Assignment 1

Ivan Efremov

October 2022

1 Task description

Task Formulation

- Define a function to execute the reference path following controlling strategy as follows:

```
def reference_path_follower_diff_drive (duration=50,
control_points=np.array([[3,0], [6,4], [3,4], [3,1], [0,3]]), k_p=0.4,
k_theta=3)
```

  , where initial reference path, denoted *control_points*,and proportional control gains of linear and angular velocities are given *k_p*, and *k_theta*, respectively
- You are asked to use the provided simulator `https://github.com/GPrathap/autonomous_mobile_robots/tree/master/hagen/hagen_gazebo`
- Your submission should include **the report** and the **source code**

Figure 1: Task description

This task was tested on Hagen's robot in Gazebo simulation. The installation folder is here.

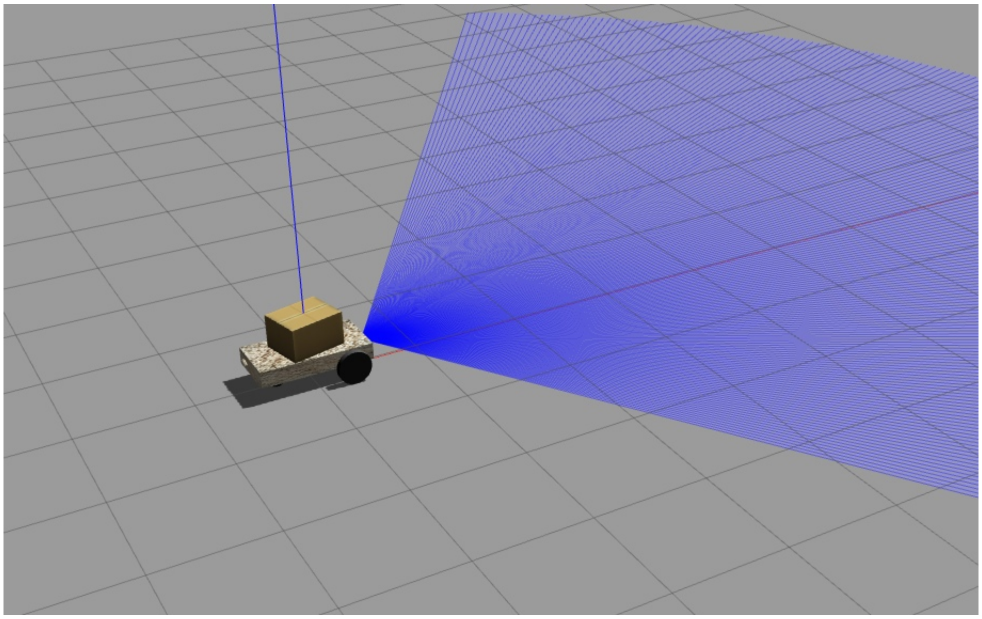

Figure 2: Example of Gazebo simulation

2 Theory

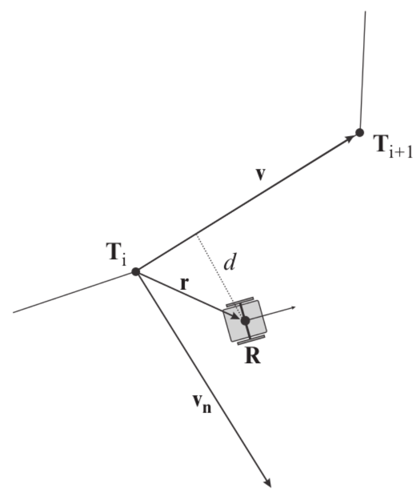

Figure 3: Control on segmented continuous path determined by a sequence of points. The reference path between the neighboring points is the straight line segment that connects those two points.

Reference path control

- The reference path is given by a set of control points. Hence, control strategy is driven to drive on a set of straight lines with proper orientation. However, this causes nonsmooth transition between neighboring line segments
- Consider the path is given by a set of points $\mathbf{T}_i = [x_i, y_i]^\top$, where $i \in 1, 2, .., n$ and n is the number of points. Orientation between two consecutive line segment is defined by taking orientation of vector $\mathbf{T}_{i+1}, \mathbf{T}_i$
- Let the direction vector be $\mathbf{v} = [\Delta x, \Delta y]^\top$ along the $\mathbf{T}_i$. The vector $\mathbf{v}_n = [\Delta y, -\Delta x]$ is orthogonal to the vector $\mathbf{v}$
- To check within which line segment robot is located at time t,

$$u = \frac{\mathbf{v}^\top \mathbf{r}}{\mathbf{v}^\top \mathbf{v}} \begin{cases} \textit{Follow the current segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \textit{if } 0 < u < 1 \\ \textit{Follow the next segment}(\mathbf{T}_i, \mathbf{T}_{i+1}) & \textit{if } u > 1 \end{cases} \tag{1}$$

Reference path control

- The normalized orthogonal distance between current pose and the line segment that robot should be

$$d = \frac{\mathbf{v}_n^\top \mathbf{r}}{\mathbf{v}_n^\top \mathbf{v}_n} \tag{2}$$

, where d is zero if the robot is on the line segment and positive if the robot is on the right side vice verse and $\mathbf{r} = q - \mathbf{T}_i$, where q is the current position of the robot

- Orientation of line segment that robot drives

$$\Phi_{lin} = arctan2(\mathbf{v}_y, \mathbf{v}_x)$$

- In case robot is far from the line segment, it needs to drive perpendicularly to line segment in order to reach the segment faster

$$\Phi_{rot} = atan(k_r \cdot d)$$

, where $k_r \in \mathbb{R}^+$ is a small constant

Reference path control

- Reference orientation and orientation error

$$\begin{aligned} \Phi_{ref} &= \Phi_{lin} + \Phi_{rot}, \\ e_\Phi &= \Phi_{ref} - \Phi, \ \omega = K_2 e_\Phi \end{aligned} \tag{3}$$

- Then the controller,

$$\begin{aligned} v &= k_p \cdot cos(e_\Phi) \\ \omega &= k_\Phi \cdot e_\Phi \end{aligned} \tag{4}$$

, where $k_\Phi, k_p \in \mathbb{R}^+$ are constants

3 Results

The code for this assignment you can find in my GitHub repository.

Algorithm is implemented according to the theory. Results you can see in figures 4 and 5.

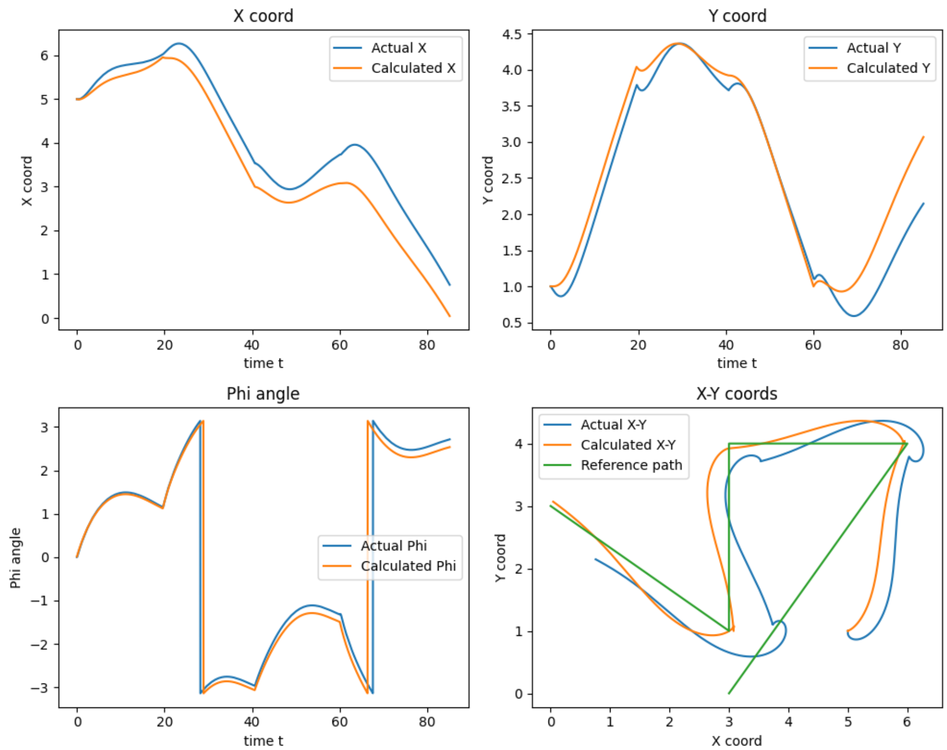

Figure 4: Example of the reference path control. Robot starting point is (5,1)

There can be quite a big error between the calculated and actual odometry. We can reduce this error by taking into account the physical parameters of the robot.

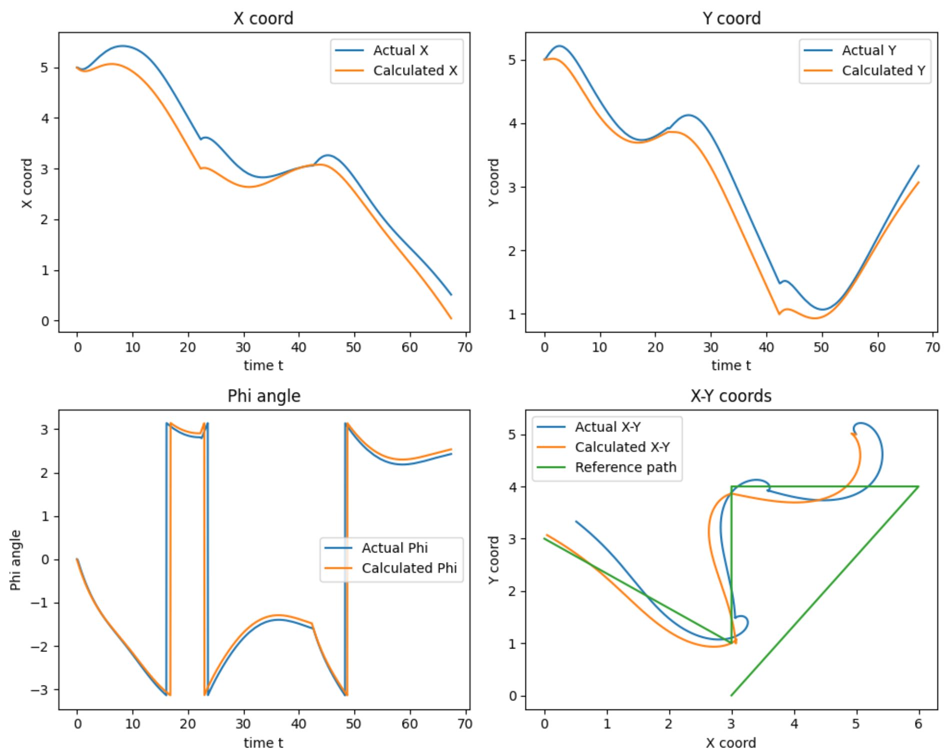

Figure 5: Example of the reference path control. Robot starting point is (5,5)