

# Introduction to Unmanned Ground Vehicles

Instructor: Micho Radovnikovich

Email: mtradovn@oakland.edu

ECE 495/595 – Winter 2017

## Homework #3 – Vehicle Kinematics

Due: Wednesday, February 8<sup>th</sup>

The purpose of this assignment is to implement the kinematics of the three vehicles discussed in lecture. This assignment also serves as further practice in writing simple publisher/subscriber nodes.

### Provided Files

The **homework3** package will be initialized with the following files, which will be pushed into the central homework repository when Homework #3 is assigned:

- > **CMakeLists.txt** and **package.xml**. You will have to modify **CMakeLists.txt** to compile your nodes, however.
- > Test scripts to test your three nodes.
  - **homework3/test/test\_diff.cpp**
  - **homework3/test/test\_mecanum.cpp**
  - **homework3/test/test\_ackermann.cpp**
- > Launch files to run the tests. You will have to edit these to add your nodes to them.
  - **homework3/launch/homework3\_diff.launch**
  - **homework3/launch/homework3\_mecanum.launch**
  - **homework3/launch/homework3\_ackermann.launch**

The following resources from the released course source code will be used:

- > **description** and **gazebo** packages for each simulated robot. You shouldn't even have to look at these files. Feel free to do so if you'd like though :)

- > The **ugv\_course\_gazebo** package, which contains pre-defined Gazebo world files.
- > Launch files to run Gazebo and spawn robot models.
  - **ugv\_course\_launch/launch/roundbot\_homework3.launch**
  - **ugv\_course\_launch/launch/omnibot\_homework3.launch**
  - **ugv\_course\_launch/launch/audibot\_homework3.launch**

## Step 1 – Write the Nodes

For this assignment, three nodes will be written. Each will subscribe to a **geometry\_msgs/Twist** message, and publish necessary topics to control Gazebo models of each robot. Create separate **.cpp** files for each node, and put them in the **src** folder of the **homework3** package.

### Differential Drive

Create a node called **diff\_drive** that does the following:

- > Subscribe to the **/twist\_cmd** topic, which will contain **geometry\_msgs/Twist** messages.
- > In the subscriber callback, use the received twist message to compute the corresponding left and right wheel speeds, referring to the kinematics equations in the notes. The particular parameters for the robot are:

$$r_w = 0.2 \qquad W = 1.0$$

- > Publish the left wheel speed command as a **std\_msgs/Float64** message on the **/roundbot/left\_speed\_cmd** topic.
- > Publish the right wheel speed command as a **std\_msgs/Float64** message on the **/roundbot/right\_speed\_cmd** topic.

### Mecanum Drive

Create a node called **mecanum\_drive** that does the following:

- > Subscribe to the **/twist\_cmd** topic, which will contain **geometry\_msgs/Twist** messages.

- > In the subscriber callback, use the received twist message to compute the corresponding wheel speeds, referring to the kinematics equations in the notes. The particular parameters for the robot are:

$$r_w = 0.15 \quad W = 0.5 \quad L = 0.5$$

- > Publish the wheel speed commands as a single four-element array within a **std\_msgs/Float64MultiArray** message on the **/omnibot/wheel\_speed\_cmd** topic.
- > The array indexes corresponding to each wheel are as follows:
  - 0: Front left (wheel 1)
  - 1: Front right (wheel 2)
  - 2: Rear left (wheel 3)
  - 3: Rear right (wheel 4)

## Ackermann Drive

Create a node called **ackermann** that does the following:

- > Subscribe to the **/twist\_cmd** topic, which will contain **geometry\_msgs/Twist** messages.
- > In the subscriber callback, use the received twist message to publish speed and steering commands to implement the requested speed and yaw rate. To do this, pass through the speed command directly, and compute the necessary steering wheel angle by referring to the kinematics equations in the notes. The particular parameters for the robot are:

$$L = 3.0 \quad \gamma = 17.3$$

- > Publish the speed command in the **linear.x** field of a **geometry\_msgs/Twist** message on the **/audibot/cmd\_vel** topic.
- > Publish the steering wheel angle command as a **std\_msgs/Float64** message on the **/audibot/steering\_cmd** topic.

## Step 2 – Compile the Nodes

Next, edit **CMakeLists.txt** to compile your nodes. Do not change anything that is there already; just add the required text to compile your nodes. The executables that are

already added there are the three test scripts, which will be used to validate your nodes' behaviors. Finally, open a terminal, navigate to your ROS workspace folder, and run **catkin\_make** to compile.

### Step 3 – Run the Nodes

To run the nodes, use the following launch files to start Gazebo and spawn the robot models. They can be found in the **ugv\_course\_launch** package in the provided code:

- > **roundbot\_homework3.launch**
- > **omnibot\_homework3.launch**
- > **audibot\_homework3.launch**

Remember: the syntax for launching a launch file is:

```
roslaunch ugv_course_launch roundbot_homework3.launch
```

With Gazebo running, run your node that corresponds to the particular robot. Finally, using the **Message Publisher** RQT plugin, publish a 50 Hz **geometry\_msgs/Twist** message on the **/twist\_cmd** topic for your node to process.

### Step 4 – Test the Nodes

When you are satisfied with the performance of your nodes, you can run the test scripts that will be used to grade the homework. To do so, modify the following launch files to run your nodes:

- > **homework3\_diff.launch**
- > **homework3\_mecanum.launch**
- > **homework3\_ackermann.launch**

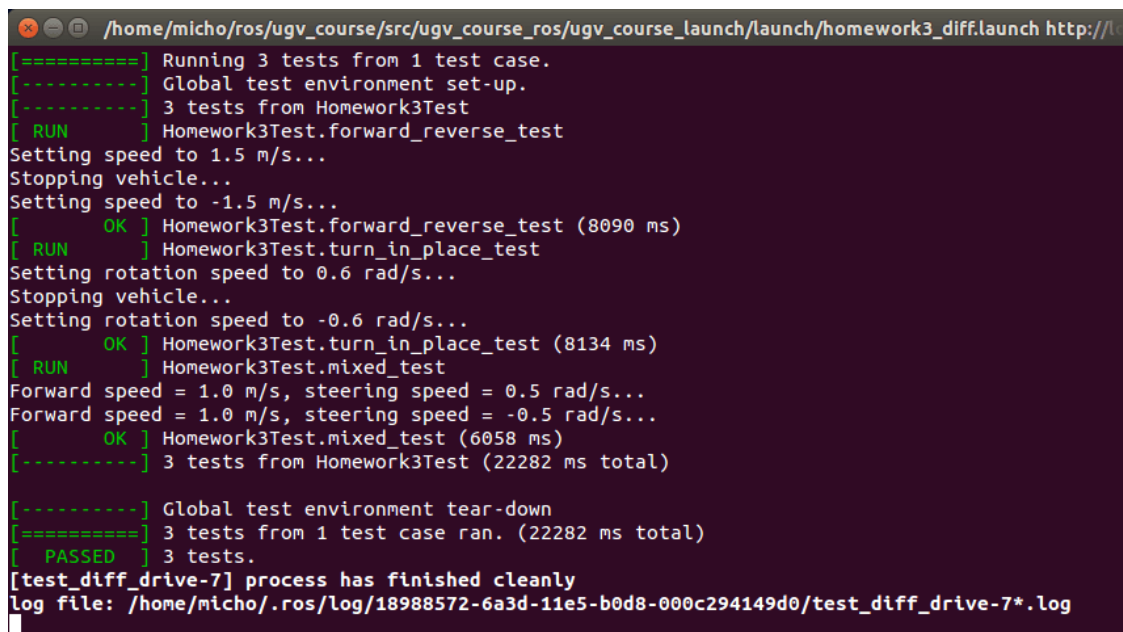
There is a helpful comment in each file to help you :)

```

1 <?xml version="1.0"?>
2
3 <launch>
4
5   <include file="$(find homework3)/launch/roundbot_homework3.launch" />
6   <node pkg="homework3" type="test_diff_drive" name="test_diff_drive" output="screen" />
7
8   <!-- Run your node here! -->
9
10 </launch>

```

Once you add your nodes to each launch file, launch them to run the tests. Each launch file will start Gazebo, spawn the appropriate robot model, and run your node and the test script. If everything is successful, you should see something like this for the differential drive test:



```

/home/micho/ros/ugv_course/src/ugv_course_ros/ugv_course_launch/launch/homework3_diff.launch http://l...
[=====] Running 3 tests from 1 test case.
[-----] Global test environment set-up.
[-----] 3 tests from Homework3Test
[ RUN      ] Homework3Test.forward_reverse_test
Setting speed to 1.5 m/s...
Stopping vehicle...
Setting speed to -1.5 m/s...
[ OK      ] Homework3Test.forward_reverse_test (8090 ms)
[ RUN      ] Homework3Test.turn_in_place_test
Setting rotation speed to 0.6 rad/s...
Stopping vehicle...
Setting rotation speed to -0.6 rad/s...
[ OK      ] Homework3Test.turn_in_place_test (8134 ms)
[ RUN      ] Homework3Test.mixed_test
Forward speed = 1.0 m/s, steering speed = 0.5 rad/s...
Forward speed = 1.0 m/s, steering speed = -0.5 rad/s...
[ OK      ] Homework3Test.mixed_test (6058 ms)
[-----] 3 tests from Homework3Test (22282 ms total)

[-----] Global test environment tear-down
[=====] 3 tests from 1 test case ran. (22282 ms total)
[ PASSED  ] 3 tests.
[test_diff_drive-7] process has finished cleanly
log file: /home/micho/.ros/log/18988572-6a3d-11e5-b0d8-000c294149d0/test_diff_drive-7*.log

```

You will see something similar when you run the other two tests.

## What to Submit

Submit Homework #3 by pushing the following to your repository:

- > The three source files for your three kinematics nodes.
- > Modified **CMakeLists.txt** that compiles your nodes.
- > Modified test launch files that run your nodes.