Sheet 2 solutions

May 10, 2019

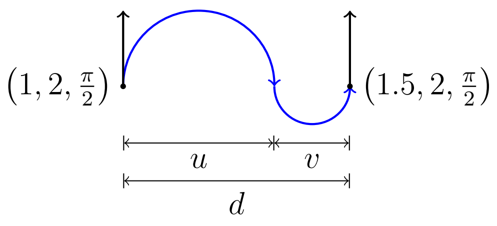
Exercise 1: Locomotion

A robot equipped with a differential drive starts at position $x = 1.0m$, $y = 2.0m$ and with heading $\theta = \frac{\pi}{2}$. It has to move to the position $x = 1.5m$, $y = 2.0m$, $\theta = \frac{\pi}{2}$ (all angles in radians). The movement of the vehicle is described by steering commands (v_l = speed of left wheel, v_r = speed of right wheel, t = driving time).

(a) *What is the minimal number of steering commands (v_l, v_r, t) needed to guide the vehicle to the desired target location?*

Each command of a differential drive only allows us to follow a portion of a circular trajectory (or to go straight, which is in the limit of infinite radius). While it is possible to reach the second location (x, y) with a single half-circle, it is impossible to do so with the correct orientation. We thus require at least two commands to reach it.

One of the possible ways to do so is to follow two half-circles, one with diameter u, the other with diameter v, such that their sum equals the distance between the poses, i.e. $u + v = d = 0.5\text{m}$.



Do note that this is valid even when setting $u = 0$ or $v = 0$ (rotation on the spot).

(b) *What is the length of the shortest trajectory under this constraint?*

The robot performs two half-circles, one with diameter u and arc length s_1 and one with diameter v and arc length s_2. The diameters u and v can be arbitrarily chosen, as long as they add up to d.

$$s_1 = \frac{\pi u}{2} \qquad s_2 = \frac{\pi v}{2} \qquad u + v = d$$

The length of the trajectory is the sum of the arc lengths of both half-circles:

$$s = s_1 + s_2 = \frac{\pi u}{2} + \frac{\pi v}{2} = \frac{\pi(u+v)}{2} = \frac{\pi d}{2} = \frac{\pi}{4} m$$

The trajectory length is independent of the particular u and v chosen.

(c) *Which sequence of steering commands guides the robot on the shortest trajectory to the desired location if an arbitrary number of steering commands can be used? The maximum velocity of each wheel is v and the distance between both wheels is l.*

The traveled distance cannot be smaller than the Euclidean distance between the two poses. We can achieve this lower bound by rotating the robot right by $\frac{\pi}{2}$, going forward for $0.5\mathrm{m}$ and rotating left by $\frac{\pi}{2}$, for a total of three commands. For the turning commands on the spot, both wheels need to turn at opposite speeds (to the right: $v_l = v$ and $v_r = -v$. To the left, $v_l = -v$ and $v_r = v$). To go straight, both wheels need to turn at the same speed.

From the differential drive kinematics, we know that the translational and rotational speeds v_{rob} and ω_{rob} of the robot depend on the wheel speeds v_l and v_r and the distance l between both wheels:

$$v_{\text{rob}} = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{t} = \frac{v_r + v_l}{2} \qquad \omega_{\text{rob}} = \frac{\Delta\theta}{t} = \frac{v_r - v_l}{l}$$

We can use the equations above to calculate the required time t to travel a certain rotation $\Delta\theta$ or distance $\sqrt{\Delta x^2 + \Delta y^2}$.

The shortest trajectory can be achieved by the following steering commands (v_l, v_r, t):

(1) $\left(v, -v, \frac{\pi l}{4v}\right)$

(2) $\left(v, v, \frac{d}{v}\right)$

(3) $\left(-v, v, \frac{\pi l}{4v}\right)$

(d) *What is the length of this trajectory?*

It's the Euclidean distance between the two poses, $0.5\mathrm{m}$.

Exercise 2: Differential Drive Implementation

Write a function in Python that implements the forward kinematics for the differential drive as explained in the lecture.

(a) *The function header should look like*

```
function [x_n y_n theta_n]=diffdrive(x, y, theta, v_l, v_r, t, l)
```

where x, y, and θ is the pose of the robot, v_l and v_r are the speed of the left and right wheel, t is the driving time, and l is the distance between the wheels of the robot. The output of the function is the new pose of the robot x_n, y_n, and θ_n.

```python
import numpy as np

def diffdrive(x, y, theta, v_l, v_r, t, l):

    # straight line
    if (v_l == v_r):
        theta_n = theta
        x_n = x + v_l * t * np.cos(theta)
        y_n = y + v_l * t * np.sin(theta)

    # circular motion
    else:
        # Calculate the radius
        R  = l/2.0 * ((v_l + v_r) / (v_r - v_l))

        # computing center of curvature
        ICC_x = x - R * np.sin(theta)
        ICC_y = y + R * np.cos(theta)

        # compute the angular velocity
        omega = (v_r - v_l) / l

        # computing angle change
        dtheta = omega * t

        # forward kinematics for differential drive
        x_n  = np.cos(dtheta)*(x-ICC_x) - np.sin(dtheta)*(y-ICC_y) + ICC_x
        y_n  = np.sin(dtheta)*(x-ICC_x) + np.cos(dtheta)*(y-ICC_y) + ICC_y
        theta_n = theta + dtheta

    return x_n, y_n, theta_n
```

(b) *After reaching position $x = 1.5m$, $y = 2.0m$, and $\theta = \frac{\pi}{2}$ the robot executes the following sequence of steering commands:*

(a) $c_1 = (v_l = 0.3m/s, v_r = 0.3m/s, t = 3s)$

(b) $c_2 = (v_l = 0.1m/s, v_r = -0.1m/s, t = 1s)$

(c) $c_3 = (v_l = 0.2m/s, v_r = 0m/s, t = 2s)$

Use the function to compute the position of the robot after the execution of each command in the sequence (the distance l between the wheels of the robot is $0.5m$).

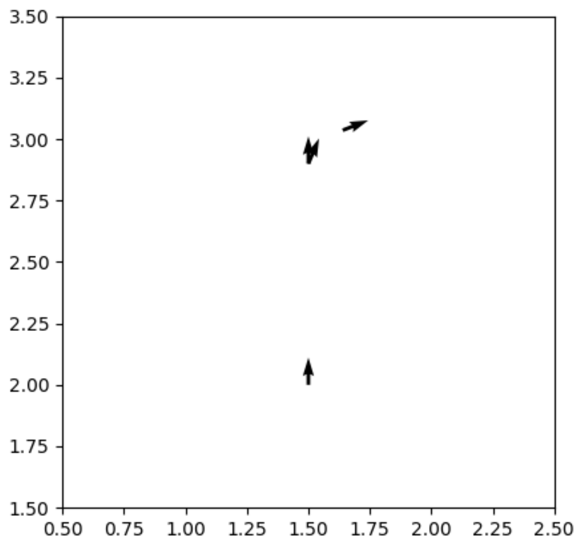
```python
#!/usr/bin/env python

import numpy as np
import matplotlib.pyplot as plt
from diffdrive import diffdrive

```

```python
plt.gca().set_aspect('equal')

# set the distance between the wheels and the initial robot position
l = 0.5
x, y, theta = 1.5, 2.0, (np.pi)/2.0

# plot the starting position
plt.quiver(x, y, np.cos(theta), np.sin(theta))
print("starting pose:  x: %f, y: %f, theta: %f" % (x, y, theta))

# first motion
v_l = 0.3
v_r = 0.3
t = 3
x, y, theta = diffdrive(x, y, theta, v_l, v_r, t, l)
plt.quiver(x, y, np.cos(theta), np.sin(theta))
print("after motion 1: x: %f, y: %f, theta: %f" % (x, y, theta))

# second motion
v_l = 0.1
v_r = -0.1
t = 1
x, y, theta = diffdrive(x, y, theta, v_l, v_r, t, l)
plt.quiver(x, y, np.cos(theta), np.sin(theta))
print("after motion 2: x: %f, y: %f, theta: %f" % (x, y, theta))

# third motion
v_l = 0.2
v_r = 0.0
t = 2
x, y, theta = diffdrive(x, y, theta, v_l, v_r, t, l)
plt.quiver(x, y, np.cos(theta), np.sin(theta))
print("after motion 3: x: %f, y: %f, theta: %f" % (x, y, theta))

plt.xlim([0.5, 2.5])
plt.ylim([1.5, 3.5])

plt.savefig("poses.png")
plt.show()
```

The resulting poses are:

```
(1.500000,2.900000,1.570796)
(1.500000,2.900000,1.170796)
(1.639676,3.035655,0.370796)
```

Represented as a set of vectors, together with the initial pose:


3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2.25
2.50