Objective
The objective of this project was to implement a method to detect the glass walls of the Computer Science Labs. This was extended to detect all walls and the final goal is to use the detection of these walls to improve the corobots general wall avoidance and to improve localization by publishing data related to the position of the wall and the corobots orientation and distance to the wall.

Introduction
The corobots of the CS Robotics lab are not currently able to detect the glass walls of the labs in the department. So using the one of the onboard sensors, the depth sensor of a Microsoft Kinect, we have set out to detect the intersections between the floor and the kick guards of the glass walls and to take this knowledge an publish it on the robots operating system(ROS) in a manner that is usable by other ROS nodes.

Depth Images
Below is a colorized example of an input image that we will be working with, the data that we get in is a gray scale image where the lighter the color the farther away the pixel is. The pixels that were too far away from the Kinect to detect and the pixels that were too close to the camera are both represented as NaN values in the input data, therefore all pixels with the value of NaN are useless to any calculations we would like to make.

Figure 3: The depth image, colored

Linear Formula
The linear function takes in the raw depth data and runs each pixel through a simple formula.

\[ aX + bY + cZ + d = 0 \]

In the formula, \( a \) and \( b \) represent constants related to the X, Y distances to the center of the censor, \( c \) is related to Z is the recorded depth and \( d \) is the height of the robot. One limitation we discovered during this phase was that the support structure cuts down the depth image on the right hand side, so we ignore values right of a horizontal pixel value of 480.

Figure 1: The raw image

Hough Line Transformation
Hough transform is a common method for determining important edges in images. First we take in the output array from the linear function and run OpenCVs canny edge detection, the output of which is shown as the thin white line in the figure below. We then take that new image and run it through OpenCVs Hough transform function which gives the thick white lines which have been laid over the canny edge output. Those lines are then converted back to real world coordinates and published with robots angle and distance with respect to the wall.

Figure 5: The detected lines

Results
The Linear function was able to detect the floor with a high level of accuracy and was able to due so in an average of 0.6 seconds, the canny edge detection and hough transform run in an average of 0.04 seconds. The detected floor matches up well with the input image when checked by eye, however it is not yet perfect and some fine tuning of the constants used in the formula is still needed.

Figure 4: The output

Conclusion
The algorithms that were implemented in this project are well suited for detecting glass walls, however there are many hardware limitations and software limitations. One of the most critical components of the algorithm was determining the proper constants for the linear formula as well as how to implement the formula and how to calculate the X and Y coordinates.

In the end the project works well with detecting the edges of the floor and publishing the data corresponding to the wall segments in a reasonable amount of time, about 1 second per frame.

Future Work
Features and optimizations we would like to make but did not have time to finish or implement:
- Implement a check to see if the endpoints had NaN for depth before publishing.
- Move the constants to a settings file so that it is easier to deploy updates to multiple robots.
- Finish fine tuning the constants for the linear formula, canny edge detection and the hough transform.
- Run tests on the robot of the code and the finished future work.
- Upgrade the existing wall avoidance node to use the published data to avoid walls.
- Implement a node to read in the published data and use it to improve localization.
- Implement a node to read in the published data and use it to improve the robots Odometry.

Contact Information
- Email: nbeckboa@msn.com

Acknowledgments
- Advisor: Professor Butler: zjb@cs.rit.edu