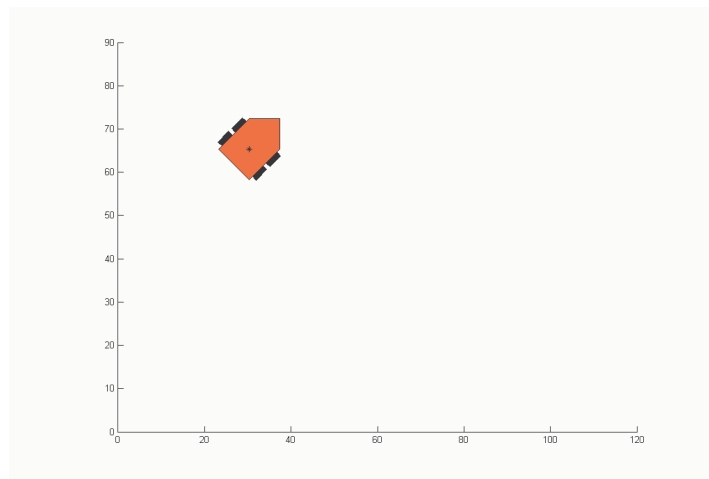


CSE 397/498-013
Introduction to Mobile Robotics

Homework: Rigid Transformations, Kinematics, and intro to Odometry
Report Due Date: Thursday, 15 Sep 05 submitted via Blackboard PRIOR TO THE START OF CLASS



A. Objectives:

1. Review rotation matrices
2. Introduction to Matlab (possibly)
3. Review kinematic models for vehicle motion
4. Investigate the effects of small biases on odometry performance.

B. Requirements:

1. You **MUST** use Matlab to complete this assignment.
2. This is an individual assignment. Each student is required to submit his/her own work in order to receive credit.

C. Part 1: Rotation Matrixes

1. For each of the following matrixes, determine if it is a valid rotation matrix.
2. You **MUST** justify your responses. If you answer yes/no, you will get a 0/0 for the question

a.
$$\begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

b.
$$\begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

c.
$$\begin{bmatrix} a & b \\ b & a \end{bmatrix}$$

$$d. \begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix} \quad e. \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}$$

D. Make/Move Robot Functions:

1. You are to make `makeRobot` and `moveRobot` functions. The prototypes for each should be as follows:

```
function robot = makeRobot(x, y, theta)
function robot = moveRobot(robot, x, y, theta)
```

The arguments correspond position (x,y) and orientation (theta) in the inertial/global reference frame. `robot` is itself a structure. You can instantiate this dynamically in Matlab without having to declare the variable. For example, I could write:

```
robot.x = x;
robot.y = y;
robot.theta = theta; % orientation
```

and the instance of `robot` is automatically created. The structure now has elements, `x`, `y`, `theta`.

2. When `makeRobot` is called, the robot will be created on the active figure at the position and orientation specified. You can use whatever level of detail you choose to draw your robot. However, I MUST be able to infer its orientation.
3. When `moveRobot` is called, the original robot is deleted from the figure and then redrawn to the new location/pose.
4. Some helpful functions:
 - a. `h = patch()` creates a polygon with coordinates specified as arguments. `h` is the handle to the object. You would need this for something like...
 - b. `delete(h)` which deletes the object with the specified handle

E. Kinematic Modeling:

1. You are to generate a kinematic model for a differential drive robot.
2. The robot is to have a wheel radius of 10cm, and a drive axle of 0.5 meters from wheel center to wheel center.
3. You will then write a function with the following prototype:

```
function hwk1d(x,y,theta,inputFile)
```

where the initial three elements are the global pose of the robot in the inertial frame, and `inputfile` is the name of a file consisting of the wheel velocities (in rad/sec) for the right and left wheels. A sample file format can be found here.

4. Your function will:
 - a. Create the robot at the initial pose specified in the function arguments using the `makeRobot` function from part C.
 - b. Read in the wheel velocities (sampled at 1 second intervals).
 - c. Use these to update the pose of the robot using the `moveRobot` function written in part C above.
5. Some helpful Matlab functions:
 - a. `importdata`
 - b. `pause(x)` will pause the program for x seconds. This is useful for animating the robot motion.
6. You can use the sample input file to test the functionality of your program. If it is working correctly, the robot should move at constant speed in a circle going counterclockwise.
7. You should also test your program with other input files to ensure that your kinematic model is working correctly. For example, how would you test pure translation? pure rotation? etc.

F. Odometry with bias:

1. In this exercise, we will test the effects of wheel bias on encoders used for odometry. The role of the encoders is to capture the wheel velocity - that is, how much the wheel has rotated in a given amount of time.
2. Let us assume that the right wheel above is 1% over spec in radius (10.1cm). While we will assume that the encoders capture the amount of wheel rotations perfectly, they can do nothing to compensate for the wheel bias.
3. Your robot is to start at initial pose (10,10,0). It is then to:
 - a. Move straight 50 meters
 - b. Turn clockwise 90 degrees
 - c. Move straight 50 meters
 - d. Turn clockwise 90 degrees
 - e. Move straight 50 meters
 - f. Turn clockwise 90 degrees
 - g. Move straight 50 meters
 - h. Turn clockwise 90 degrees

If the odometry were perfect, then the final pose should also be (10,10,0)

4. You must:
 - a. Generate the necessary input file to model the motions above assuming the correct wheel size (you may assume the wheel velocities are constant over each motion segment).
 - b. Use this as input to function `hwk1d`.
 - c. Determine the final pose of the biased robot. How large is the error? What are your thoughts on how you could compensate for this error?

- d. Repeat the above exercise with the bias now placed on the left wheel. Run the same input file. What is the pose error now? Explain the differences in performance.

G. Turn in:

1. A write-up, to include images from all simulation trials as well as your Matlab source code.
2. The input file you generated for the odometry test.
3. Discussion of the differences in the pose error results when the left or right wheel was biased.