Experiment 4

Obstacle Avoidance Control

Time to Complete: To be announced in class Marks: 20 (prelab 15, lab 5)

Purpose

The problem of navigating the 3WD robot to a goal position while avoiding collision with obstacles is considered in this experiment.

Objectives

By the end of this experiment, you will be able to:

1. Convert the raw readings of the IR sensors into distance measurements.
2. Implement a control strategy for the 3WD robot to do simple obstacle avoidance while navigating to a goal position.

4.1 Obstacle Avoidance Controller Design

We will begin with the following obstacle avoidance strategy [1]:

- Move the robot towards its final goal position until an obstacle is detected by the IR sensors.
- If an obstacle is detected, change the goal position temporarily until the obstacle is cleared.

- Restore the goal position once the obstacle is cleared and repeat the cycle until the final goal position is reached.

For simplicity, the final orientation of the robot will not be considered here.

Consider Figure 4.1, where it is assumed that the robot is moving from the current position (x, y) towards a final goal position of (x_f, y_f) with linear velocity $V = [\dot{x}, \dot{y}]^T$ and angular velocity $\omega = \dot{\theta}$.

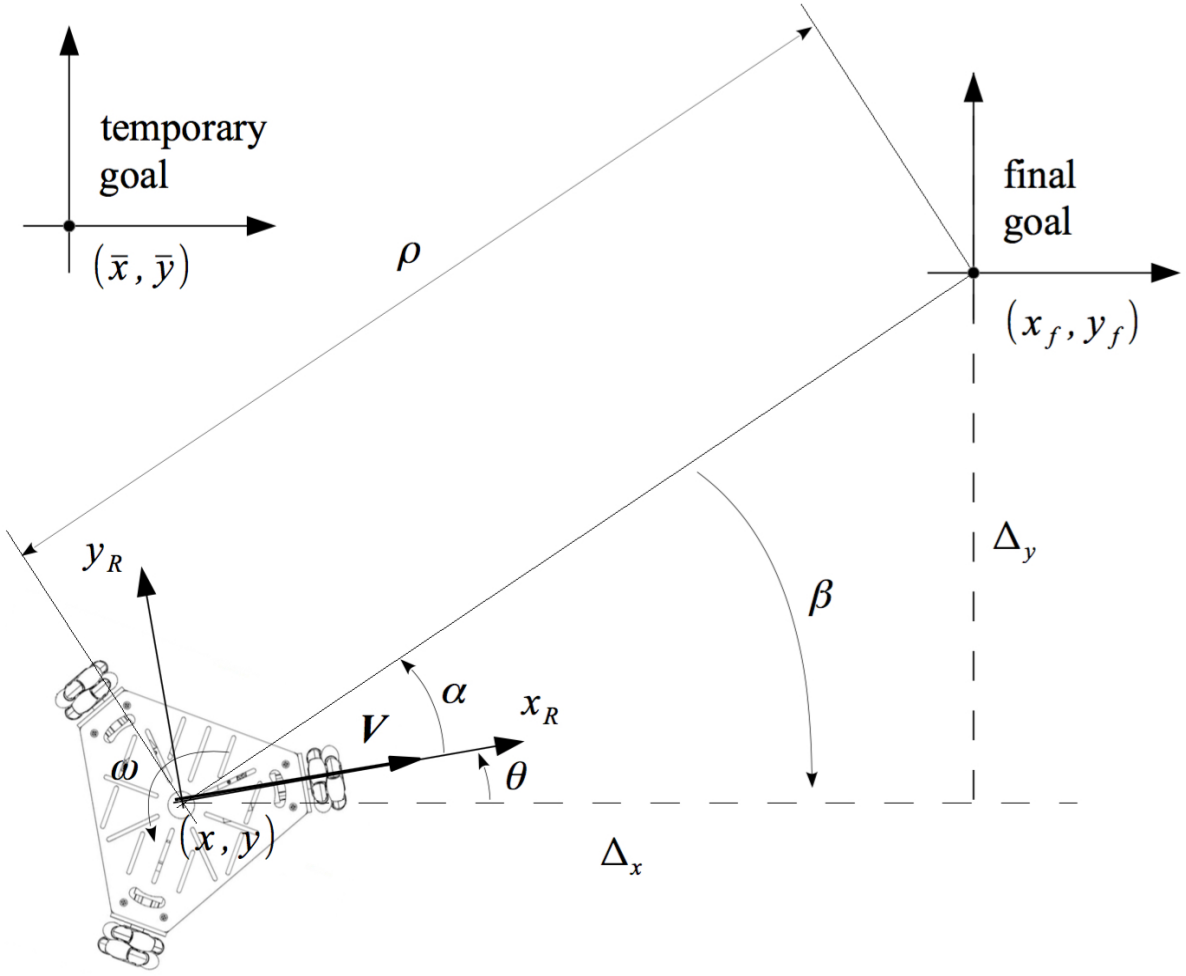

Figure 4.1: 3WD Robot and reference frames.

Using the notations in Figure 4.1, define the *distance error*, ρ, as the distance of the robot from its current position to the final goal position, i.e.

$$\rho = \sqrt{\Delta_x^2 + \Delta_y^2} = \sqrt{(x_f - x)^2 + (y_f - y)^2} \tag{4.1}$$

and the *heading error*, α, is defined as

$$\alpha \;=\; \text{atan2}(\Delta_y, \Delta_x) \;-\; \theta. \tag{4.2}$$

Also, the angle β is

$$\beta \;=\; -\theta - \alpha. \tag{4.3}$$

If an obstacle is detected then the final goal position (x_f, y_f) in the above equation will be replaced a temporary goal position $(\bar{x}, \bar{y})$ and the heading error α will be replaced by temporary heading error $\bar{\alpha}$.

The obstacle avoidance controller will be designed using the following steps:

1. Determine the temporary goal position once an obstacle is detected, and to switch between the final and temporary goal positions.

2. Design a controller to drive ρ and α to 0 given any goal position (final or temporary).

4.1.1 Step 1: Collision Avoidance

The distance measurements from the IR sensors will be used to determine the temporary goal position $(\bar{x}, \bar{y})$ and heading error $\bar{\alpha}$. The 3WD robot is equipped with 6 IR sensors, $S_0, \ldots, S_5$. These sensors are mounted on a circular plate on top of the robot as shown by the sensor frames $\mathcal{F}^{S_i} = o_{S_i} x_{S_i} y_{S_i},\ (i = 0, \ldots, 5)$ in Figure 4.2.

Let the distance measurement returned by sensors be d_i $(i = 0, \ldots, 5)$, then we assume the robot is at risk of collision if

$$\min_i d_i \;\leq\; d_{min}, \tag{4.4}$$

where d_{min} is a pre-defined *minimum safe distance* from obstacle. The value of d_{min} depends on several factors, such as the range of the sensors, how fast the robot can move, how fast the sensor readings are processed, etc. In our case, the value of d_{min} is set to 0.08m or 8cm. Similarly, we assume that the robot is free from collision if

$$\min_i d_i \;\geq\; d_{free}, \tag{4.5}$$

where d_{free} is a pre-defined *collision-free distance*. In our case, the value of d_{free} is set to be $1.25 \times d_{min}$.

Consider a typical sensor frame $\mathcal{F}^{S_i}$ in Figure 4.3 where $\delta_i^{S_i}$ is defined to be a vector of length d_i along the x_{S_i} axis and is represented by the following homogeneous coordinate:

$$\delta_i^{S_i} \;=\; \begin{bmatrix} d_i \\ 0 \\ 1 \end{bmatrix} \quad \text{(w.r.t. the sensor frame } \mathcal{F}^{S_i}\text{)}. \tag{4.6}$$

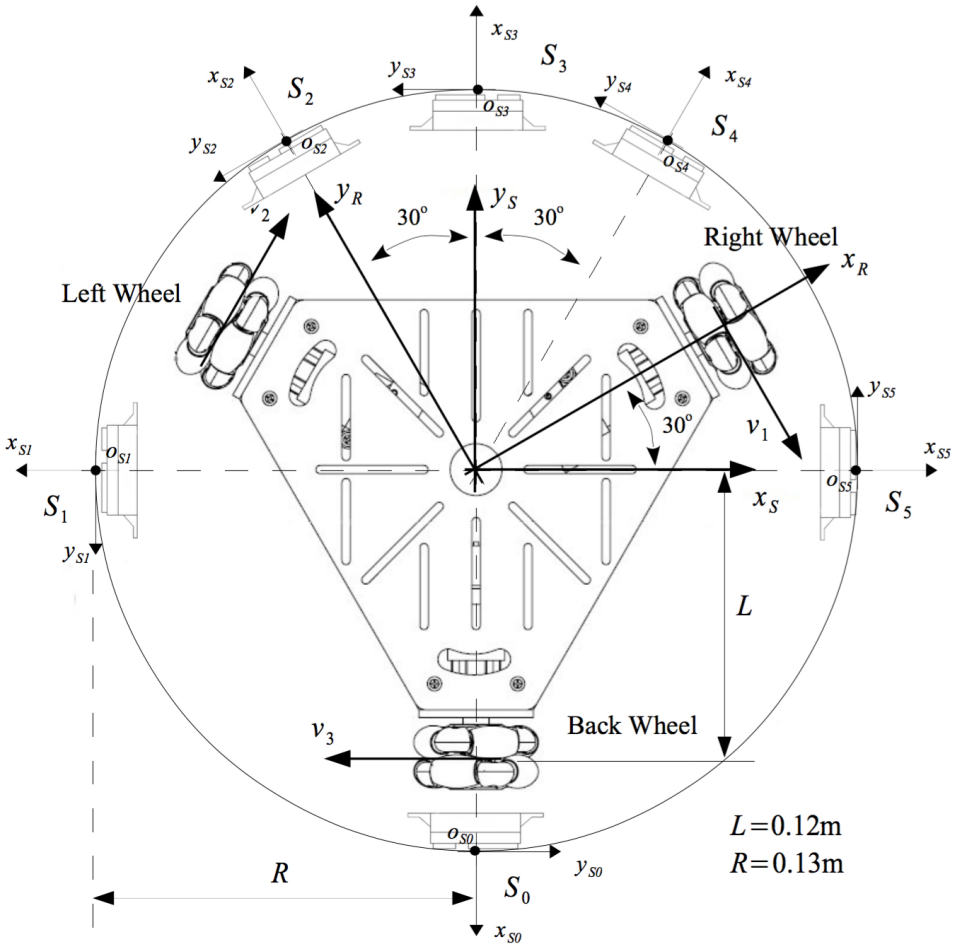

Figure 4.2: IR sensors locations.

The z component of the homogeneous coordinate is omitted here because the motion of the robot is planar only. Now, if $d_i \leq d_{min}$ then the vector $\delta_i^{S_i}$ is regarded as a direction for which an obstacle is present and the robot should not head toward that direction. On the other hand, if $d_i \geq d_{free}$ then the vector $\delta_i^{S_i}$ is regarded as a collision-free direction for the robot. Therefore, if an obstacle has been detected then a *temporary goal vector* δ_{goal}^0 (w.r.t. the base frame $\mathcal{F}^0$) can be determined as the vector sum of all the collision-free direction vectors $\delta_i^{S_i}$ as follows:

$$\delta_{goal}^0 := \begin{bmatrix} \bar{x} \\ \bar{y} \\ 1 \end{bmatrix} = H_R^0 \delta_{goal}^R, \tag{4.7}$$

where H_R^0 is the H-matrix from the robot frame $\mathcal{F}^R$ to the base frame $\mathcal{F}^0$, δ_{goal}^R is the

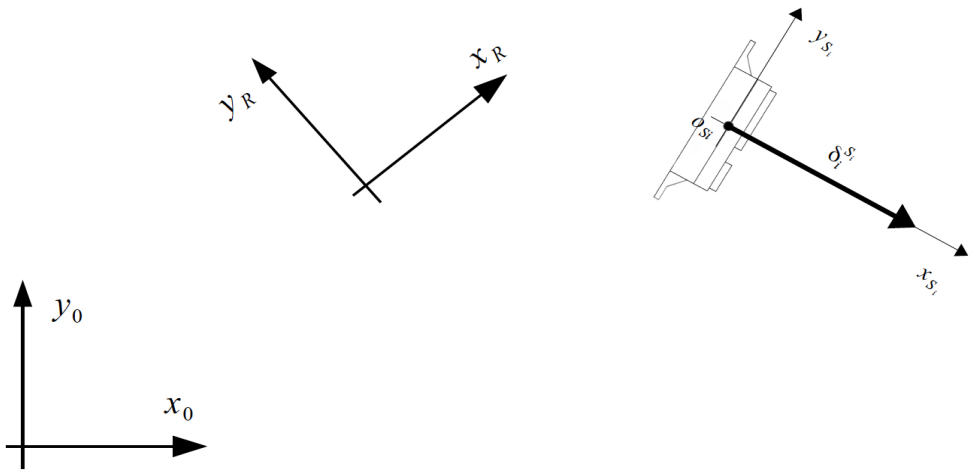

Figure 4.3: A typical sensor frame.

temporary goal vector w.r.t. the robot frame $\mathcal{F}^R$ and can be determined from:

$$\delta^R_{goal} \;=\; \sum_{\{\delta_i^R \;s.t.\; d_i \geq d_{free}\}} \delta_i^R \;=\; \sum_{\{\delta_i^{S_i} \;s.t.\; d_i \geq d_{free}\}} H^R_{S_i}\delta_i^{S_i}, \tag{4.8}$$

where $\delta_i^{S_i}$ are sensor measurement vectors defined in Eq. (4.6) and $H^R_{S_i}$ are the H-matrices from the sensor frames $\mathcal{F}^{S_i}$ to the robot frame $\mathcal{F}^R$. Since the sensors are fixed on the robot, the matrices $H^R_{S_i}$ $(i = 0, \ldots, 5)$ are constant and can be pre-determined (see Item 1 in Section 4.2). The matrix H^0_R is simply the H-matrix from the robot frame robot frame $\mathcal{F}^R$ to the base frame $\mathcal{F}^0$ and can be determined from the robot's current posture $p = [x, y, \theta]^T$ as follows:

$$H^0_R \;=\; \begin{bmatrix} \cos(\theta) & -\sin(\theta) & x \\ \sin(\theta) & \cos(\theta) & y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.9}$$

The collision avoidance algorithm is summarized below:

1. Obtain the sensor measurements $d_i,\ i = 0, \ldots, 5$.

2. Use Eq. (4.4) to determine if there is a risk of collision.

3. If there is risk of collision, then determine a temporary goal position $(\bar{x}, \bar{y})$ using Eq. (4.7) as well as the temporary distance error $\bar{\rho}$ using

$$\bar{\rho} \;=\; \sqrt{\bar{\Delta}_x^2 \,+\, \bar{\Delta}_y^2} \;=\; \sqrt{(\bar{x} - x)^2 \,+\, (\bar{y} - y)^2} \tag{4.10}$$

4. Determine the temporary heading error $\bar{\alpha}$ using

$$\bar{\alpha} \;=\; \text{atan2}(g_y, g_x) \;-\; \theta, \tag{4.11}$$

where g_x and g_y are, respectively, the x and y components of the vector δ^R_{goal} defined in Eq. (4.8).

5. Use $\bar{\rho}$ and $\bar{\alpha}$ to determine the control values to move the robot until the obstacle is clear.

4.1.2 Step 2: Exponential Stabilizing Controller

In this section a controller is introduced to drive the robot to its goal position (final or temporary). This controller has the property that it makes the closed-loop system to be *exponentially stable* [2]. Details on this controller have been discussed in the lecture and will not be repeated here.

Using the results in [2], the desired angular velocities to drive the robot to its goal position are determined as follows:

1. The desired linear and angular velocities *of the robot* are first determined using the following linear control law:

$$v \;=\; k_\rho \rho \tag{4.12}$$
$$\omega \;=\; k_\alpha \alpha \;+\; k_\beta \beta \tag{4.13}$$

where ρ, α are defined as before, and $\beta = -\theta - \alpha$. The constants k_ρ, k_α and k_β are scalar controller gains such that

$$k_\rho > 0, \quad k_\beta < 0, \quad k_\alpha > \frac{2}{\pi} k_\rho \;-\; \frac{5}{3} k_\beta.$$

If an obstacle is detected, then $\bar{\rho}$ and $\bar{\alpha}$ are used instead to determine v and ω.

2. Once v and ω are determined, the desired wheel angular velocities $\dot{\Phi}$ are determined from the inverse kinematics equation (2.1) with $\dot{p}$ determined from:

$$\dot{p} \;=\; \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \;=\; \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix}.$$

3. The desired wheel angular velocities $\dot{\Phi}$ are then executed using the `set_angular_velocities()` method.

4.2 Preparatory Work

1. The collision avoidance algorithm discussed earlier requires the computation of the temporary goal vector δ^0_{goal}. In order to do this, the vector δ^R_{goal} must be transformed to the base frame $\mathcal{F}^0$ and the vectors $\delta^{S_i}_i$ transformed to the robot frame (see Eqs. (4.7) and (4.8)). This requires knowledge of the H-matrix H^0_R and $H^R_{S_i}$. The H-matrix H^0_R can be determined during run time using Eq. (4.9). The matrices $H^R_{S_i}$ are constant and depend only on how the sensors are mounted on the robot. Use the sensor locations in Figure 4.2 as a guide, show that the origin of the sensor frame $\mathcal{F}^{S_i}$, i.e. o_{S_i} and the angle from x_R to x_{S_i}, namely θ_{S_i}, with respect to the robot frame $\mathcal{F}^R$ are given by the following Table:

Table 4.1: Sensor frame locations (with respect to robot frame $\mathcal{F}^R$)

Sensor S_i	x coord. of o_{S_i} $x_{o_{S_i}}$	y coord. of o_{S_i} $y_{o_{S_i}}$	θ_{S_i}
S_0	$-\frac{1}{2}R$	$-\frac{\sqrt{3}}{2}R$	$-\frac{2}{3}\pi$
S_1	$-\frac{\sqrt{3}}{2}R$	$\frac{1}{2}R$	$\frac{5}{6}\pi$
S_2	0	R	$\frac{1}{2}\pi$
S_3	$\frac{1}{2}R$	$\frac{\sqrt{3}}{2}R$	$\frac{1}{3}\pi$
S_4	$\frac{\sqrt{3}}{2}R$	$\frac{1}{2}R$	$\frac{1}{6}\pi$
S_5	$\frac{\sqrt{3}}{2}R$	$-\frac{1}{2}R$	$-\frac{1}{6}\pi$

2. The homogeneous transformation matrix for a planar motion of a rotation of amount $\Delta\theta$ about the z-axis and a translation of Δx along the x-axis and a translation of Δy along the y-axis can be represented by the following 3×3 matrix:

$$H := \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & \Delta x \\ \sin(\Delta\theta) & \cos(\Delta\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.14}$$

   Let $q = [\Delta x, \Delta y, \Delta\theta]^T$, extend the `myRobot` module by adding a function `HMatrix(q)` that returns the H-matrix in Eq. (4.14) given q. This function can then be used to obtain the matrices H^0_R and $H^R_{S_i}$ required Eqs. (4.7) and (4.8). For example,

```
HR0 = HMatrix([x, y, theta])
```

3. The IR sensors on the robot are analog distance measurement sensors. These sensors provide an analog voltage output based on the measured distance. Unfortunately,

the relationship between the sensor's output voltage and the measured distance is nonlinear. In addition, the analog voltage reading V_S from the sensor must also be converted using an analog to digital converter (ADC) before it can be used by the digital controller.

In order to determine the relationship between the sensor's output voltage and measured distance d, an experiment using the setup in Figure 4.4 was carried out. The

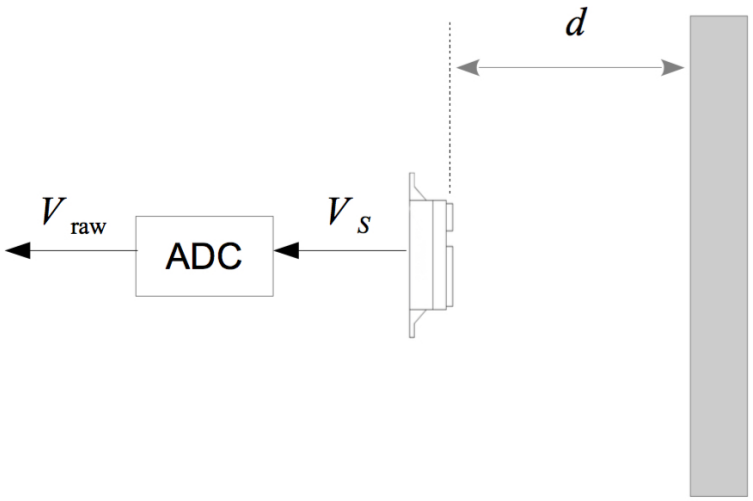

Figure 4.4: Sensor experiment.

distance d was varied from 3 cm to 40 cm and the corresponding values of V_{raw} were obtained in Table 4.2 and plotted in Figure 4.5.

Table 4.2: Distance versus raw sensor values

V_{raw}	3106	2727	1622	1084	662	425	284	186	121	76
d (m)	0.03	0.04	0.07	0.10	0.15	0.20	0.25	0.30	0.35	0.40

Figure 4.5 suggests that d is some kind of exponential function of V_{raw}. In fact, the following function was found to be an excellent fit to the data:

$$d \;=\; c_1 e^{-c_2\sqrt{V_{\text{raw}}}}. \tag{4.15}$$

Now, use the data in Table 4.2 and the Method of Least Squares, determine the constants c_1 and c_2 in Eq. (4.15).

4. Extend the `myRobot` module by implementing Eq. (4.15) as the function

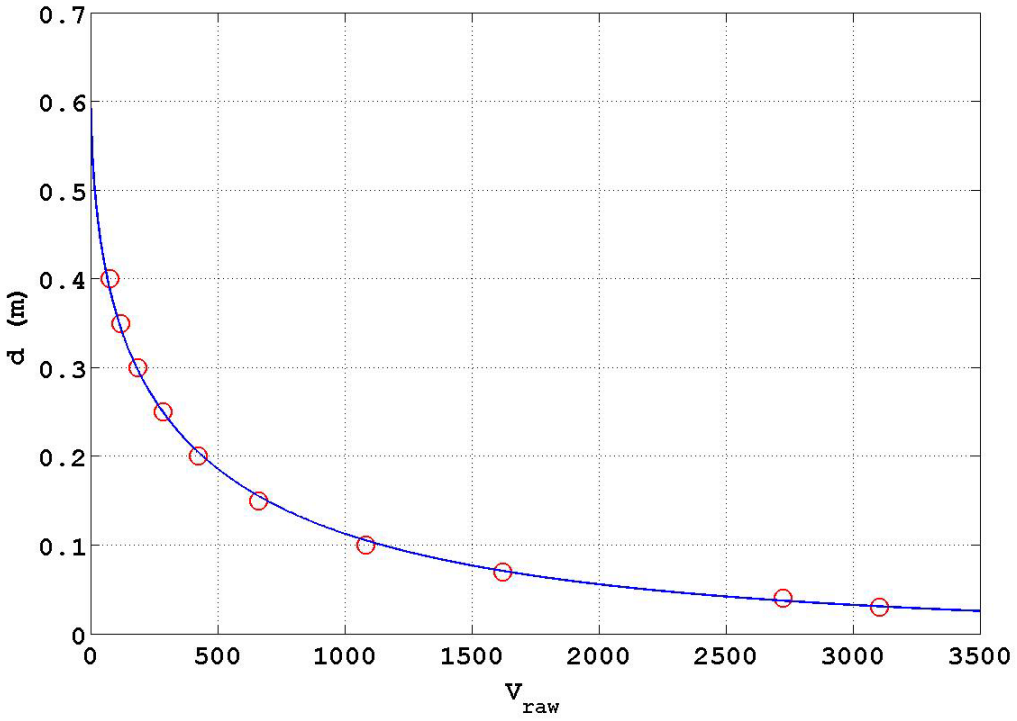

Figure 4.5: d versus V_{raw}

```
distance = Vraw_to_distance(Vraw)
```

Once this function is defined, it can then be used to obtain distance measurements from the IR sensors as follows:

```
for i in range(0,6):
   d[i] = Vraw_to_distance(myRobot.ir_sensors_raw_values[i])
```

5. Implement the exponentially stabilizing controller and the obstacle avoidance strategy using the program template `lab4demo.py` available from the course web page as a guide.

4.3 Laboratory Work

Demo 4.1 *Obstacle Avoidance with Exponentially Stabilizing Controller*
Using $k_\rho = 1.0$, $k_\beta = -1.0$, run your controller program to move the robot to the final goal position of $(0.7, 1.0)$. Run the program first with no obstacle present and record the path taken by the robot. Next put an obstacle at the location specified by the laboratory instructor and run the program again. To be successful, the robot must not collide with the obsatcle and must reach the final goal position within ± 1 cm in the final x and y position.

4.4 What to Submit

A report is required for this experiment. Requirements for the report are specified in Appendix A. In addition, archive the Python code developed as a .tar file (using your last name as the file name) and email it as an attachment to `ychen@ee.ryerson.ca`. For example,

```
tar cvf lastname.tar lab4demo.py myRobot.py
```

The .tar file must be sent by the end of the laboratory session.

4.5 Reference

1. M. Egerstedt, "Control of Autonomous Mobile Robots," in *Handbook of Networked and Embedded Control*, D. Hristu and B. Levine, Eds., Birkhauser, Boston, MA, 2005.

2. C. Canudas de Wit and O.J. Sordalen, "Exponential Stabilization of Mobile Robots with Nonholonomic Constraints," in *IEEE Transactions on Automatic Control*, 37(11), 1992.