Lab class: Autonomous robotics
Exercise sheets

Institut für Neuroinformatik

August 15, 2018

Student information

Name

ID number (German: Matrikelnummer)

Grade overview

Problem	**Tutor signature**
1.1	
1.2	
1.3	
1.4	
1.5	
2.1	
2.2	

1 Controlling the e-puck

1.1 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 does not have to be smooth, but 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

1.2 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

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

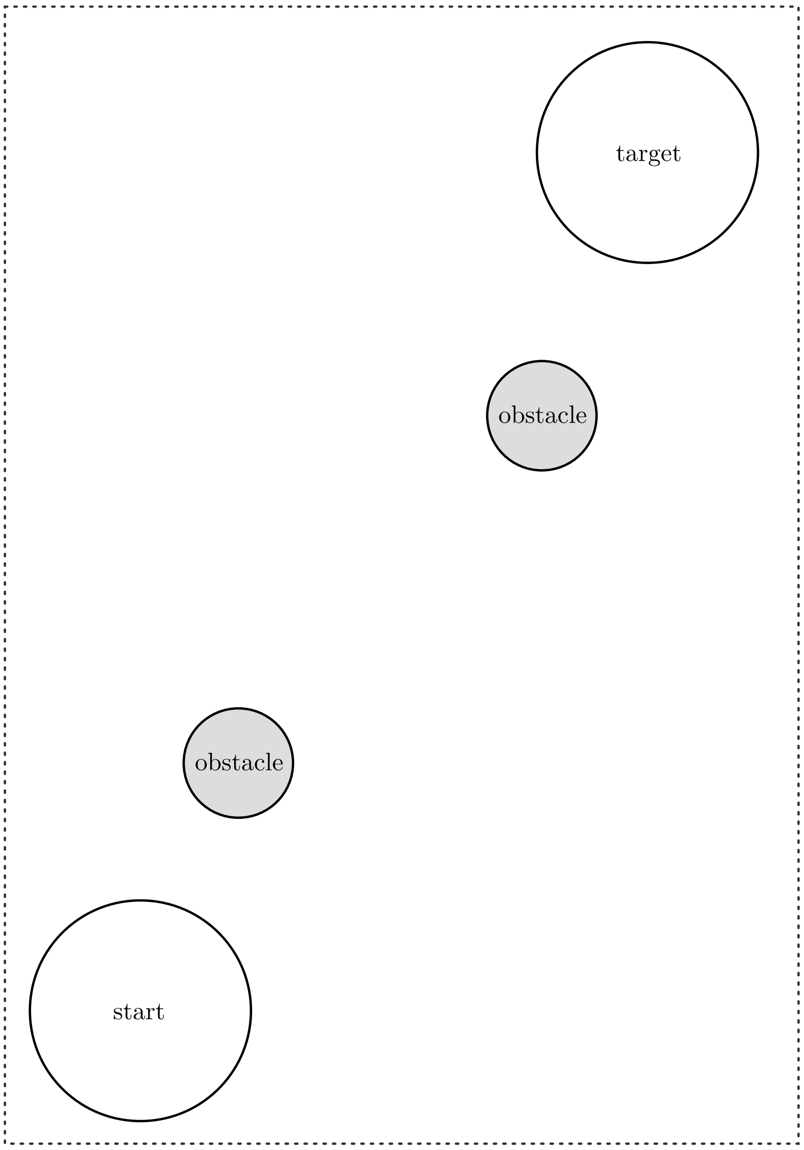

Figure 1: The environment for the first problem.

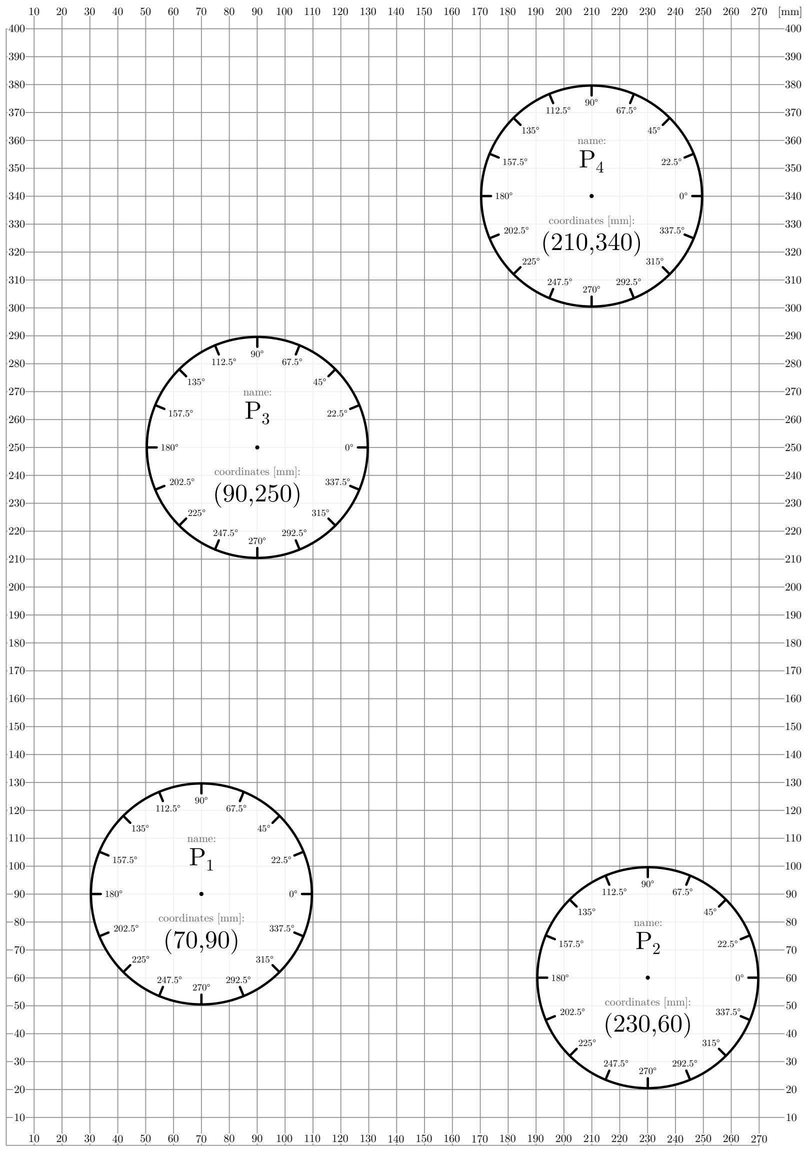

Figure 2: The environment for the second problem.

- Understanding how to rotate the robot by a given angle
- Understanding how to make the robot drive a given distance

Theoretical questions:

- In both problems, your robot finds its way to the target with the help of your program. Would you call the robot autonomous? If yes, explain what makes it autonomous. If no, explain why not and what is missing.
- Explain how you can give the robot coordinates in a metric unit (e.g., millimeters)—how does the robot know where and how far to drive?
- Explain the technical term 'kinematics' and its connection to the problem. Did you have to program a 'forward kinematics' or 'inverse kinematics'? Explain.
- Could the program replan if we moved the target along the way? If yes, explain how this works. If no, explain why this does not work.

Educational objectives of the theoretical questions:

- Establish an understanding of what autonomy means
- Understanding how much knowledge the robot has about its surroundings and about itself.
- Understanding the term 'kinematics' and its relevance to this problem.
- Understanding what information the robot needs in order to interact with the world.

1.3 Odometry

Problem: Write a program that displays the current position and previous path of the robot in a live 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

Theoretical questions:

- Explain the technical term ‘odometry’.
- If they were available, could you calculate the current position from the generated motor commands instead of the encoder values? Does it have advantages over calculating the position from the encoders? If so, name them. Does it have disadvantages? If so, what are they?
- Are there other ways of estimating the current position? Name at least one and sketch how it would work. You may assume that you could equip the robot and the environment with sensors.

Educational objectives of the theoretical questions:

- Understanding odometry and its limitations
- Thinking about other methods of estimating the current position

1.4 Detecting obstacles with sensors

Problem: We will now make the environment more difficult by placing an obstacle between the starting position and the target (see Figure 3). Write a program that makes the robot drive from a (variable) starting position (e.g., one of $A_1, \dots, A_3$) and a (variable) initial orientation toward a (variable) end position (e.g., one of $B_1, \dots, B_3$). Do not try to avoid the obstacle yet. For now, it is fine if the robot runs into it. Generate a live plot of all the robot’s infrared sensors so that you can observe their output as the robot moves in the environment. Implement a function that uses the sensor readings to approximate the distance of the robot to an obstacle (for example, in cm). Create a live plot for this as well. Observe both live plots as the robot approaches the obstacle. Try it with obstacles made from different materials.

Once all of this works, make the robot stop at a distance of 2 cm before it runs into the obstacle.

Educational objectives of the problem:

- Understanding how to read out and integrate the infrared sensors of the e-puck robot
- Understanding how the material of the obstacle influences the infrared sensors
- Programming live plots in Matlab

Theoretical questions:

- Explain how you use the infrared sensors to detect obstacles.
- What kind of influence does the material of the obstacle have on the robot's behavior? Describe and explain the effect you observed.
- How well does the approximation of the distance to obstacles work? Explain and show plots.

Educational objectives of the theoretical questions:

- Understanding the sensors and their integration
- Understanding the approach for obstacle avoidance
- Understanding how much knowledge the robot has about its surroundings and about itself

1.5 Obstacle avoidance

Problem: Write a program that makes the robot drive from a starting position (e.g., one of $A_1, \dots, A_3$) to an end position (e.g., one of $B_1, \dots, B_3$), 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

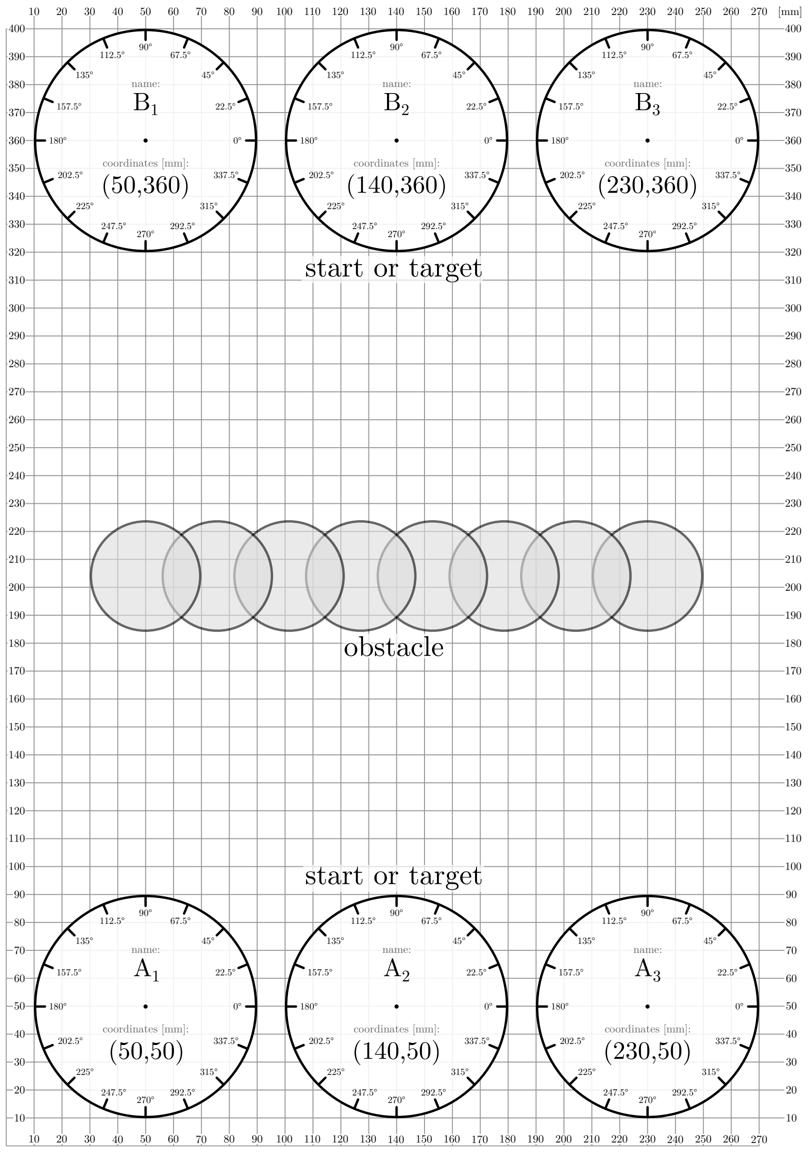

Figure 3: The environment for the rest of the lab class.

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

Theoretical questions:

- Why is odometry vital for obstacle avoidance?
- Explain in detail how you have programmed the obstacle avoidance. How and when does the robot avoid an obstacle? In which direction does it drive to avoid it? How long does it keep avoiding the obstacle?
- How does your obstacle avoidance depend on the position of the obstacle in the path of the robot?
- Write about what you notice when the robot drives back and forth between the two positions. For how long does it work? If there are any problems, explain how they might come about. Can you think of a way of resolving these issues?

Educational objectives of the theoretical questions:

- Understanding the link between odometry and obstacle avoidance

2 Attractor dynamics

2.1 Target approach

Problem: You will now solve the last problem 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

Theoretical questions:

- 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.

Educational objectives of the theoretical questions:

- Understanding the difference between the time course of a dynamics and the dynamics itself, and how these map to the robot's behavior
- Understanding the importance of parameterization
- Thinking through what makes attractors attractive

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.

Theoretical questions:

- 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 solution for the previous problem. In doing so, also compare plots of the robot's trajectory generated with the two approaches.

Educational objectives of the theoretical questions:

- Understanding the different components of the obstacle part of the dynamics.
- Observing emergent behavior that was not programmed in explicitly.
- Understanding the relation between bifurcations and behavior (decisions), and how these emerge from the chosen dynamical system