ME 132B - LIDAR Scan Matching

Released: Friday 05/05/2017 Due: Friday 05/12/2017

In this lab we will be using the LIDAR scan data from our recent hardware demonstration to estimate the position of the robot compared to the world frame. We will be using a ROS package named 'laser_scan_matcher' (that incidentally was written by a Caltech PhD student) to process all of the distance measurements from the LIDAR sensor, convert them into a point cloud, then attempt to match each successive scan with previous scans and track motion of the sensor.

Playing back sensor data

1. To play back all of the sensor data in ROS you will need to start up the Ubuntu virtual machine you used in ME 132a. If you have deleted the virtual machine then you can find instructions on how to set it up on the class wiki for last quarter.

2. Load the .bag file you received from the hardware demo onto the virtual machine's 'home' folder, which can be navigated to in the terminal with the command: $ cd ~

3. You will also need to download the configuration file for rviz which sets up the GUI to play back the sensor data from our test which be found here, and load that onto the virtual machine.

4. We can now check the .bag file to make sure it has all the data we need to run the test. To do this, open a terminal with *CTRL + ATL + T* and navigate to wherever you placed the .bag file, then run the command (replacing bag_name with whatever yours is called) which should produce an output similar to the figure below:

   $ rosbag info bag_name.bag

```
me132@me132-VirtualBox:~$ rosbag info figure8.bag
path:        figure8.bag
version:     2.0
duration:    28.3s
start:       May 03 2017 21:15:29.00 (1493871330.00)
end:         May 03 2017 21:15:58.33 (1493871358.33)
size:        499.8 MB
messages:    48265
compression: none [419/419 chunks]
types:       kobuki_msgs/SensorState [430a4bfd78449c8740bfef32b26613a6]
             nav_msgs/Odometry       [cd5e73d190d741a2f92e81eda573aca7]
             sensor_msgs/CameraInfo  [c9a58c1b0b154e0e6da7578cb991d214]
             sensor_msgs/Image       [060021388200f6f0f447d0fcd9c64743]
             sensor_msgs/LaserScan   [90c7ef2dc6895d81024acba2ac42f369]
             tf/tfMessage            [94810edda583a504dfda3829e70d7eec]
             tf2_msgs/TFMessage      [94810edda583a504dfda3829e70d7eec]
topics:      /camera/depth/camera_info     837 msgs    : sensor_msgs/CameraInfo
             /camera/depth/image_raw       837 msgs    : sensor_msgs/Image
             /mobile_base/sensors/core    1417 msgs    : kobuki_msgs/SensorState
             /odom                        1416 msgs    : nav_msgs/Odometry
             /scan                         283 msgs    : sensor_msgs/LaserScan
             /tf                         43475 msgs    : tf2_msgs/TFMessage
```

5. If any of the topics have 0 messages or are missing then I may have misspelled or neglected to include them during the hardware demo, in which case you may instead choose one of the back up bags of a figure8 or square with the position error at the end of those tests found in the text files figure8 error or square error.

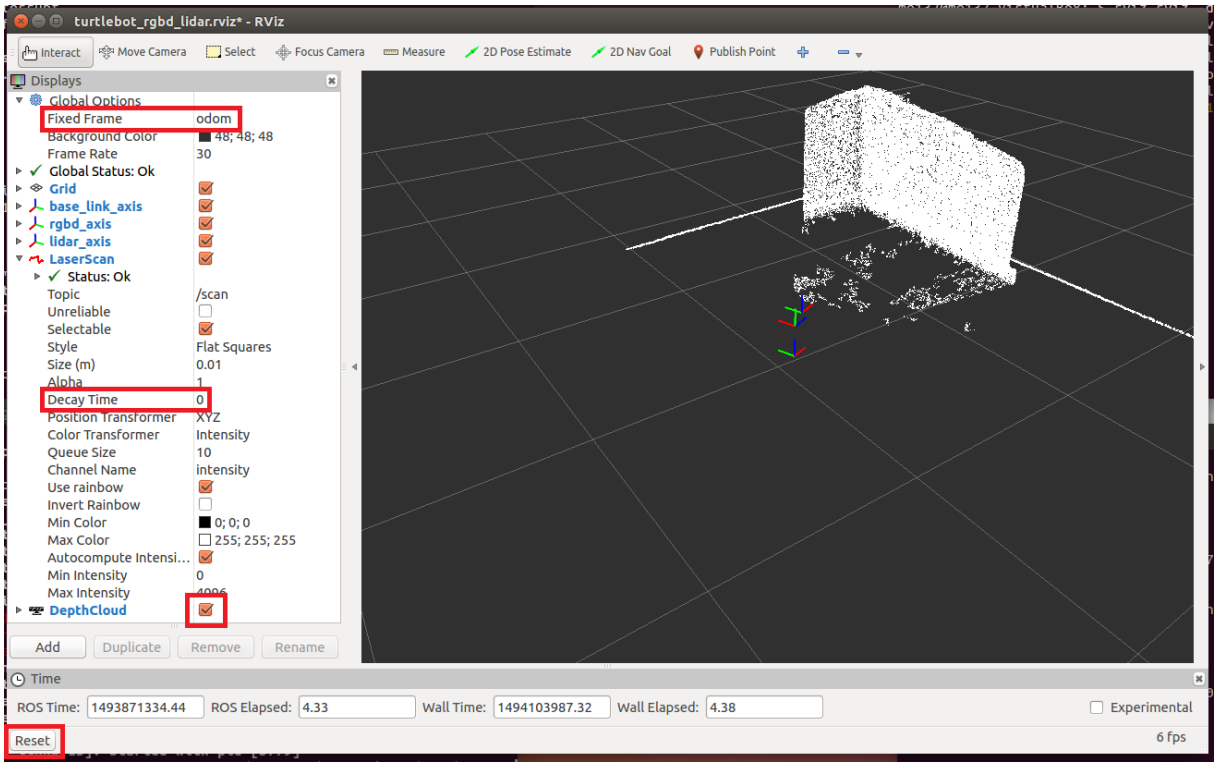
6. Once all of the files in place we can start ROS by opening a new terminal and using the command: $ roscore

7. Next we can open the rviz GUI in another terminal by navigating to the location of the turtlebot_rgbd_lidar.rvz file and then using the command:

   $ rviz rviz –d turtlebot_rgbd_lidar.rviz

8. Finally, we can play back the .bag file to see what the turtlebot saw as it was driving around during the demonstration, by navigating to the location of the file in a new terminal and running the command:

   $ rosbag play --clock bag_name.bag



9. In order to check the position of the odometry estimate against our ground truth measurement we'll need to print out the transform between 'base_link' and 'odom' by running the following command in a new terminal WHILE the .bag file is being played back:

   $ rosrun tf tf_echo /odom /base_link

How does the final location compare to the position we measured manually? Compare the two values in your submission. If you want to play back the wheel odometry referenced file again then click Reset in rviz and run the rosbag play command again.

Interpreting the data

As you play back the bag file you should see two points clouds in rviz much like the figure above, the thin line comes from the 180° sweep of the lidar, and the rectangle from the RGBD camera. The three sets of axes represent the coordinate frames of the robot's footprint, the RGBD camera, and the lidar. These are all relative to the global fixed frame named odom which the turtlebot calculates by fusing data from its encoders and IMU. We can see how this drifts by unchecking the box next to

DepthCloud (seen in red above), and changing the Decay Time variable for the LaserScan from 0 to 60 seconds.

We can now play the bag file back again by clicking Reset at the bottom left of rviz, then running the 'rosbag play ...' command again. Pay particular attention to how the apparent location of the walls around the turtlebot drifts, you should see the points diverge from a thin to a thick edged square as the odometry position estimate worsens over time.

Take a screenshot of the pattern produced by the accumulated lidar pointcloud using odom as the fixed reference and submit it along with your homework.

Using LIDAR to localize

The goal of this lab is to have you use the lidar data to produce a better estimate of where the robot is relative to the world. In the process we hope for you to also gain some more experience using ROS packages.

First you'll need to install the ROS lidar SLAM package recommended by Joel that hppens to have been scripted by a Caltech student. This is named laser_scan_matcher and can be loaded with the following terminal command:

$ sudo apt-get install ros-indigo-laser-scan-matcher

We are also going to have to create a version of our .bag file that does not contain the transformation frames 'odom' and 'base_footprint' due to the fact that ROS does not allow each node if the tf tree to have more than one parent. Previously the wheel odometry topic 'odom' was providing an estimate of the robot's position (base_link) relative to the fixed external frame, but now it will be the 'world' tf produced by the laser_scan_matcher node.

To produce this new .bag file open a new terminal, navigate to your bag file, and run the command (changing bag_name to the name of your file, can copy the command from this text file):

$ rosbag filter bag_name.bag no_odom.bag "topic == '/camera/depth/camera_info' or topic == '/camera/depth/image_raw' or topic == '/mobile_base/sensors/core' or topic == '/odom' or topic == '/scan' or topic == '/tf' and m.transforms[0].header.frame_id != 'odom' and
m.transforms[0].header.frame_id != 'base_footprint'"

While this process runs we can prepare rviz to display the localization estimate from the laser scan matching. To do this click Reset at the bottom left corner, and then change Fixed Frame under Global Options from 'odom' to 'world' by typing this in manually and hitting enter.

Next, in order to launch the laser scan matcher and play the filtered bag file we'll need to use a special launch file that can be downloaded from here and placed in the home folder. This assumes your filtered bag file is called 'no_odom.bag' so if you changed it you'll need to modify the launch file to account for this in the text editor ('$ gedit file_name' from the terminal).

Finally, we need to let the laser_scan_matcher node know that it will be processing data from

in the past, which is done by opening a new terminal and entering the command:

$ rosparam set use_sim_time true

In this same terminal we can now run our simulation with the command:

$ roslaunch lidar_estimate.launch

Watch the accumulated lidar pointclouds in rviz. Do they seem to be more consistent than when they were referenced against the wheel odometry position estimate? Take a screenshot of the accumulated point cloud using 'world' as the Fixed Frame and submit it along with your homework. Once again, we can watch the transform between the fixed and robot frames by using the following command in a new terminal WHILE the bag is being played back:

$ rosrun tf tf_echo /world /base_link

How does the final position error compare to the error we saw when using the wheel odometry to keep track of the reference frame?

The lidar localized pointcloud was likely better grouped than when referenced with wheel odometry but it still isn't perfect. In particular it has a habit of drifting when the turtlebot makes rapid changes of direction (as seen at the inflection point of the figure8 example). Much as we discussed in class, how would you go about getting an even better estimate of our current position using all of the sensors we have available?

If you wish to play back the lidar referenced data again you'll need to click Reset in rviz and *CTRL + C* in the lidar_estimate.launch terminal before running the launch file again.

Extra credit

If you happen to want to subject yourself to more fun with ROS then try to find and apply a ROS package that will run a slam algorithm on the RGBD depth video stream. Some to look at may be 'rtabmap' or 'rgbdslam'. You may find the command '$ roswtf' useful in debugging connections between different nodes and topics. Document your efforts for submission as you will receive credit even if you do not get the complete SLAM up and running.