Programming Exercises

Programming Exercises (Optional)

These seven programming exercises (one for each week) are **optional**, and will focus on implementing what you have learned in the video lectures. You will be provided with a robot simulator, which will allow you to design and test controllers for a mobile robot. This robot simulator is based on MATLAB, so you will need to install MATLAB to complete the programming exercises. You can buy MATLAB directly from MathWorks here, and make sure to pay attention to these two details:

1. Buying MATLAB will require you to create an account. When you create this account, you **must** select "Student Use of MATLAB and Simulink Student Version" when prompted for "How will you use MathWorks software?"
2. In the "University" field, you **must** list "Coursera" as your university.

If you register your account in this way and upload proof that you are taking this course during registration, you will be eligible to buy the MATLAB Student version. A screen shot of the courses you are taking on Coursera is acceptable proof. Click on your name in the top-right corner of the Coursera website and select "Courses."

The robot simulator ...

- will work with both the Student version and the standard version.
- will not require any additional toolboxes.
- will also work with older versions of MATLAB (R2009a or later), if you already own an older version.

Note: For some of the weeks, a standalone version of the simulator will be provided that does not require MATLAB. This standalone simulator will allow you to tweak the design of controllers via a XML file without any programming.

Week 1

This week's exercises will help you learn about MATLAB and robot simulator:

1. Since the programming exercises involve programming in MATLAB, you should familiarize yourself with MATLAB and its language. Point your browser to http://www.mathworks.com/academia/student_center/tutorials, and watch the interactive MATLAB tutorial.
2. Familiarize yourself with the simulator by reading the manual posted here and downloading the robot simulator posted here.

Week 2

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-2.zip (Updated: 2013-02-04)
2. manual-coursera-sp13.pdf (Updated: 2013-02-01)

Before you can design and test controllers in the simulator, you will need to implement three components of the simulator:

1. Implement the transformation from unicycle dynamics to differential drive dynamics, i.e. convert from (v, ω) to the right and left angular wheel speeds (v_r, v_l).
2. Implement odometry for the robot, such that as the robot moves around, its pose (x, y, θ) is estimated based on how far each of the wheels have turned. Assume that the robot starts at $(0, 0, 0)$.
3. Read the "IR Range Sensors" section in the manual and take note of the function $f(\Delta)$, which maps distances (in meters) to raw IR values. Implement code that converts raw IR values to distances (in meters).

Please make sure read the "Week 2" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 2 to test a PID controller that is designed to steer the robot to a desired orientation.

Week 3

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-3.zip (Updated: 2013-02-09)
2. manual-coursera-sp13.pdf (Updated: 2013-02-10)

Before you can drive to robot to different waypoints in the simulator, you will need to implement three components of the go-to-goal behavior:

1. Calculate the heading (angle), θ_g, to the goal location (x_g, y_g). Let u be the vector from the robot located at (x, y) to the goal located at (x_g, y_g), then θ_g is the angle u makes with the x-axis (positive θ_g is in the counterclockwise direction).
2. Calculate the error between θ_g and the current heading of the robot, θ.
3. Calculate the proportional, integral, and derivative terms for the PID regulator that steers the robot to the goal.

Please make sure read the "Week 3" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 3 to test a PID controller that is designed to drive the robot to a goal location.

Week 4

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-4.zip (Updated: 2013-02-15)
2. manual-coursera-sp13.pdf (Updated: 2013-02-15)

This week you will be implementing the different parts of a controller that steers the robot successfully away from obstacles to avoid a collision. This is known as the avoid-obstacles behavior. The IR sensors allow us to measure the distance to obstacles in the environment, but we need to compute the points in the world to which these distances correspond. You will be implementing the following strategy for obstacle avoidance:

1. Transform the IR distances to points in the world.
2. Compute a vector to each point from the robot, $u_1, u_2, \ldots, u_9$.
3. Weigh each vector according to their importance, $\alpha_1 u_1, \alpha_2 u_2, \ldots, \alpha_9 u_9$. For example, the front and side sensors are typically more important for obstacle avoidance while moving forward.
4. Sum the weighted vectors to form a single vector, $u_o = \alpha_1 u_1 + \ldots + \alpha_9 u_9$.
5. Use this vector to compute a heading and steer the robot to this heading.

This strategy will steer the robot in a direction with the most free space (i.e., it is a direction *away* from obstacles). For this strategy to work, you will need to implement three parts of an avoid-obstacles controller:

1. Transform the IR distance (which you converted from the raw IR values in Week 2) measured by each sensor to a point in the reference frame of the robot.
2. Transform the point in the robot's reference frame to the world's reference frame.
3. Use the set of transformed points to compute a vector that points away from the obstacle. The robot will steer in the direction of this vector and successfully avoid the obstacle.

Please make sure read the "Week 4" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 4 to test a PID controller that is designed to avoid collisions with obstacles.

Week 5

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-5.zip (Updated: 2013-02-22)
2. manual-coursera-sp13.pdf (Updated: 2013-02-22)

This week you will be making a small improvement to the go-to-goal and avoid-obstacle controllers and testing two arbitration mechanisms: blending and hard switches. Arbitration between the two controllers will allow the robot to drive to a goal, while not colliding with any obstacles on the way.

1. Implement a simple control for the linear velocity, v, as a function of the angular velocity, ω.

Add it to both *+simiam/+controller/GoToGoal.m* and *+simiam/+controller/AvoidObstacles.m*.
2. Combine your go-to-goal controller and avoid-obstacle controller into a single controller that blends the two behaviors. Implement it in *+simiam/+controller/AOandGTG.m*.
3. Implement the switching logic that switches between the go-to-goal controller and the avoid-obstacles controller, such that the robot avoids any nearby obstacles and drives to the goal when clear of any obstacles.
4. Improve the switching arbitration by using the blended controller as an intermediary between the go-to-goal and avoid-obstacles controller.

Please make sure read the "Week 5" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 5 to test the blending and switching arbitration between the go-to-goal and avoid-obstacles behaviors.

Week 6

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-6.zip (Updated: 2013-03-01)
2. manual-coursera-sp13.pdf (Updated: 2013-03-02)

This week you will be implementing a wall following behavior that will aid the robot in navigating around obstacles. Implement these parts in *+simiam/+controller/+FollowWall.m*.

1. Compute a vector, $u_{fw,t}$, that estimates a section of the obstacle (``wall'') next to the robot using the robot's right (or left) IR sensors.
2. Compute a vector, $u_{fw,p}$, that points from the robot to the closest point on $u_{fw,t}$.
3. Combine the two vectors, such that it can be used as a heading vector for a PID controller that will follow the wall to the right (or left) at some distance d_{fw}.

Please make sure read the "Week 6" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 6 to test the wall following behavior.

Week 7

Start by downloading the robot simulator and update manual for this week:

1. simiam-coursera-week-7.zip (Updated: 2013-03-18)
2. manual-coursera-sp13.pdf (Updated: 2013-03-15)

This week you will be combining the go-to-goal, avoid-obstacles, and follow-wall controllers into a full navigation system for the robot. The robot will be able to navigate around a cluttered, complex environment without colliding with any obstacles and reaching the goal location successfully. Implement your solution in *+simiam/+controller/+khepera3/K3Supervisor.m*.

1. Implement the *progress_made* event that will determine whether the robot is making any progress towards the goal.
2. Implement the *sliding_left* and *sliding_right* events that will serve as a criterion for whether the robot should continue to follow the wall (left or right) or switch back to the go-to-goal behavior.
3. Implement the finite state machine that will navigate the robot to the goal located at $(x_g, y_g) = (1, 1)$ without colliding with any of the obstacles in the environment.

Please make sure read the "Week 7" section in the manual for directions on where to add your code in the robot simulator and some additional help.

OR

Navigate to the standalone simulator for week 7 to test the full navigation system.

The End

Congratulations on working through all of the programming exercises. For your reference, you can download the simulator with a full navigation system for the mobile robot: simiam-coursera-sp13.zip. Thank you!

Created Sat 26 Jan 2013 1:53 PM PST -0800

Last Modified Fri 22 Mar 2013 9:21 AM PDT -0700