CS 309
Autonomous Intelligent Robotics (FRI I)
HW 4: Following the AR Tag Due: April 21, 2020

You can find the latest version of this PDF at
`http://justinhart.net/teaching/2020_spring_cs309/homework/HW4/hw4.pdf`

For this homework you will work with your final projects groups. I have provided code on my website which can act as a starting point here: `http://justinhart.net/files/homework/2020_spring_cs309/HW4/hw4_pkg.tar.gz`

Part A The first part of this assignment is about familiarizing yourself with frames and transformations from a practical perspective. The lessons you learn doing Part A will help you significantly in writing Part B.

Download the hw4.bag, which can be found at `http://justinhart.net/files/bags/hw4.bag`.

Be prepared! *This is a 12 GB bag.*

Open a few terminal windows. Unpack hw4_pkg.tar.gz in a catkin workspace under the src directory.

Fire up roscore. Run *rosbag play hw4.bag*. Run rviz.

First, you need to choose a new *Fixed Frame*, which appears under *Global Options* on the left. Let me recommend *camera_rgb_optical_frame*.

You will want to render a few ROS topics. I recommend going into *Add* and *By topic*, and picking *ARmarker_points*, *camera/depth_registered/points*, and *visualization_marker*. You will then want to go to *By display type* and choose *TF*.

Hit *ctrl + s* to save your rviz configuration.

For some reason, bags of TF data simply do not work well. When the TF data vanishes from the screen, simply stop and restart rosbag, roscore, and rviz. A workaround for this would make a great final project.

You'll want to familiarize yourself with how to turn on and off TF frames for this homework. Having too many on the screen can clutter things and make it difficult to get work done. Explore the bar on the left hand side of the screen to see how to do this.

Once you're comfortable with the data in the bag. It's time to knock out Part A.

Look at *hw4_node.cpp*, and see how *ROSINFOPoseRecipient* interacts with *AlvarMarker* by inheriting from *PoseRecipient*. Understanding how to make this work for your own *PoseRecipient*s will be important for completing the homework.

You should make a few things happen.

- Write a *PoseRecipient* that publishes TF frames.
- Publish a tf frame that is offset 1 meter in front of ar_marker_0, facing in the same orientation. Call it "offset_frame". Verify this in rviz.
- Publish a tf frame that is the same position as "offset_frame" but rotated 180 degrees so it is now facing the Alvar tag. It should be rotated about the axis that runs vertically with respect to the Alvar marker.

This is all you need to accomplish for Part A. Most of you are likely to find this very challenging, as it will involve some concepts that have just been introduced in the lecture material.

Also, for fun, check out what happens when you change the fixed_frame to *ar_marker_0*.

Part B

Download the homework-support.tar.gz, which can be found at `http://justinhart.net/teaching/2020_spring_cs309/homework/HW4/homework-support.tar.gz`.

Unpack into the src directory of your workspace, and run catkin build.

source devel/setup.bash
roslaunch homework-support ar_tag_hallway.launch
roslaunch homework-support nav_kinect_alvar.launch

Go into Gazebo, which was just launched. Click the triangle to expand "Models." Right click "Marvin." Click "Move to." You will now be facing a wall. Middle click your mouse and drag down, until you are facing vertically. Wheel your mouse wheel to zoom out. Click on the map to drag it around and pan. You should see two BWIBots.

Go to rviz. Go to "Global Options." Choose "marvin/base_link" as your fixed frame.

Add marvin's point cloud camera, TF frames, and the ARMarker to be rendered in rviz.

If you run "rosrun homework-support lead_robot_node," this program will try to stay a fixed distance away from marvin. It will stop when marvin gets too far away and do a little dance to try to show its markers. This is to help you test.

If you run "rosrun homework-support teleop_twist_keyboard," this will let you drive the ar_tag robot manually. You will want to killall -9 lead_robot_node before running it, so you have manual control and the lead_robot_node software doesn't fight you.

If you use "2D Nav Goal" in rviz, your command will move marvin.

Your goal is to write software that makes marvin follow ar_tag, at a distance of 1.0 meters. A strong recommendation to make this happen is to also end your motion such that marvin is facing ar_tag. I would also use MoveBaseGoal to make this happen. You do **not** need to assure that your robot does not get stuck when lead_robot is running, but you **do** need to assure that if someone is making a good faith effort to help your robot follow it by running teleop_twist_keyboard that your robot will indeed follow. Good luck. I am giving you until Tuesday, April 21 to complete this task, though we will be proceeding with project proposals to keep final projects on schedule.