# EE565: MOBILE ROBOTICS

## LAB # 4: USING IROBOT CREATE IN ROS AND IMPLEMENTING ODOMETERIC MOTION MODEL WITH NOISE

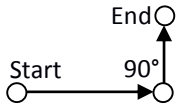
### IN-LAB WORK

This lab is composed of two components, Hardware and Simulation, that can be performed independently. All students will work on both aspects of a real commercial differential drive mobile robot (*iRobot Create*) i.e. the hardware usage and simulation testing. At any given time during the lab, all groups will either be working on the Hardware part or the Simulation part.

#### HARDWARE COMPONENT:

During the hardware experiment students shall learn how real robots interact with ROS. Mobile robots can use existing ROS components to interact with robot actuators and to acquire sensor data. IRobot Create communicates with ROS using an RS232 serial connection. We can navigate the robot using teleop_keyboard node. The robot's odometry data is being published on /odom topic. Use the following steps to create a dataset which shall be used to implement robot odometry motion model.

1. Install turtlebot packages (Debs Installation only) found at {url}.
2. Follow Robot configuration robot from {url} for Create base (contrast with Kuboki).
3. Configure serial port and launch turtlebot_bringup after configuring it for Create base. {url} Only run TurtleBot Bringup (not Workstation bringup).
4. Navigate the robot with turtlebot_teleop package for keyboard. {url}
5. Echo odometery. Get familiar with rosbag.
6. Visualize the odometery in RViz.
7. Now perform the following task 6 times, and record each run in rosbag for latter analysis
    1. Mark two points on the ground (use two opposite corners of the ground tiles in lab as start and end points).
    2. Place the robot at start point. Align it along the edge of tile.
    3. Start recording the topics
    4. Keyboard teleop the robot along the first edge, rotate 90 degrees and move along the second edge and stop at corner. [Since don't have an externally localization system, therefore, visually observe, if the robot has acquire the target pose]
    5. Stop the recording.
    6. Do this experiment 6 times and save the data.




8. Return your equipment.
9. Now make a table of the actual and expected pose $(x, y, \theta)$ when the robot had reached the final point. State reasons for the discrepancies.

#### SIMULATION COMPONENT:

During the simulation component students shall learn to use and modify existing iRobot Create model in Gazebo. Gazebo can be used to model the friction and slippage between robot wheel and ground plane. In this simulation

experiment students shall modify existing iRobot Create Gazebo model to incorporate wheel slippage and then record a dataset.

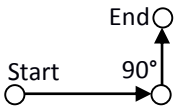
1. Spawning iRobot Create in Gazebo. [Launch gazebo with gazebo_ros package]
2. Incorporate differential-drive plugin to get the iRobot odometery data published as ROS topic (like done in Lab 2). Also add the following tags within the odometery plugin tag:

```
<wheelAcceleration>0</wheelAcceleration>
<odometrySource>encoder</odometrySource>
```

3. Modify the left and right wheel mu and mu2 inside the <ode> tag. These two parameters represent the coulombs friction coefficient of friction. The value can be between 0 and Infinity, where zero means a frictionless surface (Maximum slippage).
4. Using turtlebot_teleop, navigate the robot along a L-shaped path. [Translate, Rotate]

```
rosrun turtlebot_teleop turtlebot_teleop_key
```

5. Record and replay the simulated experiment dataset (odometery) in RViz to get graded.




## LAB ASSIGNMENT

For the modified simulation model of iRobot Create in Gazebo determine the odometry motion model's probability distribution.

### STEPS:

1. Implement a ROS node to timely publish /cmd_vel topic for trajectory following (L-shaped) 1m x 1m. Move the robot using this node and visualize in Gazebo. [You will need to implement ros::Timer so that you can keep track of duration of publishing data]
2. Read published (noisy) odometry and convert the data into $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ convention, in the same node.
3. Also read create model's position and orientation states from /gazebo/model_states as ground truth. At this point, you will have *(x, x', u)* as learned in Lecture 3.
    - *x : Initial pose of robot [from model states]*
    - *x' : Final pose of robot [from model states]*
    - *u : Odometry (rot, trans, rot) [from step 2]*
4. Publish the final pose of robot (*u*) as a Points type marker in visualization_msgs::Marker. {url}
5. Write code (inside the same node) for repeating steps 1-4 (run L-shaped trajectory 100 times, collect data and put final pose as points in the same visualization_msgs::Marker message).
    - In each iteration, there will be some error between ground truth pose (obtained through model_states) and odometeric data (obtained from noisy /odom)
6. Plot the 2D points where the robot arrives at end of each trajectory in RViz. [Use *MarkerArray*] [You will get a scatter of points, as well as the desired 2D point]
7. Find the mean and variance of $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ from sampled data.
8. **BONUS:** Graphically plot an oval gray region on the 2D plot of scattered points, that shows mean point and two-standard-deviations confidence region around the mean.