2D Simultaneous Localization and Mapping

Henry Grittner , HyeongJoo Hwang, Joel Amert, Kent Sommer
Department of Computer Science, University of Minnesota

May 11th, 2016

Abstract

Simultaneous localization and mapping (SLAM) is the process of a robot determining its current position relative to its starting position and developing a map of its surroundings while navigating in an unknown environment. This is necessary for robots that operate in environments in which there is no prior map knowledge and where navigation beacons or similar infrastructure is unable to be installed. In this paper SLAM is implemented in a two dimensional environment using a differential drive Pioneer robot equipped with a laser scanner. The independent wheel velocity readings were used to propagate the robot's position and orientation, and the laser scanner readings were used for updates. The extended Kalman filter was used for both the propagation and update steps of SLAM. The robot successfully navigated the test course of a rectangular hallway while using the laser scanner to update its position and orientation estimates.

1 Introduction

As the cost of sensors and robots continue to drop, robots that are capable of navigating and mapping their environment are increasing in popularity. With this increase in popularity arises the need for robots that are capable of accurately navigating and mapping their environment without having prior knowledge of the area and without having a method of sensing its position directly. While it is possible to estimate the location of the robot using information such as speed and turn rate, this method of estimating the position of the robot can quickly lead to errors. As time increases, the errors in the position of the robot will continue to compound if no method of decreasing the error is available. This can lead to position and orientation uncertainties that are well outside the acceptable margin for an autonomous robot.

The problem of accurately estimating a robots position and mapping the environment around it is what simultaneous localization and mapping (SLAM) tries to solve. SLAM is the process of a robot simultaneously creating a map of its surroundings as well as determining its location in an unknown environment. This is especially important for the movement of a robot indoors, since without the aid of a compass, GPS, or prior map, a robot must still be able to determine its relative location and generate a map of its surroundings.

There are two main steps to SLAM[5]: propagation and update. Propagation occurs the same way that it would without using SLAM at all; odometry information can be used to estimate the robot's trajectory over time using information about its linear and rotational velocities. These measurements are integrated over time, and are used to determine how the robot's position and orientation change from one time-step to the next. An update occurs when there is additional information from the environment that can be used to estimate the location of the robot, such as the relative location of a landmark near the robot. This estimate is then combined with the propagation estimate allowing for a reduction in the error of the robots trajectory estimate.

Obtaining the information used in the updating step consists of processing information from the environment in order to extract features near the robot. An example of this method would be a robot equipped with a laser scanner or a camera, from which the robot gets data such as a laser scan or a picture of the environment. After receiving this information the robot can extract features, such as hallway corners or distinctive markings on the walls, which can be used for the SLAM update step.

2 Related Work

Simultaneous localization and mapping (SLAM) has been a known problem since the 1986 IEEE Robotics and Automation Conference in San Francisco. Since the 1986 conference, many researchers including Smith and Cheesman, and Durrant-Whyte published papers establishing the basis for SLAM as we know it today[2]. An important part of their work was determining that there must be a large degree of correlation between estimates of different landmarks in the map and that over time these correlations would grow.

The largest break-through was the discovery that combined mapping and localisation, once formulated as a single probabilistic estimation problem, actually converged. Many researchers had been working to minimize the correlation between landmarks, however, a large breakthrough came when it was realised that these correlations were actually the key. The more these correlations could grow, the more accurate the solution[3]. Since this point in time, multiple different approaches to SLAM have been researched and studied including the Extended Kalman Filter (EKF) SLAM model we chose for this project.

3 Problem Description

Given a Pioneer 3 robot (Figure 1) *in an unknown but structured environment, navigate through hallways making left turns whenever possible, all while localizing the robot and creating a map of the environment.*

The Pioneer 3 is a differential-drive robot and is steered by controlling the velocities of the left and right wheels separately. To estimate the velocity of the robot and the turning rate, each wheel is equipped with 500-tick wheel encoders. The Pioneer 3 also has 8 sonar sensors, a 1.6 m/s maximum velocity, and is equipped with a SICK LMS-200 laser scanner. The SICK LMS-200 laser scanner has a 180 field of view, an 80m maximum range, a 10m typical range, and a standard deviation of 5mm when the reading is within 8m.

Using information from the wheel encoders, it is possible to estimate how the position and orientation of the robot change by propagating at every time step. This will result in a higher than desirable error in the robot position estimate, however, so it is necessary to use the readings from the laser scanner to update the estimate of the robot location.

Figure 1: Pioneer with Sick Laser Scanner. *www.ti.uni-bielefeld.de*

4 Solution and Analysis

4.1 Kalman Filter

For the following analysis, let N be the number of known Landmarks and M the number of landmark measurements at a given time step $k+1$. N is assumed to grow larger than M as new landmarks are discovered and saved.

The first step is propagation. The Kalman Filter propagates a prior state vector (containing the position and orientation of the robot and the location of known landmarks in the global frame) and its covariance matrix based on the previous posterior state vector, covariance matrix, and control input which consists of instantaneous linear and rotational velocities. Since the robot's position and orientation estimates are the only things changing during propagation, only the first column and the first row change from the previous posterior covariance matrix. By applying block operations for calculating the covariance matrix of the new state vector, propagation can be done in $O(N)$ time (if block operations are not used, however, the total time becomes $O(N^3)$).

$$P_{k+1|k} = \begin{bmatrix} P_{RR_{k+1|k}} & \Phi_{R_k} P_{RL_{1,k|k}} & \cdots & \Phi_{R_k} P_{RL_{N,k|k}} \\ P_{L_1 R_{k|k}} \Phi_{R_k}^T & P_{L_1 L_{1,k|k}} & \cdots & P_{L_1 L_{N,k|k}} \\ \vdots & \vdots & \ddots & \vdots \\ P_{L_N R_{k|k}} \Phi_{R_k}^T & P_{L_N L_{1,k|k}} & \cdots & P_{L_N L_{N,k|k}} \end{bmatrix} \Rightarrow O(N)$$

After Propagation, the Kalman Filter obtains the set of inferred measurements from the feature extraction process. Since each of the measurements has its own corresponding covariance matrix, the Kalman Filter forms a set of covariance matrices which correspond to the set of measurements.

Given the prior state and its covariance from the propagation step as well as the set of inferred relative position measurements and their set of covariance matrices, the update procedure estimates the posterior state vector and its covariance matrix. The update step takes $O(M * N^2)$, going over the following tasks:

For one measurement z_j, the Kalman Filter calculates the Mahalanobis distances between that measurement z_j and all landmarks in the state vector in order to classify z_j as a new landmark or as a known landmark. To calculate the Mahalanobis Distance, the Kalman Filter needs the inverse of the residual covariance matrix S. If S is calculated in a brute force manner, it takes $O(N^2)$ time per landmark, which is $O(N^3)$ for all landmarks. The calculation of S then becomes the dominant part of the update step in terms of time complexity.

$$S_{k+1|k} = H_{k+1} P_{k+1|k} H_{k+1}^T + R_{k+1}$$

$$= \begin{bmatrix} H_R & 0 & \cdots & 0 & H_{L_i} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} P_{RR} & P_{RL_1} & \cdots & P_{RL_i} & \cdots & P_{RL_N} \\ P_{L_1R} & P_{L_1L_1} & \cdots & P_{L_1L_i} & \cdots & P_{L_1L_N} \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ P_{L_iR} & P_{L_iL_1} & \cdots & P_{L_iL_i} & \cdots & P_{L_iL_N} \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ P_{L_NR} & P_{L_NL_1} & \cdots & P_{L_NL_i} & \cdots & P_{L_NL_N} \end{bmatrix} \begin{bmatrix} H_R^T \\ 0 \\ \vdots \\ 0 \\ H_{L_i^T} \\ 0 \\ \vdots \\ 0 \end{bmatrix} + R_{k+1}$$

$$\Rightarrow O(N^2)$$

If block operations are applied, however, the calculation takes only $O(1)$ time per landmark, and $O(N)$ if repeated for all landmarks:

$$S_{k+1|k} = H_R P_{RR} H_R^T + H_R P_{RL_i} H_{L_i}^T + H_{L_i} P_{L_iR} H_R^T + H_{L_i} P_{L_iL_i} H_{L_i}^T + R \Rightarrow O(1)$$

While calculating Mahalanobis distances, the Kalman Filter saves the index of the landmark which has the smallest Mahalanobis distance with z_j and saves the corresponding distance d_{zj}, S, and measurement jacobians H_R and H_{Li}, in order to avoid redundant calculations later on. The result of the Mahalanobis Distance test on z_j, can be inferred as follows:

1. If d_{zj} is larger than the upper bound threshold $Gamma_{max}$, the Kalman filter treats z_j as a new landmark and re-sizes the state vector and covariance matrix. Then, N is increased by 1, z_j is put in the new state vector, and the new covariance matrix is calculated. Calculating the new covariance matrix takes $O(N)$ time. In this case, the previous equation for updating the covariance matrix and state vector cannot be applied since the filter has no information on the new landmark and assumes its variance goes to infinity. The new update equation of covariance matrix becomes:

$$P_{k+1|k+1} = \begin{bmatrix} P_{RR} & \cdots & P_{RL_N} & -P_{RR}H_R^T H_{L_{N+1}} \\ \vdots & \ddots & \vdots & \vdots \\ P_{L_N R} & \cdots & P_{L_N L_N} & -P_{L_N R}H_R^T H_{L_{N+1}} \\ -H_{L_{N+1}}^T H_R P_{RR} & \cdots & -H_{L_{N+1}}^T H_R P_{RL_N} & H_{L_{N+1}}^T (H_R P_{RR} H_R^T + R) H_{L_{N+1}} \end{bmatrix} \Rightarrow O(N)$$

2. If d_{zj} is smaller than the lower bound threshold $Gamma_{min}$, the Kalman Filter treats z_j as a re-detected landmark. S, H_R, and H_{Li} can be reused to calculate the updated state vector and covariance matrix since they were already calculated in the Mahalanobis distance test. Calculating the Kalman gain takes $O(N)$ time using block operations and updating the state vector takes $O(N)$ time as well.

$$K_{k+1|k} = P_{k+1|k} H_{k+1}^T S_{k+1|k}^{-1} = \begin{bmatrix} P_{RR}H_R^T + P_{RL_i}H_{L_i}^T \\ P_{L_1 R}H_R^T + P_{L_1 L_i}H_{L_i}^T \\ \vdots \\ P_{L_i R}H_R^T + P_{L_i L_i}H_{L_i}^T \\ \vdots \\ P_{L_N R}H_R^T + P_{L_N L_i}H_{L_i}^T \end{bmatrix} S_{k+1|k}^{-1} \Rightarrow O(N)$$

However, updating the covariance matrix in this case takes $O(N^2)$ time, which makes it the dominant term.

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} S_{k+1|k} K_{k+1|k}^T \Rightarrow O(N^2)$$

3. If d_{zj} is between $gamma_{min}$ and $gamma_{max}$, we do not know know if z_j is a known or new landmark. So, z_j is discarded, and the measurement is not used to update the state vector or covariance matrix.

In the worst case, each of the M measurements passed into the update function is a known landmark. This results in a total time complexity of $O(M * N)$ for the Mahalanobis distance test plus a time complexity of $O(M * N^2)$ for the update step, which is a total complexity of $O(M * N + M * N^2) \Rightarrow O(M * N^2)$ in a given time step $k+1$. Without block operations, the total time complexity becomes $O(M * N^3)$. Therefore, by utilizing block operations, our Kalman Filter gets a significant speedup over the brute-force method.

4.2 Feature Extraction

In order to find features to use during the update stage of the Kalman Filter, we extract the relative position from the Pioneer to corners in the environment with the following process:

1. Transform distance and bearing measurements from SICK laser scan data into relative position measurements
2. Perform a Hough transform[1] with the relative position measurements to find lines describing walls around the Pioneer
3. Use the lines extracted from the Hough transform and the relative position measurements to find line segments in the scan data
4. Find intersections of line segments close to one another to extract corners from the scan measurements
5. Use the corners extracted from the scan data as relative feature measurements for the Kalman Filter update function

Figure 2 shows the results obtained from a typical feature extraction for a single laser scan. Note that not all distinct line segments are detected and that some corners are missed during feature extraction for single scans. Given a few scans, though, the feature extraction algorithm will eventually find all the corners in an area, which can then be used to update the Pioneer's position and orientation estimate. The entire process is accurate enough to extract features from corners as small as door-frames in a hallway, which are generally the smallest corners encountered in our environment.

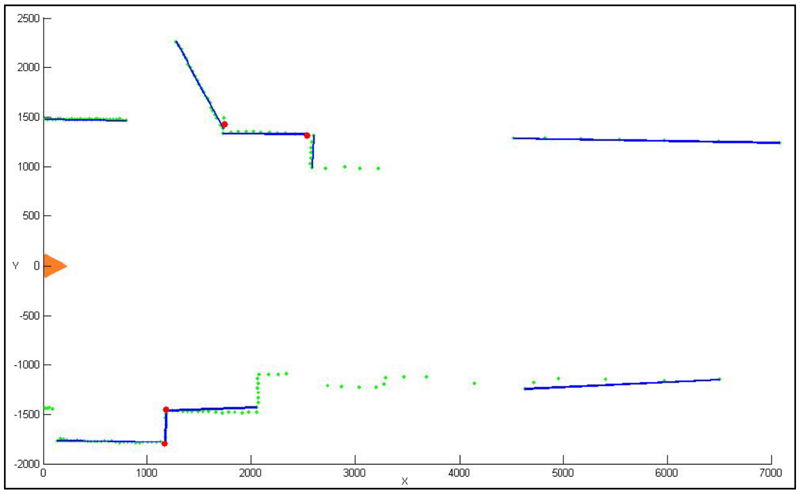

Figure 2: A typical feature-extraction result for a single laser scan. Scan points are shown in green, line segments are shown in blue, and extracted features are shown in red. The Pioneer's position is marked by the orange triangle.

Additionally, although noise in the laser scan measurements may cause the extracted corners to appear in slightly different locations between scans, the Mahalanobis distance test takes this uncertainty into account. This allows us to match new measurements with previously detected features despite the fact that they may not line up perfectly.

4.3 Structural Compass

By using the lines obtained from the Hough Transform during the feature extraction process, we are also able to synthesize a structural compass measurement that gives us the Pioneer's orientation relative to the cardinal directions of the building we are navigating through. This structural compass measurement can be used in a Kalman Filter update function to improve the estimate of the Pioneer's orientation. To find the structural compass measurement, we performed the following steps:

1. Use lines extracted from the Hough Transform to find groups of perpendicular walls around the Pioneer
2. Find the group of perpendicular walls with the longest combined line length, and assume these are the main walls of the hallway
3. Calculate the Pioneer's orientation relative to the main wall group

4. Based on the Pioneer's current heading, determine if the main wall group is oriented at 0°, 90°, 180°, or 270° relative to the Pioneer's starting orientation

5. Use the resulting structural compass measurement to update the Pioneer's orientation estimate

Since this structural compass measurement only relies on the lines extracted from the laser scans, we are able to synthesize it even when there are no corners around the Pioneer. This means that we can continue to update the Pioneer's orientation estimate even in areas with few features to extract. This helps keep the orientation error from growing unreasonably in long stretches of hallway with no corners or door-frames to update the position estimate on.

Figure 3 shows an example of how structural compass updates can improve the propagated path estimate from the Kalman Filter. The left half of the image shows the propagated path estimate with an artificial offset added to the rotational velocity measurement of the Pioneer, resulting in a "curved" path and hallway even though the Pioneer was driving in a straight line. The right half of the image shows the same path estimate with structural compass updates enabled. Note that the measurements straighten out the path estimate significantly, despite the artificial noise added to the rotational velocity measurements. Some of the noise is still present in the second image, however the resulting estimate is much better than the purely propagated one.

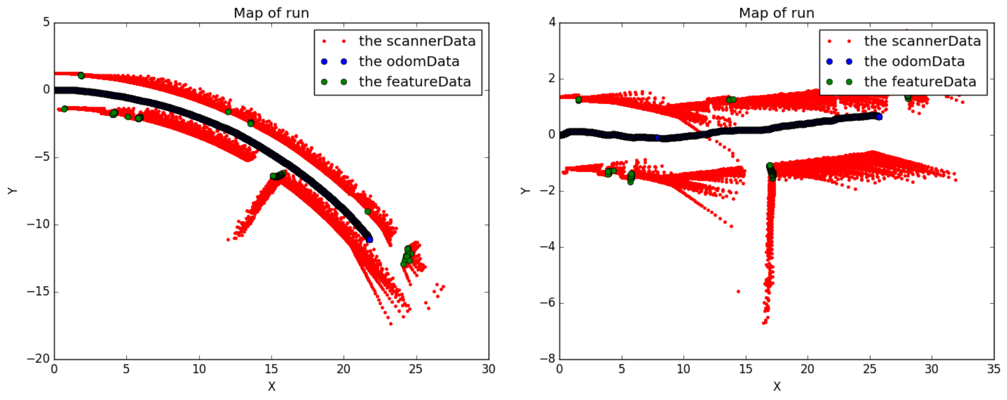

Figure 3: A comparison of propagated path estimates without (left) and with (right) structural compass updates. An artificial offset was added to the Pioneer's rotational velocity measurements as it drove down a straight hallway in order to illustrate how the compass can improve the map.

4.4 Motion Controller

While the environment is unknown in the sense that there is no prior map given, it is highly structured. This allows for a specific motion controller to be designed that matches the test area very well. For our application, the robot attempts to turn left whenever possible, otherwise aligning itself to the left wall, the right wall, or both and driving in a straight line. From the beginning, the motion controller was designed to operate separately from the rest of SLAM since run-times were unknown and it would have been risky to wait extra time for part of the SLAM loop to finish before the robot could check if it needed to stop or turn. Due to this design decision, the motion controller does not have access to the lines extracted from the Hough transform because it runs in a separate thread. Therefore, a different method was required for detecting walls around the Pioneer.

In order to determine if there are walls that can be used to align to on either side of the robot, three relative positions are determined on each side of the laser scan (Figure 4). From these points, three slopes are calculated (Figure 4). If the standard deviation between these three slopes (M1, M2, and M3) on each side is small, it is assumed that they are a line and the robot aligns to them.

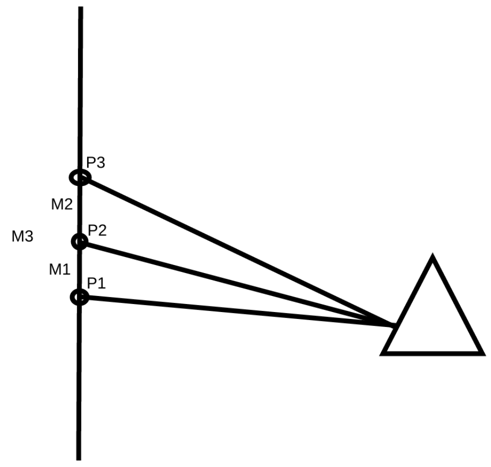

Figure 4: Wall Finding illustration

If the robot can detect lines on both sides, it will attempt to center itself by adjusting its rotational velocity until it is adequately centered in the hallway (Figure 5) before aligning itself. If only one line is available, the robot will attempt to maintain a set distance away from the known wall again by adjusting its rotational velocity until it is in a safe zone, before fixing its alignment to travel parallel to the known wall.

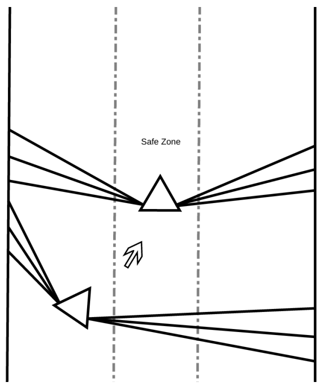

Figure 5: Alignment illustration

Turning is handled by averaging the distance between 0° and 40° (Figure 6). If the average distance is higher than a threshold, a turn is performed. To do this, the robot slows its forward velocity and sets a rotational velocity until a new wall on the left or right is detected, at which point alignment takes over again.

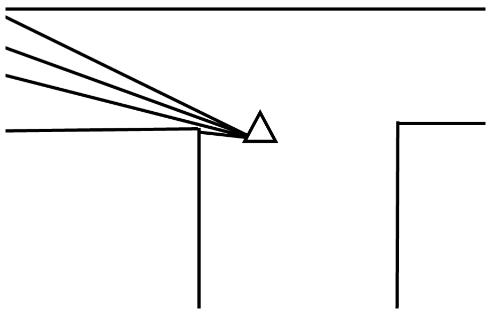

Figure 6: Turning illustration

In order to avoid possible damage to the robot if any portion of the motion controller fails, a forward check was implemented. This was accomplished by averaging distances from 90° to 100° and from 100° to 110° in the laser scan. If the average of either side is lower than a given threshold, the robot stops in order to avoid causing possible damage to itself.

These three components: stopping, turning, and aligning are run in the same continuous loop. Turning has the highest priority followed by the stopping mechanism, with aligning being given the lowest priority since waiting to check if we can align is less important than making sure we turn or stop.

5 Results

Our implementation performed very well in the environment shown in Figure 7 when both feature and structural compass updates were enabled (Figure 8). If, however, only propagate is used, the position and orientation estimates tends to diverge rather quickly (Figure 9). As seen in Figure 8, the robot does not close the loop right away when it returns to its starting point in the bottom hallway because there are few features around that point. Once we make another turn into the hallway on the right side of the image, however, the wall errors shrink down as we re-detect features we have already seen instead of marking them as new features.

Despite taking some time to close the loop, compared to the true map of the test area (Figure 7) our implementation does a good job of accurately performing SLAM.

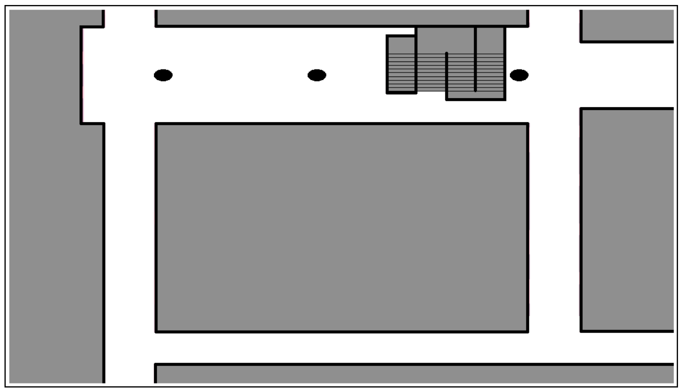

Figure 7: True map of test environment. For Figures 8 and 9, the Pioneer started in the bottom hallway and performed one and a half counter-clockwise loops around the map.

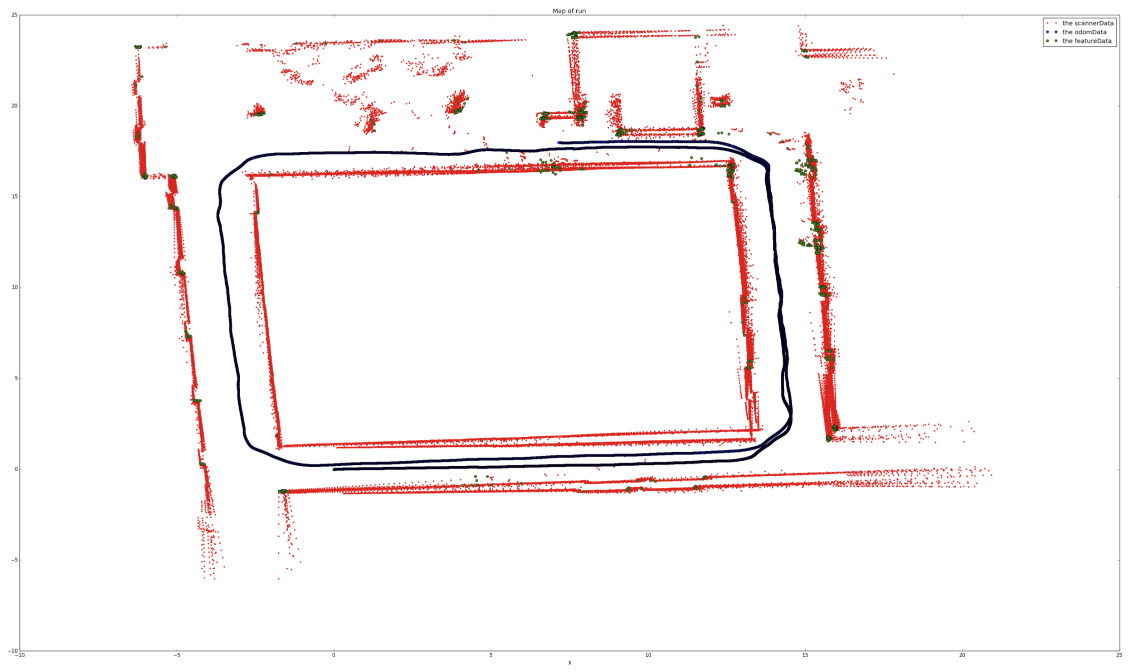

Figure 8: SLAM with updates. The Pioneer's path is shown in black, the laser scan points are shown in red, and all detected features are shown in green. Most of the scan noise in open areas was caused by people, chairs, and tables scattered around the environment.

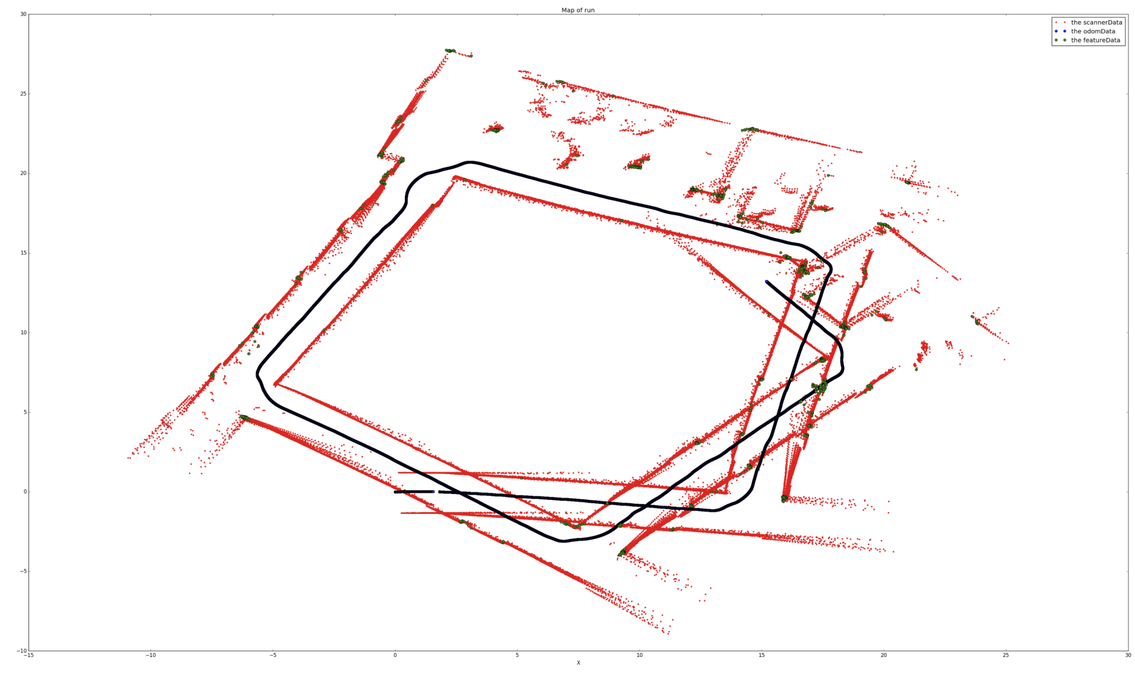

Figure 9: Propagation only. The Pioneer's path is shown in black, the laser scan points are shown in red, and all detected features are shown in green. Most of the scan noise in open areas was caused by people, chairs, and tables scattered around the environment.

6 Conclusion and Further Work

The results of this project turned out as well as we had hoped, however there is still more work that can be done. Ideally, if more time was available, we would refine the Mahalanobis distance thresholds to allow for a faster and more accurate loop-closure. The feature detection algorithm could also be refined to detect features with less positional error in order to make loop-closure more likely. We would also like to rebuild the motion controller to fit more dynamic environments, including those that are entirely different from the environment that it was built for. Being able to fully explore an area rather than following a specific control loop would make the solution more useful in later projects.

References

[1] R. Duda and P. Hart. Use of the hough transformation to detect lines and curves in pictures. *Comm. ACM*, 15:11–15, Jan 1972.

[2] H. Durrant-Whyte. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE Robotics Automation Magazine*, 15:99–110, June 2006.

[3] H. Durrant-Whyte, D. Rye, and E. Nebot. *Robotics Research: The Seventh International Symposium*, chapter Localization of Autonomous Guided Vehicles, pages 613–625. Springer London, London, 1996.

[4] B. Siciliano and K. Oussama. *Springer Handbook of Robotics.* Springer, 2008.

[5] D. Simon. *Optimal State Estimation.* John Wiley Sons, 2006.