ECE 4703
Mobile Autonomous Robots

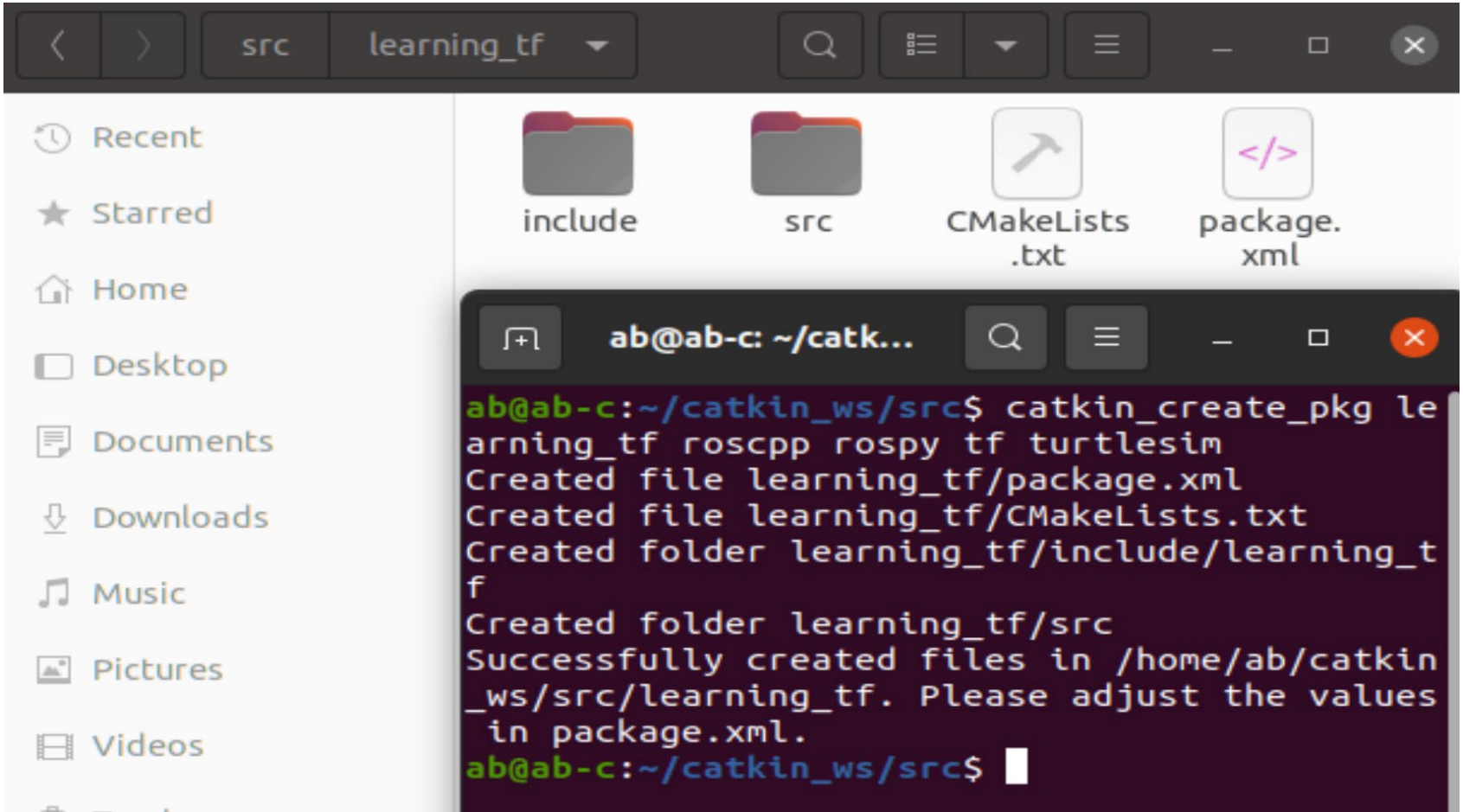
Jenny Zhen Yu
zhenyu@cpp.edu
Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Lecture 13: ROS Coordinate System Program

Create Package

$ cd ~/catkin_ws/src

$ catkin_create_pkg learning_tf roscpp rospy tf turtlesim



Transform Broadcaster

turtle_tf_broadcaster.cpp

```cpp
#include <ros/ros.h>
#include <tf/transform_broadcaster.h>
#include <turtlesim/Pose.h>

std::string turtle_name;

void poseCallback(const turtlesim::PoseConstPtr& msg)
{

        static tf::TransformBroadcaster br;


        tf::Transform transform;
        transform.setOrigin( tf::Vector3(msg->x, msg->y, 0.0) );
        tf::Quaternion q;
        q.setRPY(0, 0, msg->theta);
        transform.setRotation(q);


        br.sendTransform(tf::StampedTransform(transform, ros::Time::now(), "world", turtle_name));
}

int main(int argc, char** argv)
{

        ros::init(argc, argv, "my_tf_broadcaster");

```

Transform Broadcaster –cont'd

turtle_tf_broadcaster.cpp

```
27
28
29          if (argc != 2)
30          {
31                  ROS_ERROR("need turtle name as argument");
32                  return -1;
33          }
34
35          turtle_name = argv[1];
36
37
38          ros::NodeHandle node;
39          ros::Subscriber sub = node.subscribe(turtle_name+"/pose", 10,
  &poseCallback);
40
41
42          ros::spin();
43
44          return 0;
45 };
46
47
```

Transform Listener

turtle_tf_listener.cpp

```
1 #include <ros/ros.h>
2 #include <tf/transform_listener.h>
3 #include <geometry_msgs/Twist.h>
4 #include <turtlesim/Spawn.h>
5
6 int main(int argc, char** argv)
7 {
8
9         ros::init(argc, argv, "my_tf_listener");
10
11
12        ros::NodeHandle node;
13
14
15        ros::service::waitForService("/spawn");
16        ros::ServiceClient add_turtle =
  node.serviceClient<turtlesim::Spawn>("/spawn");
17        turtlesim::Spawn srv;
18        add_turtle.call(srv);
19
20
21        ros::Publisher turtle_vel = node.advertise<geometry_msgs::Twist>("/-
  turtle2/cmd_vel", 10);
22
23
24        tf::TransformListener listener;
25
26        ros::Rate rate(10.0);
```

Transform Listener-cont'd

turtle_tf_listener.cpp

```
27          while (node.ok())
28          {
29
30                  tf::StampedTransform transform;
31                  try
32                  {
33                          listener.waitForTransform("/turtle2", "/turtle1",
    ros::Time(0), ros::Duration(3.0));
34                          listener.lookupTransform("/turtle2", "/turtle1",
    ros::Time(0), transform);
35                  }
36                  catch (tf::TransformException &ex)
37                  {
38                          ROS_ERROR("%s",ex.what());
39                          ros::Duration(1.0).sleep();
40                          continue;
41                  }
42
43
44                  geometry_msgs::Twist vel_msg;
45                  vel_msg.angular.z = 4.0 * atan2(transform.getOrigin().y(),
46
    transform.getOrigin().x());
47                  vel_msg.linear.x = 0.5 *
    sqrt(pow(transform.getOrigin().x(), 2) +
48
    pow(transform.getOrigin().y(), 2));
49                  turtle_vel.publish(vel_msg);
50
51                  rate.sleep();
52          }
53          return 0;
54 };
```

Compiling Code

add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)

target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})

add_executable(turtle_tf_listener src/turtle_tf_listener.cpp)

target_link_libraries(turtle_tf_listener ${catkin_LIBRARIES})

```
## Specify libraries to link a library or executable target against
# target_link_libraries(${PROJECT_NAME}_node
#   ${catkin_LIBRARIES}
# )

add_executable(turtle_tf_broadcaster src/turtle_tf_broadcaster.cpp)
target_link_libraries(turtle_tf_broadcaster ${catkin_LIBRARIES})

add_executable(turtle_tf_listener src/turtle_tf_listener.cpp)
target_link_libraries(turtle_tf_listener ${catkin_LIBRARIES})

#############
## Install ##
#############

```

Compiling Code

```
$ cd ~/catkin_ws
$ catkin_make
$ source devel/setup.bash
$ roscore
$ rosrun turtlesim turtlesim_node
$ rosrun learning_tf turtle_tf_broadcaster __name:=turtle1_tf_broadcaster /turtle1
$ rosrun learning_tf turtle_tf_broadcaster __name:=turtle2_tf_broadcaster /turtle2
$ rosrun learning_tf turtle_tf_listener
$ rosrun turtlesim turtle_teleop_key
```

space

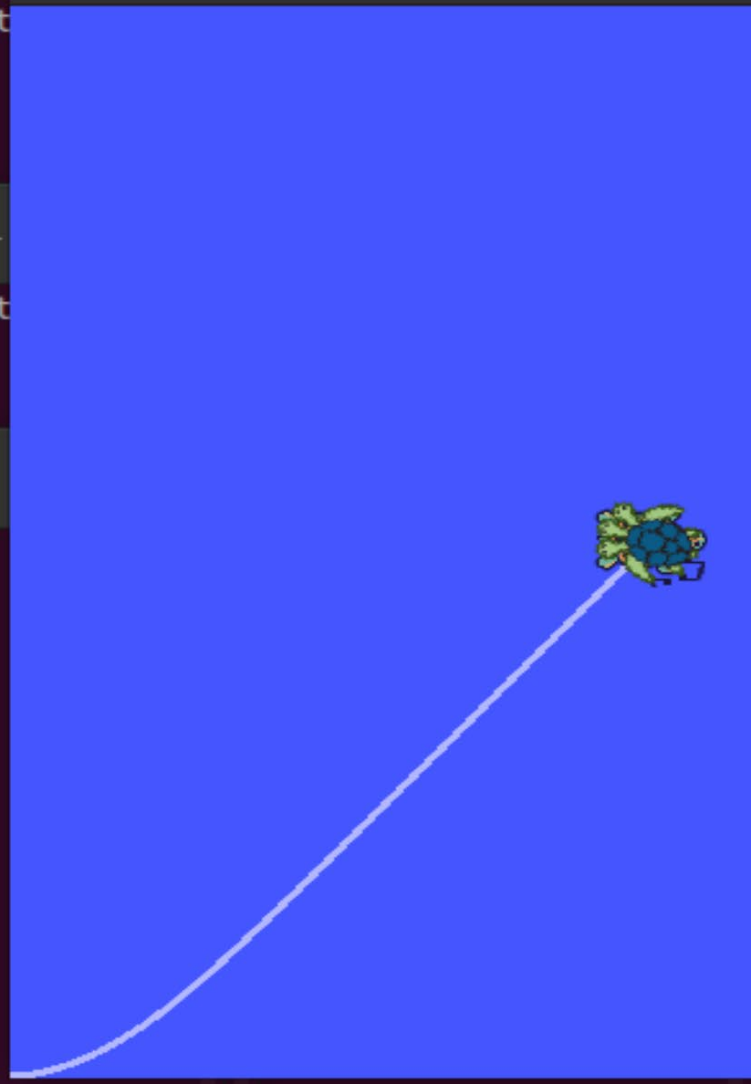
space

Rosrun

```
ab@ab-c: ~
ab@ab-c:~$ rosrun learning_tf turtle_tf_broadcast
:=turtle1_tf_broadcaster /turtle1

ab@ab-c: ~
ab@ab-c:~$ rosrun learning_tf turtle_tf_broadcast
:=turtle2_tf_broadcaster /turtle2

ab@ab-c: ~
b@ab-c:~$ rosrun learning_tf turtle_tf_listener
```



turtle_tf_broadcaster.py

```
1 import roslib
2 roslib.load_manifest('learning_tf')
3 import rospy
4
5 import tf
6 import turtlesim.msg
7
8 def handle_turtle_pose(msg, turtlename):
9     br = tf.TransformBroadcaster()
10     br.sendTransform((msg.x, msg.y, 0),
11                      tf.transformations.quaternion_from_euler(0, 0,
  msg.theta),
12                      rospy.Time.now(),
13                      turtlename,
14                      "world")
15
16 if __name__ == '__main__':
17     rospy.init_node('turtle_tf_broadcaster')
18     turtlename = rospy.get_param('~turtle')
19     rospy.Subscriber('/%s/pose' % turtlename,
20                      turtlesim.msg.Pose,
21                      handle_turtle_pose,
22                      turtlename)
23     rospy.spin()
24
```

turtle_tf_listener.py

```
1 import roslib
2 roslib.load_manifest('learning_tf')
3 import rospy
4 import math
5 import tf
6 import geometry_msgs.msg
7 import turtlesim.srv
8
9 if __name__ == '__main__':
10     rospy.init_node('turtle_tf_listener')
11
12     listener = tf.TransformListener()
13
14     rospy.wait_for_service('spawn')
15     spawner = rospy.ServiceProxy('spawn', turtlesim.srv.Spawn)
16     spawner(4, 2, 0, 'turtle2')
17
18     turtle_vel = rospy.Publisher('turtle2/cmd_vel',
   geometry_msgs.msg.Twist,queue_size=1)
19
20     rate = rospy.Rate(10.0)
21     while not rospy.is_shutdown():
22         try:
23             (trans,rot) = listener.lookupTransform('/turtle2', '/turtle1',
   rospy.Time(0))
24         except (tf.LookupException, tf.ConnectivityException,
   tf.ExtrapolationException):
```

turtle_tf_listener.py - cont'd

```
            continue

        angular = 4 * math.atan2(trans[1], trans[0])
        linear = 0.5 * math.sqrt(trans[0] ** 2 + trans[1] ** 2)
        cmd = geometry_msgs.msg.Twist()
        cmd.linear.x = linear
        cmd.angular.z = angular
        turtle_vel.publish(cmd)

```

Reference

- R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd Edition, 2011, ISBN-10: 0262015358.
- Y. Pyo, H. Cho, R. Jung, and T. Lim, ROS Robot Programming, ROBOTIS Co., Ltd., 2017, ISBN 979-11-962307-1-5
- J. O'Kane, A Gentle Introduction to ROS, CreateSpace Independent Publishing Platform, 2013, ISBN-13: 978-1492143239