

Signals, instruments & systems



E-puck line following project

MIAZZA Raphaël & PIGNERES Marie

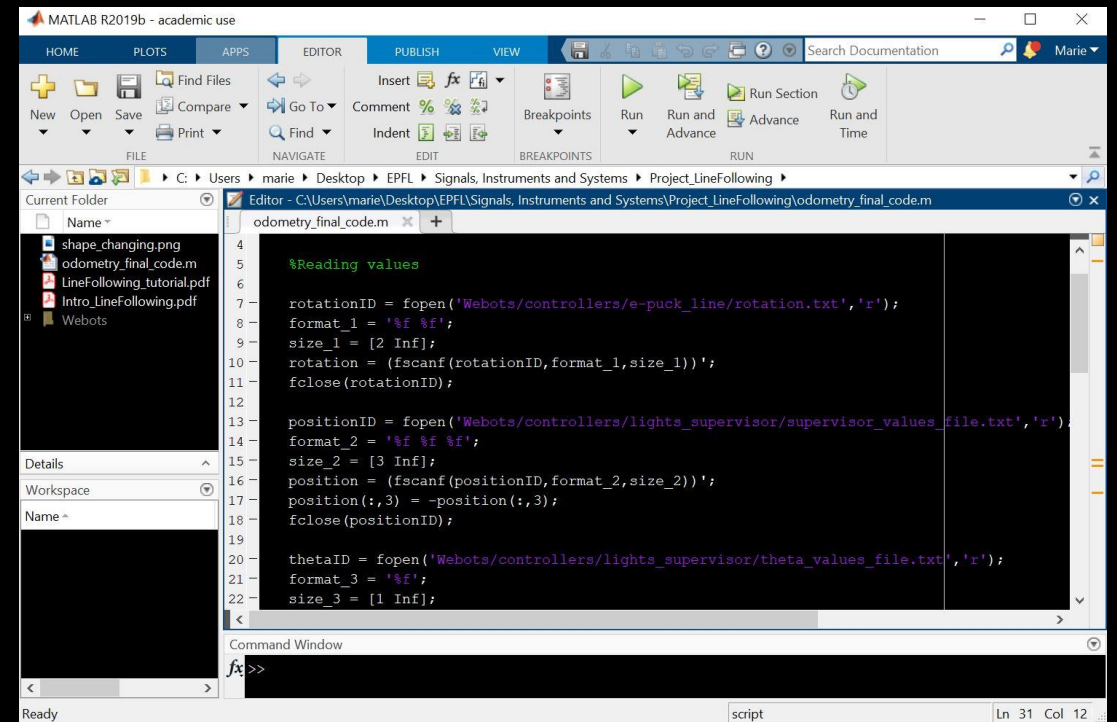
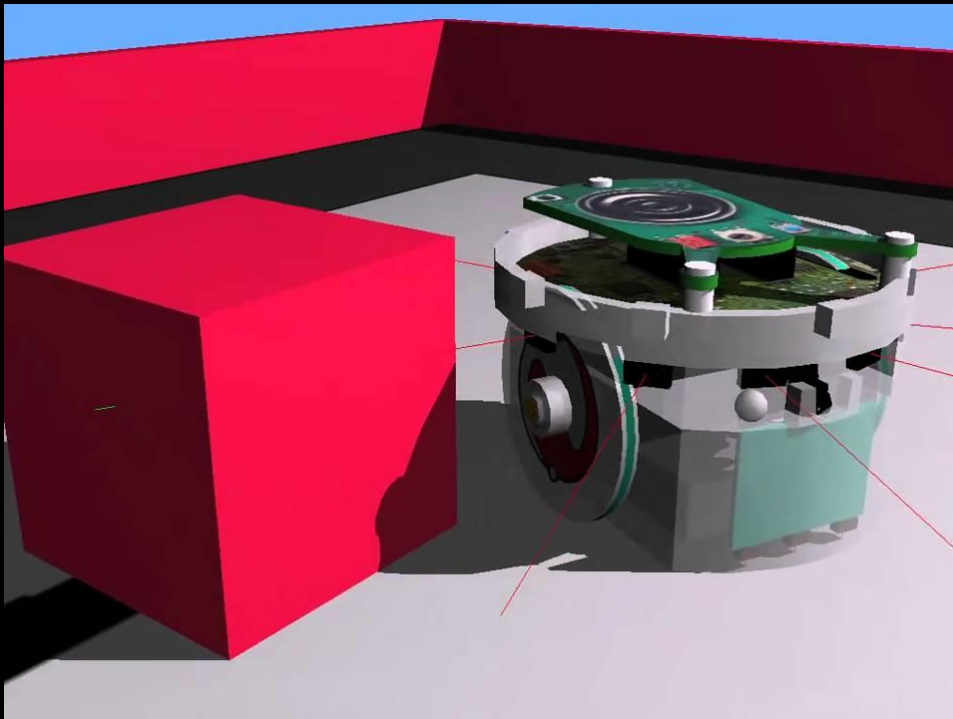
Introduction

Goals of this project:

- Perform line following
- Reacquire line after losing it
- Avoid obstacle on top of line following
- Implement odometry

Introduction

Mediums : Webots & Matlab

A screenshot of the MATLAB R2019b interface. The window title is "MATLAB R2019b - academic use". The interface shows a menu bar (HOME, PLOTS, APPS, EDITOR, PUBLISH, VIEW), a toolbar with various icons, and a command window at the bottom. The main editor window displays the following MATLAB code:

```
4 %Reading values
5
6
7 rotationID = fopen('Webots/controllers/e-puck_line/rotation.txt','r');
8 format_1 = '%f %f';
9 size_1 = [2 Inf];
10 rotation = (fscanf(rotationID,format_1,size_1)');
11 fclose(rotationID);
12
13 positionID = fopen('Webots/controllers/lights_supervisor/supervisor_values_file.txt','r');
14 format_2 = '%f %f %f';
15 size_2 = [3 Inf];
16 position = (fscanf(positionID,format_2,size_2)');
17 position(:,3) = -position(:,3);
18 fclose(positionID);
19
20 thetaID = fopen('Webots/controllers/lights_supervisor/theta_values_file.txt','r');
21 format_3 = '%f';
22 size_3 = [1 Inf];
```

The Command Window at the bottom shows the prompt `>>`. The status bar at the bottom indicates "Ready" and "script" with line and column numbers "Ln 31 Col 12".

Methods

E-puck devices used:

- ❑ Camera

Color camera with 52x39 resolution

- ❑ Proximity sensors

8 infra-red sensors measuring ambient light and proximity of obstacles in a 4 cm range

- ❑ Motors

2 stepper motors with 20 steps per revolution and a 50:1 reduction gear

Methods

General structure of e-puck's behavior assignment

```
switch(obstacle)
{case true: //obstacle detected
  obstacle avoidance code
  break;

  case false: //no obstacle detected
    if (line detected) {//line following
      line following code}
    else if(line lost, test_dead_end = 1
      and no far obstacle) {//turn around
      when meeting dead end
      turn around when dead end code}
    else { // e-puck is lost somewhere in
      the world
      keep going ;}
  break;}
```

Methods

Odometry

1. Export data from Webots into Matlab
2. Forward Euler's method -> rotational speed of each wheel
3. Forward kinematic model -> linear speed
4. Integration of speed -> position

Experiments

Line following and obstacle avoidance

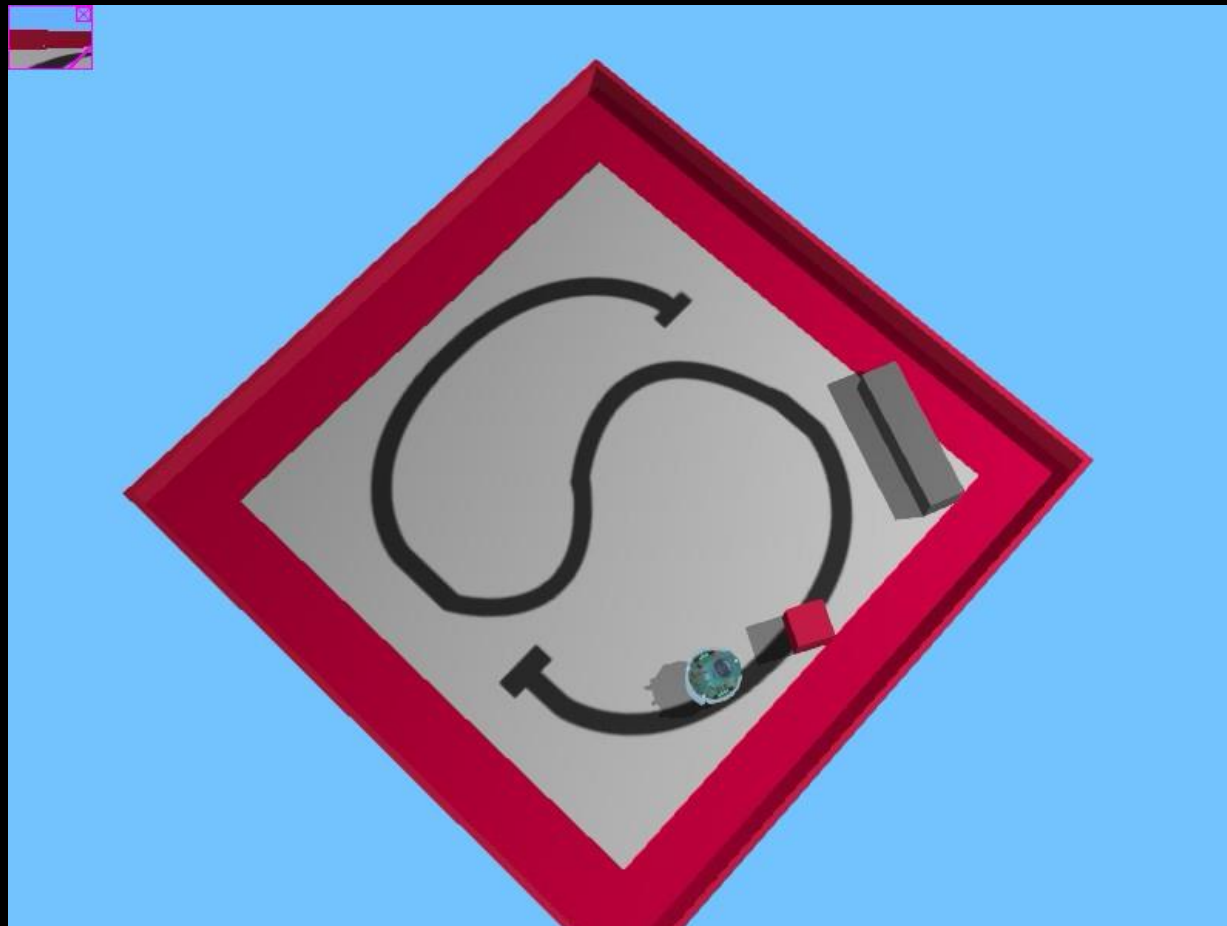
- Basic obstacle avoidance → Basic world
- Line following and obstacle avoidance → Oval line
- Dead end detection and final implementation of the code → Funky line

Odometry

- While following the line → Oval line
- While performing U-turns and avoiding obstacles → Funky line

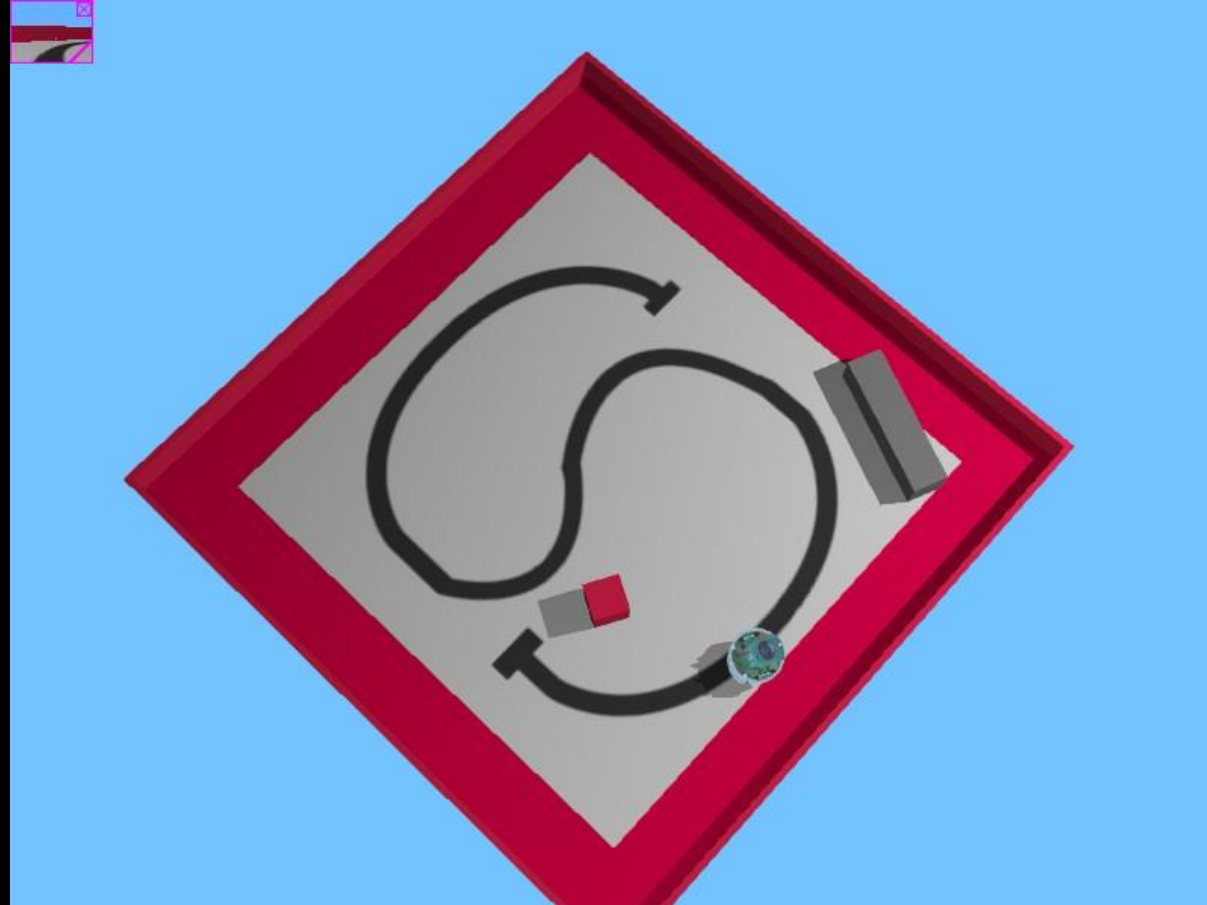
Results

Line following and obstacle avoidance



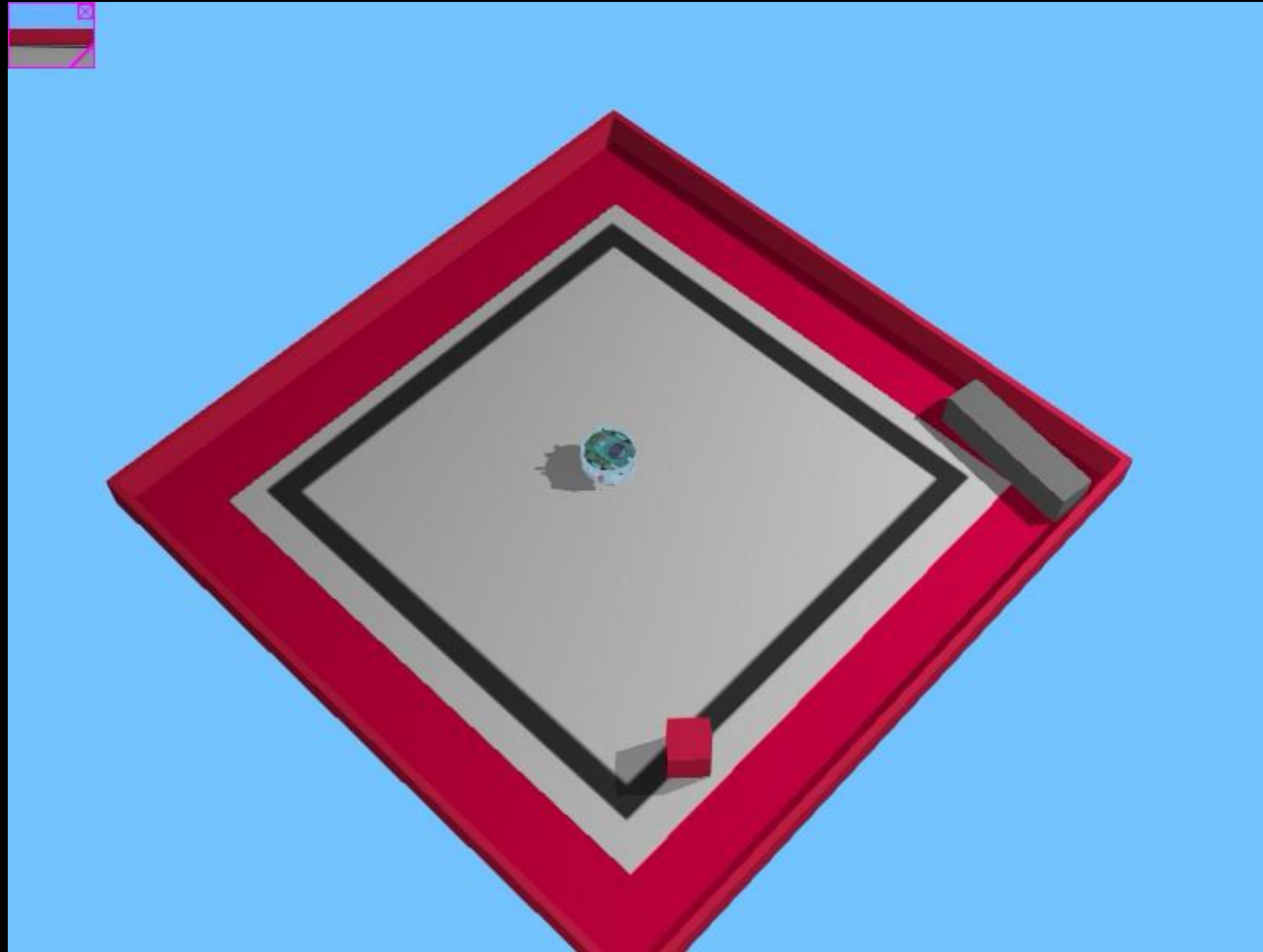
Results

Problems that may occur for dead end detection



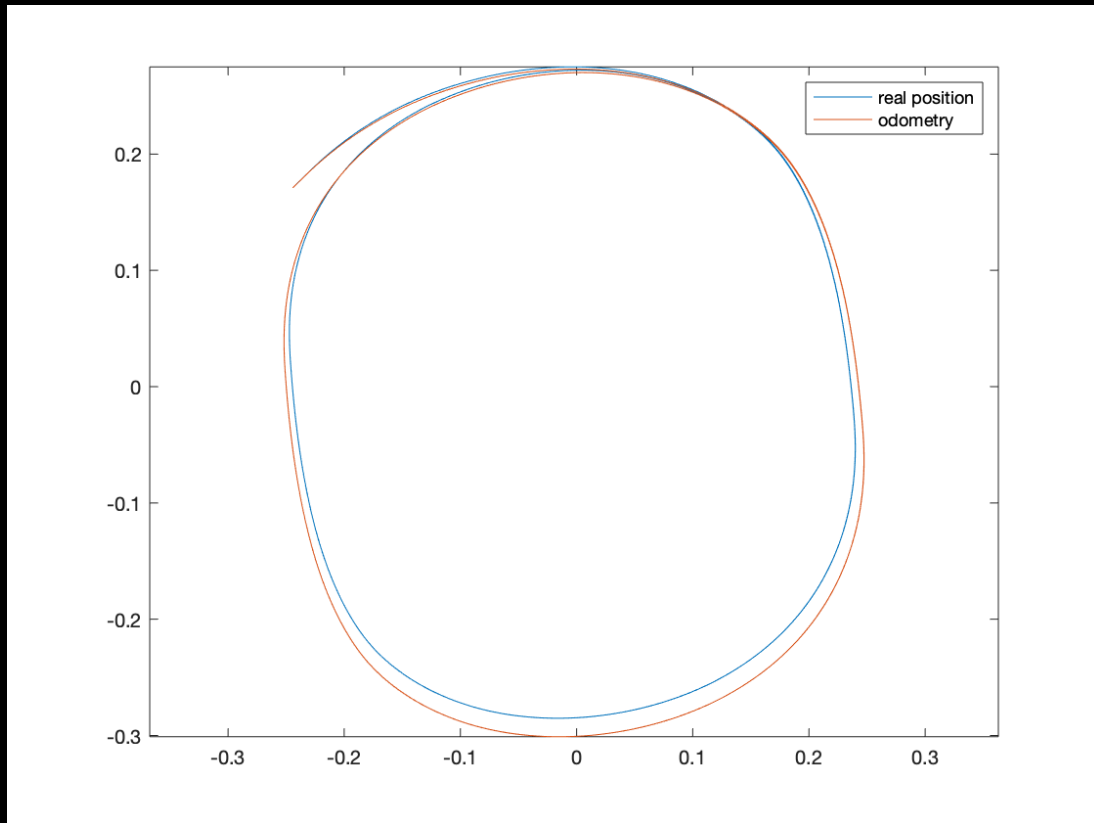
Results

Problems that may occur when crossing a line perpendicularly

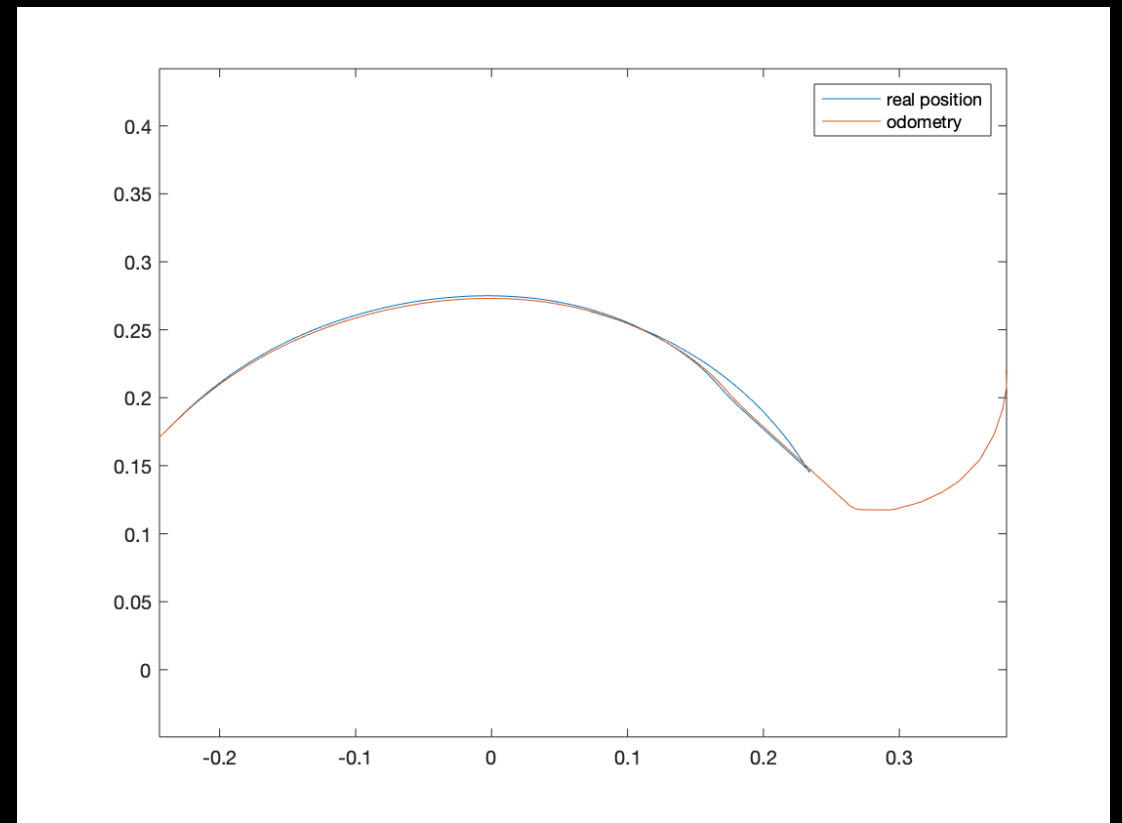


Results

Odometry



Trajectory comparison while turning four times around the circuit



Trajectory comparison while performing a U-turn

Conclusion

- ❑ Overall, each behavior worked pretty well, as well as the switching between each of them
- ❑ Odometry results weren't perfect but pretty close to the actual trajectory
- ❑ We got a deeper understanding of coding in the context of a robot's simulations