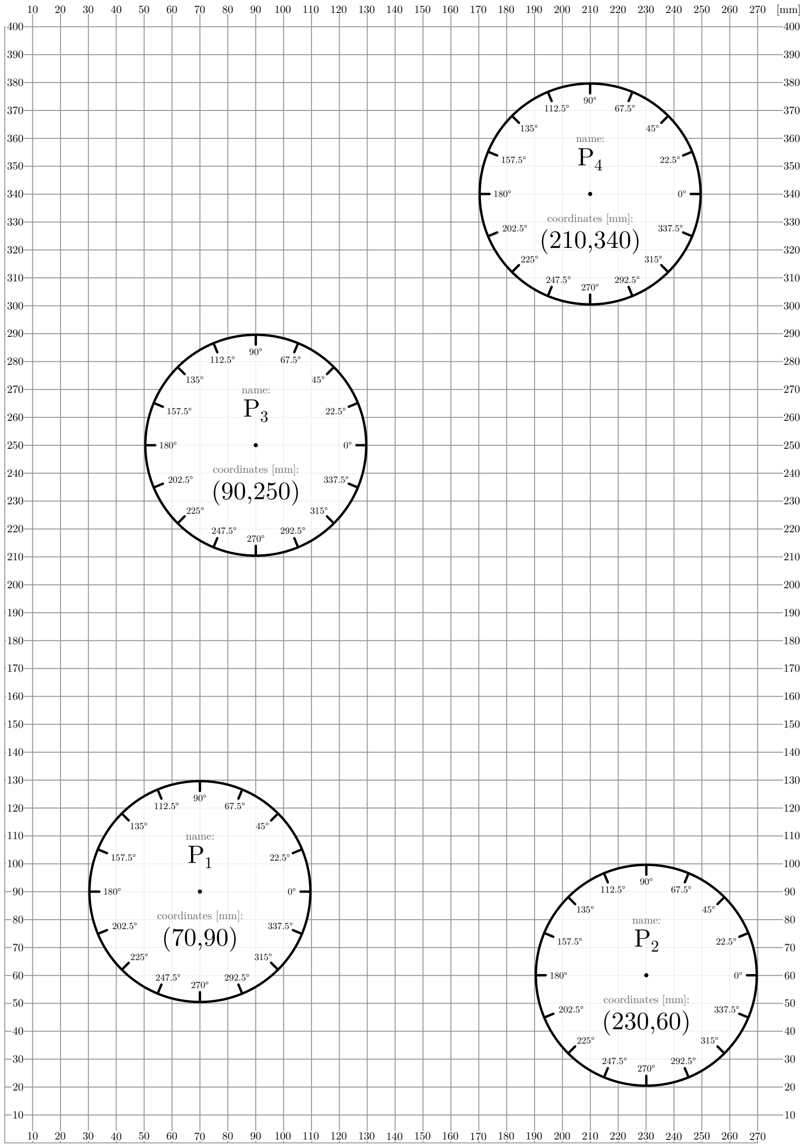

Figure 2: The environment for the second problem.

1 Controlling the robot

1.1 Odometry

Problem: Write a program that displays the current position and previous path of the robot in a live plot. Choose an appropriate coordinate system and starting position for your plot. You will receive a program from us that will control the robot. This means that you will not know what commands are being sent to the robot.
(Hint: The current position and orientation of the robot can be determined by integrating sensor readings, i.e., encoder values, over time.)

Educational objectives of the problem:

- Programming a live plot of position
- Understanding the equations and trigonometry involved in odometry

1.2 Detecting obstacles with sensors

Problem: If we introduce obstacles into the environment (see Figure 3) the robot will need to estimate their location to appropriately avoid them. In this task you will evaluate how the robot's infrared sensors react to obstacles and how sensor information may be used for robot control. Systematically analyze how a single infrared sensor reacts to a single obstacle at different distances while the robot remains still. Take multiple measurements and analyze the mean as well as standard deviation for each distance. Visualize your findings in a plot. Repeat the analysis for an obstacle made of a different material.
Use your knowledge about the sensor behavior to implement a function that converts the sensor readings into an approximation of the distance between the robot and an obstacle (for example, in cm). Verify your conversion through a live-plot of the measured distances while the robot is standing still. Write a program that makes the robot drive forward and reliably stops it once an obstacle is observed at a distance of 2 cm (4 cm).

Educational objectives of the problem:

- Understanding how to read out and integrate the infrared sensors of the e-puck robot
- Understanding how the distance and material of the obstacle influences the infrared sensors

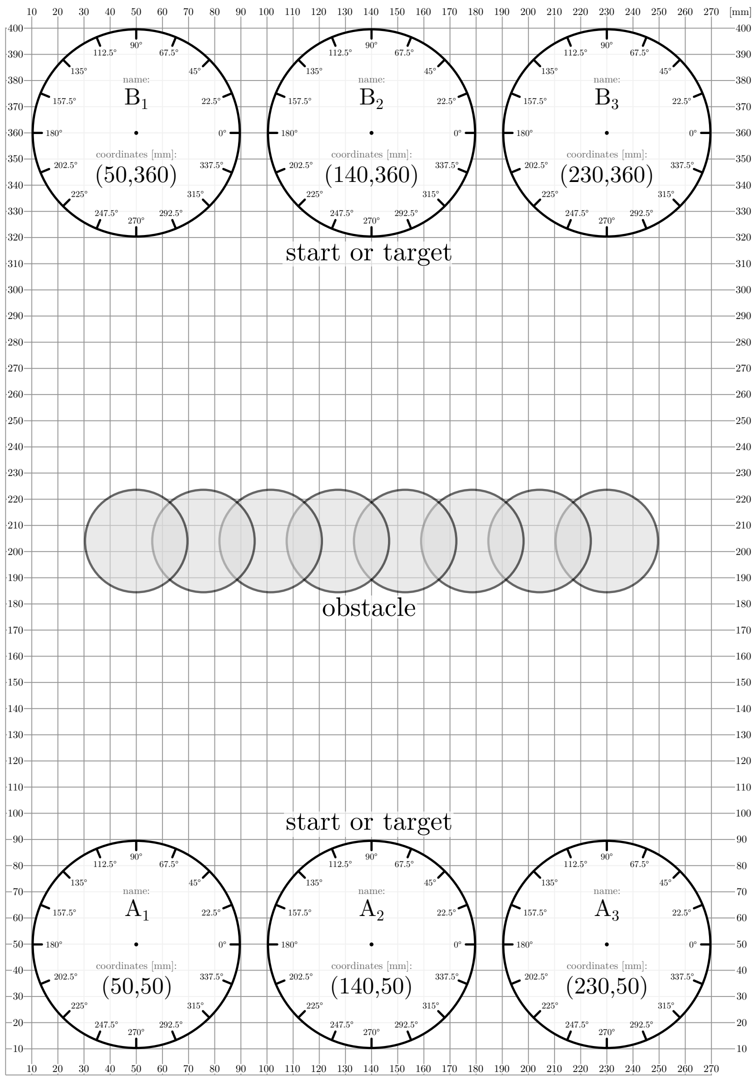

Figure 3: The environment for the rest of the lab class.

1.3 Obstacle avoidance

Problem: Write a program that makes the robot drive from a starting position (e.g., one of $A_1, \ldots, A_3$) to an end position (e.g., one of $B_1, \ldots, B_3$), while avoiding an obstacle in the environment (Figure 3). While navigating the environment, the robot may not touch the obstacle. Do not hard-code the obstacle's position into your program. Instead, use the infrared sensors to detect when the robot is close to an obstacle and then avoid it by changing course. The robot has to reach the target after avoiding the obstacle. (It is fine if the robot drives off the paper for a while.) Track the robot's position in a live plot.
Make the obstacle avoidance dependent on the position of the obstacle, that is, if the obstacle is right in the middle of the robot's path, avoid it more strongly than if it is off to the side of the path.
Once this works, extend the program further so that the robot drives back and forth indefinitely between the starting position and the ending position.

Educational objectives of the problem:

- Creating your own approach to obstacle avoidance

2 Attractor dynamics

2.1 Target approach

Problem: You will now solve the problem of section B again, but this time, using an attractor dynamics approach.
Write an attractor dynamics that rotates the robot on the spot toward the target. Use a sine dynamics that is defined over the orientation of the robot. Once turning on the spot works, add a constant forward speed to drive the robot to the target while turning.

Educational objectives of the problem:

- Understanding how attractor dynamics can orient the robot toward the target
- Understanding a numerical method for solving dynamics

- Investigating the properties of dynamics as a mechanism for controlling a robot

Hints for report:

- Explain the dynamical system and why it makes the robot turn toward the target. Create *at least two figures* (i.e., a phase plot of the dynamics and a plot that shows how the system develops over time) and refer to the plots in your explanation. What does each figure mean with respect to the robot?
- Over which variable is the dynamical system defined?
- Explain the concepts of an attractor and a repellor using the figures you created. Mark attractors and repellors in the phase plot.
- In which cases does the robot fail to reach its target? Explain how this depends on the chosen parameter values using *multiple exemplary plots* of the robot's trajectory.

2.2 Obstacle avoidance

Problem: Extend your program so that the robot can avoid obstacles while it is driving toward the target. The robot should still move forward and turn at the same time. Additionally, it should be repelled from obstacles and avoid them in smooth trajectories. Solve the obstacle avoidance by modifying the dynamical system you have implemented for the last problem.
Hint: Use the force-lets described in the background material for obstacle avoidance. You will need to choose values for various parameters; make sure you understand the equations involved here first.

Educational objectives of the problem:

- Understanding the equation for obstacle force-lets and its parameters.
- Understanding the properties of a combination of different influences on the heading direction.

Hints for report:

- In the environment the robot is navigating, which elements represent attractors and which represent repellors? Why? Explain.
- Explain the equations you use to generate the influence of the obstacles. Explain how *each parameter of the equation* influences the shape of the function and how this impacts the robot's behavior. Make plots where appropriate.
- Discuss the robot's perception of obstacles: Does it perceive them as a discrete set of obstacles? (How) does this correspond with the explanation of the attractor dynamics in the background material?
- Explain the bifurcation that the dynamics undergoes between Figures 12 and 13 in the background material. Why is there a repellor for each obstacle in Figure 12, while there is only a single repellor in Figure 13? What does this mean for the robot's behavior? Make drawings and explain.
- Compare the dynamic obstacle avoidance approach to your algorithmic solution. In doing so, also compare plots of the robot's trajectory generated with the two approaches.