TurtleBot - Gesture Recognition

Weiyu Liu

Goals & Plan

- Long-term Goal

Make TurtleBot understand series of meaningful gestures and act correspondingly.

- Short-term Goals

1. Be familiar with Robot Operating System
2. Implementing basic gesture recognition in TurtleBot by using its Kinect camera

- Plan

1. To be familiar with ROS commands and tools.
2. To test the existing gesture recognition package, and confirm its output by using visualization tools.
3. To integrate gesture recognition package in my own source code to control Turtlesim.
4. To print detailed transformation information to console for debugging and finally to develop a successful Turtlesim program.

Background

- TurtleBot

TurtleBot combines popular off-the-shelf robot components, the iRobot Create, Yujin Robot's Kobuki and Microsoft's Kinect into an integrated development platform. Behind the TurtleBot, ROS acts as its operating system.

- ROS

ROS is an open-source framework, which contains collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

TurtleBot™

Transform & Packages

- tranform (tf)

  transform (tf) is a package that maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

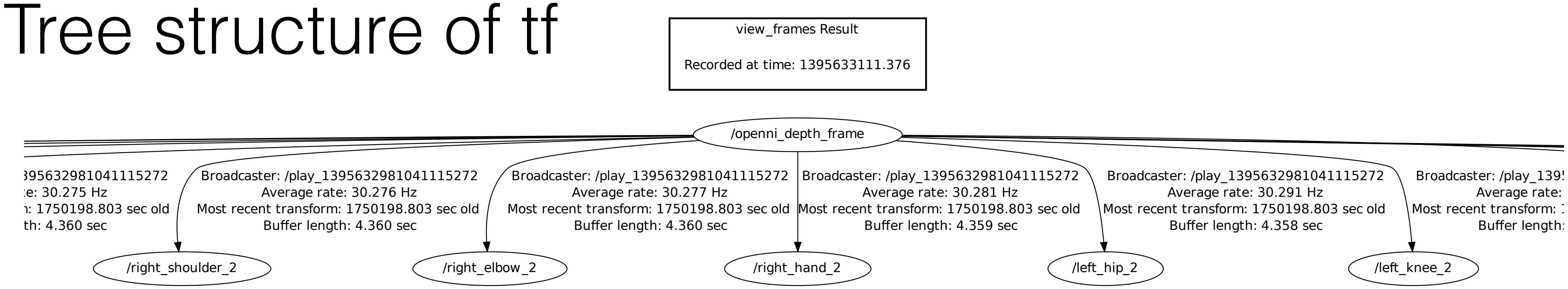
- Packages output tf

  By using existing gesture recognition package <openni_kinect > and <openni_tracker>. User's pose will be published as a set of transforms as listed in the right

- /head
- /neck
- /torso
- /left_shoulder
- /left_elbow
- /left_hand
- /right_shoulder
- /right_elbow
- /right_hand
- /left_hip
- /left_knee
- /left_foot
- /right_hip
- /right_knee
- /right_foot

Structure of tf

- Tree structure of tf

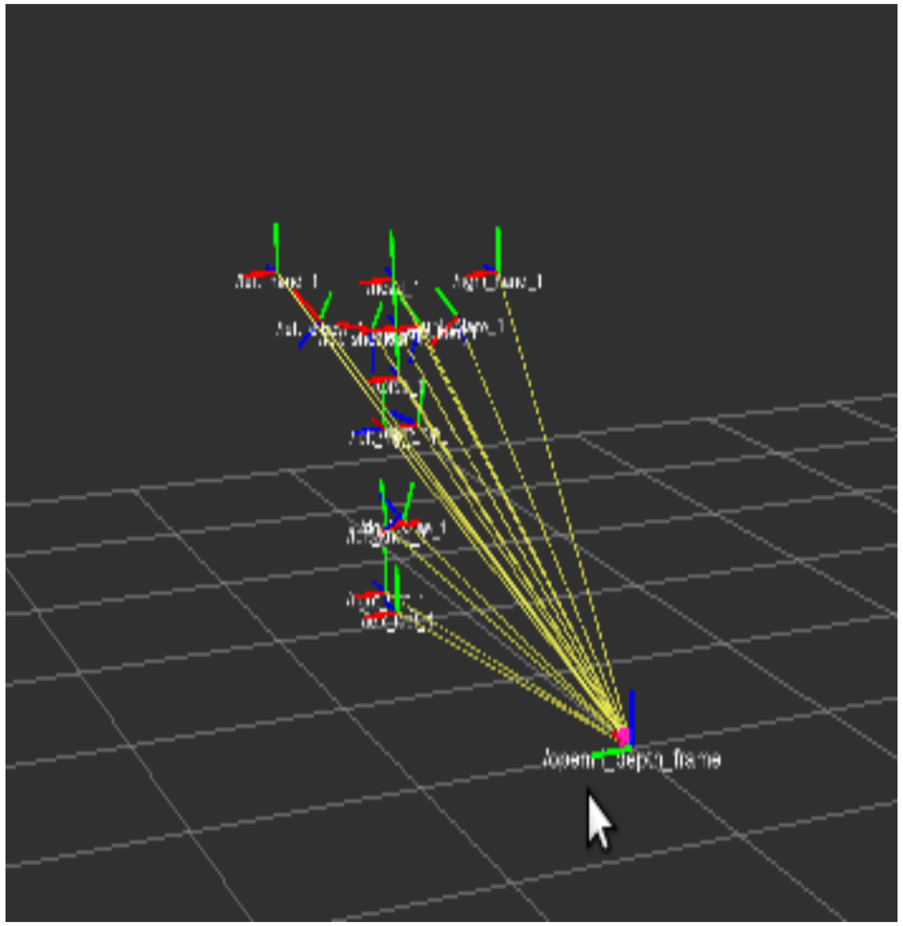


```
395632981041115272
:e: 30.275 Hz
n: 1750198.803 sec old
th: 4.360 sec

Broadcaster: /play_1395632981041115272
Average rate: 30.276 Hz
Most recent transform: 1750198.803 sec old
Buffer length: 4.360 sec

Broadcaster: /play_1395632981041115272
Average rate: 30.277 Hz
Most recent transform: 1750198.803 sec old
Buffer length: 4.360 sec

Broadcaster: /play_1395632981041115272
Average rate: 30.281 Hz
Most recent transform: 1750198.803 sec old
Buffer length: 4.359 sec

Broadcaster: /play_1395632981041115272
Average rate: 30.291 Hz
Most recent transform: 1750198.803 sec old
Buffer length: 4.358 sec

Broadcaster: /play_139!
Average rate:
Most recent transform: :
Buffer length:
```

- rviz

1. ROS tool for visualizing tf. rviz uses the tf transform system for transforming data from the coordinate frame it arrives in into a global reference frame.

2. Reference frame is set to the parent frame /openni_depth_frame. Coordinate frame is set to all the child frame, like /head and /torso



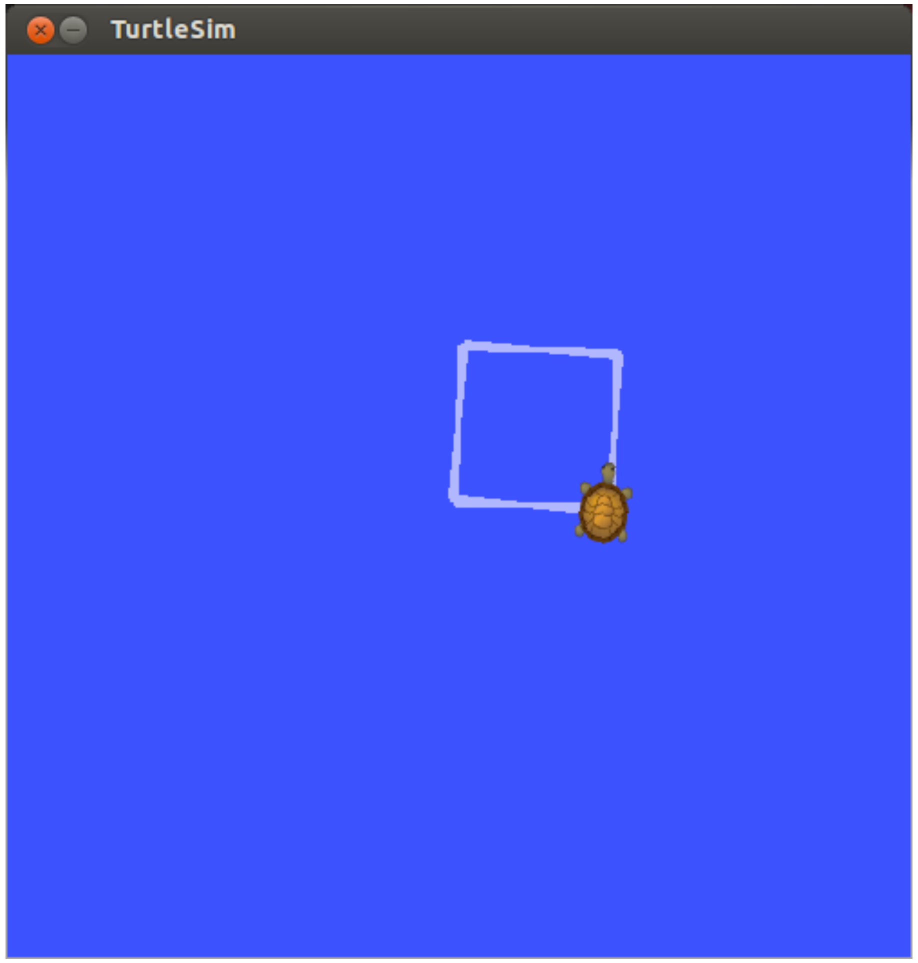
Turtlesim & tf_listener

- Turtlesim

  Package <turtlesim> is a simple GUI application which contains a simulated turtle moving around in the background.

- tf_listener

  tf_listener is a program I wrote which use the tf provided by gesture recognition packages to control the linear and angular velocity of the simulated turtle



Codes of tf_listener

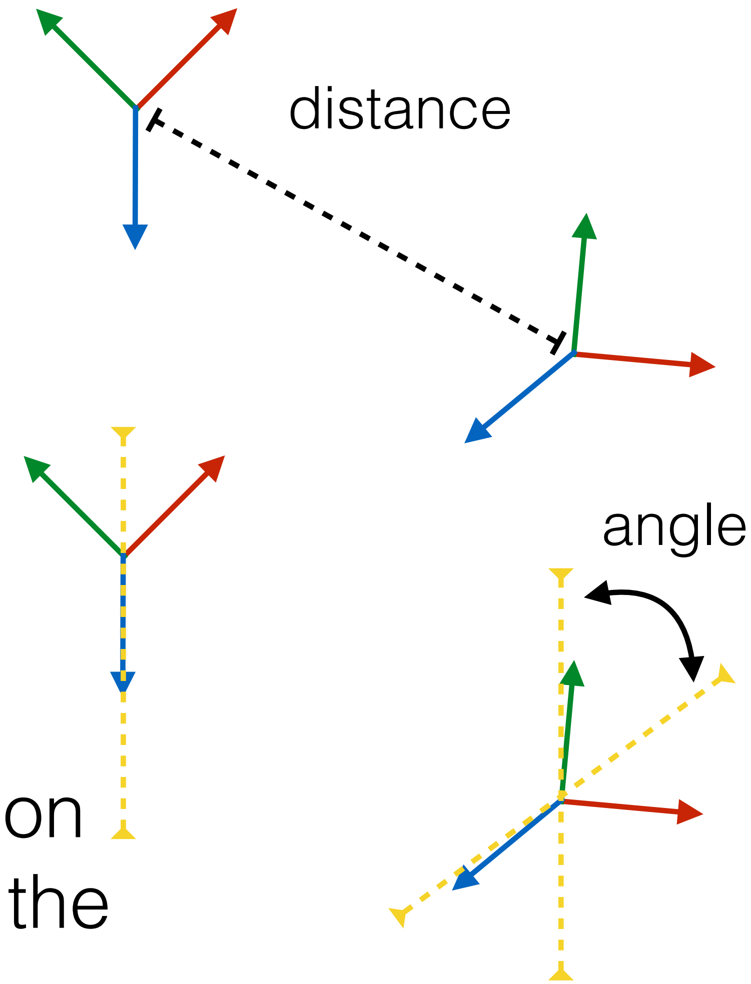
1. Get the transform between two frames

```
listener.lookupTransform("torso_1", “left_hand_1", ros::Time(0), transform);
```

2. Use the transform in x, y, z three directions

```
transform().getOrigin()
// This will use the distance between two points
```

```
transform().getRotation()
// This will use the angle between two orientations
```



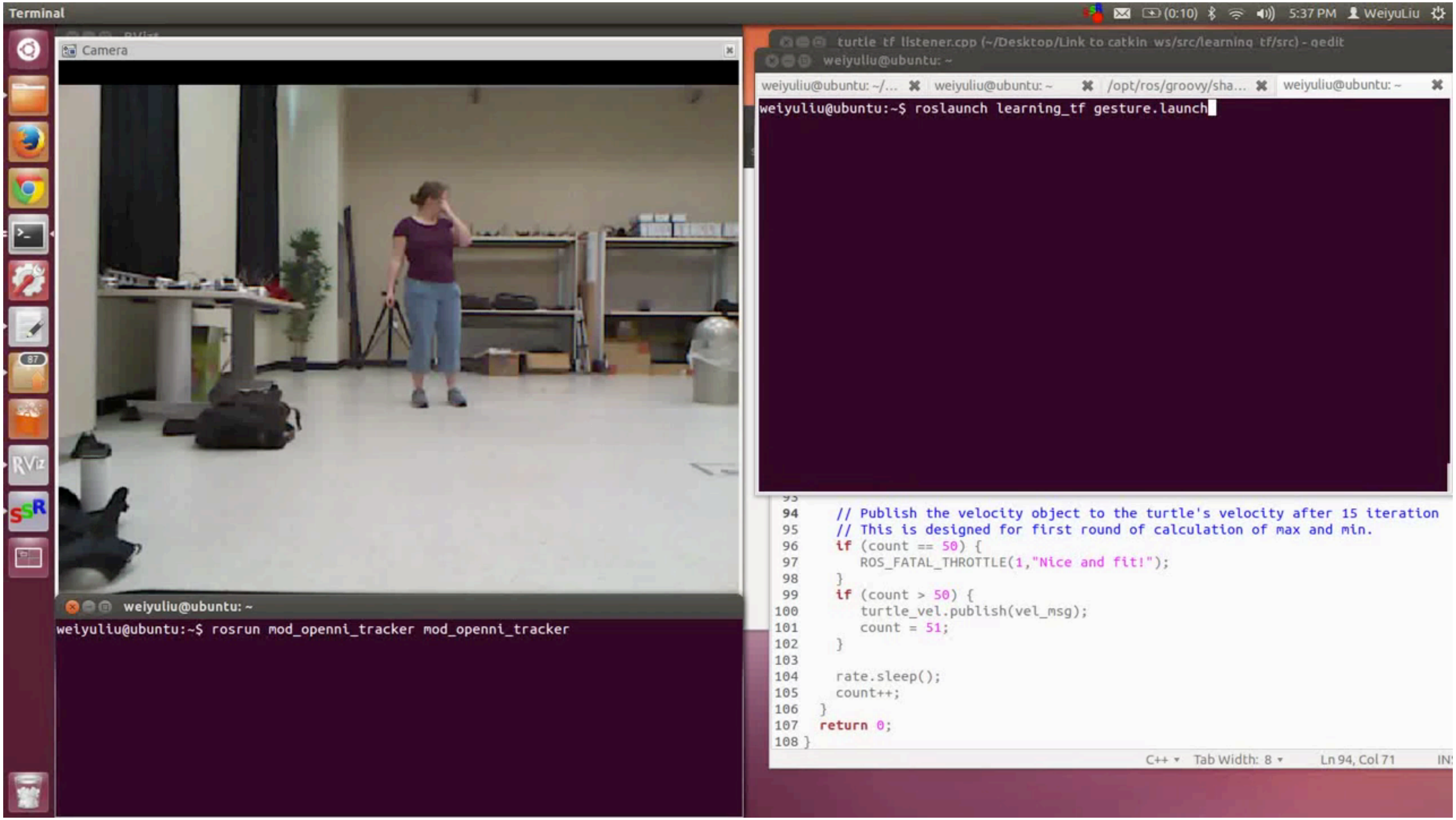
3. Distance is being used. By experiments, x direction in tf corresponds to the movement of left hand to the left side of torse, y corresponds to up, and z corresponds to forward.

Calibration

In order to allow the program to work with different users with different body dimensions, the algorithm will constantly record the maximum and minimum displacement and use these two values to scale the velocity of simulated turtle.

```
y = transform.getOrigin().y(); // stores tf in y direction to y
if (y > max_y) {  // actively determine the maximum of y and record it.
   max_y = y;
}
if (y < min_y) {  // actively determine the maximum of y and record it.
   min_y = y;
}
// below use max and min y to scale linear velocity
double tempy = (y - min_y) / (max_y - min_y) * (MAX_LINEAR_SPEED - 0);
if (tempy < 0.2) {
   vel_msg.linear = 0;
} else {
   vel_msg.linear = tempy;
```

Presentation



```
weiyuliu@ubuntu:~$ roslaunch learning_tf gesture.launch
```

```
weiyuliu@ubuntu:~$ rosrun mod_openni_tracker mod_openni_tracker
```

```
    // Publish the velocity object to the turtle's velocity after 15 iteration
    // This is designed for first round of calculation of max and min.
    if (count == 50) {
       ROS_FATAL_THROTTLE(1,"Nice and fit!");
    }
    if (count > 50) {
       turtle_vel.publish(vel_msg);
       count = 51;
    }

    rate.sleep();
    count++;
  }
  return 0;
}
```

Future Work

1. To build more interesting gaming application.
2. To move the program to control TurtleBot.
3. To make TurtleBot understand series of meaningful gestures.
4. To make TurtleBot dance with the user!