Lab class: Autonomous robotics
Exercise sheets

Institut für Neuroinformatik

March 7-11, 2022

A Basic movement commands

Problem: Let the robot drive from the starting position to the target (see Figure 1). The trajectory of the robot has to remain within the arena and may not touch any obstacles or walls. For this task and for this task only, assume that you know where the obstacles are. Use only wheel velocity commands to achieve this goal.

Educational objectives:

- Understanding basics of webots
- Getting to know the webots python controller interface

Presentation objectives:

- The robot drives from start to target position without any collision and stays at the target

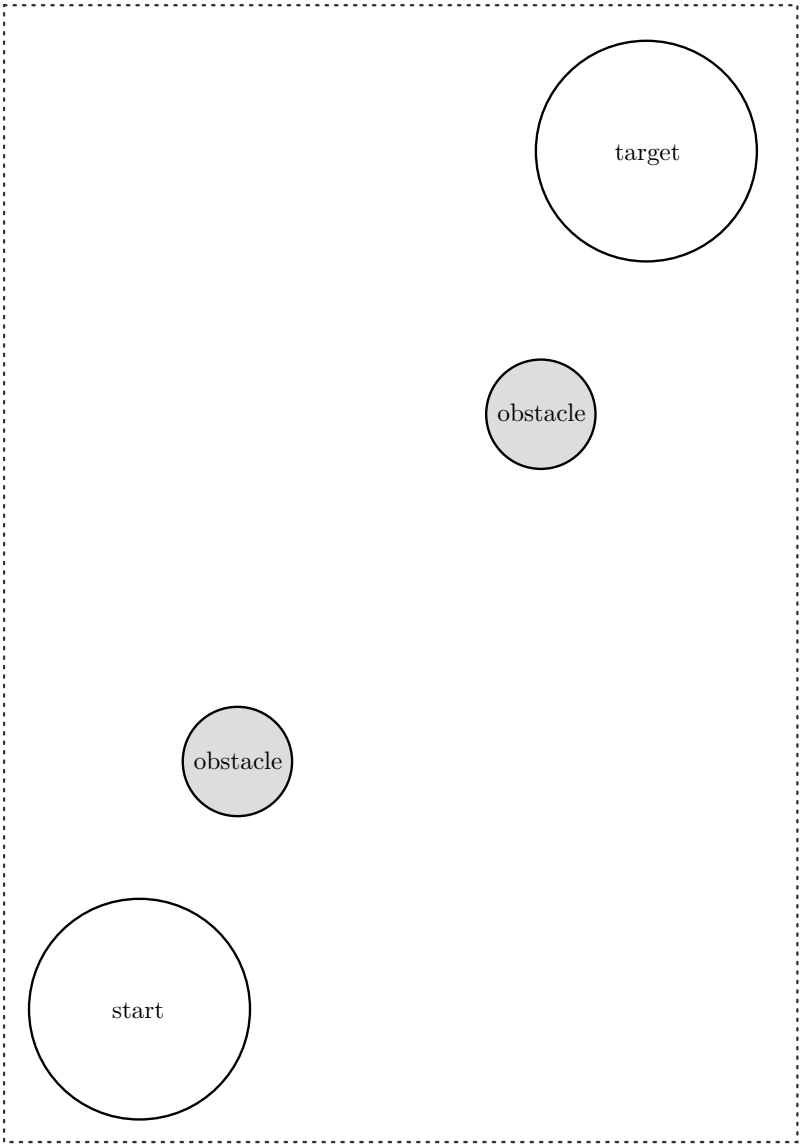

Figure 1: The environment for the first problem.

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 in the global (allocentric) coordinate frame defined by webots. 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 several initial orientations to make sure there are no errors in your code.[1]
(Hint: Turn first, drive later.)

Educational objectives:

- 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

Presentation objectives:

- The tutor chooses an arbitrary start point, end point, and initial orientation
- The robot should drive from start to end point and stop there
- Repeat for a second set of locations and orientation

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

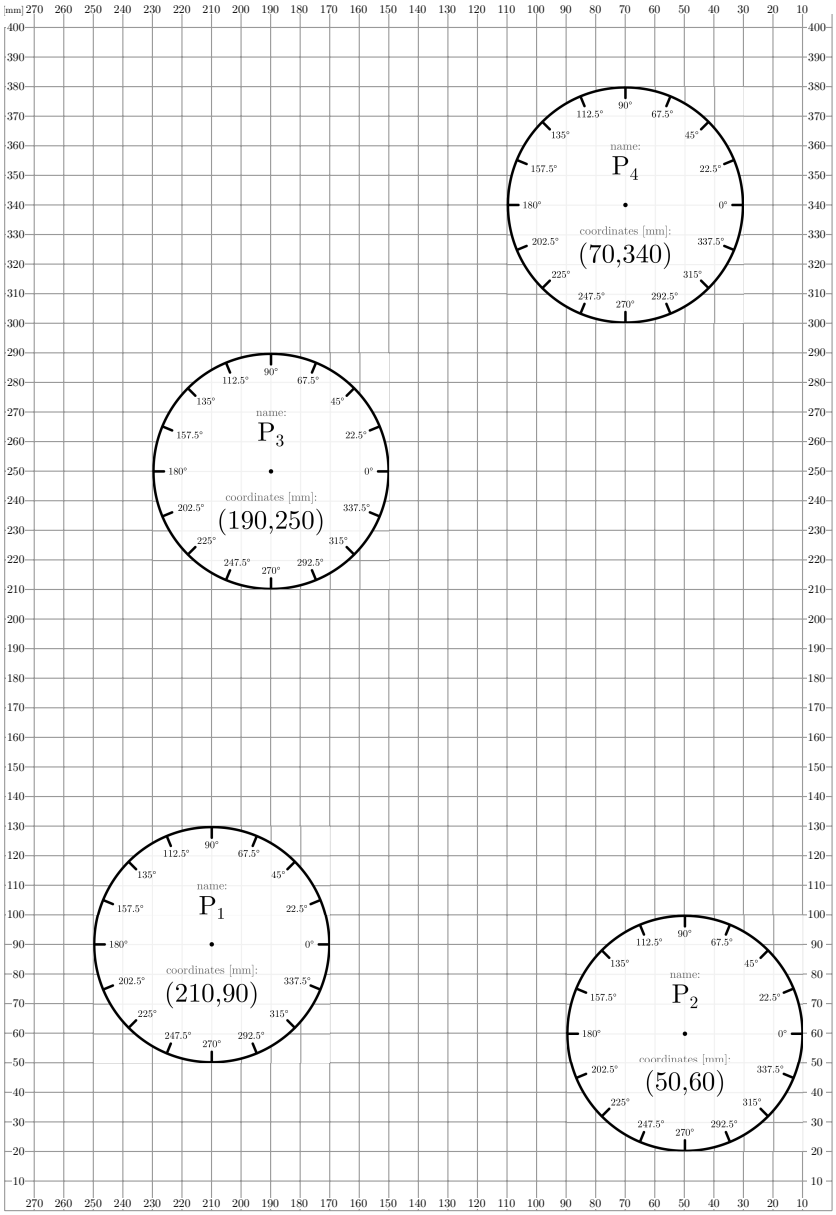

Figure 2: The environment for the second problem.

1 Controlling the robot

1.1 Odometry

Problem: Write a program that determines the robot's current position and plots its path. The controller in this template already executes a series of random movements over a period of time. Do not change anything about this random movement generator and assume that you do not know the commands given to the robot. You, however, know starting position and orientation of your robot. Given those determine how the location and orientation of the robot changes while it is driving. At the end of the movement plot the robot's trajectory, e.g. with *matplotlib*. Please choose an appropriate coordinate system for your plot.
(Hint: The current position and orientation of the robot can be determined by integrating sensor readings, i.e., encoder values, over time.)

Educational objectives:

- Plotting data with python
- Understanding the equations and trigonometry involved in odometry

Presentation objectives:

- Let the controller run
- Print the current location and orientation to the console in each time-step
- Show a trajectory plot at the end of the movement
- Repeat three times for different random trajectories

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 frontal infrared sensor reacts to a single obstacle at

different distances while the robot remains still (Choose sensor $p0$ or $p7$). Take multiple measurements and analyze the mean as well as the standard deviation for each distance. Visualize your findings in a plot.
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 by printing the live-values of the measured distances to the console 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 1 cm (2 cm).

Educational objectives:

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

Presentation objectives:

- Print the current sensor values and their conversion into distance to the console
- While the robot is standing still, slowly move an obstacle towards the robot to verify your conversion
- Let the robot drive in a straight line towards an obstacle. It should stop 1 cm before the obstacle.
- Repeat the step above for 2 cm.

Tasks for the report:

- Plot the mean sensor readings against different distances for a single front sensor
- Plot your measured distance against different distances for a single front sensor

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. After the movement plot the robot's trajectory.
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:

- Creating your own approach to obstacle avoidance

Presentation objectives:

- The tutor chooses an arbitrary starting, end, and obstacle position
- The robot should drive back and forth two times between start and end position without touching the obstacle
- Repeat for a different set of start, end, and obstacle position
- The robot should demonstrate a stronger avoidance for an obstacle directly in its path

Tasks for the report:

- Plot the trajectory your robot took through the obstacle course

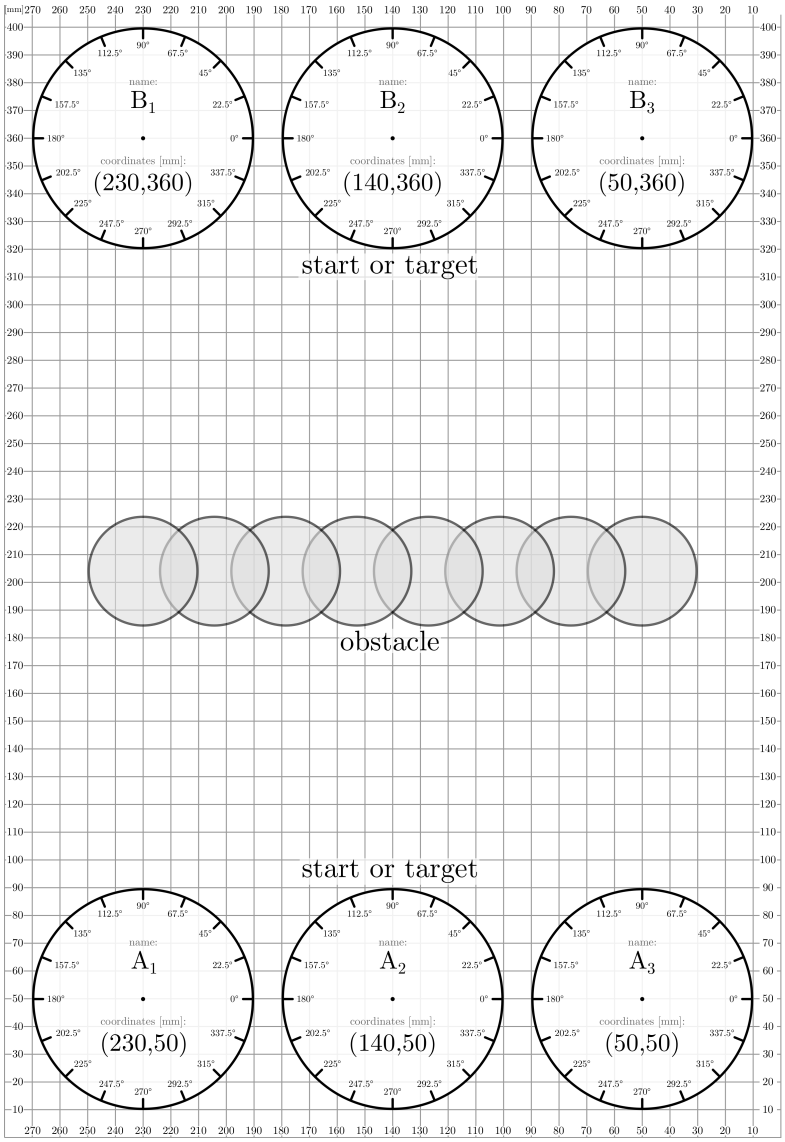

Figure 3: The environment for the rest of the lab class.

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:

- 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

Presentation objectives:

- The tutor chooses an arbitrary start point, end point, and initial orientation
- The robot should drive from start to end point and stop there
- Repeat for a second set of locations and orientation

Tasks for the 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. 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.
Hint: Plot the weight function λ_{obs} with your choices for β_1 and β_2 for different distances, d, to understand their effect on the avoidance behavior.

Educational objectives:

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

Presentation objectives:

- Explain which values you chose for β_1 and β_2 and why.
- Choose an arbitrary start and end position and place two obstacles in between such that they form a gap of roughly 4 cm.
- While driving from start to goal the robot should drive around the pair of obstacles.
- Widen the gap to roughly 8 cm. While driving from start to goal the robot should now pass directly through the gap.

Tasks for the 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.