ISTANBUL TECHNICAL UNIVERSITY
ELECTRICAL – ELECTRONICS ENGINEERING FACULTY

INDOOR LOCALIZATION, PATH PLANNING AND CONTROL OF MOBILE ROBOTS

BSc Thesis by
Abdullah Ömer Sevil
Erim Vural

Department of Control and Automation Engineering
Control and Automation Programme
Supervisor: Prof. Dr. Hakan TEMELTAŞ

June 2015

FOREWORD

This thesis was written for our Bachelor's degree in Control and Automation Engineering Department at Istanbul Technical University. The aim of this thesis is to make autonomous indoor mobile robots to travel without problem.

We would like to thank our supervisor, Professor Hakan TEMELTAŞ, for his guidance and great help during the development of this thesis.

We also want to thank a friend of ours, Emre AY, for his great support and willingness.

Finally, we appreciate and thank to our families for everything.

June 2015

Erim Vural
Abdullah Ömer Sevil

TABLE OF CONTENTS

Page

ABBREVATIONS

2D: Two dimensional
LMS 200: Laser Measurement System 200
DC: Direct current
D&A: Distance & Angle
S&M: Split & Merge

LIST OF TABLES

LIST OF FIGURES

INDOOR LOCALIZATION, PATH PLANNING AND CONTROL OF MOBILE ROBOTS

SUMMARY

For mobile robots to accomplish their tasks successfully; localization, path planning and control which are interactive stages should be implement in best way. Since a fault in one of these parts may cause to robot cannot perform its task or perform it with a mistake. Localization which is the first of these stages is the subject of measuring the robot's position by itself. Because of this reason it is the most important stage between these three stages since a mistake in localization may cause the knowledge of robot about its position will be incorrect. In this situation it is not possible for robot to accomplish its tasks like path following or reaching target position. Stage that comes after localization is the path planning which is the reaching of robot from start position to target position by avoiding obstacles. After these two stages, control is the stage that ensuring these two stages perform in best way. In control stage, distance of robot's current position from desired position is measured and an error knowledge is generated thereafter this error is minimized by using control theory.

In this thesis a study is performed which consists of indoor 2D localization techniques and a path planning technique. In this study, a laser sensor is used to scan the environment. The laser sensor scans the environment with 180^o scanning range and 1^o resolution. Therefore after the scan, data of 181 points are obtained. These data are transferred to MATLAB by using serial communication. Three different localization methods are used to find the localization information.

For the first method Split & Merge Algorithm is used to do feature extraction and with a Distance & Angle Algorithm the roto – translational matrix is found.

For the second method RANSAC Algorithm is used to do feature extraction and Genetic Algorithm is used to find the optimal roto –translational matrix which minimizes an error function.

In the final method, line equations of walls or objects those are long enough are obtained and position of robot by using distances of robot from these walls or objects is measured.

After localization studies, path planning is explained. To do path planning, a potential field algorithm is used. This method uses the principle that target position

generates attractive force and obstacles in the environment generate repulsive forces to robot to find the most appropriate path for robot. In contrast to localization methods this method is only simulated.

INDOOR LOCALIZATION, PATH PLANNING AND CONTROL OF MOBILE ROBOTS

ÖZET

Mobil robotların kendilerine verilen görevleri başarıyla tamamlayabilmeleri için birbirleriyle ilişkili aşamalar olan lokalizasyon, yön planlaması ve kontrolün en iyi şekilde uygulanması gerekmektedir. Çünkü bu aşamalardan herhangi birinde yapılacak olan hata, robotun kendisine verilen görevi hatalı olarak gerçekleştirmesine veya gerçekleştirememesine yol açacaktır. Bu aşamalardan ilki olan lokalizasyon robotun kendi konumunu bulmasıdır. Bu sebeple bahsedilen üç aşamadan en önemlisi lokalizasyondur. Çünkü lokalizasyonda gerçekleşecek bir hata robotun kendi konumunu yanlış bir şekilde bilmesine yol açacaktır. Bu durumda robotun kendisinden istenilen yön takibi veya hedef noktaya ulaşma gibi görevleri gerçekleştirebilmesi de mümkün olmayacaktır. Lokalizasyondan sonraki aşama olan yön planlaması robotun başlangıç noktasından hedef noktaya engellerden sakınarak ulaşmasıdır. Bu iki aşamadan sonraki kısım ise bu iki aşamanın olabilecek en iyi şekilde gerçekleşmesini sağlayacak olan kontroldür. Bu aşamada robotun o anda kendisinden olması istenilen noktadan ne kadar uzakta olduğu ölçülerek bir hata bilgisi oluşturulur ve bu hata kontrol teorisi uygulanarak minimize edilir.

Bu tez çalışmasında mobil robotlar için iç ortamlarda 2 boyutlu lokalizasyon teknikleri ve yön planlaması için çalışmalar yapılmıştır. Bu çalışmada bir adet lazer sensörü kullanılmıştır. Lazer sensörü 180^o lik bir çevreyi 1^o lik çözünürlükle taramaktadır. Bu sebeple tarama sonucunda 181 nokta bilgisi elde edilmektedir. Tarama sonucu elde edilen bu veriler seri bağlantı kullanılarak MATLAB programına aktarılmıştır. Bu tezde üç farklı yer saptama yöntemi anlatılmıştır.
Bu yöntemlerden birincisi, Split & Merge Algoritmasıyla öznitelik çıkarımı yapılıp elde edilen yerel haritayı kullanarak Distance & Angle Algoritmasıyla dönme ve öteleme matrisinin elde edildiği yöntemdir.
İkinci yöntemde öznitelik çıkarmak için RANSAC Algoritması kullanılmıştır. Bunun sonucunda elde edilen yerel haritalar ile Genetik Algoritma kullanarak hata fonksiyonunu en iyi şekilde minimize eden dönme ve öteleme matrisi elde edilmiştir.

Son olarak anlatılan yeni yöntemde ise ortamdaki duvarların veya duvar yerine kullanılabilecek uzun nesnelerin doğru denklemleri bulunur ve robotun bu duvarlara veya nesnelere olan uzaklığından yola çıkarak robotun konumu tesbit edilir.

Lokalizasyon çalışmalarından sonra yön planlaması yapılmıştır. Yön planlamasında potansiyel alan algoritması kullanılmıştır. Bu yöntem belirlenen hedef noktanın mobil robotu kendisine doğru çeken bir kuvvet uygulayarak ve ortamdaki engellerin ise mobil robotu iten bir kuvvet uygulayarak mobil robot için en uygun yönün planlanması prensibine bağlı olarak çalışır. Lokalizasyon yöntemlerinin aksine bu yöntem sadece simülasyon ile test edilmiştir.

1. INTRODUCTION

Autonomous mobile robots are robots that have the ability to find a path and to follow that path without any human interaction. Mobile robots are designed to have much more different abilities depending on their properties such as object recognition, manipulation, environment scan and analysis of the different properties of the environment. They are designed to work in different environments such as in hospitals, libraries, warehouses, and factories or at outsides. They can even work on the walls of buildings to do cleaning of the walls' and glasses' of the building or to do surveillance. The initial step to design an autonomous mobile robot is to make localization. Localization is the ability of the robot to find the position of itself with respect to a reference point. A mobile robot cannot go to a goal position without knowing its location, otherwise it can hit the obstacles and this can damage the robot and/or the environment. There are many methods to do mobile robot localization. In this graduation project three different method of robot localization for indoor environments are explained and examined. First two methods, which are explained in the sections 3. and 4. of this document already existed in the literature. The third method, which is explained in the section 5. is a new approach to mobile robot localization. These three localization methods have their advantages and disadvantages. Their advantages, disadvantages, and experimental results of the each localization method are given in this thesis. After localization is made path planning should be made to find the best path between a goal and a start position for the mobile robot. Potential field algorithm, which is explained in the section 6., is used as a path planning method.

2. INTRODUCTION TO LOCALIZATION

Localization is the ability of the robot to know the position of itself with respect to a reference coordinate frame. Localization is done by obtaining data from a sensor or couple of sensors. Sensors generally do not provide exact results; they have sensitivities and resolutions. This implies that the localization information may not show the exact position of the robot but the error between the exact and the calculated position of the robot can be made very small by using different kind of sensors and localization algorithms. Depending on the environment that the vehicle is intended to be used various sensors can be chosen because some sensors can produce better results in some specific environments than others. The following cases that, effects the appropriate sensor choice can be given as an example.

If the unmanned ground vehicle is intended to be used in large areas, it is better to select sensors that are capable of measuring long distances; sensors that have large ranges.

If the unmanned ground vehicle is intended to be used in confined spaces with considerable amount of obstacles, it is better to select sensors that have high resolution and produce more reliable results because the risk of collision is high.

The choice of sensor or sensors is one of the most important aspects of the design of unmanned vehicles and this choice among with the other choices can greatly increase or decrease the success of the unmanned ground vehicle.

In our project we have not participated in the design of the mobile robot that, we have used.

Odometry is one of the most basic methods of localization. It is explained in detail below.

2.1. What is Odometry?

Odometry is a method, which is used to find the position of the robot by integration. In our mobile robot we have 2 motors. So we can find the position of the robot by integrating the velocity of the vehicle and the rate at which the vehicle is turning. We can use different sensors to find the angular position/angular velocity of the motors [1].

Finding the position of the robot using the odometry generally produces very erroneous results because the errors are summed up in every movement of the robot. The errors of this method arise from the following reasons;

- Limited resolution of the measurement devices
- Unequal wheel diameter
- Variation of the contact point of the wheel
- Slipping due to unequal floor contact and variable friction

Some of the aforementioned reasons are deterministic such as unequal wheel diameter. Deterministic errors can be removed by making appropriate calibration but non-deterministic errors such as slipping can be reduced but it cannot be completely eliminated [2].

In our project we have not used odometry because of the given reasons instead we have used SICK LMS 200 laser scanner and tried to develop several localization algorithms in order to find the mobile robots position.

2.2. Mobile Robot

The mobile robot that we have used is designed to be used in indoor environments and it has differential drive system. It has 4 wheels. Two of the wheels are in each side of the robot and one of them is in the front and the other is the back of the robot. The wheels that are in each side of the robot are driven independently with two motors. The front and the back wheel are non - driven wheels and they are used to balance the robot. The motion vector of the robot is the sum of the motion vector of the side wheels. Differential drive robots can be also made with 3 wheels. The Fig. 2.1 shows the motion of 3 wheeled differential drive robot. The robot in the Fig. 2.1 has 2 side wheels that are driven with independent motors and the blue arrows represent the magnitude and the direction of the velocity of each wheel.

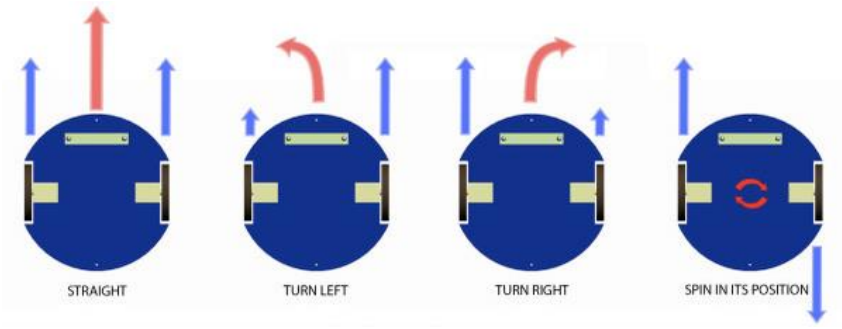

Figure 2.1 Differential drive vehicle

The mobile robot that we have used has a rectangular cuboid shaped metal frame. Maxon Epos EC45 250 W brushless DC motors with Maxon Epos 70/10 drivers are used to drive the two side wheels of the mobile robot. The mobile robot also has 2 batteries and Texas Instruments C2000 Series DSP. Two SICK LMS 200 laser scanners are used to obtain two dimensional scan of the environment.

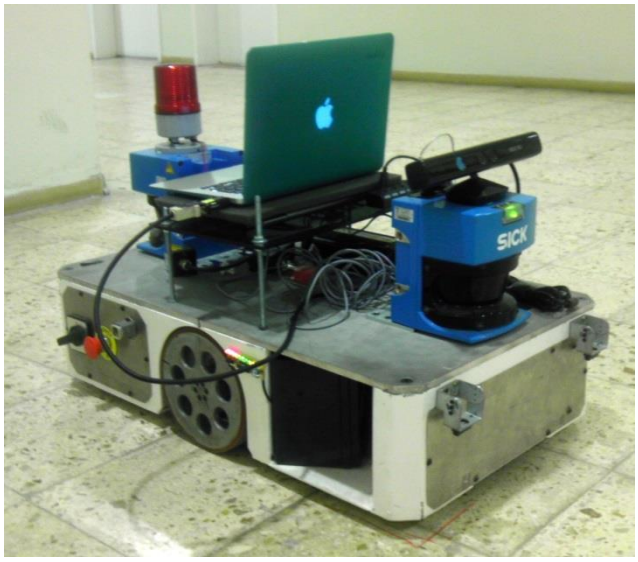

Figure 2.2 Mobile Robot

2.3. Data Acquisition

2.3.1. SICK LMS 200 Laser Scanner Properties

SICK LMS 200 laser scanner is an indoor type laser scanner; in outdoors the measurements can be very erroneous because of the sunlight. This laser scanner takes measurements by using the time of flight principle; it sends a light pulse and at the same time a counter starts to count until that the exact light pulse returns due to reflection. The distance between the laser scanner and the object, which reflected the light pulse, is found using the value of the counter. SICK LMS 200 should be supplied with 24V DC($\pm$ 15%) voltage. The laser scanner can scan the environment with 0.25°, 0.5°, or 1° angular resolutions and it scans the 180° field, that the scanner is directed by using a continuously rotated mirror. The mirror is rotated with a motor and 1 complete rotation of the mirror takes 13.32 ms which corresponds to a 75 Hz. When the angular resolution is selected as 1°, the mirror should rotate 1 time but for the angular resolutions to be 0.5° and 0.25° the mirror should rotate 2 and 4 times respectively. So when the angular resolution increases, the amount of time that's

necessary to obtain 180° scan of the environment also increases. The laser scanner has maximum scanning range of 80 meters. In our project we have set the laser to have 1° angular resolution. Operating range diagram of the SICK LMS 200 is given in the Fig. 2.3 [3].

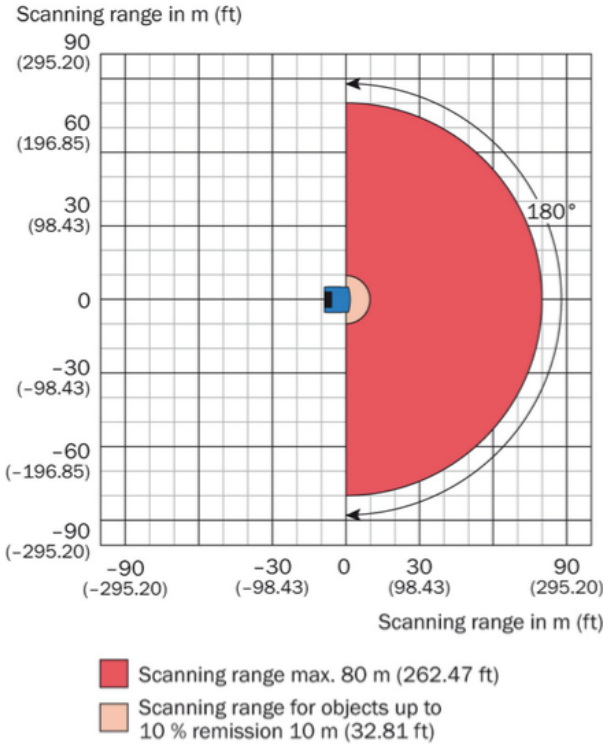

Figure 2.3 Operating range diagram of the SICK LMS 200

2.3.2. Laser Scanner Communication

It is possible to connect the laser scanner to computer using RS232 or RS422 serial communication interface. SICK has software dedicated to read measurements but in our project we have not used that software instead we used MATLAB to read measurements using RS232 serial interface. Matlab has very easy to use commands to define a serial object, to read from that object and to write to that object. To be able to do communication, the telegram structure of the laser scanner should be used. The steps to do RS232 serial communication between the laser scanner and the Matlab are given below.

1. First we have created a serial object in Matlab and opened it.
2. Then it is possible to change the properties of the serial object such as timeout, baud rate, and input buffer size… The specified baud rate for the

LMS 200 laser scanner after power up is 9,600Bd and we have kept it as it is. But we decreased the timeout value 0.01 sec because it was very long.

3. Then the laser scanner should be switched to installation mode. To do this the telegram which is given in the Table 2.1 are sent to the laser scanner [3].

Table 2.1 Installation mode telegram structure

Description	STX	Address	Length		Command	Data	Checksum	
Byte position	1	2	3	4	5	6 to 14	15	16
Hex. value	02	00	0A	00	20	00 53 49 43 4B 5F 4C 4D 53	5F	B2

4. When the LMS 200 successfully receives the command, it confirms it with the acknowledge, 06H and then sends the telegram which is given in the Table 2.2 to the computer [3].

Table 2.2 Response mode telegram structure

Description	STX	Address	Length		Response	Data		Checksum	
						Data	LMS status		
Byte position	1	2	3	4	5	6	7	8	9
Hex. value	02	80	03	00	A0	00	10	16	0A

5. After the installation is made succesfuly, the laser scanner should be switched to the monitoring mode in order to scan the environment.
6. The telegram which is given in the Table 2.3 is sent to the laser scanner to switch to monitoring mode [3].

Table 2.3 Monitoring mode telegram structure

Description	STX	Address	Length		Command	Data	Checksum	
Byte position	1	2	3	4	5	6	7	8
Hex. value	02	00	02	00	30	01	31	18

7. After the the telegram in Table 2.3 is sent, the measurement values are read from the laser scanner and they should be saved in a varible in the Matlab environment.
8. Then for each new scan, steps 6 and 7 should be repeated.
9. When the scanning of the environment is finished completely, the Matlab serial object should be closed.

Actually we have used Matlab while we were developing and testing the algorithms. In the mobile robot, Robot Operating System is used which is shortly called ROS. ROS can be installed in any computer, which has Unix based platforms. It is not possible to run the Matlab code in the ROS environment. Matlab code

should be converted to Python, C++ or Lisp code in order to run it in the ROS environment.

2.4. What is ROS and why is it used?

ROS is an open-source, meta-operating system robots. It provides the services that can be expected from any ordinary operating system, such as including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocos, CARMEN, Orca, MOOS, and Microsoft Robotics Studio.
The ROS runtime "graph" is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server.

ROS is not a real time framework, though it is possible to integrate ROS with real time code. ROS also has seamless integration with the Orocos Real-time Toolkit.
The primary goal of ROS is to support code *reuse* in robotics research and development. ROS is a distributed framework of processes (aka *Nodes*) that enables executables to be individually designed and loosely coupled at runtime. These processes can be grouped into *Packages* and *Stacks*, which can be easily shared and distributed [4].

ROS is used as a central framework for the elements such as laser scanner, micro controller, remote control etc., which are used in our project.

2.5. The Environment of the Robot

In this project the mobile robot is desired to work in the corridors of the Faculty of Electrical and Electronic Engineering building. It is not designed to work inside of the rooms or classrooms unless they have wide obstacle free areas and appropriately large doors. The mobile robot also cannot go upstairs or downstairs.

Since the mobile robot is going to be working in an indoor environment, it would be sufficient for us to denote its position and orientation in the (x, y, θ) format.

3. LOCALIZATION METHOD 1

In this method, initially Split & Merge algorithm is used to extract features for each of the two consecutive scans. Then these two feature sets are associated using Distance & Angle algorithm to find the position of the mobile robot with respect to the scan, which was used in the data association process. This process of scanning the environment and finding the position of the robot between two consecutive scans is done periodically. The final position of the robot with respect to its initial frame is found by a special summation of the positions, which are found using consecutive scans. Whole localization process is explained in detail below.

3.1. Feature Extraction Algorithm: Split & Merge

This algorithm is the combination of the Top – Down Polyline Splitting Algorithm and the Bottom – Up Merging Algorithm. To be able to use this algorithm the points, which are obtained in the data acquisition process should be ordered. To explain this more thoroughly, let's assume that the coordinates of the points are stored in an array. So if the point set is ordered the elements which are closest to the kth element is stored in the (k-1)th and/or (k+1)th element in the array.

Split algorithm is used to define the walls roughly in the scanned environment and merge algorithm is used to improve the quality of the split algorithm. Split algorithm tries to find lines, which should represent the point groups. So the number of features which are obtained in the measurement process are decreased to much smaller value.

3.1.1. Polyline Splitting

The split algorithm is a recursive algorithm.

1. At first an initial line is generated between the first and the last element of the ordered point array.
2. Then the maximum perpendicular distance between the generated line and all the points between the end points of this line is found.
3. If that distance exceeds a predefined threshold value than that point (the point which had the maximum distance to the generated line) is selected as an end

point. Let's call that point as middle point. Thus at this step 2 lines are formed using three points.

4. Then new line is formed using the first element of the initial point array and the newly found middle point as end points and Step 2 is repeated again. If there exists a point which exceeds the distance threshold value Step 4 is repeated again until no point is found that exceeds the threshold value.
5. When 2 points P1 and P2 are found, which describe a line and when there is no point in between these 2 line end points that exceed the distance threshold value, the first element of the ordered point array is updated. The point among P1 and P2 that has bigger index number becomes the new first element of the ordered point array and the whole algorithm is repeated from Step 1. The algorithm is terminated when the point among P1 and P2 that has bigger index number has an index number equal to the length of initial measurement array.

   Please note that the order of the measurement set is never changed in any of the steps.

The Fig. 3.1 shows the split algorithm.

Figure 3.1 Split Algorithm graphical explanation

The vertices that are found in the Polyline Splitting algorithm define the walls of the scanned environment roughly. So merging algorithm is used to improve it's quality such that the vertices that are found using Split and Merge algorithm best describes the scanned environment and it also decreases the number of features to a more processable value.

3.1.2. Merging Algorithm

It is possible to apply different merging algorithms to improve the quality of the split algorithm. In our project we applied the exact same algorithm in the split algorithm as merging algorithm. Instead of using the measurement results which are

obtained using the laser scanner, we used the points which are obtained after running the polyline splitting algorithm and run the split algorithm for those points.

3.2. Experimental Results of Split & Merge Algorithm

Results of the Split & Merge Algorithm for the laser scan data which is shown in the Fig. 3.2 are shown in the Fig. 3.3, and Fig. 3.4. The Fig. 3.2 shows the laser scan of the Control and Automation Department corridor and in the figure there are 181 points, in other words 181 features. Fig. 3.3, and Fig. 3.4 show the result of the Split & Merge Algorithm for the distance thresholds 20 and 10 respectively. As you can see, when the distance threshold value decreases, the local maps, which are obtained by running the Split & Merge Algorithm, become more detailed. For this specific scan of the environment, when the distance threshold value is 20, the number of features which are obtained by the algorithm decreases to 16 and when the distance threshold value is 10, the number of features which are obtained by the algorithm decreases to 26.

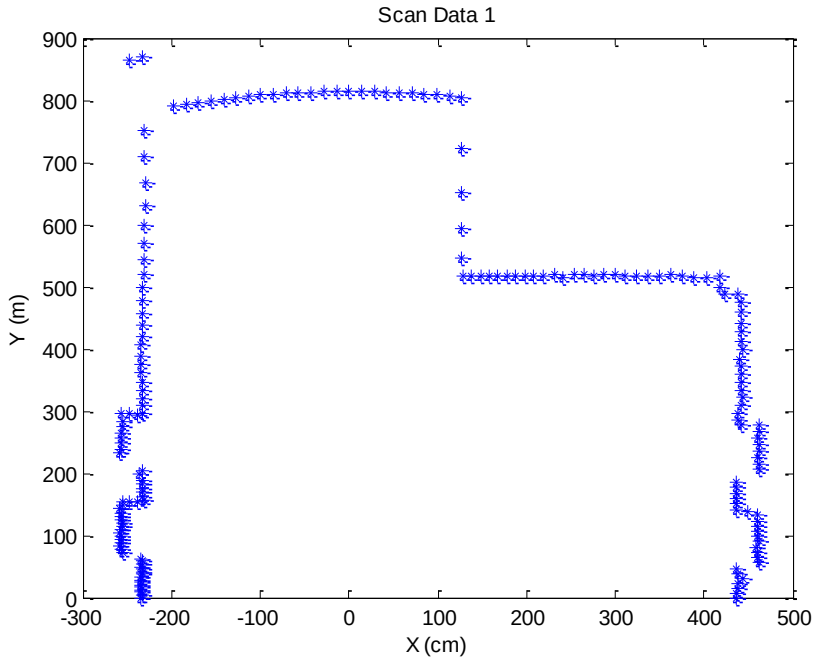

Figure 3.2 Laser scan of the corridor

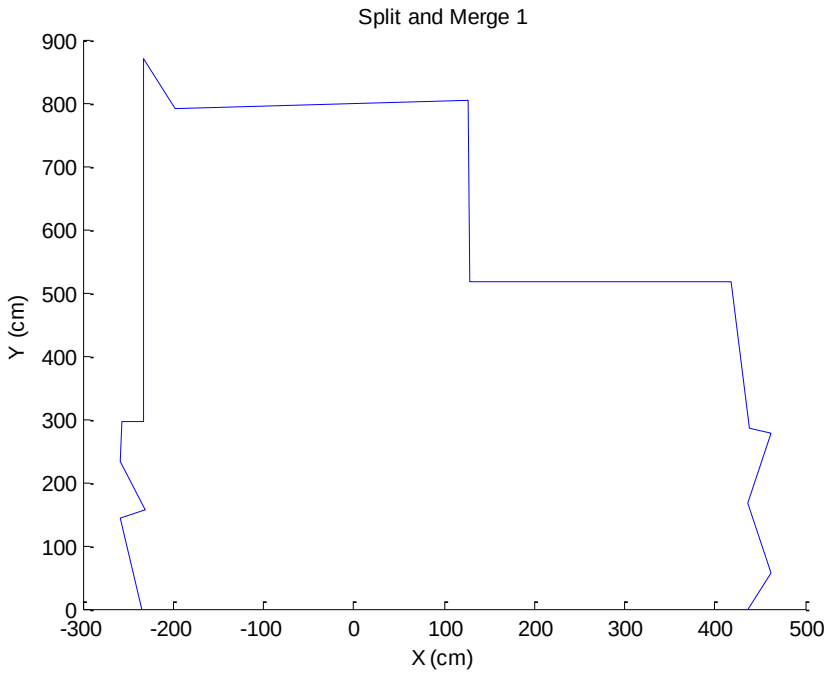

Figure 3.3 Result of the S&M Algorithm for distance threshold 20

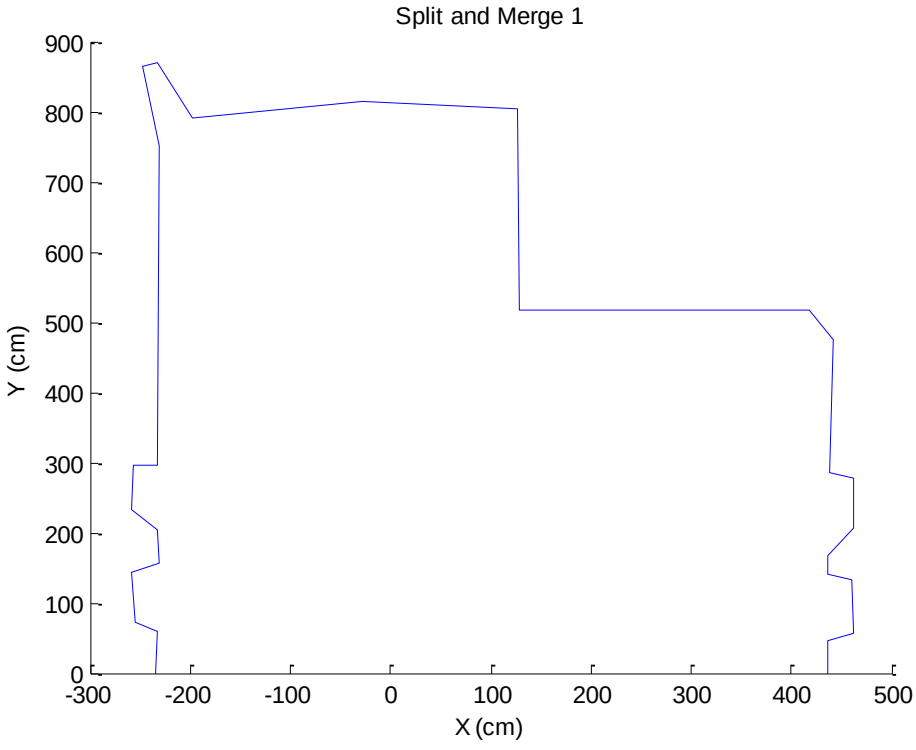

Figure 3.4 Result of the S&M Algorithm for distance threshold 10

3.3. Properties of Split & Merge Algorithm

- It is easy to apply and is very fast but when the environment has considerable amount of obstacles the results may not be reliable.

- The point set should be ordered in order to apply Split & Merge Algorithm.
- The every line segment, which are obtained after running the algorithm are connected to each other; there is no gap between the lines.
- It does not involve probabilistic processes; there is no randomness in the algorithm. The algorithm will produce the same results for the same measurements each time the algorithm is used. So in that case, the Split and Merge algorithm can be considered as a robust segmentation algorithm.

3.4. Data Association

For each laser scan, feature extraction is made using Split & Merge algorithm. Split & Merge algorithm gives the local map of the each scan. Each consecutive local map should be associated to find the position of the mobile robot. This association process is called scan matching. To be able to do scan matching, common features should be found between two consecutive scans. Scan matching can also be made with predefined global map of the environment and the currently obtained local map of the environment.

3.5. Basics of Roto – Translational Matrix

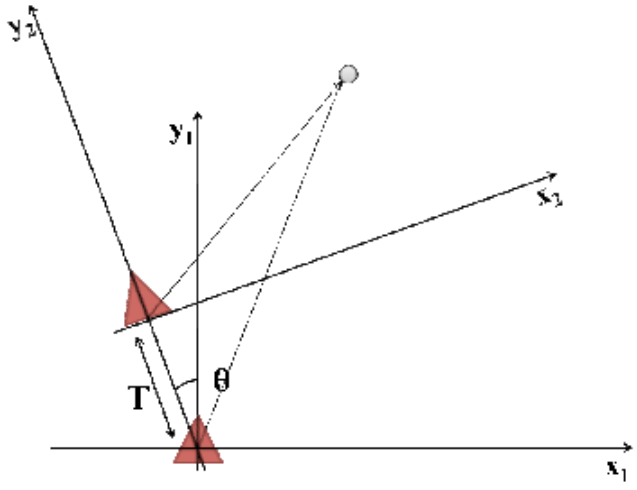

Figure 3.5 Two coordinate frames corresponding to two consecutive robot positions

The fig. 3.5 shows the two different points which are represented with triangular shapes. Let P_1 be the point, which is the origin of the x1y1 frame and P_2 be the point, which is the origin of the x2y2 frame. Let's assume that P_1 shows the initial position of the robot and P_2 shows the final position. Then T is the translation between the

two point and θ is the angle between the two frames; the orientation of the robot with respect to its initial frame. The following relation exist for points P_1, and P_2.

$$P_2 = R(\theta, T).P_1 \quad \textbf{(3.1)}$$

Here R is the roto – translational matrix, which defines the relation between P_1, and P_2. R has the form;

$$R(\theta, T)\begin{bmatrix} cos(\theta) & -sin(\theta) & T_x \\ sin(\theta) & cos(\theta) & T_y \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{(3.2)}$$

In our project Distance & Angle Algorithm is used to find the roto – translational matrix between two consecutive local maps.

3.6. Distance & Angle Algorithm

When the robot moves some new features can be extracted through scanning and some of the features, which is found before the robot has moved, can be lost. We have to find the features, which are seen in both of the scans/local maps.

The Distance and Angle (D&A for short) algorithm tries to find the roto – translational matrix. Initially D&A algorithm tries to find segments in each local map that have approximately equal segment lengths so that the absolute value of the length difference is below a predefined threshold value. The longest pair of segments, which satisfy the length threshold, are selected as a reference for local maps. Then another pair of segments is searched that satisfy the length criteria. After finding another pair of segments, the angle between one of the newly found segments and the reference segment for that local map is found. Then the angle between the matching segments for the consecutive local map is found. If the angle difference is below a predefined threshold value than the algorithm is terminated. When the algorithm is terminated total of 4 segments for 2 local maps are obtained and these 4 segments are used to find the roto – translational matrix and hence θ, T.

D&A algorithm works as follows [5]:

1. A first d_R-d_R association between two maps is found searching firstly between longer segments (purple segments in Fig. 3.6). The first association is initially considered as reference.

2. Following *d*-*d* association are searched (green segments in Fig. 3.6).
3. The angle between each segment *d* and the reference d_R for its map should be "close" to the one in the other map to let association to be considered as valid.
4. If any *d*-*d* association were not found, the reference is considered as wrong so it is changed and the algorithm starts again.

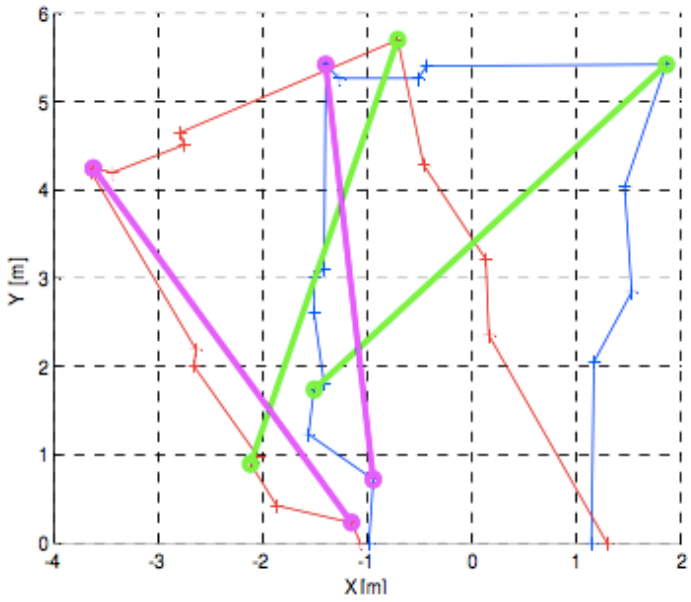

Figure 3.6 D&A Algorithm reference line selection

Purple segments represent the references while the green ones represent the second data association. The difference of length between the two green segments is less than a given threshold and, as stated in point 3, the difference of angle, between each segment and the reference one, is less then another given threshold. Only in this case the association is considered as valid.

3.7. Finding Final Position With Respect to Initial Frame

We find the change of position between two consecutive scans when we run the D&A Algorithm but our aim is to find the position of the robot with respect to initial frame.

Suppose x_0y_0 is the initial(first) frame; the point where the first laser scan is made. Suppose that the robot starts to move after the first scan and passes the point (x_1, y_1, θ_1) when the second scan is made and passes another point (x_2, y_2, θ_2) when

the third scan is made. The point $(x_1, y_{1,}, \theta_1)$ can be found by running the Split & Merge algorithm and the D&A algorithm with the first and the second scans. The point (x_2, y_2, θ_2) can be found again using the same algorithms with the second and third scans but (x_2, y_2, θ_2) would be defined with respect to the second frame. To find the final position with respect to the initial frame we applied the following formula;

$$[\alpha, \rho] = cart2pol(x_2, y_2) \quad \textbf{(3.3)}$$

$$x_2^0 = \rho cos(\alpha) \quad \textbf{(3.4)}$$

$$y_2^0 = \rho sin(\alpha) \quad \textbf{(3.5)}$$

$$T_x = x_1 + x_2^0 \,; total\ x \quad \textbf{(3.6)}$$

$$T_y = y_1 + y_2^0 \,; total\ y \quad \textbf{(3.7)}$$

$$T_\theta = \theta_1 + \theta_2 \,; total\ \theta \quad \textbf{(3.8)}$$

The cart2pol expression in Eq. 3.3 symbolizes the conversion of Cartesian coordinates to polar coordinates. The formulation given above can be applied to each result of the D&A algorithm to find the final position of the mobile robot with respect to its initial frame.

3.8. Experimental Results of Distance & Angle Algorithm

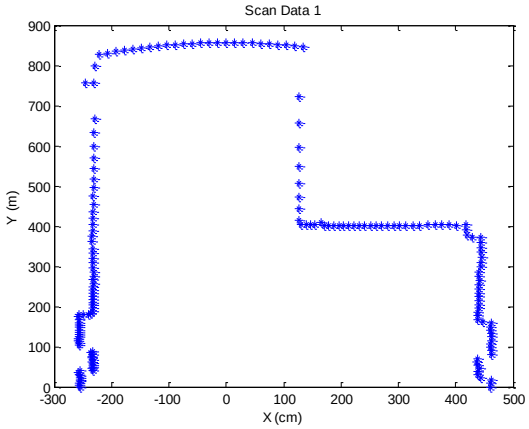

Figure 3.7 First scan data

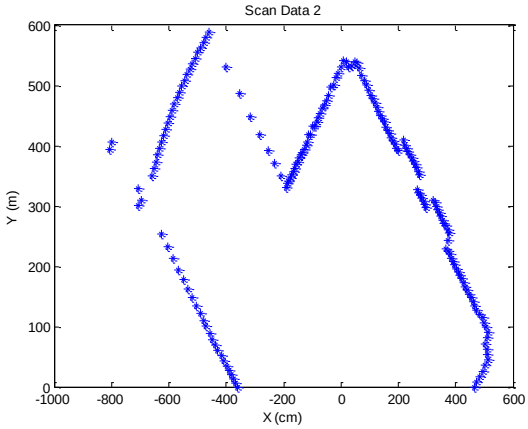

Figure 3.9 Second scan data

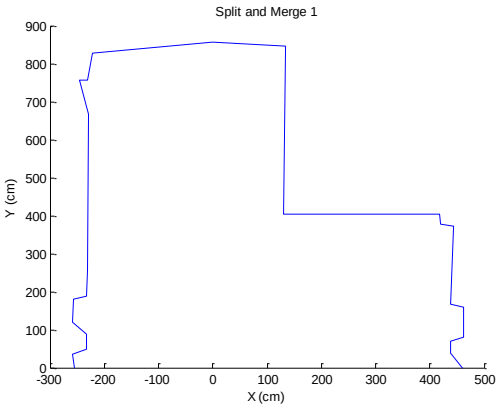

Figure 3.8 First local map

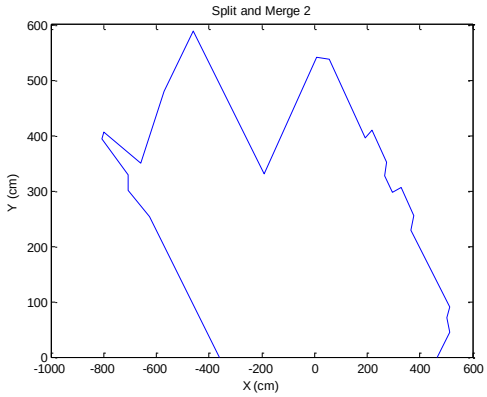

Figure 3.10 Second local map

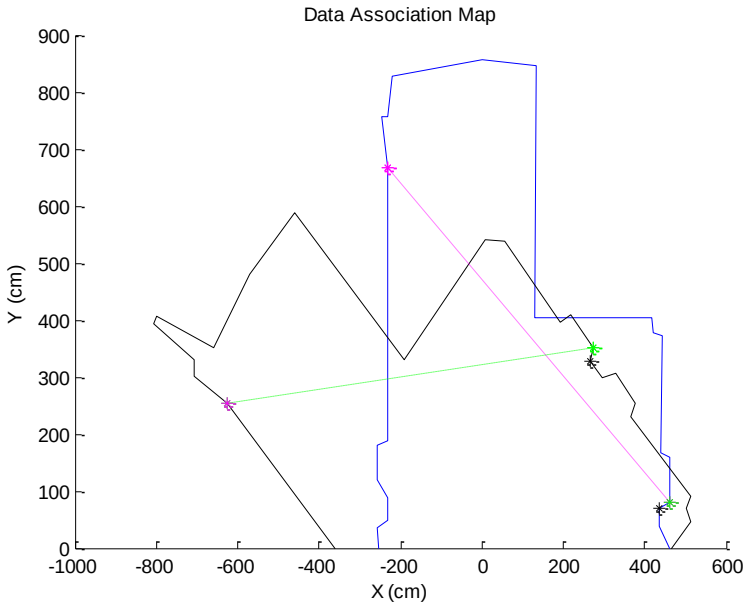

Figure 3.11 Data association map

Fig. 3.7, and Fig. 3.9 shows the two different scan of the environment. As it can be seen the orientation of the robot is slightly changed between two scans. The corresponding local maps are shown in the figures 3.8, and 3.10. Here distance threshold is set 10 for the Split & Merge algorithm. Fig. 3.11 shows the result of the Distance & Angle Algorithm for the two local maps that are shown in the figures 3.8, and 3.10. In the Fig. 3.11 pink dashed line is the reference of the first local map. The end points of this reference line are selected randomly. Then the corresponding line is found in the other local map. To find it the lengths and the angles of the reference line and its correspondent are compared. The following relation is used;

$$|L - L_1| < L(\frac{T_L}{100}) \quad \textbf{(3.9)}$$

In Eq. 3.9, L is the length of the reference line of the first local map which is the pink colored line in Fig. 3.11, L_1 is the length of the green colored line in Fig. 3.11, and T_L is the length threshold which is equal to 1 for this specific case.

When two matching lines are found as a result of the D&A Algorithm, "estimateGeometricTransform" function of the Matlab Image Processing Toolbox is used to calculate the x,y and θ values of the mobile robot with respect to the first scan.

When the value of the T_L is increased the quality of the results decreases because wrong lines can be chosen as a matching lines in the two consecutive local maps. When the value of the T_L is decreased the processing time of the D&A algorithm increases and it may not be possible for the algorithm to find a matching line because of the length threshold criteria which is given in the Eq. 3.9.

3.9. Results of the Split & Merge and D&A Algorithm

The result of the localization depends on the quality of the local maps that are obtained with S&M Algorithm. Split & Merge Algorithm does not produce good results when there is too much objects in an environment. When all of the threshold values are selected appropriately for a specific environment and for a specific amount of movement the results obtained with the S&M and D&A algorithms can be good but this is not generally the case. This may be due to the randomness involved in the D&A algorithm. As a result, it is possible to say that the localization algorithm that we have implemented is not robust. That's why we tried to develop different algorithms that produced better results.

4. LOCALIZATION METHOD 2

In this method, RANSAC Algorithm is used as a segmentation algorithm and with it the number of points obtained during measurements is decreased to a much processable value. Two local maps are obtained using RANSAC Algorithm for two different laser scan data of the environment and then Distance & Angle algorithm is used to find the matching points in the two local maps. Finally, optimal roto-translational matrix $R(\theta, T)$ is found using the Genetic Algorithm for the selected matching points which are found using D&A Algorithm.

4.1. Feature Extraction Algorithm: RANSAC

The RANSAC Algorithm (Random Sample and Consensus) was first introduced by Fischler and Bolles in 1981, as a method to estimate the parameters of a certain model starting from a set of data contaminated by large amount of outliers [6]. The important feature of RANSAC is the incorporation of certain probabilistic characteristics of a scan to determine the run-time length of the algorithm [7].

In general it consists of two major steps [7]:

- Estimation of the number of random tries to search a line, taking into account the predefined probabilities
- The iterative searching of potential segment point groups

In our code, the estimation of the number of tries is skipped; instead the number of tries to a search a line is predefined.

The steps of the RANSAC algorithm are given below.

1. At first the outlier points in the initial measurement set, which is obtained using laser scanner, are eliminated and a new set is obtained which is named as Pn.
2. Then a random line is generated by selecting two different random points in the set Pn.
3. Then the distances of all the points in the Pn to the line which is generated in Step 2 are found.
4. The points which have smaller distance than a predefined threshold value are found and they are collected in another set named Pc. This predefined distance threshold value is shown as T_D.

5. The length of the line is found and the number of elements in the Pc is found. Line point density is calculated using the Eq. 4.1.

$$Line\ point\ density = \frac{Number\ of\ elements\ in\ Pc}{Line\ length} \qquad \textbf{(4.1)}$$

6. If line point density and the number of elements in the set Pc is bigger than or equal to threshold values, the elements of the Pc are extracted from the elements of the set Pn. Otherwise algorithm is repeated starting from Step 2. The line which is generated in the Step 1 represents the corresponding set Pc.
7. The algorithm is repeated until the number of elements in the set Pn is smaller than the predefined element number value which is shown as T_{MIN}.
8. The end points of lines which pass the criterion explained in the Step 6 are collected as result.

a. Experimental Results of RANSAC Algorithm

The output of the RANSAC algorithm for the laser scan which is shown in Fig. 4.1 is shown in the Fig. 4.2 below.

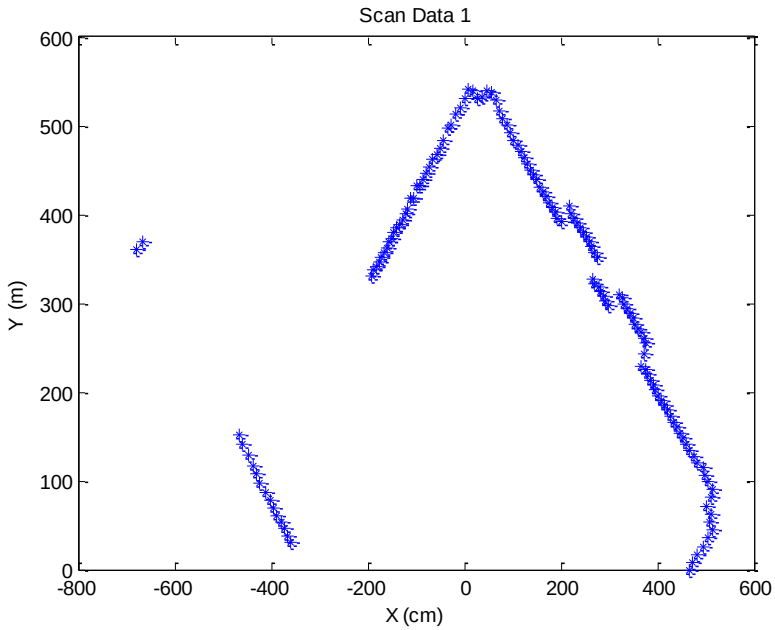

Figure 4.1 Laser scan of the corridor

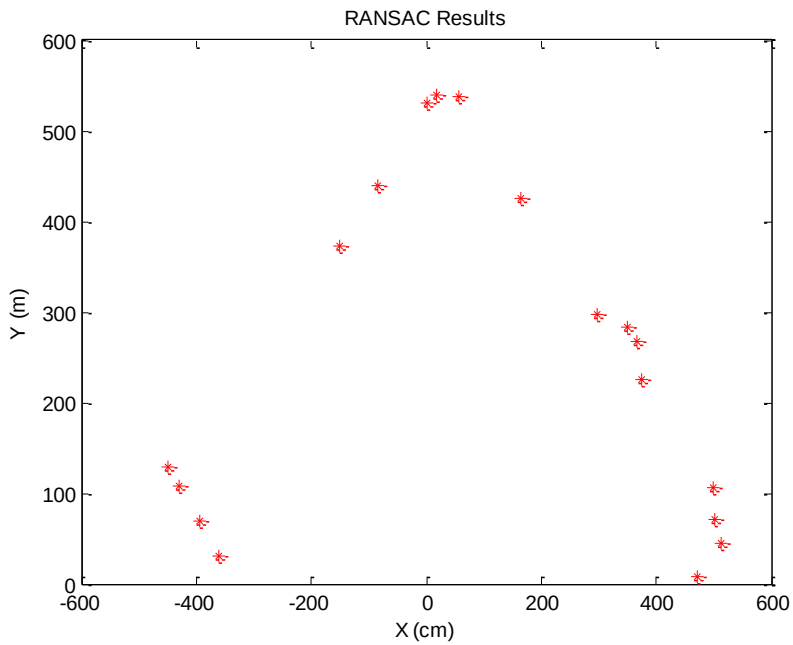

Figure 4.2 Result of the RANSAC Algorithm

In the Matlab code, T_{MIN} is set equal to 5 and T_D is set equal to 20. As a result of RANSAC algorithm, the number of features obtained with these threshold values are decreased to 18.

4.3. Properties of RANSAC Algorithm

- RANSAC is a non-deterministic algorithm; it involves randomness. It's results for a specific scan data can change.
- It produces better results; in other words local maps than Split & Merge algorithm when there are outliers because of the existence of the line density criteria which is explained thoroughly in the steps of the algorithm.
- It can also be used for different purposes other than segmentation.
- It is easy to apply and is very fast but it is slower than the Split & Merge Algorithm.
- The measurement set should not supposed to be ordered to apply the RANSAC algorithm.
- It is easy to change the results of the algorithm by changing the threshold values. In this way, this algorithm is more adaptable to different tasks and environments than Split & Merge Algorithm.

4.4. Data Association

After obtaining two local maps for the two consecutive scan of the environment, data association is made; matching points in the two local maps are found. To do data association, Distance & Angle algorithm is used which was explained in the section 3.6 of this document.

4.5. Genetic Algorithm

Genetic Algorithms, which is invented by John Holland in the early 1970's, were invented to mimic some of the processes observed in natural evolution [8]. The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. It is possible to apply genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear. The genetic algorithm uses three main types of rules at each step to create the next generation from the current population [9]:

- Selection rules select the individuals, called *parents*, that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.
- Mutation rules apply random changes to individual parents to form children.

4.5.1. Steps of the Genetic Algorithm

1. Randomly initialize population.
2. Determine fitness values of the initial population for the given objective function.
3. Select parents from population
4. Perform crossover on selected parents to create the next population.

5. Perform mutation of population.
6. Determine fitness values of the newly generated population.
7. Repeat the algortihm starting from the step 3 until the change of the objective function value becomes smaller than a predefined value or until the maximum number of generations, which is also predefined, is exceeded.

Genetic Algorithm is used to minimize the error function which is given in the Eq. 4.2.

$$E(\theta, T_x, T_y) = \frac{1}{N}\sum_{i=1}^{N} \|x_i - R(\theta)y_i - T\|^2 \quad \textbf{(4.2)}$$

Here, R is the rotation matrix, and T is translation vector given in Eq. 4.3 and Eq. 4.4 respectively.

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \quad \textbf{(4.3)}$$

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix} \quad \textbf{(4.4)}$$

In the Eq. 4.2; x_i, and y_i are the matching points which are found using the Distance & Angle Algorithm and, N is the number of matching point pairs. The number of matching point pairs can be easily increased and the result of the Genetic Algorithm will be more accurate when more point pairs are used.

Genetic Algorithm is implemented using the Matlab's Optimization Toolbox and "ga" function is used.

5. NEW APPROACH FOR INDOOR LOCALIZATION

In this method position of the robot is measured from its distance from the walls or objects that are long enough. For this reason line equations of walls are gotten to measure the distance from the robot. To make exact measurements it is very important to get line equations without a mistake. To do this several criteria are applied to the scanned points. First, points that are not belong to walls or long objects are eliminated. After that each wall in the environment are categorized as right wall, left wall and opposite wall. Thereafter most appropriate walls are determined from each category. Finally robot's position is measured from most appropriate walls. These steps are given below.

1. Eliminate irrelevant points.
2. Categorize each wall as right wall, left wall and opposite wall.
3. Select most appropriate walls in each category.
4. Obtain line equations of most appropriate walls.
5. Measure and update the robot's position by using line equations.

5.1. Irrelevant Point Elimination

To eliminate irrelevant points following procedure is applied.

1. Assign each point an importance value equal to zero.
2. For each point calculate the distance from that point to all of the other points.
3. When a distance is shorter than designated distance threshold increase the importance value of related point by '1'.
4. Compare each point importance value with designated importance threshold.
5. If an importance value of a point is smaller than importance threshold eliminate related point.

5.2. Wall Identification

It is very important to identify walls correctly. Any error in wall identification causes incorrect position values. To minimize the error two different procedures applied according to the robot's direction.

5.2.1. Determining the Direction

To identify walls robot's direction is need to be known. There are three possible directions for robot. It can be heading for right side, left side or moving parallel to side walls. In these situations robot's direction angle is positive, negative or approximately zero respectively. If robot's direction angle is approximately zero then different procedures are followed from positive or negative direction angle situations.

5.2.2. Identifying Walls When Robot Moves Parallel to Side Walls

When robot goes straight it means that it moves parallel to side walls. For this reason x axis position is determined from the wall that lies opposite of the robot. Afterwards, robot's y axis position is determined from walls which lie either sides of the robot. Finally robot's theta angle is determined from slopes of these two sidewalls.

5.2.2.1. Identifying Opposite Walls

Opposite walls can be easily identified when robot goes straight. First of all slopes between consecutive points are calculated. Because of robot's direction is known opposite walls slopes are approximately known. For this reason a slope range is designated and points belong to slopes which are not in the designated slope range are eliminated. After this elimination difference between remained consecutive slopes are compared with predetermined slope threshold. If difference between consecutive slopes is smaller than slope threshold points belong those slopes are accepted as belong to opposite walls. This procedure is explained in following steps.

1. Calculate slopes between consecutive points.
2. If the slopes of consecutive points are not in the designated slope range eliminate relevant points.
3. Calculate differences between remained consecutive slopes.
4. If difference between consecutive slopes are smaller than slope threshold accept those points are belong to opposite walls.

5.2.2.1.1. Identifying Most Appropriate Opposite Wall

After all of the opposite walls are identified most appropriate opposite wall can be determined according to the purpose. For example this can be the longest wall or can be the most distant wall from the robot.

5.2.2.2. Identifying Left Walls

As same in identifying opposite walls all of the slopes between consecutive points are calculated. Because of robot's direction is known left walls slopes are approximately known. For this reason a slope range is designated and points belong to slopes which are not in the designated slope range are eliminated. After that because of left walls y axis position is expected to be negative, points those have positive y values are eliminated. This inference is based on a truth that left walls always lie left side of the robot. This make left walls have negative y values. This inference is valid only if robot's theta angle is approximately zero which means that robot moves parallel to side walls. After the elimination of points those have positive y values, remained points are determined as which point belongs to which left wall according to their y position values. This is done by designating a threshold value for y values. Consecutive points those y position value differences are smaller than threshold are accepted as belong to same wall. This method consists of following steps:

1. Calculate slopes between consecutive points.
2. If the slopes of consecutive points are not in the designated slope range eliminate relevant points.
3. Eliminate points those have positive y position values.
4. Compare y values of remained consecutive points.
5. If difference of their y values are smaller than a designated threshold accept them as they belong to same wall. Identify each wall this way

5.2.2.2.1. Identifying Most Appropriate Left Wall

After all of the left walls are identified most appropriate left wall can be determined according to the purpose. In this study longest left wall is selected as most appropriate left wall.

5.2.2.3. Identifying Right Walls

Right walls are identified like left walls except of one difference. At step 3, instead of eliminating points with positive y values, points with negative y values are eliminated. Steps of identifying right walls are given below.

1. Calculate slopes between consecutive points.
2. If the slopes of consecutive points are not in the designated slope range eliminate relevant points.
3. Eliminate points those have negative y position values.
4. Compare y values of remained consecutive points.
5. If difference of their y values are smaller than a designated threshold accept them as they belong to same wall. Identify each wall in this way

5.2.2.3.1. Identifying Most Appropriate Right Wall

After all of the right walls are identified most appropriate right wall can be determined according to the purpose. In this study longest right wall is selected as most appropriate right wall.

5.2.3. Identifying Walls When Robot Heads For Left Side or Right Side

When robot heads for left side or right side it means that its theta angle is not equal to zero. Because of this reason walls cannot identified like explained before. To identify walls a line generation algorithm is written which is explained below. After the identification of walls by using the Eq. 5.2 robot's x axis position is determined from the wall that lies opposite of the robot. Robot's y axis position is determined from walls which lie either sides of the robot. Finally robot's theta angle is determined from slopes of these two sidewalls.

5.2.3.1. Line Generation Algorithm

To identify walls when robot heads for left side or right side an algorithm is written that generates all of the possible line equations that satisfied opposite and side walls. In this algorithm line equation at below is used.

$$y = mx + k \tag{5.1}$$

In this equation ‘y’ and ‘x’ are position values of scanned points, ‘m’ is the slope of line and ‘k’ is the point which the line crosses the y axis. To generate all of the possible lines nested for loop is used to iterate ‘m’ and ‘k’ values and also iterate the array which stores the x and y values of scanned points. The range of ‘m’ and ‘k’ are determined according to robot’s direction and wall type (opposite or side wall) to be identified. It will be explained in details in later chapters. After the determination of ‘m’ and ‘k’ iteration ranges, nested for loop is iterated for all of the scanned points. When line equation is satisfied for a point this point is added to an array which stores the possible points those belong to any wall. Until the for loop finished all of the points those satisfied line equations are added to arrays. Each array stores the points of each wall. Afterwards most appropriate walls among these are chosen for purpose. In this study generally longest walls are chosen as most appropriate wall.

5.2.3.1.1. Determining the Parameter Ranges

As explained above line generation algorithm is based on the iteration of ‘m’ and ‘k’ values which are constants of the line equation. Because of robot’s direction is known wall slopes are approximately known. For this reason different slope ranges are designated for the line generation algorithm. When robot is heading to left side positive m ranges are defined for both of the left wall and right wall. On the contrary for opposite walls negative m ranges are designated. The reason of this choice is based on the truth that when robot turns to left side it scans points with positive slope for left and right walls but negative slope for opposite wall. When robot is heading to ride side, for right and left walls negative m ranges and for opposite walls positive m ranges are defined. After the designation of m ranges, ranges for k are designated. In line equation ‘k’ is the point which the line crosses y axis. In line generation algorithm, ranges those consists of both positive and negative k values are designated. The reason of that is ‘k’ depends on the both of slope of walls and robot’s distance to walls. If the robot is distant from a wall and slope of wall is small then ‘k’ value of that wall be positive. In contrast when robot is close to a wall and slope of that wall is big in that case ‘k’ value of that wall be negative.

Designation of ‘m’ and ‘k’ ranges should be flexible. Because they depends on the environment which the robot is scan. In addition their ranges highly affect the speed of the algorithm. If their ranges will be too large and iteration range will be too small than the algorithm will work very slowly. In contrast if their ranges will be not

large enough and if the iteration range will be bigger than it must be at that case algorithm will not identify walls.

5.3. Obtaining Line Equations of Walls

After the identification of walls it is very easy to obtain line equations of these walls. As explained in "Wall Identification" two different methods are used to identify the walls.

First method is designed for the situation that the robot moves parallel to sidewalls. In this situation robot's position can be measured without obtaining line equations of walls. The reason of that is opposite and sidewalls are parallel to y and x axes of robot. Because of most appropriate opposite wall's theta angle is zero this wall lies parallel to robot's y axis. Thus the distance of robot from that wall is equal to x value of the wall. As a reminder robot is located at the origin. Afterwards the y position of robot is obtained from most appropriate sidewalls. These sidewalls are parallel to robot's x axis. Thus distances of robot from these walls are equal to y values of walls.

In second method, by using line generation algorithm 'm' and 'k' values of walls are obtained. In this way line equations of walls are already obtained when identifying walls.

5.4. Measuring Robot's Position

After line equations of wall are obtained robot's position can be measured by using distance from a point to line equations. By measuring robot's distance from walls periodically, change of the robot's position can be measured by comparing these distances periodically. As explained in "Obtaining Line Equations of Walls" when robot moves parallel to sidewalls there will be no need to use distance from a point to a line equations. However in other situations these formulas are given the robot's current position. To measure robot's distance from walls, equation at below is used.

$$d = \sqrt{\left(\frac{x_0 + my_0 - mk}{m^2 + 1} - x_0\right)^2 + (m\frac{x_0 + my_0 - mk}{m^2 + 1} + k - y_0)^2} \qquad \textbf{(5.2)}$$

In this equation 'x_0' and 'y_0' represents the coordinates of the point, which are equal to zero because of robot is located at the origin.

After the robot's distance from walls is measured, robot's current position is measured by adding changes of wall distances to previous position. By repeating these measurements, robot's position is known periodically.

5.5. Experimental Results

In this experiment ITU Control and Automation Laboratory corridor is scanned with LMS 200 laser sensor. Then the "New Approach for Localization" is performed.

5.5.1. Scan Data

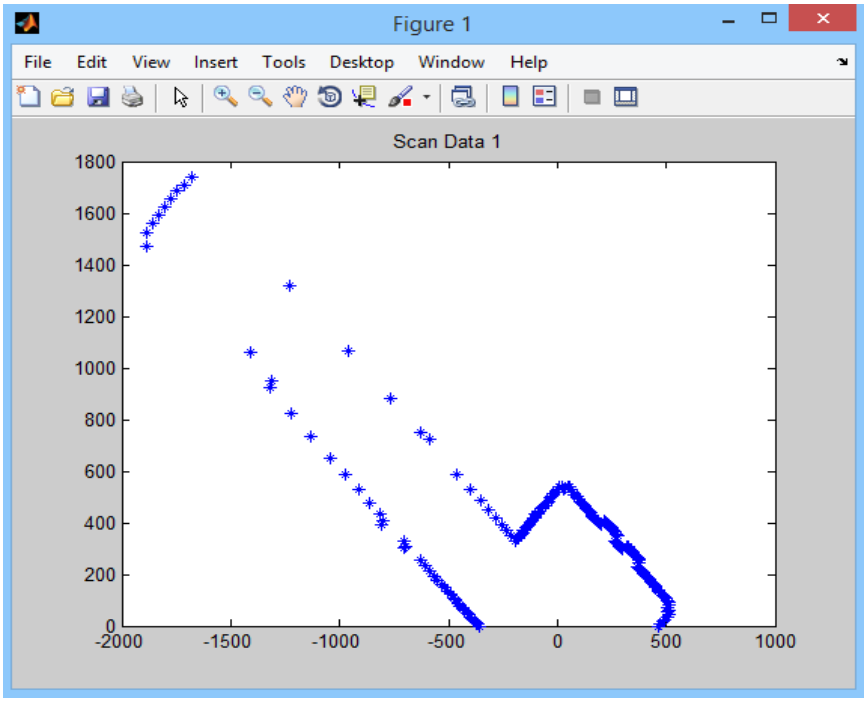

Figure 5.1 First scan data

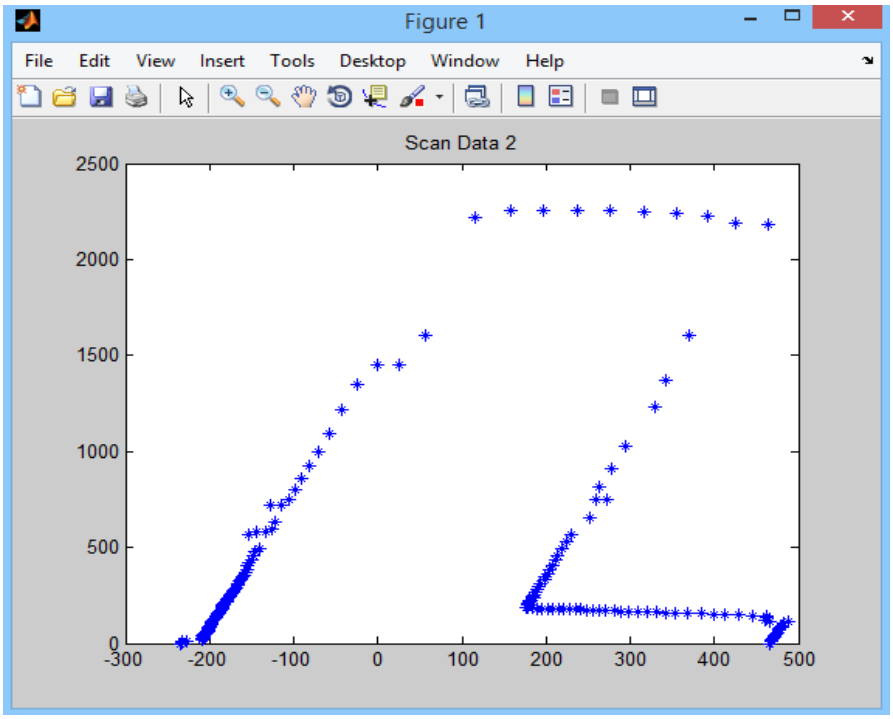

Figure 5.2 Second scan data

5.5.2. Opposite Wall Identification

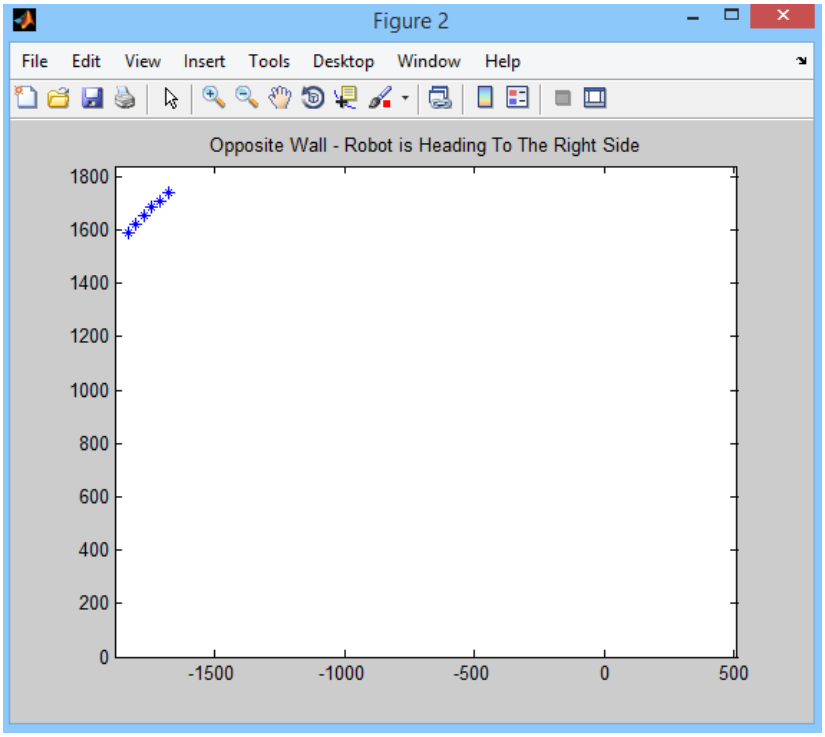

Figure 5.3 Opposite wall of first scan data

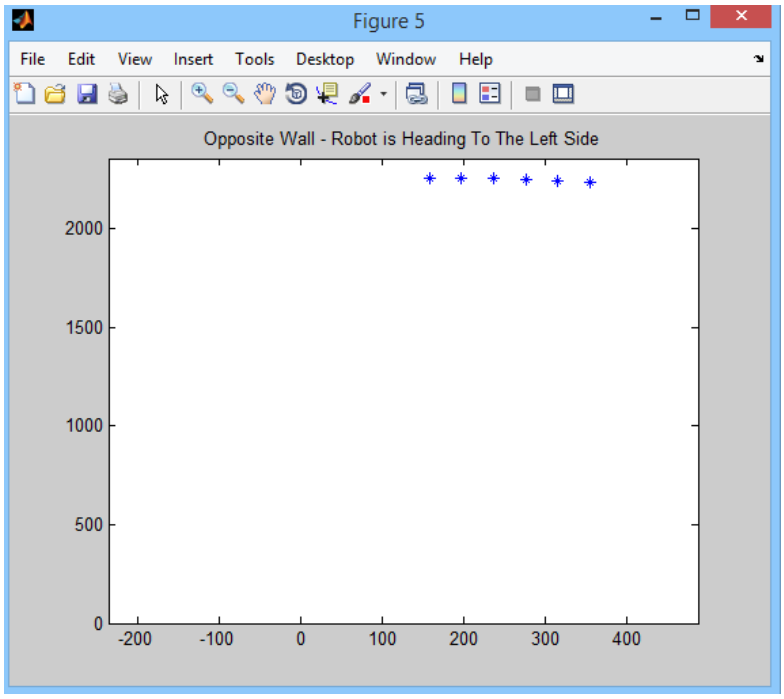

Figure 5.4 Opposite wall of second scan data

5.5.3. Right Wall Identification

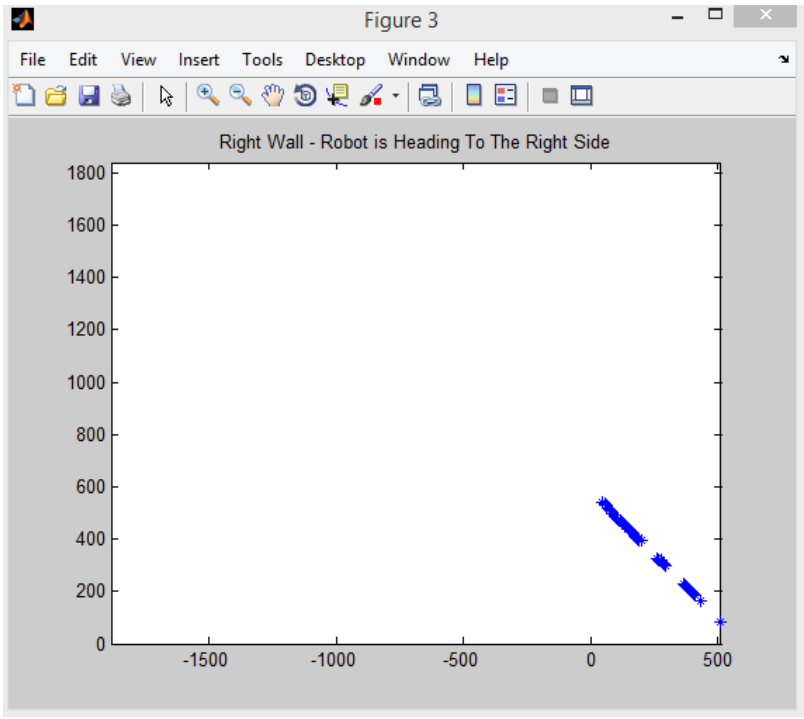

Figure 5.5 Right wall of first scan data

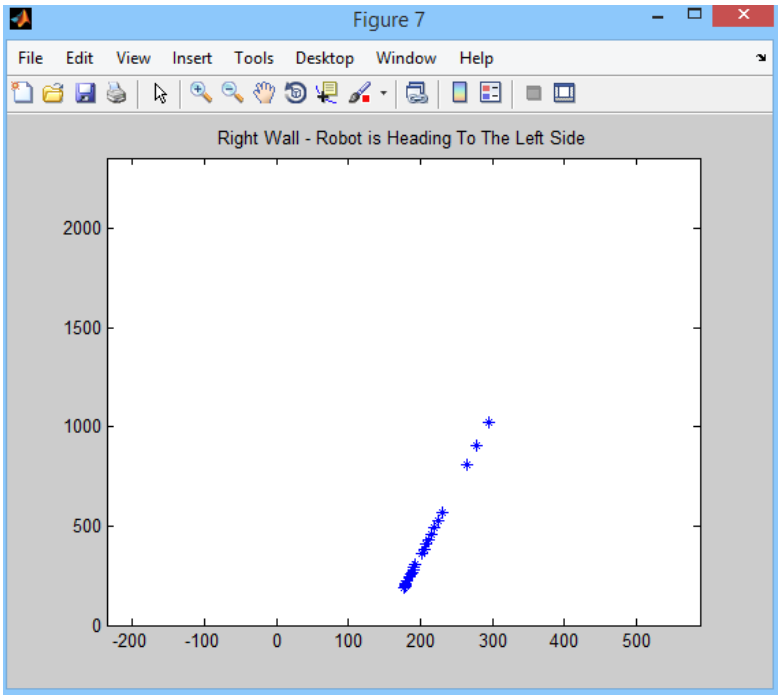

Figure 5.6 Right wall of second scan data

5.5.4. Left Wall Identification

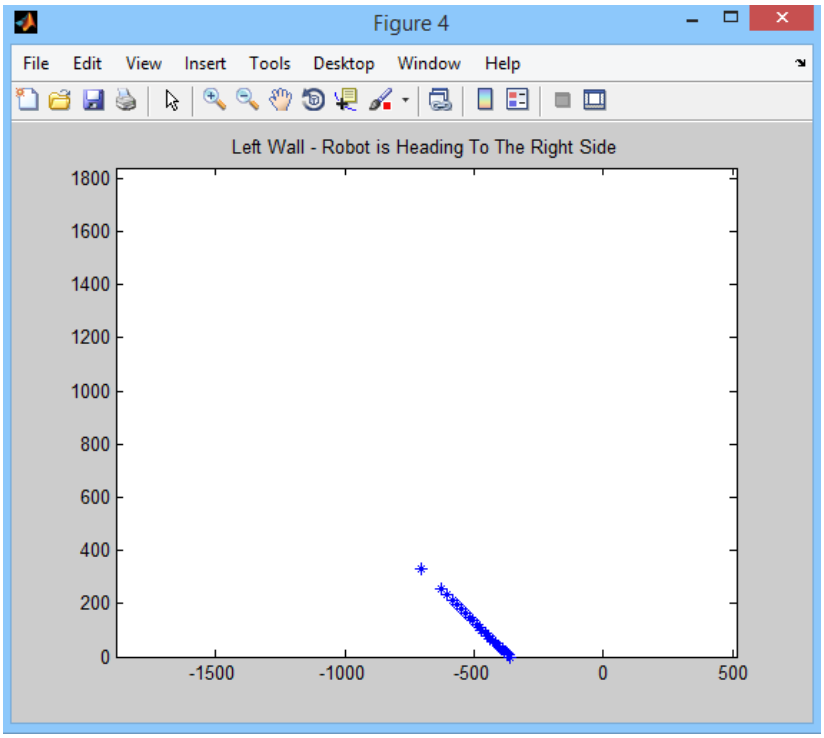

Figure 5.7 Left wall of first scan data

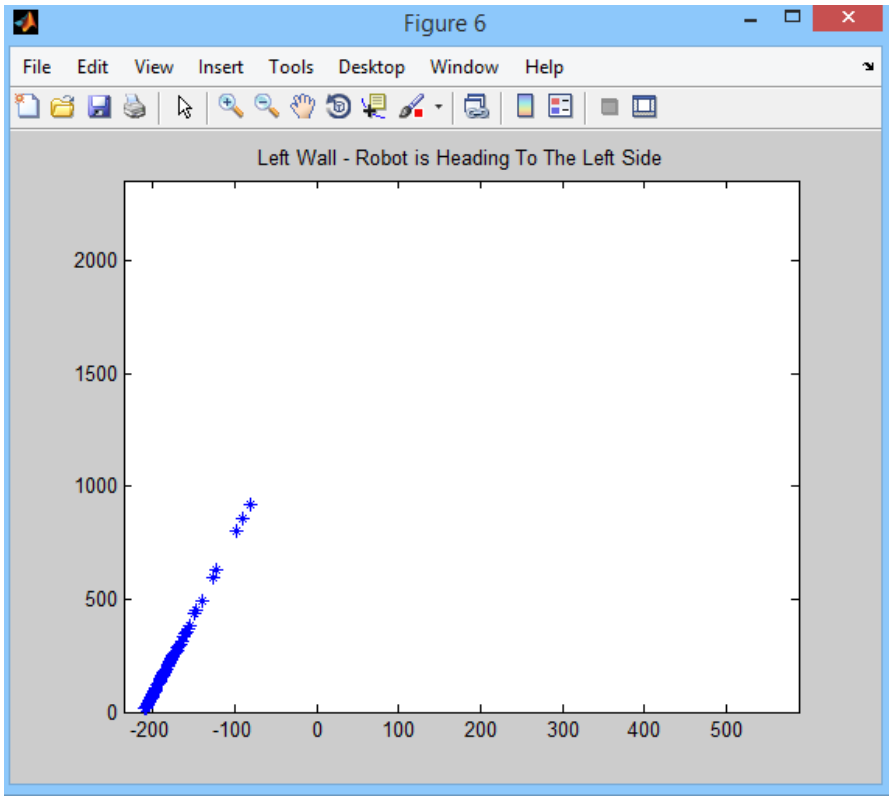

Figure 5.8 Left wall of second scan data

6. INTRODUCTION TO PATH PLANNING

After localization is made using the data that are obtained from the laser scanner, the robot should be able to find a path between its initial and goal position. The obstacles which are sensed using laser scanner should be avoided not to damage the robot.

In this project, before the robot moves initial scan is obtained. So the robot knows its initial position in the environment. Then we enter a goal position for the robot with respect to its initial position. So after running the path planning algorithm the robot should find a path between the initial and goal position or it should find the best step that should be followed in order to reach to the goal position. Then the robot should start to follow the path. After starting to move, the position of the robot with respect to its initial position is continuously found using the Localization algorithm. For each new position information, new path or new step to reach the goal position is calculated. This localization, path planning and path tracking algorithm should run in this order until the robot reaches the goal position.

Up to now several different localization algorithms to find the actual position of the robot with respect to initial frame are explained. In the next section a path planning method will be explained. There are several ways to plan a path for robots. In our project we have used the Potential Field approach.

6.1. Potential Field Method

In potential field approach, obstacles are defined as points like goal position and the path is planned step by step. More clearly, in each step direction and the speed of the robot is calculated independent of the previous step and the next step. As its name suggests a potential function is used to find the path. Different potential functions can be used depending on the application. Potential function is used to find the imaginary forces, which are assumed to be applied to the robot by the obstacles and the goal position. Obstacles exert repulsive forces to the robot and goal position exerts attractive force. The speed and the direction of the next step are found using the net force that is assumed to be applied to the robot.

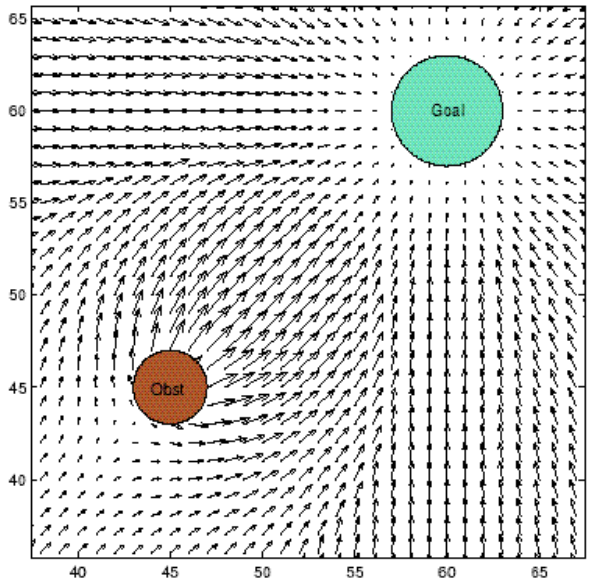

Figure 6.1 Attractive and repulsive forces for a field with a goal and an obstacle

Let's assume a potential function $U : R^m \rightarrow R$.

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{6.1}$$

Here, $U_{att}(q)$ is the attractive potential and $U_{rep}(q)$ is the repulsive potential. We want to find the minimum potential. That's why we find the derivative of the potential function. The derivative of the potential function is given below [10].

$$\nabla U(q) = [\frac{\partial U}{\partial q1}, \frac{\partial U}{\partial q2} \dots \frac{\partial U}{\partial qm}]^T \tag{6.2}$$

Here, $q = [q1\ q2 \dots qm]$ is the point in R^m.Since we are working in 2D space q is in R^2.

Potential function $U(q)$ will have one global minimum because there is only one goal position. The net force, which is assumed to be applied to the robot, is found using the equation below.

$$F_{net} = -\nabla \mathrm{U} = -\nabla U_{att} - \nabla U_{rep} \tag{6.3}$$

6.2. Attractive Potential Field

6.2.1 Conical Potential

The potential filed grows linearly with respect to the distance between the goal and the robot. The potential function is given below [10].

$$U_{att}(q) = \zeta d(q, q_{goal}) \tag{6.4}$$

$$\nabla U_{att}(q) = \frac{\zeta}{d(q, q_{goal})}(q - q_{goal}) \tag{6.5}$$

In the above equations $d(q, q_{goal})$ is the norm of the vector, which is $(q - q_{goal})$.

When the robot is in the goal position, the gradient of the potential function becomes undefined/indeterminate. That is why conical potential is not used solely to define the attractive potential field.

6.2.2. Quadratic Potential

Unlike conical potential, quadratic potential is a continuously differentiable function. The potential function is given below [10].

$$U_{att}(q) = \frac{\zeta d^2(q, q_{goal})}{2} \tag{6.6}$$

$$\nabla U_{att}(q) = \nabla\left(\frac{\zeta d^2(q, q_{goal})}{2}\right) \tag{6.7}$$

$$= \frac{1}{2}\zeta\nabla(d^2(q, q_{goal})) \tag{6.8}$$

$$= \zeta(q - q_{goal}) \tag{6.9}$$

The only downside of using the quadratic potential is that, the force -$\nabla U_{att}(q)$ becomes very large when the robot moves away from the goal position. That is why generally the combined version of conical and quadratic potential function is used as a potential function in path planning applications.

6.2.3. Combined Potential

$$U_{att}(q) = \begin{cases} \dfrac{\zeta d^2(q, q_{goal})}{2} & , d(q, q_{goal}) \le d^*_{goal} \\ d^*_{goal}\zeta d(q, q_{goal}) - \dfrac{(d^*_{goal})\text{^}2\zeta}{2} & , d(q, q_{goal}) > d^*_{goal} \end{cases} \quad \textbf{(6.10)}$$

$$\nabla U_{att}(q) = \begin{cases} \zeta(q - q_{goal}) & , d(q, q_{goal}) \le d^*_{goal} \\ \dfrac{d^*_{goal}\boldsymbol{\zeta}(q - q_{goal})}{d(q, q_{goal})} & , d(q, q_{goal}) > d^*_{goal} \end{cases} \quad \textbf{(6.11)}$$

The d^*_{goal} defines the transition between conical potential function to quadratic potential function.

6.3. Repulsive Potential Field

$$U_{att}(q) = \begin{cases} \dfrac{\eta(\frac{1}{D(q)} - \frac{1}{Q^*})\text{^}2}{2}, & D(q) \le Q^* \\ 0, & D(q) > Q^* \end{cases} \quad \textbf{(6.12)}$$

$$\nabla U_{rep}(q) = \begin{cases} \eta(\frac{1}{Q^*} - \frac{1}{D(q)})\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^* \\ 0, & D(q) > Q^* \end{cases} \quad \textbf{(6.13)}$$

Q^* is the radius of influence and D(q) is the shortest distance between the robot and an obstacle. When the distance between the robot and an obstacle is bigger than the radius of influence that obstacle will not exert any force on the object.

6.4. Properties of Potential Field Approach in Path Planning

It's an easy to apply and simple method. Implementing the potential field approach as an algorithm is not time consuming.

The method is based on finding the global minima of the gradient of the potential function. Sometimes the gradient of the potential can become 0 even if the robot is not in its desired position. Those points where the attractive and repulsive forces are balanced are called the local minima of the potential function. At local minima points, the next step that should be followed in order to reach the goal position cannot be calculated.

It is possible to use potential field approach to avoid moving obstacles by updating the potential field frequently.

The robot can get into a dead end because the path is planned step by step. Each step is found only to make the gradient of the potential function minimum.

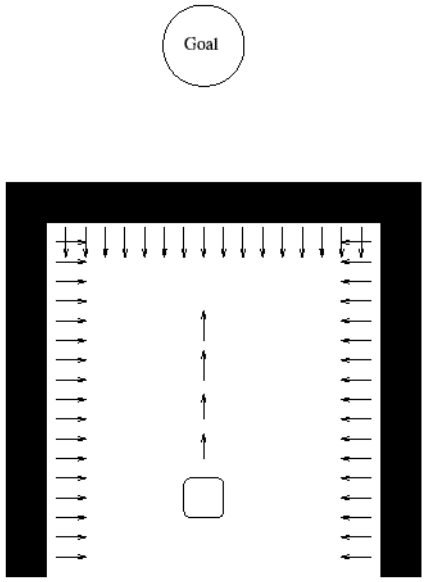

Figure 6.2 Local minima configuration when potential field approach is used

7. CONCLUSION

In this senior design project two different indoor localization methods for 2D environment were performed. In each localization method a laser sensor was scanned the environment. By using these scanned data robot's position was measured. The reason of applying two different methods was each method have its own advantages and disadvantages. In addition to this, a new localization method was designed which is an alternative to aforementioned localization methods. This new approach was also used laser sensor to gather the scan data of environment. After localization studies were finished a potential field path planning program was written. This program was not tested on mobile robot but successfully simulated on Matlab.

To conclude, a comprehensive study was performed which had an objective to finding the best solution for indoor mobile robots self-localization problem.

8. REFERENCES

[1] Olson, E. (2004). *A Primer on Odometry and Motor Control.* Retrieved from http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-186-mobile-autonomous-systems-laboratory-january-iap-2005/study-materials/odomtutorial.pdf

[2] Clark, C. (2011*). Autonomous Robot Navigation Lecture Notes.* Retrieved from https://www.cs.princeton.edu/courses/archive/fall11/cos495/COS495-Lecture5-Odometry.pdf

[3] SICK AG. (2006). *Telegrams for Configuring and Operating the LMS2xx Laser Measurement Systems.* Retrieved from http://sicktoolbox.sourceforge.net/docs/sick-lms-telegram-listing.pdf

[4] Thomas, D. (2014). Retrieved from http://wiki.ros.org/ROS/Introduction

[5] Bonaccorso, F., Muscato, G., & Baglio, S. (2012). *Laser range data scan-matching algorithm for mobile robot indoor self-localization*. Paper presented at Mexico. Retrieved from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6320883&isnumber=6326320

[6] Zuliani, M. (2011). *RANSAC for Dummies.* Retrieved from http://www.ic.unicamp.br/~rocha/teaching/2012s1/mc949/aulas/ransac-4-dummies.pdf

[7] Nock, G. (2003). *Segmentation Algorithms for 2D-Laserscans in an indoor environment, Zaragoza.* Unpublieshed doctoral dissertation. Retrieved from http://opus.htwg-konstanz.de/files/55/diploma_thesis.pdf

[8] Genetic Algorithms. (n.d.). Retrieved from the Imperial College website http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html#selection

[9] The Mathworks. (n.d.). *What is the Genetic Algorithm?*. Retrieved from http://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

[10] Choset, H. (n.d.). *Robotic Motion Planning: Potential Functions*. Retrieved from http://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf