Lab 5

Controlling Puma Using Leap Motion Device

Objective
In this experiment, students will use “Leap Motion” device to achieve the following goals:
- Get familiar with the Leap Motion device
- Experience ROS tools i.e. tf, rviz
- Control the Puma robot to grab an object using their own hands.

Reference Materials
- tf broadcaster and listener: http://wiki.ros.org/tf/Tutorials sections 1-3
- tf and Rviz http://wiki.ros.org/tf/Tutorials/Introduction%20to%20tf
- http://wiki.ros.org/rviz/DisplayTypes/TF
- Learning ROS for Robotics Programming by Aaron Martinez, Enrique Fernandez: chapter 6, 7, 8
- Leap Motion website: https://leapmotion.com
- Leap_motion ROS driver: http://wiki.ros.org/leap_motion

Prelab
Read reference material and background below and be prepared for a quiz.

Background
Leap Motion Device:
Leap Motion Device gives the pose/gesture of hands and fingers which can be used to control the Puma 260 robot. The coordinate frame of the LM and its configuration w.r.t robot coordinate frame are shown in Figure 1. As it can be seen, a translation (-215, 305, -185) and a rotation (90 degrees around moving x) are required. Considering the origin on the table, the transformation of hand w.r.t robot base can be found as:

\[ R_B T_{Hand} = (O_{TB})^{-1} \times O_T L_M \times L_M T_{Hand} \]
Once you found transformation of hand w.r.t robot, you can use that as the pose of end effector and control the robot to pick an object. You may use different methods (i.e. two hands) to open or close the gripper.

**Transformations in ROS:**
ROS has a nice library (called tf) for defining different coordinate frames and transformations between these frames. A tf has all required information (time stamp, translation, rotation, parent frame, child frame) which can be broadcasted by `tf::TransformBroadcaster` in one node and listened by `tf::TransformListener` in another node (see tf broadcaster/listener tutorials).

**ROS Rviz:**
Rviz is a visualization tool which can help you to debug and visualize your robot and its sensor data (i.e. image, 3D point cloud, pose, and etc.). For example, you can visualize all coordinate frames (i.e. robot link frames, robot base frame, object, LM frame and etc.) and debug your program appropriately. You can launch Rviz by typing:

```
$ rosrun rviz rviz
```

This will show a GUI which you can add different type of data to be visualized.

---

Figure 1: (a) coordinate frame of leap motion, (b) The top view of leap motion and robot base coordinate frames with respect to each other

Figure 2: Rviz (a) 3D visualization of robot and sensor data, (b) list of all modules displayed on a, (c) to add/remove a new/existing module i.e. 3D point cloud
Lab Procedure
Part 1: Getting Started with Leap Motion
Leap motion driver is installed on Puma computers and you can see how the device can detect your hand by running:
$ LeapControlPanel
Now you should see a tray icon on top-left corner of desktop, right-click and choose Diagnostic Visualizer. Figure 3 shows the GUI where you can see depth images and your augmented hand(s).

Figure 3: Leap Motion Diagnostic Visualizer

In order to run ROS driver you can run below commands:
$ export PYTHONPATH=$PYTHONPATH:/usr/local/src/LinuxPUMAs/src/LeapSDK/lib/x64
$ roscore &
$ rosrunc leap_motion sender.py
You will see messages with number of hands, fingers and pose of the right hand. Try to understand roll, pitch and yaw angles. What messages were published by the node? What is the type of message? (Hint: try $rostopic list, $rostopic info <name_of_topic>, $rosmsg show <type_of_msg>)

You can also see raw images and the point cloud by:
$ roslaunch leap_motion leap_camera.launch
$ roslaunch leap_motion leap_stereo.launch
This will publish left/right raw images and point cloud. You can see all published topics by:
$ rostopic list
In order to see point cloud in Rviz, you need to broadcast a transformation between world (map) and leap motion device (leap_optical_frame). You can do this in command line by:
Try to understand different topics and how you can control the robot by your hand.

Part 2: Getting Started with tf
In this part, you will write a tf broadcaster to publish coordinate frames of origin, robot base, leap motion device, and your hand(s). All these frames should be defined w.r.t world coordinate frame (map) or origin using information provided by leap motion device and transformation mentioned in Background section. Try to visualize coordinate frames by adding TF in Rviz. You can also use following command to debug your program:

$ rosrun tf view_frames
$ rosrun tf tf_monitor

Include the transformation tree in your report.

Part 3: Controlling Puma with Leap Motion
In this part, you will use the pose of your hand as pose of end-effector and try to control the arm using your hand. That is, you need to write a wrapper between ROS and Puma_API. You can either do system call to binary functions or directly use c++
functions provided under `/usr/local/src/LinuxPUMAs/src/Puma_API` to control the robot (Extra 10 Credits if C++ functions used).

**Note:** make sure you have if statement to check calculated XYZ to be in a free space to avoid robot hitting camera or table.

**Extra 20 Credits** (Mandatory for Grad Students): visualizing puma links coordinate frames in Rviz using tf. You need to find coordinate frame of each link using forward kinematics after finding joint angles using inverse kinematics and broadcast all transformations w.r.t previous link.

The transformation tree will be similar to figure below:

![Transformation Tree Diagram](image)

**Figure 5: Structure of Transformation Tree**

**Post Lab Questions:**
1. Derive transformation $^{RB}_{Hand}$
2. Describe data returned by leap motion ROS driver (`/leapmotion/data`)
3. Explain your approach of opening and closing the gripper.