TurtleBot: Crossing

Vertically Integrated Project: Robosense

Fall 2014

Hongrui Zheng

Weiyu Liu

Professor: ***Fumin Zhang***

Graduate Student Advisor: ***Carol Young***

Introduction

The objective of the project group this semester is to design an algorithm for robots with limited field of vision to have the ability to cross the junction of hallways in workplace. There are a lot of researches dealing with obstacles avoidance. However, crossing the junction is a more interesting problem that involves robot controlling, data processing, and ultimately robot learning. We believe this topic can provide insight into the human-robot interaction. We created programs using Turtlebot as the platform to perform the following: centers itself in the hallway, stop right before crossing the junction, locate a moving object and determine its distance and velocity with respect to the Turtlebot. These are the groundworks for developing programs in the next phase dealing with real life conditions. In our approach to this problem, we used Matlab to process and visualize sample data, and developed programs according to our results in Matlab. We also studied several research papers about HRI and object detection algorithm to understand the background of this topic and bring insight into our solutions.

Background

In the aspect of Human Robot Interaction, a robot should know what matters and what action to take[1]. The robot need to have an internal simulation of the human environment in parallel of the environment it perceives. This gives the robot the ability to rehearse many actions or manipulate environmental factors without disturbing and causing damages to the true world. The internal simulation has to be constantly updated by new information the robot perceives, and it should also be able to predict the changes according to guidelines such as physical laws if new information are interrupted. In the crossing behavior, this predictive power allows robot to find the appropriate time to cross even if it does not have real time information about both sides of the junction.

In order to have a fast visual sensing and small data size, we used laser range finder in Microsoft's Kinect. This laser range finder provides about 37 degree of field of view and a maximum operating distance of about 7 meters. This very limited capability has caused a lot of difficulties in our project. However, this choice is low-cost and a good starting point. As our robot platform, TurtleBot combines popular off-the-shelf robot components, the iRobot Create, Yujin Robot's Kobuki and Microsoft's Kinect into an integrated development platform. Behind the TurtleBot, ROS acts as its operating system. ROS is an open-source framework, which contains collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Our work in TurtleBot can be simply transferred to other robotic platform using ROS such as the wheelchair project in another project group.

Results

Structure of the Crossing program：

1. Center in hallway and stop before junction
2. Turning behavior
3. Detecting moving objects and measuring speed
4. Repeating 2&3 for another side
5. Making decision
6. Repeating 2&3 while crossing
7. Crossing the junction
8. Learning

The centering program, we use the difference between the ranges taken from the _msg.ranges[] array from laser scan data. Our original idea was to take two data points with same spacing on each side of the center laser beam. In order to determine which area the robot is in with respect to the centerline of the hallway or which direction the robot is facing, we subtract the two data points on each side compare the absolute value of the differences. Greater difference indicates which direction the robot should be

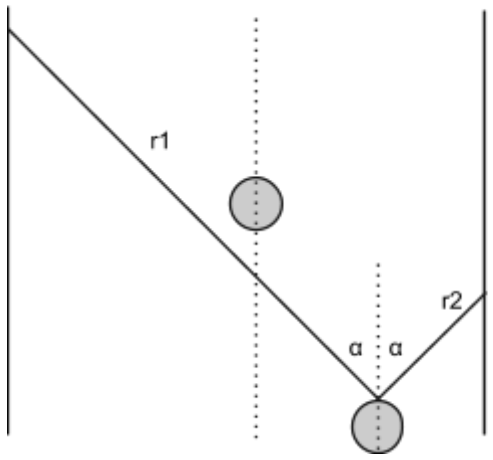

Figure 1

going (i.e. if r1 > r2, robot moves to the left). After making the turn and move forward, the robot will continue to read in the data from the kinect and repeat the calculation in each loop of the main method. The ideal goal is to make constant turns and eventually moving forward on the centerline. The idea of using the differences didn't work as well as we expected. First of all, data points with value of NaN will appear, the difference in ranges between the two data points with spacing of 5 is too small, the subtraction in our calculation will only magnify the error in the measurements. We came up with a simple idea to only use two laser beams which have the same angle between them and the center beam. We then simply compare the two data points and determine which way the robot should turn (i.e. if the right range is larger then the robot turn to the right). We also implemented turning gain on the angular speed. It will allow the robot to turn faster when the difference between the two data points is larger (i.e. the offset angle between the center beam and the hallway's center line is larger). We also included a tolerance to determine whether the robot is facing the center of the hallway to eliminate the vibration of the robot when the difference between the two data points become smaller.

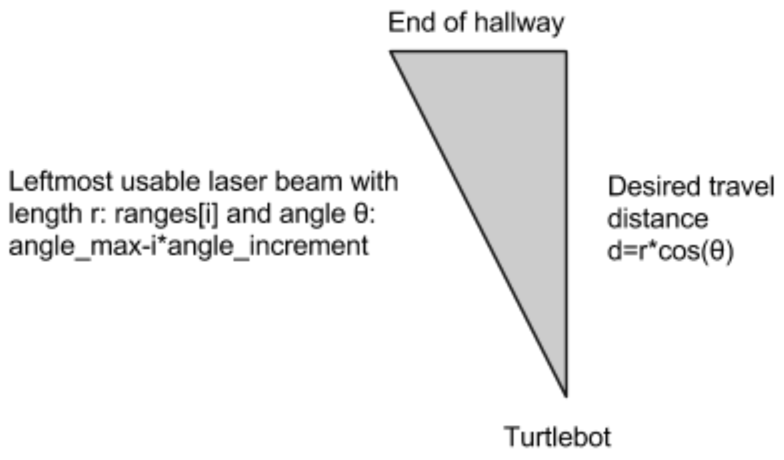
End of hallway
Leftmost usable laser beam with length r: ranges[i] and angle θ: angle_max-i*angle_increment
Desired travel distance d=r*cos(θ)
Turtlebot

Figure 2

The stopping program is designed to prepare the robot for the next step. The position of the robot when it stops moving determines whether the robot will involve in an accident. We used the rightmost non-NaN data point to do the calculations. The robot will constantly take measurements while it moves until it finds a jump in the rightmost beam. Then it will use the data before the jump to calculate the distance it should move in order to stop right at the end of the hallway. We used simple trigonometry to calculate this distance. The index of the jump in the array also indicate the angle from the jump to the center beam, which is calculated by: θ = _msg.angle_max - i*_msg.angle_increment. With the angle and the data before the jump, we can calculate the distance the robot should travel by: d = r*cos(θ). After the calculation, we used odometry to measure the distance the robot travelled from the point it finds the jump. When it hits the calculated distance with a tolerance, it stops publishing the linear velocity, which stops the robot.

The turning behavior of the robot used a counter with the loop rate of the main method. After the robot stops at the end of the hallway, it starts publishing angular velocity to turn to the right. In the mean time, the counter will increase. When the counter hits the upper limit, the program will publish angular velocity in another direction. In the mean time, the same counter will decrease until it hits the lower limit. In the ideal situation, the turning behavior will happen simultaneously with the object detection and speed measuring program. We experienced some inconsistency in the turning behavior due to the inconsistent performance of the differential drive of Turtlebot. This cause the robot to turn to different angles in the two directions. It also caused problems for the robot to return to its position prior to the turning in order to cross the hallway. We decided that next semester we will implement a turning platform for the kinect similar to a lazy susan. With this feature, instead of turning the whole robot we can turn the platform and let the robot stay in the position and ready for turning. We could even turn the platform while the robot is crossing the hallway adding a safety feature to the program.

For detecting moving object, we use command “rostopic echo [rostopic] > sample.txt” to extract data from laser scan. This allow us to analyze scan data and manipulate them in Matlab. We wrote code to parse text file and put scans into a cell array where each cell contains 640 points of a single scan. The graph below is a scan where a human stand 4 meters away and to the left.

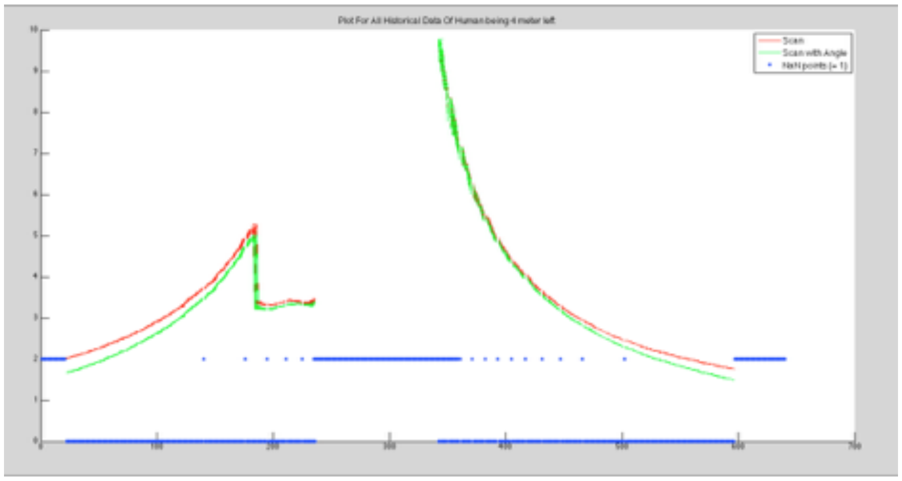

Figure 3

The two curves in the left and right side of the scan correspond to the hallway walls. The straight horizontal line that connect to the left curve is believed to be the person. Because of the limited capability of Kinect detecting object that is too far or too close, these points have the value of NaN. This is shown as the blue dots with value of 2 in the graph. The majoring of NaN occurs in the middle because that is towards the end of hallway which is too far for Kinect to detect. Other appearance of NaN values are assumed to be noise. In order to locate the segment of points that correspond to the person, we use the jump that exceeding threshold in the continuous data points to determine the start and end of the segment. The NaN values are first filled by valid values from its left-side neighbors in able for detecting jump. After we find the segment, the points where NaN used to be are automatically discarded. In this way, we developed a way of detecting object hallway. The final result is shown in the graph below, where the yellow line circles the segment of interest.

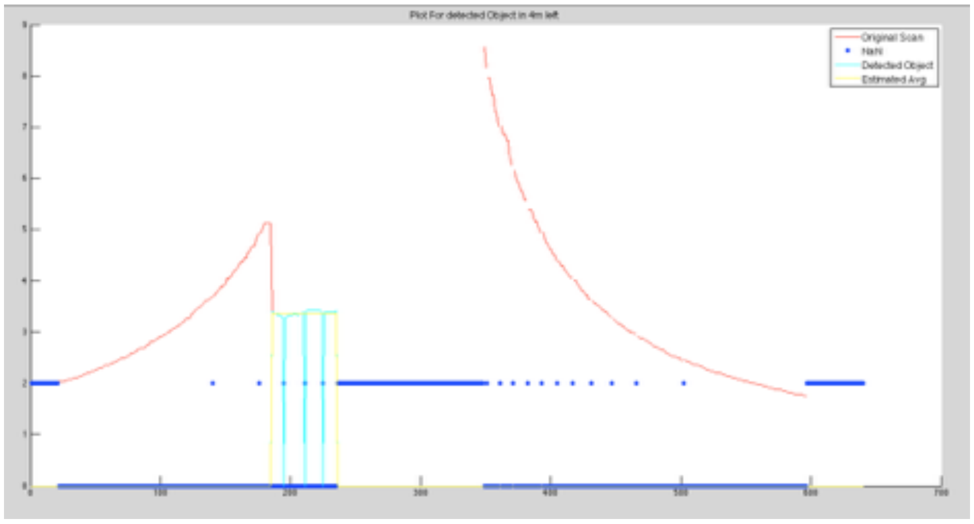

Figure 4

Then the same logic was implemented in the code for TurtleBot. After multiple trials, this algorithm shows a robust way of detecting single object and measuring distance between object and robot.

Future Work

The ultimate goal of this project is to create a crossing program imitating the crossing behavior of the human being. We have already established solid groundwork for the first half of the program. In the next semester, we will start designing the algorithms for decision making. We will also try different algorithms to locate the moving object to make the program more robust. Furthermore, the ability of detecting multiple objects is also what we would like to implement in the current program.

Future Goals and Tasks:

1. Implementing a more robust object identifying and speed measuring algorithm
2. Integrate the hallway centering, stopping, turning, object identifying, speed measuring and crossing programs

3. Design a decision making algorithm
4. Further research into HRI and machine learning

One of the problems we encountered during this semester is that we couldn’t get our hands on the Turtlebot as often as we wanted. We had to spend almost half of the meeting time we have every week to debug the programs we wrote and sometimes redesign them. If we could have meetings more often in the lab during the week we could achieve a lot more than we had this semester.

Conclusion

In all, this project is an attempt to implement object detection algorithms into a more dynamic environment and explore the HRI through the behavior of crossing junction. Although this project has not completed, we have wrote codes for wall-centering, junction-detection, and measuring speed. We have also learned different algorithms for detecting moving object in fuzzy conditions. These will prepare us to achieve our goal in following semesters.