Introduction to Unmanned Ground Vehicles

Instructor: Micho Radovnikovich
Email: mtradovn@oakland.edu
ECE 495/595 – Winter 2017

Homework #4 – Rviz Markers and TF Transforms

Due: Monday, February 27th

The purpose of this assignment is to become more familiar with Rviz visualizations, dynamic reconfigure GUIs, and TF transforms and reference frames. Each of these aspects of ROS are very commonly used, and will be a foundation upon which the rest of the course will be built.

Provided Files

The **homework4** package will be initialized with the following files, which will be pushed into the central homework repository when Homework #4 is assigned:

> **CMakeLists.txt** and **package.xml**. You will have to modify CMakeLists.txt to compile your nodes, however.

> Dynamic reconfigure file to control position and orientation from the reconfigure GUI (**homework4/cfg/Marker.cfg**). Don't change this file.

> Launch file that will run your node and the test node (**homework4/launch/homework4.launch**). You will have to modify this to include your node.

> Test script (**homework4/test/test_homework4.cpp**).

Step 1 – Write the Node

Create a new source file called **homework4.cpp** in the **src** folder of the **homework4** package. Within that source file, write a node to:

> Advertise a **visualization_msgs/MarkerArray** topic called **marker_array**.

> Set up a timer and its corresponding callback function that triggers at 20 Hz.

> Initialize a dynamic reconfigure server and bind a callback to process inputs from the GUI.

> Create a **visualization_msgs/MarkerArray** that contains three cube markers:

- Each cube should be 1 meter in all dimensions.
- One cube should be yellow (mixture of red and green) and be positioned at (0, 0, 0) with no rotation.
- Another cube should be cyan (mixture of green and blue) and be positioned at (1, 1, 1) with no rotation.
- The third cube should be magenta (mixture of red and blue) and be positioned at (2, 2, 2) with no rotation.
- Each of the cube markers should be relative to the **marker** TF frame.

> In the dynamic reconfigure callback function, use the current values of the dynamic parameters to populate a global **tf::TransformStamped** object as shown in the **marker_tf_example** example from lecture. Make sure the parent frame of the transform is set to **map** and the child frame is set to **marker**.

> In the timer callback function, declare a static **tf::TransformBroadcaster**, update the time stamp on the **StampedTransform** object, and send the transform.

> Also in the timer callback, publish the marker array message on the **marker_array** topic. Be sure to update the time stamps on each of the individual markers.

Step 2 – Compile the Node

Compile the node like usual. Be sure to do the following:

> Put the **add_dependencies** line to indicate that your node depends on the generated dynamic reconfigure header.

> Make sure the compiled node's name is **homework4**. This is necessary for the test script to function properly.

> Add your node to the provided **homework4.launch** launch file.

Step 3 – Test the Node

You can test your node manually by doing the following:

> Open Rviz, add a **MarkerArray** display that subscribes to the **marker_array** topic, as well as a **TF** display.

> Run the dynamic reconfigure GUI and play around with the settings, and make sure the three cubes move and rotate appropriately.

Once you're satisfied with the performance, you can run the formal test by launching **homework4.launch**. Make sure both tests pass, and then move on to submitting!

What to Submit

Submit Homework #4 by pushing the following to your repository:

> The **homework4.cpp** source file.

> Modified **CMakeLists.txt** that compiles the **homework4** node.

> Modified **homework4.launch**.