Advanced Special Topics  
CS 395  
Fall 2017  
Homework Final  
Due on Sunday, Dec. 10. (Please see the notes below!)

- The Programming Assignment should be submitted electronically to our course drop-box by 11:59pm on Sunday, Dec. 10. Please use the submit395 command, as usual:

  ```
  submit395 hwfinal <your-hwfinal-dir>
  ```

  to submit your work to the hwfinal directory in our dropbox. If you have any questions about submit395, please ask me well before the deadline!

  The required printouts should be turned in by the end of the day on Sunday, Dec. 10. You may turn in the printouts either by giving them to Jennie Colabella in the main CS office, giving them to me in person, or by leaving them under my office door.

  As discussed in class, time permitting, we will also schedule a meeting to demo your code.

- Please do this entire assignment inside the catkin workspace used for our HW4. When submitting your code, please submit the entire catkin workspace, and document (as on HW4) exactly what commands are needed to run your code for this assignment.

- Collaboration is restricted for this assignment. You should use only the ROS wiki pages (except for those that may tell you answers in ways that violate the course policy on Academic Integrity) and the texts A Gentle Introduction To ROS (by O’Kane) and Programming Robots with ROS (by Quigley, Gerkey, and Smart) as outside resources relating to ROS; you are not permitted to share code with classmates or look at the code of a classmate. (You may also use any Python resources for help with coding, but nothing else that is ROS-related.)

This is the assignment discussed in class on November 20.

**Programming Assignment: Add Two (or More) States!**

Add two states (behaviors) and relevant events and transitions to the FSM from HW4. (If you’d prefer, you can instead create an entirely new FSM with four or more states and events / transitions, but I recommend adding to the one from HW4.) Here are some ideas for FSM augmentations:

- Use laser scan information to do something different in a Gazebo world. For example, you might use laser scan data to determine if an obstacle is a sphere, and if it is, have the turtlebot push it for a period of time before wandering elsewhere.

- Use other sensors (e.g., bump sensor on a turtlebot in a Gazebo world) to determine new behaviors.

- Coordinate behaviors or implement obstacle avoidance among multiple turtles in a turtlesim, or multiple turtlebots in a Gazebo world.
• Use other tf transformations or other ROS topics for new behaviors. (See rostopic list for a list to choose from!)

• ...

As in previous assignments, please also submit a printout containing some documentation accompanying your work. The purpose of this documentation is to briefly explain important features of your work and how to run your code in context, as well as to include your code itself. Therefore, the printout should include:

• A brief explanation of the approach you took to designing and implementing the new states for the FSM.

• Documentation of how to run and test your code. Submit a list of all the command-line ROS commands one would need to run in order to demonstrate how your code works, any instructions on how to set up a Gazebo world in which your code might run, any instructions on how to run new launch files that you have written for this exercise, etc. If code submitted for this exercise cannot be easily tested by following the instructions in the accompanying documentation, that code will not receive full credit.

• Your python code. You need not print out any other elements of the ROS packages, just the python file you wrote for each exercise.

As discussed in class, your grade will in part be determined by how much interesting content there is in the FSM’s behavior or implementation. It is highly recommended that you meet with me to discuss your idea for your new FSM states before implementing them!