ECE 232 Lab 3

Matthew Dombroski

March 21, 2016

I affirm that I have not given or received any unauthorized help on this assignment, and that all work is my own.

Abstract

For laboratory 3 a simplified beam model was implemented within the simulator written for the previous lab. In addition a GUI to visualize the position of the robot and feedback from the beams was built. Beam data with both the simulated robot and the Turtlebot with Kinect was collected and compared to see how the simulator matches up with the real world robot.

1 Code Implementation

1.1 GUI

Within the file `gui.cpp` the functions `handle_laserscan` and `handle_odom` were written. The purpose of these function was to simply store the laser scan data and odometry data within class variables, which were declared in `gui.h`. This is then used within the `paintGL` function to draw the robot and beams in the correct location within a global system known by the simulator. Basic trigonometric calculations were used to compute the coordinates for drawn objects.

An additional line was added to the function `timer_callback` to initiate a repaint of the screen. WIthout this line the screen would only redraw when clicked on or dragged.

1.2 Simulator

The simulator replicated the physical layout given in the lab 3 instructions, with a 0.1m radius cone at 1m distance from the robot and a wall at 2m distance from the robot. The problem of replicating this physical layout was broken down into two main functions - `calcTrueDistance` and `calcNoisyDistance`. `calcTrueDistance` calculates the noise free beam ranges given a robot position and setup geometry. By sweeping over each angle for the beams the distance between wall or cone and the robot is computed. To calculate the distance to the wall simple trigonometry using parrallel lines is used. To calculate the distance between the robot and different point on the cone a more involved calculation is needed.

First the angle between the oncoming beam and the radius of the cone intersecting the beam on the outer edge of the circle is computed using the equation:

$$b = asin\left(\frac{distance\_cone * sin(angle)}{cone\_radius}\right)$$

Next the angle between the radius and line segment connecting the robot's center point to the cone's center point is computed with the equation:

$$c = 180 - angle - b$$

Finally the distance to the edge of the cone is computed by the equation:

$$C = \frac{distance\_cone * sin(c)}{sin(b)}$$

The minimum and maximum angle for which the beam will intersect with the cone can be found by the formula:

$$angle\_min\_max = atan\left(\frac{radius\_cone}{distance\_cone}\right)$$

As the distance for each angle is computed the current beam angle is compared with the bound calculated by the equation above and if within those bounds the formulas for calculating the distance to the cone are used. Outside these bounds the normal equations for computing the distance to the wall are used.

After computing the true distances the noisy distances are calculated within `calcNoisyDistances`. To save a bit of computational power the true distances are only calculated once and afterward the noisy distances are calculated with every ROS update as the noisy distances are easier to compute. The noise for each beam is computed the same way the noise for the motion model was computed in the previous lab by summing 12 random values that are bounded by the variance of the measurement and then this value back to the true distance.

After all calculations are made the computed values are then published on the topic `scan` to be read by the GUI and any other nodes requiring the data.

1.3 Data Collection

To write the beam data to a file the `ros_subscriber` program written in lab 1 was modified. Another callback function `save_scan` was added and all data contained in the `sensor_msgs::LaserScan` message are written into a text file.

2 Data and Analysis

Figures 1 and 2 show the beam data collected. Figure 3 and 4 show screenshots of the GUI and drawing of the robot and beams.

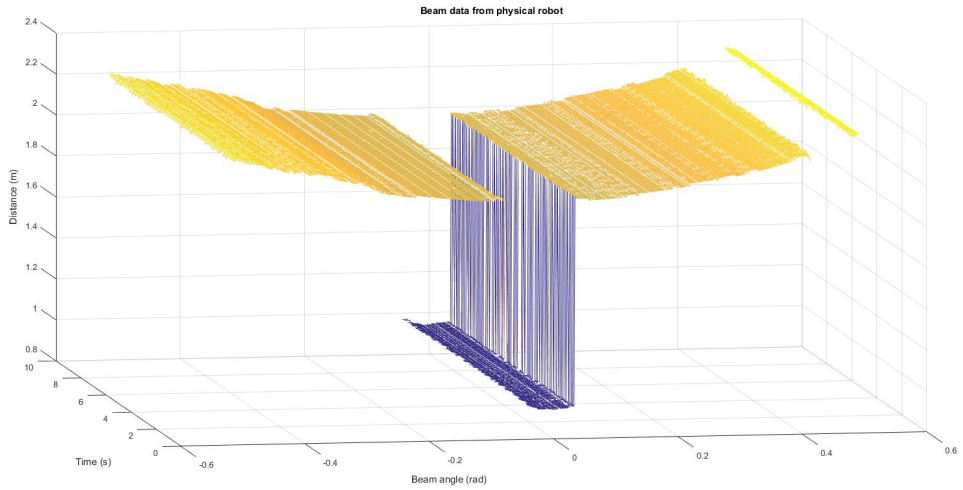

Figure 1: Beam data from lab with a physical robot and beam sensor

The most striking difference between the data collected was the noisiness of the sensor readings. The physical beam sensor has very little noise whereas the simulated beam model had a large proportion of noise. The mean distance readings from the simulator remained close the expected distance values.

In addition the cone appears to be much wider in the simulated beam model. The angular span of the lab cone is much smaller than the simulated cone. This could be from the fact that the cone from the lab had a smaller radius than the simulated cone.

Overall both the simulated and in lab beam data returned reasonable results. The sensors each gave a realistic and fairly accurate representation of what the world in front of the robot looked like.

3 Laboratory Questions

Comparing the beam data between the simulator and physical robot two main difference are immediately noticeable. First the noise of the physical beam scan was much lower than that in the simulator. Where in the simulator there was upwards of 0.4m deviation the physical beam scan had a deviation of only a few hundredths of a meter. This discrepancy is more than likely the result of the fact that the variance of the physical scanner is lower than the variance used for the simulator. Another theory is that the function used to compress the 3D point cloud into planar ranges does some form of smoothing and filtering of the data, reducing the noise of the returned ranges.

Secondly it was observed that the width of the cone in the simulator was much wider than in the physical beam scans. Using data collected from lab it was found that the angular

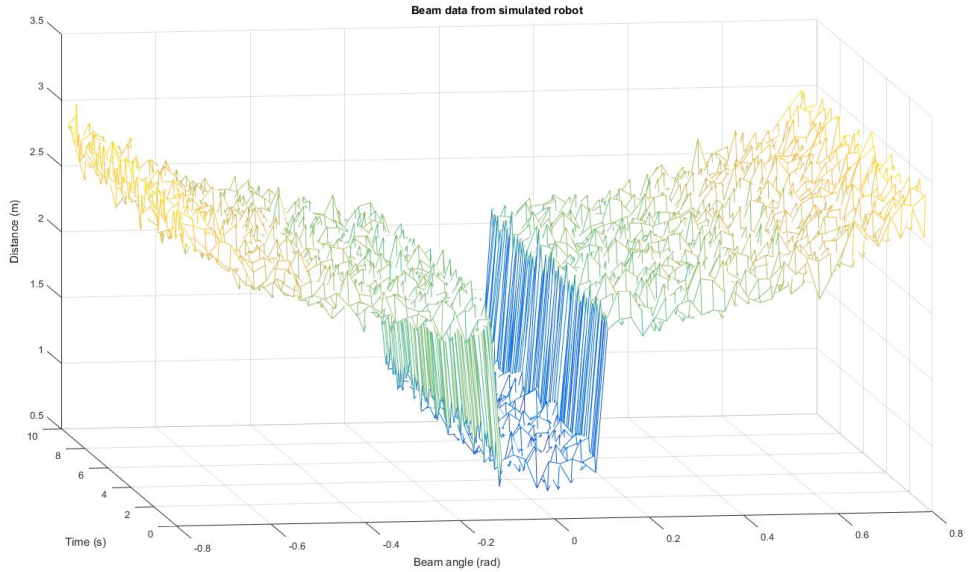

Figure 2: Beam data from the simulator

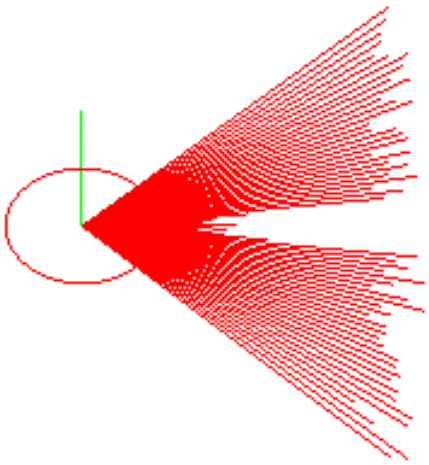

Figure 3: Screenshot of beam model in the simulator

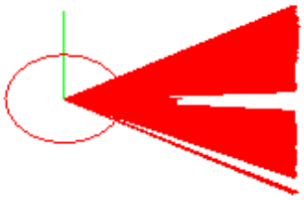

Figure 4: Screenshot of the beam model from the lab

diameter of the cone was 0.0745 radians and the distance to the closest point of the cone was 0.937m. Using the formula relating angular diameter of an object to its radius the radius of the physical cone can be computed to be roughly equal to:

$$R = \sqrt{D^2 - (Dcos(\frac{\alpha}{2})^2)}$$

$$R \approx 0.03m$$

Thus the radius of the cone in lab was approximately 30% the radius of the cone in the simulated environment.