Homework Assignment 1:
Differential Drive Kinematics

Overview

In this assignment, you will implement differential drive forward kinematics in order to control your robot with the keyboard.

Submit your code files and write-up PDF or Word Doc to Canvas.

`template.py` contains the basic structure for your programs – it sets up the graphical user interface and connection to the Sparki robot over Bluetooth. To run in simulation mode:

```
python template.py
```

To connect to Sparki:

```
python template.py –port /dev/tty.Sparki.DevB
```

or

```
python template.py –port COM4
```

Replace the port name with the appropriate port for your machine.

For your assignments, copy `template.py` to a new file and add the requested functionality.

I recommend that you create a Robot class to maintain the state of the robot and implement functionality such as drawing the robot and integrating the pose over time (forward kinematics).

I included a file `RobotLib/Math.py` which contains helper functions for creating and multiplying transformation matrices and points. `math_example.py` shows how to use it.

Note that `RobotLib/IO.py` has been updated since HW0. Please do not use the old code from HW0.

Robot dimensions

Here are some measurements of the Sparki robot which you can use in your code:

Steps per revolution of a wheel: 4096
Wheel diameter: 5 cm
Distance between wheels: 8.51 cm
Width of robot (side-to-side): about 9 cm
Length of robot (front-to-back): about 10 cm
Offset from center of robot to sonar: about 2.5 cm

The motors are controlled by a speed setting that ranges from 0-100%. 100% = 1000 steps/sec, and the relationship is roughly linear (90% = 900 steps/sec, 80% = 800 steps/sec, and so on.)

Assignment components

1. Control the robot using the keyboard and send appropriate motor settings (20 points).

i. Use the arrow keys to control the robot's linear and angular velocity.

   The up/down keys should control the linear velocity and the left/right keys should control the angular velocity.

   Don't set the velocities too high or the motor speeds will max out at 100%.

ii. Calculate the left and right wheel velocities according to the differential drive forward kinematic equations given in class.

   In the MyFrontEnd class, send the current wheel speeds and directions to the robot with `self.sparki.send_command()` in the `update()` function.

   You will need to convert the desired wheel velocities in rad/sec to steps/sec. The numbers given above will help.

2. Maintain the robot pose by integrating over time (30 points).

i. Maintain a robot-to-map transformation matrix and calculate the updates by right-multiplying the robot's transformation matrix by movement matrices as explained in class.

ii. When the angular velocity is non-zero, the robot should turn in a circle.

iii. When the angular velocity is zero, the robot should move in a straight line.

3. Draw the robot on the screen. (20 points)

i. Draw the robot on the screen as a rectangle centered at its current position and rotated with its current orientation.

   The rectangle corner points should be calculated using the robot-to-map transformation matrix.

ii. Draw a red line indicating the current sonar reading, using the chain of transformation matrices (sonar-to-robot, robot-to-map).

   In the MyFrontEnd class, the current sonar reading is stored in `self.sparki.dist`.

   The line should start from the sonar position, be oriented according to the sonar direction, and have length equal to the current sonar reading.

4. Evaluate your robot navigation system. (30 points)

In this step, you will evaluate your system and present results in a write-up document. Submit the write-up as a Word doc or PDF.

You will need a pen and some paper, a ruler, and a protractor. Stick the pen in the center of Sparki to draw lines.

i. Move the robot forward 10 cm in differing amounts of time (1 second, 0.5 seconds, etc.).

   Do this by setting the wheel velocities, waiting `t` seconds using `time.sleep(t)` and then setting the wheel velocities to zero.

   Measure the observed error in movement by measuring the line that Sparki drew with a ruler. Make multiple measurements at each velocity setting for better accuracy.

   Make a plot of error vs. velocity (e.g. using Excel or Google Sheets, or matplotlib in Python).

ii. Move the robot forward 10cm, rotate the robot 90 degrees, and move forward 10cm again. Complete the rotation in differing amounts of time.

Measure the error in the turn by measuring the angle between the two lines using a protractor. Make a plot of error vs. velocity.

iii. Analyze the results. How does error relate to velocity in your experiments? Name some possible causes for the errors and some possible improvements that would reduce error.