Simultaneous Localization and Mapping Using Extended Kalman Filter

Jon Lawson
laws0079@umn.edu

Janani Ravichandran
ravic025@umn.edu

Bryan Stadick
stadi012@umn.edu

Shneka Muthu Kumara Swamy
muthu013@umn.edu

***Abstract*—For a robot, the ability to identify and react to unknown environments is an essential prerequisite to attaining autonomous robot operation. Simultaneous Localization and Mapping(SLAM) represents the ability of a robot to build a map and simultaneously to locate itself with respect to elements in the constructed map. This process utilizes external features or landmarks observed to develop a map, while evaluation of the robot motion establishes the basis for estimating the robot position, followed by updates to the position of the robot and the observed landmarks to bring the map closer to reality.**

I. Introduction

Autonomous navigation of robots in an unknown environment has numerous applications ranging from guidance, mapping, to search and rescue and more. The common factor beneath all such applications is the ability of the robot to be able to view and navigate in unknown surroundings and use the data gathered to localize its position which is the basic essence of Simultaneous Localization and Mapping(SLAM).

The history of SLAM dates back to the mid-eighties. It was originally developed by Hugh Durrant-Whyte and John J. Leonard [2], employing a statistical basis to describing features or landmarks and their geometric relationships; and was built on the earlier seminal work of Smith, Self and Cheeseman. [6] The majorly successful approach to simultaneously tackling the position estimation and environment mapping was probabilistic, and based off of Baye's theorem in one form or another. A very popular approach to solving this high uncertainty robotics problem is the use of feature-based estimation utilizing an Extended Kalman Filter(EKF), later followed by Particle Filter as other viable alternatives, depending on the type and requirements of the application. Though earlier applications of SLAM was very much viewed as pertaining to only static environments, developments in utilizing SLAM in dynamic moving environments was achieved by combining it with the problem of Detection and Tracking of Moving Objects, called targets, were challenging the limits of applying SLAM to dynamic outdoor environments. [7]

II. Project Objective

The goal of this project was to understand and gain hands-on experience implementing simultaneous localization and mapping in a robot that can navigate autonomously in an indoors environment. The robot had to be programmed to navigate autonomously in a corridor, traveling counter-clockwise over a closed loop, avoiding collision with obstacles, if any, on its way. Traversing this fixed path, the robot is expected to detect and identify unique features and produce a usable map of the surroundings for future use. The map would be a collection of data gathered on the presence and relative locations of wall and other significant, but fixed landmarks that are identifiable and form an integral part of the environment being explored by the robot.

III. Literature Review

A basic level of understanding of the SLAM problem [5] was essential to gain perspective on how to approach the problem and the implications of the Extended Kalman Filter. It is known how computationally inconvenient a direct-hand implementation of the EKF could be, because of the huge covariance matrix that maintains $O(N^2)$ elements, data for each and every landmark, that needs to be updated every time any feature is re-observed. Use of particle filters and tree structures, that utilize the conditional independence among the consecutive poses of the robot, reduce the processing required. [3] Improvements with respect to computational cost in the EKF propagation step, by using an efficient way to propagate covariance matrix using Block Thomas approach, were also studied. [4]

IV. Robot and Software

The robot used here to implement and test our SLAM algorithm is a third generation Pioneer, P3DX-SH, with a SICK Laser Scanner acting as the exteroceptive sensor. The Pioneer P3DX-SH employs differential drive, allowing independent control of the individual wheel velocities. An odometer with a 100 ticks per revolution accuracy serves as our primary senor to estimate position according to the principles of Dead-reckoning, and using the secondary SICK scanner measurements to combat the accumulating errors in odometry. With the choice of either a SICK scanner or KINECT or a camera, the SICK scanner was chosen, keeping in mind the simplicity and computational complexity and requirements. SICK LMS200 measures the distance to objects based on the reflections of the laser directed towards them. The SICK used here offers an angular resolution of 0.5 - 1 degrees, distance accuracy of +/-15mm within its distance range of 1-8 meters and its symmetric field of view of 180 degrees in the front. The other sensors available are the 7 SONAR, extensively used

for a protective obstacle avoidance mechanism in the front and sides.

The use of the Adept ARIA library for the Pioneer robot provided a faster and simpler communication channel with the robot. The software coding was done in C++ and utilized the pre-compiled headers that were available as part of the Aria Library, for various parts of robot navigation and control and feature detection. The C++ Eigen library was used extensively for matrix calculations. The Eigen library provides a number of matrix and vector structures and operations including the ability to multiply, invert, transpose, and re-size. The Adept MobileSim simulator was used in simulating and understanding the behavior and controls of the Pioneer robot. Initial exploration of the predefined classes and functions in the Aria libraries employed MobileSim extensively.

V. Approach and Solution

The easiest way to approach the project problem was to break it down into its four main components and handle them independently:

- Navigation and Control
- Feature detection and identification
- Map generation
- Position Localization

The control problem for the robot was straight forward: the robot had to travel counter-clockwise, taking consecutive left turns to complete the loop, in addition to the obstacle avoidance constraint. Our approach to implement this kind of a control was to utilize the LIDAR to detect and avoid collision of the robot with hindrances or even the walls, and make a simple algorithm that would steer a 90 degree turn of the robot as and when an opening is encountered to the left.

The feature extraction is a vital part of the SLAM algorithm, since its characteristics and quantity determines the effectiveness of the position estimation of the. Properties of the features, of interest to us, are the uniqueness and its ability to be identified of the features, when re-identified. Since the features are to be detected by laser scanner as distance and bearing data, feature identification is solely dependent on the global position of the landmark as estimated by the robot. Map, here refers to aggregation of useful data gathered by the laser scanner, in the form of lines that define any obstacle it encounters. The map generated by the indoor navigation of the robot generates a map that is a congregation of lines that represent mainly the walls and other obstacles found, if any, within its range.

Since, the robot estimates its position by continuously assessing its motion with respect to previous pose, as done with odometry based position estimation, the error tends to accumulate with time, increasing the uncertainty surrounding the evaluated robot location. This requires employing another secondary form of external position computation logic to rid the integrated error,and in this case it is the SICK laser scanner. EKF is used as the primary tool to update the position estimation from odometry with the necessary correction.

VI. EKF and Landmark Implementation

A. Line Extraction

The Sick LMS200 laser scanner provides distance and bearing measurements with a 180 degree viewing angle in front of the robot. The laser scanner provides 181 distance and bearing measurements in 1 degree increments every 13.3ms. From the raw distance and bearing measurements, lines are extracted using the ArLineFinder class included in the Aria libraries, which uses the Hough Transform to extract lines from the raw distance and bearing measurements. The following parameters are set when extracting lines:

- The minimum line length is set to 40 mm.
- The minimum number of points per line is set to 4 points.
- The maximum distance any point could be from the line is set to 50mm.
- The average distance from the line for all points is set to 20mm.
- Lines with an average distance greater than 8m away from the robot are ignored.
- If two lines are less than 100mm apart and the angle difference between the two lines is less than 10 degrees the lines are combined into a single line.

Once the lines are extracted they are transformed to the global coordinate system and written to the .map file every five seconds for offline viewing.

B. Landmark Detection

Landmarks are detected by checking every possible line endpoint combination to determine if the two end points are within 200mm of each other. Once a corner is found then the relative line angle is checked. If the value of the angle between the two lines that share an intersecting endpoint is with in a +/-20 degrees tolerance of 90 degrees, then a valid landmark has been found. An example is given in Figure 1.

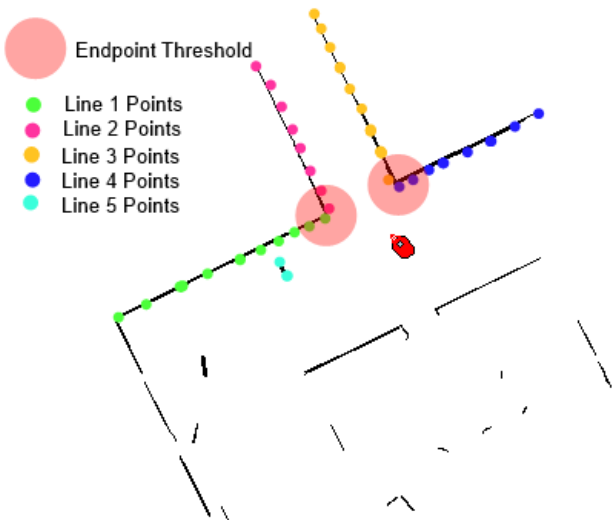

Fig. 1. Landmark detection.

The next step is to determine if the landmark is a convex or concave corner. This is accomplished by taking the average distance from each line to the robot and comparing that to the distance from the line intersection to the robot. If the distance from the intersection to the robot is less than the average distance to the lines, then the corner is labeled as convex. If not, the corner is labeled as concave. Once all possible landmarks have been found from a single scan, the landmarks are sent to update function, which computes the Mahalanobis distance to determine if the landmarks are new, previously found, or indeterminate.

C. Extended Kalman Filter

The Kalman Filter is used to find estimate of a variable using series of measurements made over a period of time. Kalman Filter is usually used when the equations are linear. For non-linear equations the approach can be used after linearizing the error function about the estimate of the mean and covariance. This nonlinear version is called the Extended Kalman Filter and is used in the project.

The most important part of SLAM is to update the robots position using the external environment and this can be done using odometry of the robot, however the readings are not very accurate due to accumulation errors and hence the laser scanner is also used. The distance and bearing measurements from the laser is used in update and the odometer is used in propagate.

D. EKF Propagate

The linear and the rotational velocity retrieved from the odometer can be used to get the approximate position of the robot at a given instance of time. This is done by using the following equations.

$$x_{k+1} = x_k + v_m dt \sin(\phi) \tag{1}$$
$$y_{k+1} = t_k + v_m dt \cos(\phi) \tag{2}$$
$$\phi_{k+1} = \phi_k + \omega_m dt \tag{3}$$

Where the measurement from the previous iterations are in k and the new calculated values are in $k+1$. The measured values v_m and ω_m represents the translational and rotational velocity measured from the odometry. The propagate step is called whenever the odometer measurements are available at an interval of dt.

The covariance matrix is updated whenever the function is called.

$$P_{K+1|k} = \phi_k P_{K+1|k} \phi_k^T + G_k Q_k G_k^T \tag{4}$$

where,

$$\phi_k = \bigtriangledown_{x_k} f(\hat{x}_{k|k}, uk, 0) \tag{5}$$
$$G_k = \bigtriangledown_{w_k} f(\hat{x}_{k|k}, uk, 0) \tag{6}$$

It was verified from the error ellipse graph the error of localization keeps increasing with the increase in the number of steps. Which is the probability of the robot misconceiving its actual position increases as we move away from the origin and which makes the use of update function mandatory.

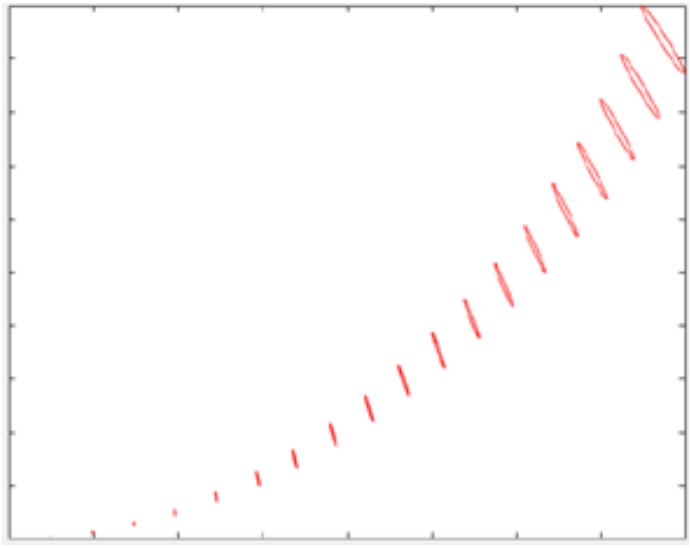

Fig. 2. Uncertainty ellipses with propagate only.

E. Update

The distance and the bearing measurement retrieved from the laser scanner is used to improve the localization accuracy of the robot.

Update is done whenever the robot finds a corner of the wall, which is considered as a landmark. The landmark is either observed for the first time or is re-observed. If the landmark is observed for the first time it is added as a state to the state space matrix. If it is re observed the state is just updated. This distinction is made by finding out the Mahalonobis Distance. If the Mahalonobis distance found is lesser than a certain threshold then Update is done and when it exceeds a certain value then Initialization is done. If the Mahalanobis distance falls between the two thresholds, the landmark is assumed to be a indeterminate and is not used.

The distance and bearing measurements are converted to position measurement.

$$x = d\cos(\alpha) \tag{7}$$
$$y = d\sin(\alpha) \tag{8}$$

This position of the landmark is compared with the estimated value. The position conversion mentioned above is in robot frame of reference and must be converted to global frame of reference. This is done using Rotation matrix. The covariance and Kalman gain are also calculated using the following equations.

$$r_{K+1|k} = z_{k+1} - \hat{z}_{K+1|k} \quad (9)$$

$$H_{k+1} = \bigtriangledown_{x_{k+1}} h(\hat{x}_{K+1|k}) \quad (10)$$

$$S_{K+1|k} = H_{k+1} P_{K+1|k} H_{k+1}^T + R_{k+1} \quad (11)$$

$$R_{k+1} = G_k R G_k^T \quad (12)$$

$$K_{k+1|k} = P_{K+1|K} H_{k+1|k}^T Sk + 1|k^{-1} \quad (13)$$

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1|k} r_{k+1|k} \quad (14)$$

$$P_{k+1|k+1} = [I - K_{k+1|k} H_{k+1}] P_{k+1|k} [I - K_{k+1|k} H_{k+1}]^T + K_{k+1|k} Rk + 1|k K k + 1|k^T \quad (15)$$

The number of landmarks found in the actual project is less than 10 and hence the computational complexity of the update process need not be taken into consideration. The information from the above sections, namely line detection and motion control is used in localization of the robot. And this localization in turn helps in motion control of the robot.

VII. Control

The path plan for the robot consists of moving forward while following the wall to the left of the robot at a fixed distance. This plan is broken down into three parts: wall detection, correction, and left turns. Each part will be discussed below separately.

A. Wall Detection

Correction involves tuning the rotational velocity to keep the robot a fixed distance from the left wall while maintaining a fixed forward translation velocity. If the robot drifts too far away from the wall the rotational velocity becomes negative to turn it back toward the wall, while the converse holds as well. When the robot is at the correct distance from the wall the rotational velocity is zero. The angular velocity is controlled using a proportional-derivative (PD) control loop [1].

$$\omega = -K_p e + K_d \Delta e + K_{prot}(\theta_{min} + \pi/2) \quad (16)$$

- ω Angular velocity of the robot
- $K_p = 0.5$ Distance proportional constant
- $K_d = 0.01$ Distance differential constant
- $K_{prot} = 0.5$ Angle proportional constant
- $e = d_{min} - d_{desired}$ Error in distance from wall
- Δe Discrete derivative of e
- θ_{min} Angle to minimum distance

The first two terms are a standard PD equation for correcting an error in the distance. The third term is a P equation for correcting the error in the angle of the robot with the wall. When the robot is parallel with the wall and at the fixed distance, all three terms become zero and the angular velocity is zero. As the errors increase the angular velocity increases and the robot ends up turning faster to correct faster. The values given for the constants were found from experimentation with the robot. They are tuned conservatively so that the robot corrects itself, but slower and over a longer distance. While these values do not give the fastest response, it gives a stable response and ensures the robot will not careen out of control.

The forward translational velocity is set at a fixed value of 0.25 of the max speed of the robot unless there is an obstacle directly in front of the robot. In the case of an obstacle in front of the robot, the translational velocity is set to zero while leaving the rotational velocity to be set to the value given by the control loop. This enables the robot to turn and follow around the forward obstacle.

B. Left Turns

Left turns around concave corners can be handled by the control loop mentioned above, but left turns around convex corners are handled separately. An example of a convex corner is shown in Figure 3. Because of the conservative turning of the control loop, by the time a left turn around a convex corner would finish the robot would have already reached the other side of the opening. Thus, the loop tuning does not allow for convex left turns through an opening. Instead left side openings are detected separately and a set routine is used to execute the turn.

Left opening detection is determined by the criterion that no wall exists within a threshold distance of the robot in the left twenty degrees of the scan (from 180 to 160 degrees). If the criterion is met, the control loop is circumvented and the robot goes into a three state state machine. The first state is the robot drives forward 0.8m, then it turns 95 degrees to the left and finally moves forward another 1m before returning to following the wall using the control loop. The turning and movements are done using the relative pose estimate of the robot, while the wall following does not rely on the pose of the robot.

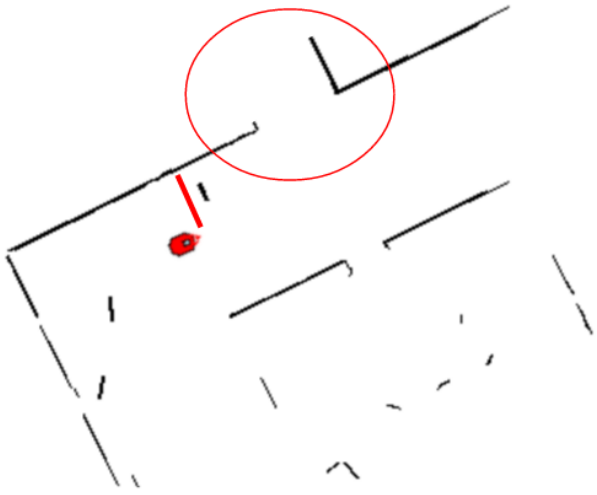

Fig. 3. Fixed wall distance and convex corner example.

VIII. Experiment

The robot was allowed to navigate at Kenneth. H. Keller Hall,University of Minnesota, Twin Cities in the hallway to follow a counter-clockwise loop to obtain a map from all features it detects through the two rounds of the loop. The inclusion of these features in their relative position, as identified by the robot, were used to construct a map of a part of the 5th floor. The second loop was completed to ensure that the robot got greater chances to encounter the recorded landmark, update its location based on the additional data and the EKF algorithm, resulting in a more accurate and certain position estimation.

IX. Results and Discussion

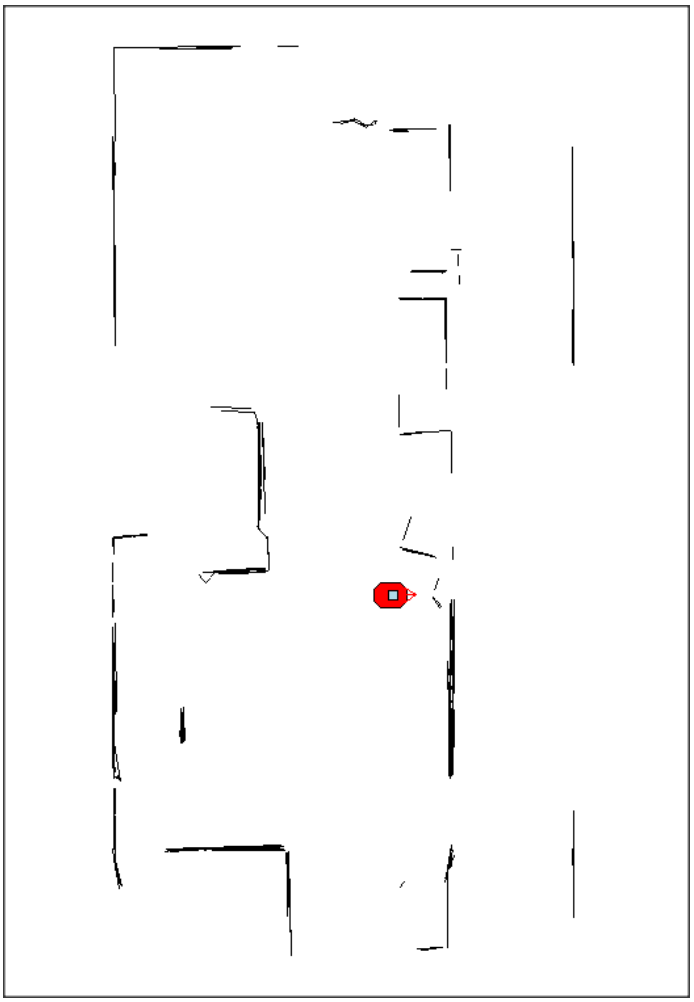

Fig. 4. Map captured after wandering in the simulator map (no drift).

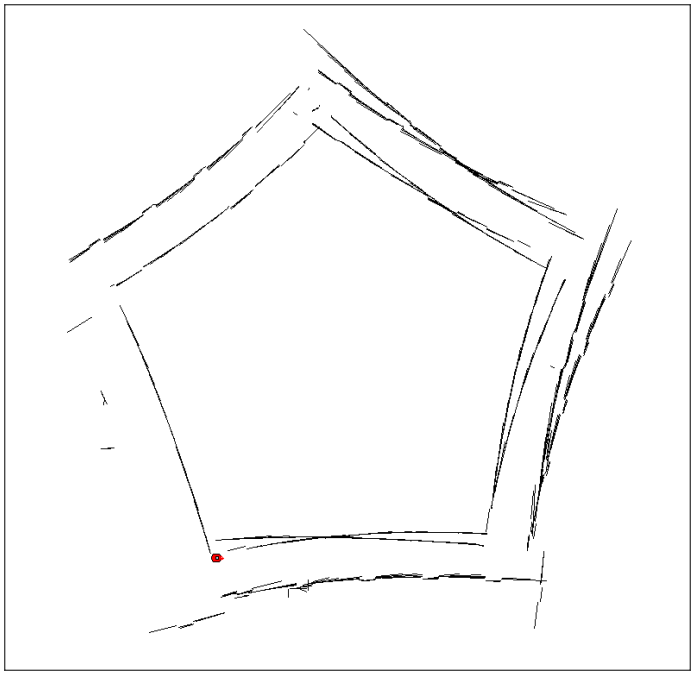

Fig. 5. Map captured with robot after two loops (drift present).

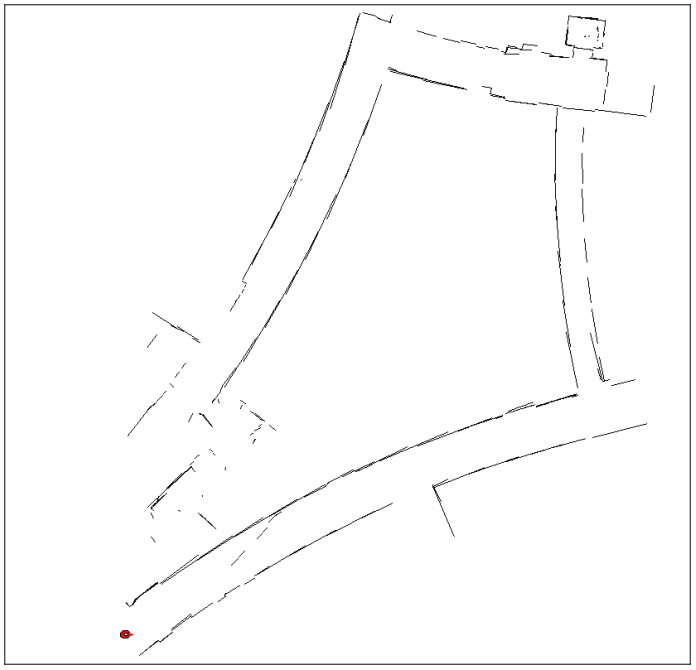

Fig. 6. Map captured with robot navigating the 5th and 1st floors of Keller (drift present).

The robot successfully navigated the hallway route following a counter-clockwise loop. The navigation control was fairly robust and could handle unexpected objects (such as groups of people in its path). During its loop, several landmarks were discovered and added to the state space vector. The hallways of Kenneth. H. Keller Hall lacked a lot of easily identifiable landmarks. Only major 90 degree convex/concave corners were identified. In the more complex columbia.map in the MobileSim application, the robot was able to identify many more landmarks shown in Figure 4. Due to the limited number of corners available as landmarks, updates to the EKF filter were separated by long periods of propagate only. The robot has a constant drift towards the right, which resulted in a skewed map. Instead of the expected rectangular hallway the

map generated looks more like a pentagon as shown in Figure 5. On the second loop the Mahalanobis distance calculations had trouble matching up landmarks due to the constant drift, which resulted in a skewing of the entire map. Figure 6 shows another example of the robot navigating Keller Hall floors 5 and 1. The persistent drift to the right can be seen in the curvature of the hallway walls.

X. Conclusion and Future Work

There are several potential areas for improvement in mapping and navigation of the robot:

1) Using corners as the only landmarks yields a very limited number of landmarks, which limits the ability for the EKF update to function optimally, since the robot travels a significant distance in between landmarks. One potential solution is to detect door frames as well, this could be accomplished by checking for line endpoints representing the beginning and ends of the door are with in a certain threshold, and the lines are parallel to each other. The corner detection method could also be modified to look for endpoints which are separated a significant distance from the next endpoint. In this situation a corner could be detected without having to "see" both sides of the corner. Both of these methods would greatly increase the number of potential landmarks available to the EKF Update function.
2) Implementing bearing compensation in the robot, would help to counteract the drift in bearing seen in the generated map generated, where the robot consistently drifts towards the right, and generates a pentagon, instead of the expected rectangle for a map. Identifying the difference in the individual wheel inflation and use it analytically obtain a drift correction factor or adding a constant bearing correction to gradually counteract the drifting and refining this correction amount by trial and error were identified as plausible solutions to keep in check the robot drift.
3) The structural compass algorithm could also be implemented, which would help improve the mapping, and prevent drift seen in the generated map.

References

[1] Ros_cpp_wallfollowing: Wall following - ros c++, "2013".

[2] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, Jun 1991.

[3] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbrei. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI National Conference on Artificial Intelligence*, 2002.

[4] Eftychios A. Pnevmatikakis, Kamiar Rahnama Rad, Jonathan Huggins, and Liam Paninski. Fast kalman filtering and forward?backward smoothing via a low-rank perturbative approach. *Journal of Computational and Graphical Statistics*, 23(2):316–339, 2014.

[5] Søren Riisgaard and Morten Rufus Blas. Slam for dummies, a tutorial approach to simultaneous localization and mapping, 2005.

[6] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. *CoRR*, abs/1304.3111, 2013.

[7] Chieh-Chih Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2918–2924, 2002.