Assignment 2

TME290Autonomous robots

Axel Johansson
axejoh@student.chalmers.se

May 1, 2023

Solution proposal

All code are found in the zip file provided. Commands to run are found ether in the readme or in this report.

Problem a)

The motion of a robot with differentially steered wheels can be represented as rotation around an Instantaneous Centre of Rotation with an angular velocity. The robot's velocity is calculated as the average of the left and right wheel velocities, while the yaw rate is the difference in wheel velocities divided by the robot diameter:

$$v_{robot} = (v_L + v_R)/2 \quad \text{and} \quad \dot{\theta} = (v_R - v_L)/(2R). \tag{1}$$

Here, R is the radius of the robot. To update the robot's position and orientation, Euler integration is used:

$$\begin{aligned} \theta_{n+1} &= \theta_n + \dot{\theta} n + 1 \Delta t & (2)\\ Xn+1 &= X_n + V_{n+1}\cos(\theta_{n+1})\Delta t & (3)\\ Y_{n+1} &= Y_n + V_{n+1}\sin(\theta_{n+1})\Delta t. & (4) \end{aligned}$$

Here, V_{n+1} is the robot's velocity at time step $n+1$, and Δt is the time step size. The implementation of this algorithm was done in Python. A trace of the robot's path is shown in Figure 1, and the code is provided in the `task_a.py` file.

Problem b)

The code is contained within a zip file, and its location is indicated in both the .rec-file and in Figure 2, which displays the path of the robot in scenario b). To run the program, a build command has been added to the simulate.yml file. To execute it, simply use the following command:

```
docker compose -f simulate-kiwi-b.yml up --build
```

The car will come to a stop near the coordinates (0.2, -0.4), although the exact position may vary depending on the frequency.

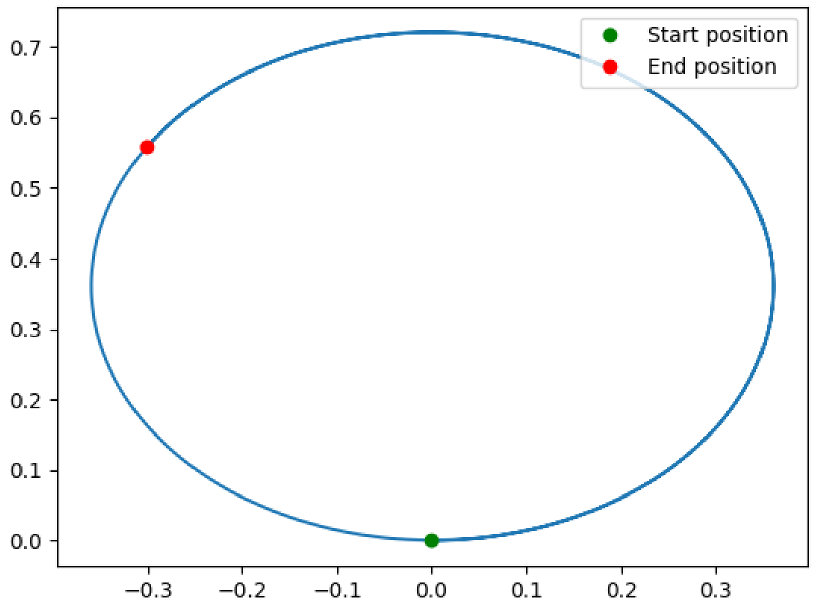

Figure 1: The robot drives 1 full lap and then stops at the red dot on the second lap.

Problem c)

The program for part c is in the folder finite-state-machine. The same virtual motor is used. For this task the distance sensors has been re added. The program is run using the following command:

```
docker compose -f simulate-kiwi-c.yml up --build
```

The FSM-diagram in words. The present system employs a finite state machine consisting of two primary states, namely the ”move forward” and ”turn” states. The robot's default behavior, upon initialization, is to move forward continuously, until an external event or stimuli activates the turning behavior. The latter state is designed with higher priority than the former, and is initiated when one or more of the distance sensors, located at the front, left, or right of the robot, detects a distance below a pre-defined threshold.

The distance sensors are triggered at a specific frequency and their outputs are updated in the system's step function, which carries out the requisite logic for the respective states. The step function's output is then used to determine the subsequent actions of the robot, whether it be to maintain its current forward trajectory or to execute a turn in response to the input from the distance sensors. This system design ensures that the robot operates efficiently, while also providing the flexibility to adapt to changing environmental conditions.

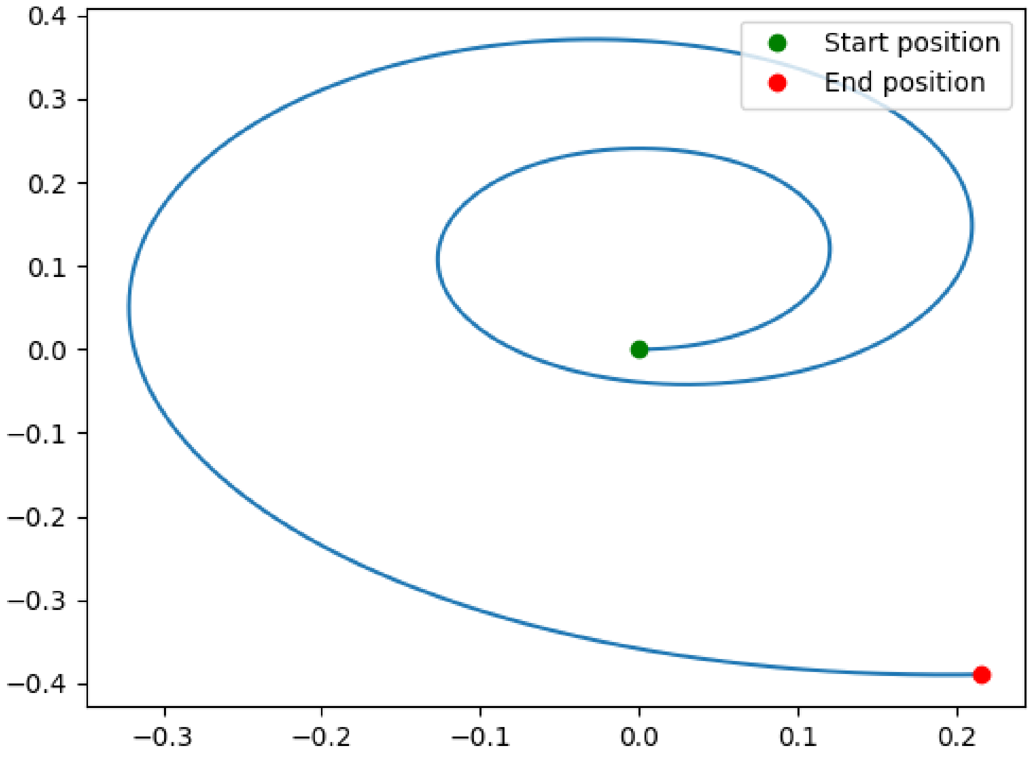

Figure 2: Path of robot in b).