ME 133 (a): Lab #2

Turtlebot Differential Drive Kinematics and Odometry

The purpose of this lab is to introduce you to the *TurtleBot* system, improve your growing familiarity with the ROS programming environment, and to start to become familiar with the principles of robot *odometry*.

The Turtlebot is essentially a Roomba vacuuming robot that has been modified to allow for programming of its motions. It is a *differential drive* robot, as its motions are empowered by the interaction of two motor-driven wheels with the ground. In this lab you will be modifying a script to command a TurtleBot (Fig. 1) to trace out a simple shape while it is being tracked by the OptiTrack system. During the test the TurtleBot will be logging readings from the encoders on each of its wheels, which you will convert into Cartesian position using the differential drive vehicle equations developed in class and in the class hand-out, and compare with the path you intended it to follow. We can then compare the intended path, the measured path from encoder odometry, and ground truth from OptiTrack to see the level of accuracy we can expect from this type of sensing.

Figure 1: The turtlebot robot

Lab Preparation:

It is assumed that you have completed the *pre-lab* assignment using ROS and the Rviz simulation package. You will turn in the pre-lab write-up when you carry out your experiment in the CAST center.

Appointments for the hardware demonstration can be made via the instructions sent by the class T.A.s in an email. Before your scheduled demo time you will need to add code to a script to make the

TurtleBot trace any simple pattern of your choice that keeps it within 1 meter of the starting location and completes within 20 seconds. The template for this script can be downloaded from the class website. Units for linear and angular velocity in the script are in m/s and rad/s which should be kept below 0.3 m/s and π/2 rad/s.

You will modify the script and test it in simulation before bringing it to your scheduled lab slot in order to make sure everything is working as expected. This can be achieved by transferring the modified script to an easily accessibly location such as the 'home' directory, open the simulator in a new terminal (CTRL + ALT + T) and running the command:

```
$ roslaunch turtlebot_stage turtlebot_in_stage.launch
```

Once the simulation has opened, you can test your script in a second terminal navigated to the location of the script with the command:

```
$ python me133a_lab2.py
```

If everything has been written correctly you should see the turtlebot trace out your desired shape and then come to a stop. If anything goes wrong, check through your code and debug any issues until the script works as expected. If you need to move the turtlebot around so that it won't collide with any walls during a test you can control it with the command:

```
$ roslaunch turtlebot_teleop keyboard_teleop.launch
```

Don't forget to stop keyboard control with CTRL + C before running the python script again. Note that the encoders and other sensors are not simulated in this program, so the csv file created by the script will be empty as no sensor messages are published.

Hardware Demonstration:

Show up to your scheduled time slot with your modified python script ready to be transferred to the lab robot's control computer. The TurtleBot will be placed on a marked starting position and then your script will be run, moving through whatever pattern you designed as OptiTrack records its position. After the program finishes, we will provide you with the data recorded by the TurtleBot and OptiTrack for use in plotting these for the homework deliverables.

Odometry Data Processing:

Use the kinematic equations from the class handout on differential drive vehicles to calculate the XY position and orientation of the TurtleBot over the course of your program, and plot these variables as a function of time. The wheel encoder values in the csv you receive after testing give the number of 'ticks' seen by the encoder on each wheel. There are 2578.33 ticks per wheel revolution, which can then can be multiplied by 2π to give ϕ, with a wheel radius of $\rho = 3.5$cm. The encoder values are 16-bit numbers which means if they exceed 65535 they will roll back to zero, which you may need to account for. The distance between the wheels or 'wheel base' is $2W = 23$cm. The encoder readings are taken at a rate of 50Hz, but when using the backward difference equation the time step is not needed as the relation is purely kinematic.

Homework Deliverables:

Submit a plot of the XY plane that includes three trajectories: your original intended path, the path traced by the odometry measurements, and the ground truth provided by the OptiTrack system. Secondly, plot the robot's orientation, θ, for each of the three paths against time (rotation about the Z axis, which will need to be converted from Quaternion for OptiTrack). Comment on both the accuracy of the odometry estimate and how well the motion of the TurtleBot adhered to your intended pattern.