COMP.4510/COMP.5490 Mobile Robotics

Fall 2018

Prof. Yanco

**Assignment 1: Tutorials and ROS Race**

Out: Thursday, 13 September 2018

Due: Thursday, 20 September 2018, with files submitted before class

***Preparation:***

Go to the CS office (Dandaneau Suite 405) and get a account on the cs.uml.edu server

You may want to look over the ROS slides form last class as well, particularly for CMakeLists.txt setup.

In your homedir, create a catkin workspace (<http://wiki.ros.org/catkin/Tutorials/create_a_workspace>).

source /opt/ros/rosdesktop\_ws/devel/setup.bash

mkdir –p catkin\_ws/src

cd ~/catkin\_ws

catkin\_make

source devel/setup.bash

Download the uml\_race starter code:

cd src

git clone <https://github.com/uml-robotics/uml_race.git>

cd ~/catkin\_ws/

source devel/setup.bash

catkin\_make

In the src folder of the workspace you’ve created (~/catkin\_ws/src), create a catkin package (catkin\_create\_pkg) to contain the node you will be writing.

Name it something containing your name.

cd ~/catkin\_ws/src

catkin\_create\_pkg -m [your first name] [your pkg name] roscpp rospy geometry\_msgs sensor\_msgs std\_msgs

cd ~/catkin\_ws

source devel/setup.bash

catkin\_make

For this assignment, you can use either python or C++, but either requires dependencies:

***geometry\_msgs sensor\_msgs std\_msgs***

If you use C++, don’t forget to catkin\_make between test+edit iterations

In each new terminal, you need to source *~/catkin\_ws/devel/setup.bash* to use your **catkin workspace.**

**What to do:**

**Part 1: Tutorials**

1. Do the Core ROS Tutorials (<http://wiki.ros.org/ROS/Tutorials>) 1-17.
   1. Make sure you do **both** the C/C++ **and** Python versions of the publisher/subscriber tutorial (don’t worry if you are not familiar with Python; we just want you to get a basic into to that language).
2. Do the rviz Text-based Tutorials (<http://wiki.ros.org/rviz/Tutorials>) 1-10.

**Part 2: uml\_race**

Write a control program to complete the race track simulation provided in the uml\_race package. Your robot should travel no faster than 5m/s. This will be enforced by a referee node that is watching you when the robot starts. The starting location of the robot may be anywhere within a 4x10m bounding box around the start location so do not hard code the turns.

The robot has a 5 meter laser scanner that can see 180 degrees in front of it. The output of this topic is published to **/robot/base\_scan** and messages are typed ***sensor\_msgs/LaserScan***.

The robot listens to commands on **/robot/cmd\_velocity**. Messages are of type ***geometry\_msgs/Twist***. Twists are composed of an angular Vecto3 and a linear Vector3. Forward velocity is stored in ***linear.x*** while rotational “yaw” velocity is stored in ***angular.z*** .

**To run the racetrack and referee:** roslaunch uml\_race racetrack.launch

In the stage window you may see the output of the laser by selecting View > Data. Alternatively, you could *rosrun rviz rviz*, and add a LaserScan display with its topic set appropriately.

It turns out that msg.ranges[0] is the right-hand laser reading. Use msg.ranges[179] to get the left-side reading. You can use any/all of the readings as you see fit to complete the task.

After launching the racetrack, start your node with rosrun in a separate terminal. The racetrack will kill itself after the robot reaches the finish line.

Completion of the race without collision is sufficient. However, if you are inspired by competition, UML alumni Eric McCann’s node runs the track in 49 seconds, and student Steve Kim’s runs in ~48.5.

***How to Get Help:***

Email [jkuczyns@cs.uml.edu](mailto:emcann@cs.uml.edu) or [Victoria\_Albanese@student.uml.edu](mailto:Victoria_Albanese@student.uml.edu) if you need assistance on this assignment.

***What To Turn In:***

**Part 1: Tutorials**

A zip file containing the following:

1. A README file containing
   1. Your name and email address
   2. Instructions for the grader on how to run it.
   3. Answer the following questions *(you may use Google to help solve these)*:
      1. How much memory is your vLabs VM allocated?
      2. How many CPUs is your vLabs VM allocated?
      3. How much disk space is your vLabs VM allocated
   4. A detailed description of any problems that you were not able to fix, and a hypothesis on what is most likely causing them. *You will* ***not*** *be deducted points for disclosing errors which would otherwise not have been detected. This is merely to help the grader diagnose any major problems, so they can give you the maximum partial credit.*
2. A ROS package containing *(these two don’t have to be “your own work”, since you’ll copy/paste from the site; however, feel free to experiment)*:
   1. Your solutions from the publisher/subscriber tutorial (both C++ AND Python versions)
   2. Your solutions from he message/service tutorial
3. Your default rviz configuration file, renamed to ```DOTrviz```. (You’ll find it as a hidden file: “~/.rviz”)

(from your src folder: *zip –r yourname\_ps0.zip your\_dir*)

Copy the zip to your homedir on mercury.cs.uml.edu, then submit it with:

**submit jkuczyns hw0** *yourname\_ps0.zip*

**Part 2: uml\_race**

A zip file containing:

1. A README file containing
   1. Your name and email address
   2. Instructions for the grader on how to run it.
   3. A detailed description of any problems that you were not able to fix, and a hypothesis on what is most likely causing them. *You will* ***not*** *be deducted points for disclosing errors which would otherwise not have been detected. This is merely to help the grader diagnose any major problems, so they can give you the maximum partial credit.*
2. ONLY your race controller package folder (not the entire workspace).

(from your src folder: *zip –r yourname\_race.zip your\_race\_pkg*)

Copy the zip to your homedir on mercury.cs.uml.edu, then submit it with:

**submit jkuczyns race** *yourname\_race.zip*