2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics
Suntec Convention and Exhibition Center
Singapore, July 14-17, 2009

Line Matching Localization and Map Building with Least Square

E. Mihankhah[†], H.D. Taghirad[‡], A. Kalantari, E. Aboosaeedan, H. Semsarilar, *Department of Mechanical Engineering, K.N. Toosi University of Technology, Tehran*

Abstract—**we introduce a very fast and robust localization and 2D environment representation algorithm in this paper. This innovative method matches lines extracted from the LASER range finder distance data with the lines that construct the map, in order to calculate the local translation and rotation. This matching is done with a simple least square with no iterations. The algorithm is suitable for any indoor environment with mostly polygonal structure and has proven high speed and robustness in the experimental tests on our innovatively designed tracked mobile rescue robot "Silver". One experimental test is presented in the last section of this paper where the outputs are presented. These outputs are: 1- drift free raster map made of points and 2- A gallery of lines providing a linear ground truth.**

Keywords: Resquake, Rescue Robot, SLAM, Least square Line Matching, Simultaneous localization and mapping, Ground truth.

I. Introduction

Among all the issues concerning mobile robotics, self localization and mapping seem to be the basis for almost all the other autonomous behaviors. Many researchers have introduced different methods of self localization and mapping during the past years. This problem has the same objectives for all SLAM algorithms. To be fast (real time), to have low computational cost (the robot has a lot other things to do), to be robust against noise, to give more information keeping less amount of data in memory, to be able to recall, use and update old information and to generate a map comparable to the ground truth.

Many localization and mapping algorithms are based on point to point matching of LASER range finder (LRF) scans [1] and [2]. Since these methods should analyze a lot of points in iterations, the computation cost and slowness is in the nature of them.

Yaqub, Tordon and Katupitiya presented a line matching algorithm in [3] to solve the problem with computational cost. They used encoder information to help data association, and conjugate gradient to solve the formulated geometrical constraints. They do not match scans with the map, but they match two consequent scans. Since they tested the algorithm in simulation, drifts are not noticeable. But matching two scans in real environment ends to very noticeable drifts after a couple of turns since errors in matching will be cumulatively added. These errors are due to the fact that lines extracted from real environment are extracted from noisy sensor data, so they are not very certain lines.

Gutmann, Weigel and Nebel solved the positioning problem in RoboCup Soccer field [4]. The robot should try to find its position by matching lines to the boundaries of the soccer field. They fused LRF data with odometery in a Kalman filter. Mapping is not an issue in the algorithm they provided since the profile of the boundaries is known.

In this paper we provide a line matching algorithm that matches each scan with the map in order to find the position and orientation of the robot by performing a least square to a set of linearized equations. Unlike the many other methods, we do not seek the maximum overlap of the points in the map but we are looking for the similarities between each new scan and the map. After each matching, map is updated with the help of the updated position and orientation. The gallery of lines which helps constructing the map in background is also updated in each scan. At the end, the length of each line in the gallery is exactly known so these lines can provide a very accurate linear ground truth besides the raster map which is constructed from the points. All this is done just by processing the distance data provided by a LRF.

Section II of this paper introduces our experimental platform which is a tracked mobile rescue robot named Silver. The environment in which the robot is supposed to move and the flow of data in the algorithm are also introduced later in the same section. Section III describes the details of the algorithm. An experiment in real environment is discussed in section IV and the paper in concluded in Section V.

II. Project at a Glance

In parts A and B of this section, platform, environment and sensors are described. Then the logic and overall procedure of line matching localization and mapping with least square is presented in part C.

A. Experimental Platform

The experimental platform is a tracked mobile rescue robot named "Silver".

Silver is a tracked mobile robot which was designed in 2004 by Resquake Robotics team at K.N. Toosi University of Technology. This robot was first introduced in RoboCup2005 -Osaka, Japan- in rescue robot league and was awarded for "2nd Place Best Design" of rescue robot [5]. Both the hardware and software of the robot was improved for RoboCup2006, 2007 and 2008 world

[†] Corresponding author, Graduate Student of K.N. Toosi University of Technology
[‡] Associate professor of K.N. Toosi University of Technology

championship competitions. The robot won the innovation award for “Best operator interface” in RoboCup2006 - Bremen, Germany. Resquake won the “3rd Place” in RoboCup2008 -Suzhou, China- in rescue robot league main competitions and “2nd Place” in “Best in Class Mobility” challenge. This robot has four independently actuated arms that help it climb obstacles up to 40cm high. Tracks and arms are driven by DC motors. Position of arms and speed of tracks are measured. An illustrative description of its main characteristics has been presented in [6] and [7].

Adding autonomous behavior to Silver and reducing the effect of human operator is our current research approach. Fast autonomous map generation and localization is one of the most important features that can enable Silver to get necessary information from the environment for making further intelligent decisions. Silver is shown in Fig.1.

B. *Sensors and Environment*

A Hokuyo URG LX04 is our LRF. We only process the distance data of this sensor in our calculations. The sensor can measure distances up to 4 meters with 1mm resolution and 5mm tolerance. It provides 770 distance data from 270 degrees sweep angle with the sampling rate of 10Hz.

Fig. 1. Silver navigating in a sample indoor environment. There are more linear objects and straight walls than rounded and non-straight object in the environment. The ground is not completely flat. Some roll-pitch ramps cover some parts of the ground. (Thanks to Mr.Raymond Sheh for taking this photo in International RoboCup2008, Suzhou, China).

The environment can be any area which contains more linear objects and straight walls than rounded and non-straight objects. Most common indoor environments have such polygonal structure. A good model of such environment is the yellow arena of Rescue Robot League of international RoboCup competitions. Many pictures and information about such environment could be found at http://www.isd.mel.nist.gov/projects/USAR/2009/index.htm.

This environment is not completely flat and some roll-pitch ramps cover some parts of the ground. Silver in such environment is shown in Fig. 1.

A stabilizing mechanism keeps the LRF horizontal at all times. To sense the amount of roll and pitch elements of orientation, a 3DOF gyro (Microstrain 3DM-GX1 orientation sensor) is used. This mechanism is shown in Fig. 2.

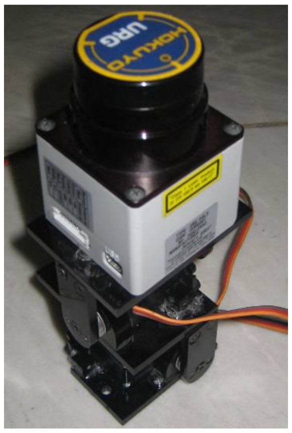


Fig. 2. LASER Range finder is mounted on a stabilizing mechanism. This mechanism keeps the LASER range finder horizontal on roll-pitch ramps. A 3DOF gyro is used to sense roll and pitch values.

C. *Introducing the Algorithm*

The general idea is to find linear objects in each new scan and find the best local transformation vector [dX dY dΘ] that makes this scan more similar to the map by applying a least square to a group of linearized equations. Each equation in group represents our attempt to fit a point taken from one of the “new lines extracted from LRF input distance data” into the equation that describes a line taken from the “gallery of lines that construct the map”. General transformation vector [X Y Θ] which keeps the position and orientation of the robot is updated with the resulting local transformation vector. Finally the “new lines extracted from the new scan” and the pair lines in line gallery are merged to update (by making corrections and adding extensions) the line gallery.

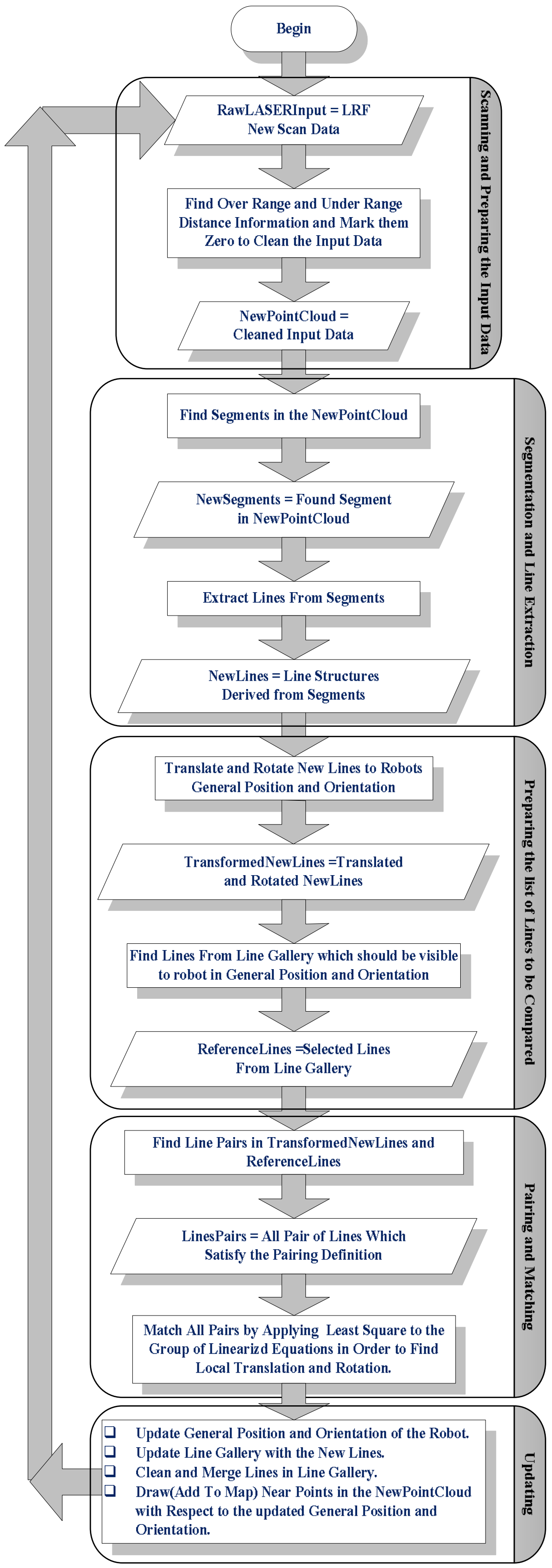
It could be learnt from Fig. 3 that the whole procedure is dividable to 5 main sub procedures. These sub procedure are described in Section III of this paper.

It should be noted that our high level programming language is Microsoft C#.Net 2008 and the operating system is Microsoft Windows XP running on a Sony VIO UX 380 laptop.

III. Algorithm Details

A. *Scanning and Preparing the Input Data*

The output of LRF is converted to an array of integer numbers that represent the distances sensed by the LRF in millimeters. Errors and over range measurements should be removed from this array (sensor marks these with values less than 20). Also there are some noisy distance data smaller than the boundaries of robot (less than 250mm) which are needed to be filtered out. All removed points are marked zero in the array. Zero values are ignored in calculations. The final array is stored in NewPointCloud arraylist.


Begin
RawLASERInput = LRF New Scan Data
Find Over Range and Under Range Distance Information and Mark them Zero to Clean the Input Data
NewPointCloud = Cleaned Input Data
Scanning and Preparing the Input Data
Find Segments in the NewPointCloud
NewSegments = Found Segment in NewPointCloud
Extract Lines From Segments
NewLines = Line Structures Derived from Segments
Segmentation and Line Extraction
Translate and Rotate New Lines to Robots General Position and Orientation
TransformedNewLines =Translated and Rotated NewLines
Find Lines From Line Gallery which should be visible to robot in General Position and Orientation
ReferenceLines =Selected Lines From Line Gallery
Preparing the list of Lines to be Compared
Find Line Pairs in TransformedNewLines and ReferenceLines
LinesPairs = All Pair of Lines Which Satisfy the Pairing Definition
Match All Pairs by Applying Least Square to the Group of Linearizd Equations in Order to Find Local Translation and Rotation.
Pairing and Matching
Update General Position and Orientation of the Robot.
Update Line Gallery with the New Lines.
Clean and Merge Lines in Line Gallery.
Draw(Add To Map) Near Points in the NewPointCloud with Respect to the updated General Position and Orientation.
Updating
Fig. 3. The algorithm that matches "lines extracted from the new scan" with lines in the "line gallery that construct the map" by applying least square to a group of linearized equations. This algorithm then updates the position and orientation of the robot while adding information to the map.

B. *Segmentation and Line Extraction*

To determine the segments in each scan, two consequent distance data are compared. When this difference is greater than a threshold, a new segment is started. The segments that contain small number of points are ignored. The rest are stored in NewSegments arraylist.

Then segments are broken to sub-segments so that a line can be fitted to the points in that sub-segment. This method is described in [8].

Finally the line that best fits the points in the sub-segment is found from (1) to (3).

$$tg(2\phi) = \frac{-2\sum_i(\bar{x}-x_i)(\bar{y}-y_i)}{\sum_i[(\bar{y}-y_i)^2-(\bar{x}-x_i)^2]}, \tag{1}$$

$$r = \bar{x}\cos\phi + \bar{y}\sin\phi, \tag{2}$$

Where

$$\bar{x} = \frac{1}{n}\sum_i x_i \text{ and } \bar{y} = \frac{1}{n}\sum_i y_i, \tag{3}$$

In these equations, r is the normal distance of the line from the origin and ϕ is angle of the normal. These line fitting formulas are taken from [9].

After finding the line, the start and end point of the segment are projected on this line and these two projected points are stored as the terminal points of this line segment in NewLines arraylist

C. *Preparing the List of Lines to be compared*

All lines that construct the map during the scans are stored in an arraylist named LineGallery. The general position vector [X Y Θ] stores the position and orientation of the robot. In an arbitrary position, it is possible to determine which of the lines in LineGallery should be visible to the robot. These lines are copied to ReferenceLines arraylist. Then the new lines that are extracted from the new scan should be translated and rotated to the general position vector (when the new scan is taken, all values are in origin), so that the set of lines taken from the gallery in the arbitrary general position would be similar and comparable to the new lines. The terminal points of new lines are transformed to arbitrary [X Y Θ] general position in (4)

$$\begin{bmatrix} X_t^b & Y_t^b & 1 \\ X_t^e & Y_t^e & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & X \\ \sin\theta & \cos\theta & Y \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X_o^b & X_o^e \\ Y_o^b & Y_o^e \\ 1 & 1 \end{bmatrix}, \tag{4}$$

Where notation b stands for beginning point of the line segment, e stands for the end point of the line segment, t stands for values in translated coordinate and o stands for values in origin. Please note that naming "beginning point" and "ending point" to the terminal points of line segment is just to separate them in calculations and one could call any of them the "beginning point" or "ending point".

As it was mentioned in part B, only terminal points of a line segment are stored in the arraylists. This means after preparing the list of lines to be compared, two arraylists containing terminal points of line segments are generated. One contains terminal points of some line segments from

LineGallery (ReferenceLines) and the other contains terminal points of the line segments extracted from the new scan and transformed to the general position of the robot (NewTransformedLines). Two comparable sets of line segments are ready to be compared in order to find the local translation and rotation of robot during the last new scan. Next step is to find the pair lines and then match the lines.

D. Pairing and Matching

In simulation, lines are exactly known objects and by matching two scans, all linear objects are ideally overlapping. But when working with real environment it is very common that two consequent scans are NOT exactly similar in features. For example it is very much possible that some newer lines are found in newer scan and/or some lines are not anymore found in the newer scan. Also in majority of cases the lines extracted from two consequent scans do not have exactly equal lengths. Also due to the measurement noise, the lines extracted from the same object in two different scans do not have exactly the same equations. Also it is possible that one line is extracted from an object in one scan, since more than one line could be extracted from the same object in the next scan. The two sets of lines generated in part C of section III have the above conditions. That means some of the line segments in NewTransformedLines are candidates to have been extracted from the same objects that some lines in ReferenceLines do. So, the pairing is not necessarily one by one pairing but we mean finding such candidates by using the word "pairing". Pair lines have the following characters:

1- When the terminal points of a line segment from NewTransformedLines are projected on a line segment in ReferenceLines, at least one of the projected points would lie between the two terminal points of the reference line.

2- The difference between the slopes of these lines is less than a threshold.

3- The distances of terminal points of the line segment in NewTransformedLines are less than a threshold from the candidate line in ReferenceLines.

If all 3 pairing conditions are met for two line segments, then a pair is chosen from NewTransformedLines and ReferenceLines.

Each line in NewTransformedLines is compared to all lines in ReferenceLines and pairs are found this way. This should be noted that it is possible for two or more lines in NewTransformedLines to be pair with one line in ReferenceLines and it is also possible that one line in NewTransformedLines would be pair with more than one line in ReferenceLines. That is because breaking a segment into lines (part B of section III) is based on a threshold and due to the existing measurement noise, multiple sets of lines can be extracted from the same object in two different scans when the line extraction algorithm works near the threshold. All pairs are accepted and used in calculations.

As it is said before, it is impossible to find a vector [dX dY dΘ] in real environment capable of fitting all pairs together after applying the transformation. All it could be done is to find a vector of translation and rotation that after applying to new lines, all pairs, in general, look best alike. Assume that there are n pairs found. We focus on each pair to see how a desired [dX dY dΘ] vector could be found. The equation of the Reference line in ith pair could be written as in (5).

$$a_i x + b_i y + c_i = 0\,, \qquad (5)$$

Terminal points of this line segment are known, so

$$a_i = (Y_b^{r,i} - Y_e^{r,i})\,, \qquad (6)$$

$$b_i = (X_e^{r,i} - X_b^{r,i})\,, \qquad (7)$$

$$c_i = Y_b^{r,i}(X_b^{r,i} - X_e^{r,i}) + X_b^{r,i}(Y_e^{r,i} - Y_b^{r,i}))\,, \qquad (8)$$

Where notation r shows the parameter is taken from ReferenceLines, notation b denotes begin point and e denotes the end point.

A good [dX dY dΘ] vector, guarantees the begin and end points of the new line segment in this pair to satisfy equations (9) to (11):

$$a_i X_b'^{n,i} + b_i Y_b'^{n,i} + c_i = 0\,, \qquad (9)$$

$$a_i X_e'^{n,i} + b_i Y_e'^{n,i} + c_i = 0\,, \qquad (10)$$

$$\begin{bmatrix} X_b'^{n,i} & Y_b'^{n,i} & 1 \\ X_e'^{n,i} & Y_e'^{n,i} & 1 \end{bmatrix} = \begin{bmatrix} 1 & -d\theta & dX \\ d\theta & 1 & dY \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X_b^{n,i} & X_e^{n,i} \\ Y_b^{n,i} & Y_e^{n,i} \\ 1 & 1 \end{bmatrix}, \qquad (11)$$

Where notation n shows the parameter is taken from TransformedNewLines. Please note that because this calculation finds the translation and rotation during the very short amount of time (100 milliseconds), cos(dΘ) is linearized to 1 and sin(dΘ) to dΘ. The values with prime sign are the transformed new terminal points that are also transformed by the local transformation matrix.

The equations (5) to (11) are supposed to persist for all pairs. That means for n pairs, 2n linear equations similar to (9) and (10) are generated where dX, dY and dΘ are the unknown values in all equations. The problem to find [dX dY dΘ] is reduced to a least square problem.

$$A\begin{bmatrix} dX \\ dY \\ d\theta \end{bmatrix} = B\,, \qquad (12)$$

$$\begin{bmatrix} dX \\ dY \\ d\theta \end{bmatrix} = A^{+}B\,, \qquad (13)$$

Where A^+ is the pseudo inverse of matrix A and

$$A = \begin{bmatrix} a_1 & b_1 & b_1 X_b^{n,1} - a_1 Y_b^{n,1} \\ a_1 & b_1 & b_1 X_e^{n,1} - a_1 Y_e^{n,1} \\ . & . & . \\ . & . & . \\ a_i & b_i & b_i X_b^{n,i} - a_i Y_b^{n,i} \\ a_i & b_i & b_i X_e^{n,i} - a_i Y_e^{n,i} \\ . & . & . \\ . & . & . \end{bmatrix}, \quad (14)$$

And

$$B = \begin{bmatrix} -c_1 - a_1 X_b^{n,1} - b_1 Y_b^{n,1} \\ -c_1 - a_1 X_e^{n,1} - b_1 Y_e^{n,1} \\ . \\ . \\ -c_i - a_i X_b^{n,i} - b_i Y_b^{n,i} \\ -c_i - a_i X_e^{n,i} - b_i Y_e^{n,i} \\ . \\ . \end{bmatrix}, \quad (15)$$

The resulting [dX dY dΘ] is the best vector which can describe the amount of translation and rotation of the robot during the last scan time (100 milliseconds) so that the profile of the new scan would be very much similar to the map. Now that the best transformation matrix is known, the mean square value in (16) can give a measure of how much this transformation is not satisfying the pair lines to overlap.

$$\text{Overlap Mismatch} = \frac{1}{n} \sum_i \begin{bmatrix} (a_i X_b'^{n,i} + b_i Y_b'^{n,i} + c_i)^2 + \\ (a_i X_e'^{n,i} + b_i Y_e'^{n,i} + c_i)^2 \end{bmatrix}, \quad (16)$$

Where X′ and Y′ values are calculated from (11) with known [dX dY dΘ].

E. Updating

First thing to be updated is the position of robot. So, [dX dY dΘ] is added to [X Y Θ]. Please note that dX and dY values should go to a coordinate system which is rotated Θ radians prior to be added to X and Y. Then dΘ is directly added to Θ. After the update, [X Y Θ] is the current general position of the robot.

Having the updated general position vector, the new lines can be transformed and merged to the lines in the line gallery. In order to do this, line pairs are again used. When updating the gallery, three important steps are followed:

1 – The best line that fits the terminal points of both lines (4 points) in a pair is found.

2 – Terminal points of both lines in the pair are projected on the line created in step 1 and the two furthest points are considered as the terminal points of the new line.

3 – The created line replaces the old line in the gallery and an integer value is incremented for this line that keep the number of times this line has been reviewed in the gallery. This integer number is used in step one to give more weight to the terminal points of line taken from the gallery when updating the direction of this line. New lines that no matches of them could have been found in the line gallery are added to the gallery and weighted one.

After updating the gallery, some internal merging is also needed. As it was mentioned in part D of section III, more than one line may be the pair for another line either in NewTransformedLines or ReferenceLines. This means that after performing the update to the gallery there will be some lines in the gallery that are so similar that can be merged to one longer line. In order to do that, an internal pairing and merging similar to what it was done in part D of section 3 is performed to the gallery.

Finally, to make sure that the newly added lines to the gallery are not mistakenly generated (for example because of the measurement noise or because of the delay in stabilizing mechanism to make the LRF horizontal after falling from a roll-pitch ramp), the gallery is cleaned up after each scan and the lines that are not reviewed at all during the certain number of scans are assumed to be noise and though removed from the gallery.

The last step is to draw the NewPointClaud on the map by applying the updated transformation matrix (17):

$$\begin{bmatrix} \cos\theta & -\sin\theta & X \\ \sin\theta & \cos\theta & Y \\ 0 & 0 & 1 \end{bmatrix}, \quad (17)$$

IV. Experimental Results

To test if the algorithm is robust enough, an arena with the similar structure to what it was explained in part B of section II was prepared. Some Roll-Pitch ramps were also added to the arena. An autonomous fuzzy obstacle avoidance algorithm was running the robot in the small arena.

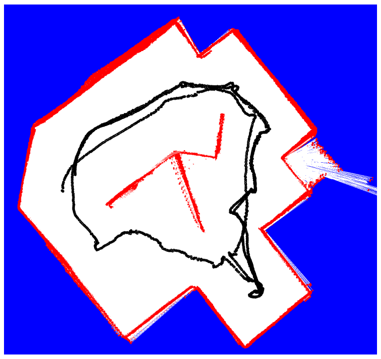


Fig. 4. Final map of a closed-loop arena. The black path is the movement path of the robot. The arena is not flat and some roll-pitch ramps are inside the arena. Walls are made of wood and the robot navigates in a 4-by-5 meters arena.

The reason why the small area is chosen is to visually check the errors and drifts which cannot be seen in the tests in large areas or long hallways, also it is needed that the algorithm checks if the robot can remember the lines it has seen before and use them hundreds of scans later again to complete, close and correct the map.

In Fig.4 the final map of a closed-loop arena is shown. The black path is the path in which the robot has moved. The top linear speed of robot was 15cm/s and the max rotational speed was 0.75rad/s. The algorithm runs at real time and no more than 20% of CPU is busy with the whole navigation and mapping algorithm. If the logged data is processed later (without the 100ms delay of sensor, no navigation process and no drawings), it is possible to process more than 120 scans per second which is a huge number in comparison with the other matching techniques. Another important result of this mapping algorithm is the gallery of lines that are used to process the map. This gallery itself is a precious output, because the lengths of gallery lines are exactly known. That means this algorithm can provide the ground truth made of lines, besides a raster map made of points (See Fig.5).

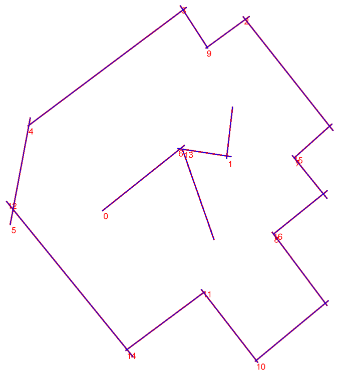


Fig. 5. Lines in Line gallery. 17 lines (the numbers are marked at the ending point of each line) have constructed the map. The sizes of these lines are exactly known. This can provide a precise linear ground truth.

The video of this map being constructed could be found at http://saba.kntu.ac.ir/resquake/Closed-Loop_Arena.wmv

The result of some other tests in a U-shaped arena and in a large RoboCup yellow arena can be seen in Fig.6. The video of these maps being constructed could be found at:

http://saba.kntu.ac.ir/resquake/U-Shaped_Arena.wmv
http://saba.kntu.ac.ir/resquake/RoboCupArena.wmv

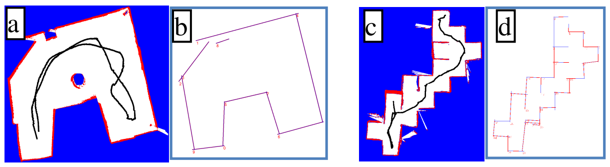


Fig. 6. a) The map of a 4 by 5 meters U-shaped arena with a rounded dustbin in the middle. b) Line gallery of U-shaped arena (10 lines).
c) The map of RoboCup yellow arena test (Iran Open 2009 Competitions). This arena is 11 by 13 meters wide and covered with roll-pitch ramps. d) Line gallery of RoboCup yellow arena test (32 lines).

V. Conclusion

In this paper we introduced a very fast and robust line matching localization and mapping algorithm based on least square. One of the advantages of using lines instead of points is reducing the amount of calculations. When for instance, 500 points in a scan are compared with the other 500 points in another scan, a 500*500 point scope is to be searched in order to find matching points and this search should be done iteratively to find the acceptable translation and rotation between the two consequent scans. But by extracting lines from these points and comparing the lines, the calculation is done over something around 10 pairs of lines (a couple of line more or less based on the environment) and problem is finally reduced to applying a least square over a group of 20 linear equations only once (no iterations) . It is also important to mention that extracting lines from points is a noise reducing step itself. So, the sensor noise is also filtered with the least possible amount of computational effort. In point to point matching algorithms searching for most number of overlapping points is the goal, so the measurement noise is very offensive in these methods, but we find the similarity between the lines extracted from a new scan and the lines that has made the map until now. So, this approach reduces the effect of noise in the calculations. Drifts are very common in consequent scan matching algorithms. Keeping lines in the memory and searching among them is very easy because of the very little amount of memory needed to be allocated for keeping 4 values to memorize the terminal points of each line segment. That means it is possible to match new scan with the map and not with the previous scan. This helps the final map to be drift-free. Also this method gives another output which is the gallery of lines with exactly known lengths. This can provide an exact ground truth based on lines, besides the raster map made of points.

References

[1] J. Nieto, T. Bailey, and E. Nebot, "Recursive scan-matching slam," in *Proc. Robotics and Autonomous Systems*, Jan 2007, vol. 55, no. 1, pp. 39–49.

[2] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. IEEE International Conference on Robotics and Automation*, Sacramento, CA, USA, 1991, pp. 2724–2729 vol.3.

[3] T. Yaqub, M.J. Tordon, and J. Katupitiya, "Line Segment Based Scan Matching for Concurrent Mapping and Localization of a Mobile Robot," in *Proc. 9th International Conference on Control, Automation, Robotics and Vision*, Singapore, Dec 2006, pp.1 – 6.

[4] J. S. Gutmann, T. Weigel, and B. Nebel, "accurate and robust self localization in polygonal environments," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems,* Oct. 1999, Kyongju, Korea.

[5] E. Mihankhah, A. Kalantari, E. Aboosaeedan, H.D. Taghirad, and S.Ali.A. Moosavian , "Autonomous Staircase Detection and Stair Climbing for a Tracked Mobile Robot using Fuzzy Controller," in *Proc. 2008 IEEE International Conference on Robotics and Biometrics*, Feb 2009, Bangkok, Thailand, pp. 1980-1985.

[6] S. Ali A. Moosavian, H. Semsarilar, and A. Kalantari, "Design and Manufacturing of a Mobile Rescue Robot," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, Beijing, China, Oct. 2006, pp. 3982-3987.

[7] S. Ali.A. Moosavian, and A. Kalantari, "Experimental Slip Estimation for Exact Kinematics Modeling and Control of a Tracked Mobile Robot," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2008, Nice, France.

[8] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, "A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics", in *Proc. Conference on Intelligent Robots and Systems*, Aug 2005, Edmonton, Canada, pp. 1929- 1934

[9] W. Burgard and M. Hebert, "World modeling", *Springer Handbook of robotics*, Part E.36, p-p 853-869.