Mobile Robot Navigation in an Unknown Environment

A. Jazayeri, A.Fatehi, H. Taghirad

Faculty of Electrical Engineering
K.N. Toosi University of Technology
Seyyed Khandan,
P.O. Box:16315-1355
Tehran, Iran

Phone: +98(21)8846-9084
Fax: +98(21)8846-2066

jazayeri@resquake.com
fatehi@kntu.ac.ir
taghirad@kntu.ac.ir

Abstract— **This paper discusses application of an intelligent system in order to navigate in real-time a small size, four wheeled, indoor mobile robot accurately using ultra-light (160 gr), inexpensive laser range finder without prior information of the environment. A recurrent neural network is used to find the best path to the target of the robot. An accurate grid-based map is generated using a laser range finder scene and location found by a modified dead reckoning system. Finally a motion control method is presented. These approaches are implemented and tested in Resquake mobile robot.**

I. INTRODUCTION

In some hazardous environment where attendance of human force might be threatened by an incident, a mobile robot can be used to manipulate a specific operation. In chemical, gas, oil or other harmful arenas, a mobile robot will be very helpful especially if it can operate autonomously. The application of autonomous robots extends to rescue, fire fighter, space robots, mine sweeper and many other applications [6].

There are three challenges in navigation of autonomous mobile robots in an unknown environment. First, the robot decides a target location according to sensor data and the aim of the robot. For instance, as a rescue robot, carbon dioxide of breathe, body temperature, sound, movement, etc. can be assumed as the victim signs. Second, the robot finds a way to go there using current position of the robot, generated map of obstacles and target location. Finally, the robot is controlled in the obtained path by the motion control strategy. In this paper, last two challenges have been focused on, including localization, map generating, path planning, and motion control of a mobile robot in an unknown environment.

Varieties of methods have been developed for navigation of a mobile robot. Some robots are equipped with a GPD or DGPS. However, for an indoor mobile robot other localization sensors are used such as dead reckoning, SLAM [12, 17, 18], inertial navigation [19], beacons, vision sensor [20, 22], geometric [9, 10] or hybrid topological and metrical [7]. Laser range finders (LRF) are widely used for both mapping and localization in two last decades. Geometric methods had more accuracy rather than the probabilistic approaches. Instead probabilistic approaches [8] such as Monte Carlo [13, 14] or Markov [11] ones have more robustness but need a map of the environment and spend plenty of time.

In this paper, mobile robot navigation problem has been broken down to four sub-problems. An autonomous robot first should know its current position (Localization). Then the robot should perceive the environment (Mapping). After that, the robot should find its way between obstacles (path planning) and finally the robot should be automatically tracked the obtained path. Practical issues and experiences in implementation of selected approach has been mentioned and emphasized in this paper. Experimental results are tested on the mobile robot of Resquake team who won the second place of the best design award in Rescue Real Robot league of the International RoboCup Competition 2005 in Osaka, Japan.

Localization of mobile robot is discussed in the next section and proposed four methods to reduce error in a dead reckoning system. Section 3 concerns about accurate and robust map generating using a laser range finder. Recurrent neural network is presented in section 4 and then it is used to find a path from the current position of the robot to the final goal. Section 5 intended to simplify the path to make it easy to be followed by the robot. Section 6 discusses motion control of the robot in the simplified path and in the final section conclusions are stated.

II. LOCALIZATION

There are so many ways for positioning of a mobile robot including GPS[23], Sonar[15.16], gyroscope[19,23], dead reckoning[19,23] or sensor fusion[21,23]. For a 2D indoor mobile robot, a modified dead reckoning method is chosen using rotary encoder for measuring traveled distance of left and right side of the robot. Straightforward

formula suggested in [1] is used to calculate the differential variation of x-y-θ in a short period.

Localization result has been used in the map generator. Since any error in the localization affects the accuracy of the map, four easy, inexpensive and efficient modifications have been applied in addition to the usual method of dead reckoning to minimize the localization error in Resquake robot [5].

First, a free wheel is added to reduce effect of slippage of the main wheel on measurement of displacement. This new wheel will not transmit the power. So robot displacement is measured rather than the shaft rotation which will be slightly inaccurate during the slipping.

Second, in this four wheels robot, encoders have been installed very close to the drive wheels otherwise, the maximum acceptable obstacle height may reduce to 1 cm from the current 7 cm.

Third, for detecting obstacles errors, two other rotary encoder and free wheeling have been added in order to find mismatches and compensate detected errors by the other couple of encoders.

Finally, to minimize the error that is the actual displacement of the robot and measured data, the free wheels of the dead reckoning system are pushed to the ground with two springs which will be maximized the friction between the free wheel and ground surface and minimize the likelihood of slippage of the free wheel on the ground.

As a results of the above four solutions, localization error of the robot reduces to an acceptable value for our experiments when it moves straight. However when it turns some more error is appeared which will be compensated as explained in the next section.

III. MAP GENERATING

There is no prior information about the environment and the robot needs a method to avoid collisions with the obstacles. Very light, inexpensive and accurate Laser Range Finder [5] is used to find the surrounding objects of the robot. It measures the distance of the nearest object to the robot in each angle as a polar coordinate. Raw data of the LRF that is shown in Figure 1 depicted the output of the sensor in the environment of Figure 2. LRF is attached to the front of the robot.

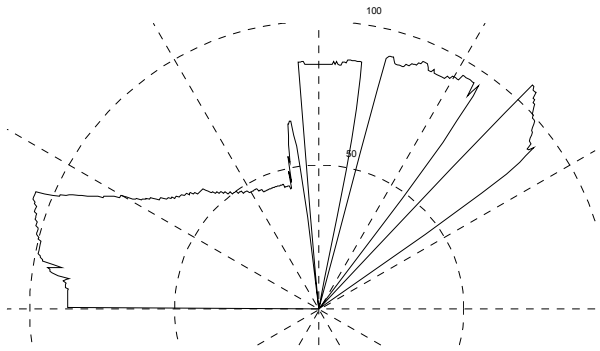

Figure 1 – Raw data coming from the HOKUYO brand LRF

Figure 2 – Real robot and the environment where the LRF observes

Global map is represented as a binary value for each cell. The cell value is one if there is an obstacle in the cell and is zero if there is no obstacle on it. This map can be outdated with every scene of the LRF knowing the exact location of the robot in the map. The LRF data is converted to Cartesian coordination and is put at the robot position in the global map.

After a while, when robot moves to unknown areas and sees what could not see before, the map of the environment will be completed step by step.

Figure 3 shows the raw data of Figure 1 in dot format. It is clear that there are some extra points in Figure 3 that are appeared when we connect sequential points to each other.

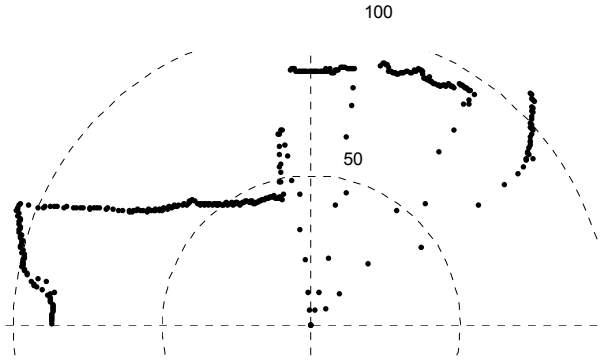

Figure 3 – Raw data coming from LRF in dot representation

This phenomenon is mainly caused by the sensor when two surfaces have a small gap between each other. It's important to exclude these points because if they exist in the global map, the map will be unclear and inaccurate.

This error should be filtered by a pre-filter on LRF data. Equation 1 is applied to the raw LRF data to omit extra points.

$$\text{Min}(|r_i - r_{i+1}|, |r_i - r_{i-1}|) < r_{\text{Threshold}} \qquad (1)$$

Where r_i is the i-th distance in the polar coordinate and $r_{\text{Threshold}}$ is the threshold distance. This equation means that points, which are alone or far from other point must not be selected. Filtered points are omitted from the vector and replaced by unknown distance. Figure 4 shows the map of Figure 3 after omitting the extra points of Figure 3.

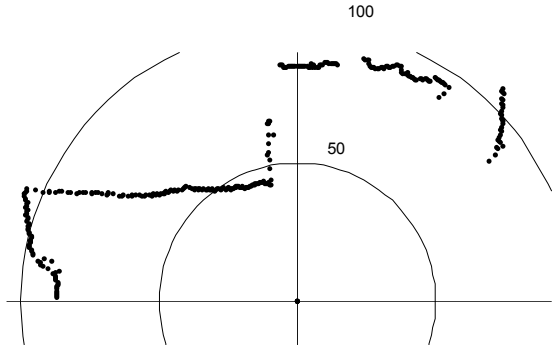

Figure 4 – Pre-filtered data of the LRF, noise has been omitted

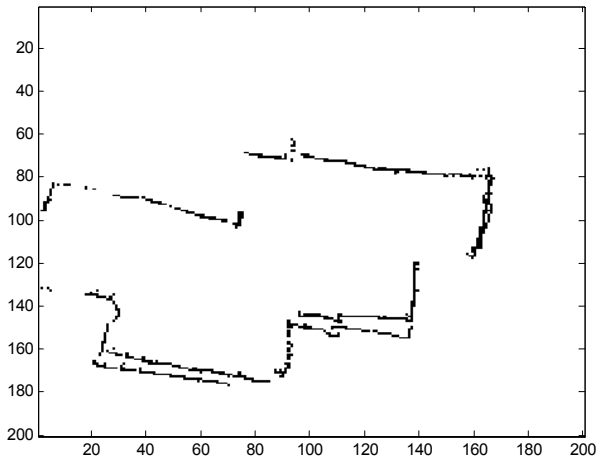

Figure 5 – Global map when the robot move around the room

In Figure 5 the resulting map is depicted when the robot turned 360 degree and different scenes of the laser range finder is put on the robot location in the map. Direction of the robot is taken account to rotate the scenes and then put them on the appropriate position.

The second map is very similar to the first map but localization error effect can be seen in Figure 5 where two walls below the figure are not superposed to each other. This figure and other experimental results show that the localization error is significant when the robot turns and negligible when the robot go straight.

IV. PATH PLANNING

In this section the generated map, current location and target location are used to find the best path that the robot should travel to achieve the target without collision with obstacles. Before using the map, there is a slight point about the robot volume and obstacles that must be considered in an experimental design. It is clear that the robot has a volume and cannot be assumed as a point in a map. The obstacle avoidance strategy should take account the robot width to find out whether the ways is wide enough for robot to pass the narrow aisle.

Since considering the robot shape for path planning will be complicated, another way is proposed. In this way, each cell of the obstacle is expanded to half of the robot's wide. The expanded obstacles map of Figure 5 is shown in Figure 6.

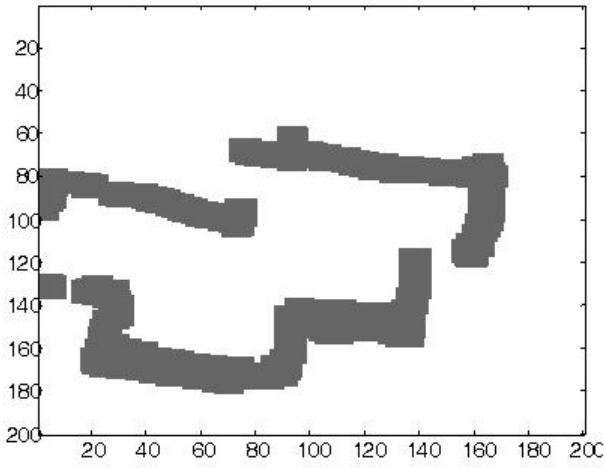

Figure 6 – A map with expanded walls

Now, the robot can be represented as a small point that should not enter to this gray area and there is no other constraint on the robot position which should apply to the robot movement to avoid obstacle collisions. There is lateral result of the obstacle expansion that the localization error due to turn movement can be eluded. It can be easily seen that the duplicated walls of Figure 5 are mixed in Figure 6.

The radius of expansion of obstacles is very important parameter that should choose cautiously. It should be large enough to make sure that the robot will not collide with the obstacle. But it can not be too large because this may close some ways to the robot while it can go through them. As a rule of thumb, it can be half-wide of the robot plus 5cm (For the presented robot), totally 25 cm.

In the generated map the position of the robot, target and obstacles are specified. A neural map is suggested in [2] for path planning. Each cell in the global map is assigned to a neuron of a recurrent neural network. For example in the map of Figure 6, a 200 by 200 occupancy grid map generates a neural network with 40000 neurons. Activation function of each neuron is a one-side sigmoid function:

$$g(x)=\begin{cases}\tanh(x) & x\geq 0\\ 0 & x<0\end{cases} \qquad (2)$$

Output of each cell is applied to every other cell with a weight *w*.

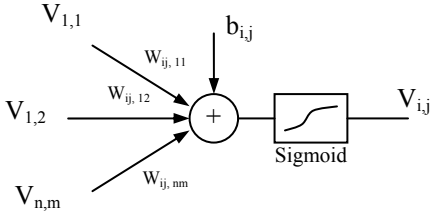

Figure 7 – (i,j)th neuron of the Neural network

The structure of a neuron is shown in Figure 7. *V*'s are the output of the network and are assigned to each cell of the global grid map. *b*'s are the bias of cells and *w*'s are weights that determine the effect of each cell to others. Target cell that assigned to the target neuron have positive

infinite bias. Obstacles have negative infinite bias and other cells have zero bias:

$$b_{ij} = \begin{cases} +\infty, & \text{Target neuron} \\ -\infty, & \text{Obstacle neurons} \\ 0, & \text{Other neurons} \end{cases} \quad (3)$$

A weight function determines the effect of each neuron to others and has chosen as Equation 4.

$$w_{ij,kl} = \begin{cases} 0 & d\{(i,j),(k,l)\} = 0 \\ f(d\{(i,j),(k,l)\}) & 0 < d\{(i,j),(k,l)\} < d_{\max} \\ 0 & d\{(i,j),(k,l)\} > d_{\max} \end{cases} \quad (4)$$

where f is a descending function and d is the distance function. f assumed to be $f(x)=1/x$ and d assumed to be a Cartesian distance of two neurons. So the weight function will be a large value for closed neurons and will be a small value for far neurons. It means that farther neurons have smaller effect and nearest neurons have the most significant effect on a cell. Also, each neuron has no effect on itself. Neurons farther than a specified radius have no effect.

The aim of the network is to find the best path in order to reach the target position. The network is unsupervised and training of this network means to recalculate the output of the network according to the current outputs. An incremental way is used to train the network. After training the network, output of the target neuron will be 1 because of its positive infinite bias passed through the sigmoid function. For the obstacle, negative infinite bias will have the output to be zero. Neurons near the target neuron affected by the target neuron and their output value will be higher than others. By going far from the target neuron, outputs will be decreased and near the obstacles, the neuron output will be forced to be declined to zero.

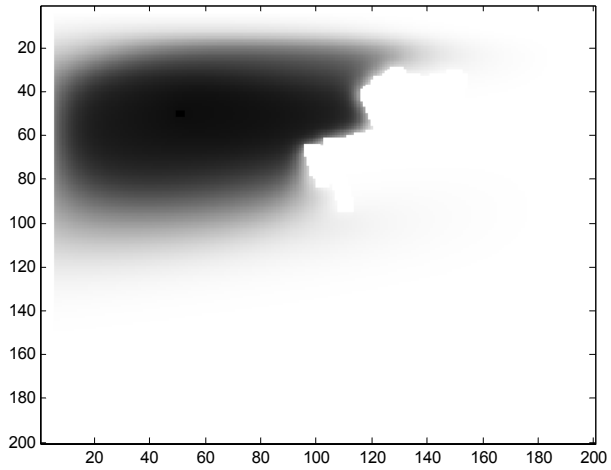

Figure 8 – Trained neural network output

An output of a generated neural map is shown in Figure 8. The target cell is the darkest point in Figure 8 and obstacles are bright white. Output values of far neurons from target one are very small but not zero. However, these small values can not be seen in this figure.

Process time for training the neural network is proportional to the number of neurons and training iterations. To minimize process time, three methods are used. First, the train radius is limited. Effect of far neurons is very small. This effect is descending according to the function $f(x)=1/x$ and for big distances the effect is negligible. Therefore, training of neuron (i,j) can be started at $(i-r,j-r)$ and ended at $(i+r,j+r)$. So the training time will be reduced highly. However, it is very important not to reduce it very much which will cause the far neurons to be unaware about the target neuron. So the train radius can not be very small. In the practical experiments, the train radius is chosen 5 and the number of iterations is 12.

Second, the train region can be limited. Those neurons, which need to be train are limited because neurons before the target neuron have not initially been affected by the target neuron and have the zero value. So, there is no need of training in some regions because all neurons in the region have the same value of zero. This is occurred in the primary iterations when the target neuron just affects very close neurons and training of far neurons can be skipped.

Finally, target effect should distribute uniformly in all direction, otherwise neurons after the target neuron will be affected more than other neurons. The direction of training alters every second iteration. Thus, the number of iterations needed to cover all the area is decreased numerously.

After training the network, it is shown in [2] that wherever the robot is located in the map, if it chooses the maximum slope when it wants to decide where to go, a path can be obtained. Result of this algorithm for path planning is shown in Figure 9.

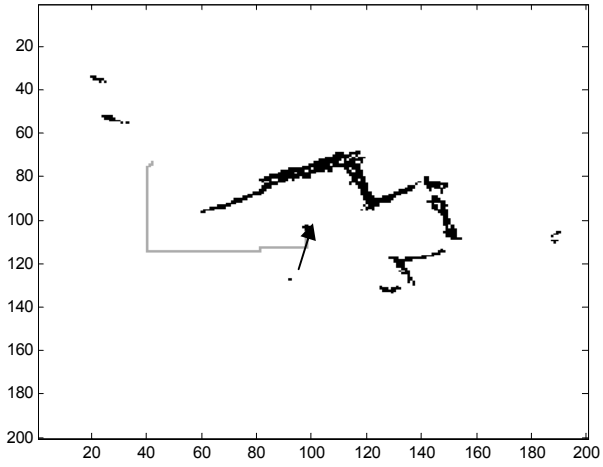

Figure 9 – Path planning result

In Figure 9 the initial robot direction is shown by an arrow to determine the robot orientation. Unexpanded walls are shown in as a black areas and obtained path is shown in gray line.

V. SIMPLIFYING THE PATH

Path planning algorithm of the last section derives a curve as a feasible robot trajectory to the target. Motion control in a general case and in an arbitrary curve needs to control each motor of the robot. Furthermore, localization error increases in robot turning movement that means that tracking a curve will increase the localization error incredibly. To avoid too many turnings, the following two techniques are implemented.

A. In every corner of the path, the robot must turn. As explained above, this makes some localization error. To avoid too many corner points in the path, the latest direction in the path is amplified with a coefficient larger than one during steepest ascend algorithm of path planning. In Figure 9, the amplification gain is 1.5 and, as a result, the number of the corners is reduced significantly. This simplification will be appreciated in the robot motion control when the robot tries to follow the path. The coefficient should not be so large because this amplification means that the decision on the direction will not be fair and the robot may approach the obstacles.

B. Let's divide the obtained trajectory to some straight lines. The robot moves from a start point of a line to the end point of it which will be the start point of the next line. Let's call these points corners. The robot should follow the corners to achieve the final target position.

To simplify the path, distance between each two corners are considered and if it is less than a threshold value, the line between them is absorbed to its neighbor line by eliminating the common corner of lines.

Also, a corner which is in the same orientation with some others can be eliminated. In this approach, for each corner the next and previous corners are connected with a straight line. If the distance of the corner to the line is less than a specified threshold, the corner can be left out.

The result of this approach to simplify the path is illustrated in Figure 10 where the blue line is the path, which is obtained from the neural map and the red circles are the final corners where the robot has to follow.

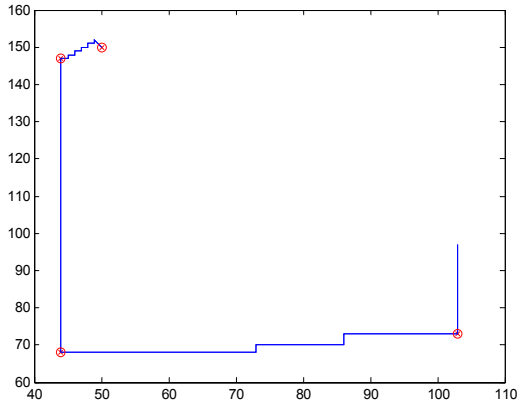

Figure 10 – The result of simplification of the path

In Figure 10 both distance thresholds are assumed to be 10cm. This value is chosen in a trade off. Larger value will increase the probability of collision of robot to obstacles and small values will increase the number of corners and path complication.

VI. Motion Control

The simplified path is easily applied to the robot as local destinations and the robot goes to each of these corners by two separate modes. First, it turns until the head of the robot points to the next corner and second it goes straight until the robot reaches to the next corner. Passing all the corners, the robot reaches the final target.

VII. Conclusion

In this paper, four effective and easy-to-use approaches are proposed to minimize the localization of dead reckoning system with rotary encoders. Laser range finder data is filtered and used on a global grid-base binary map. A recurrent neural network assigned to the map, which obtained the value of each cell of the map. Three proposed approaches reduce the process time of training to one second, programmed by C# on a 1.8 MHz Intel Centrino, which was suitable in online implementation. Three simplifications in the path make it more feasible to the robot to move with less localization error and the robot pursued corners to reach to the final target.

References

[1] G.W. Lucas, "A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators," http://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html, 2005.

[2] M. G. Lagoudakis, and A. S. Maida. "Neural Maps for Mobile Robot Navigation." *In Proceedings of the 1999 International Joint Conference on Neural Networks (IJCNN-99)*, 1999.

[3] J. Borenstein, "Where Am I? Sensors and Methods for Mobile Robot Positioning", *University of Michigan,* 1996.

[4] M. Hagan, H. Demuth, and M. Beale, "Neural Network Design," *Boston, MA: PWS Publishing*, 1996.

[5] A. Jazayeri, E. Mihankhah, and H. Semsarilar, "Team Description Paper," *International RoboCup Competitions and Symposium*, 2005.

[6] R. Siegwart , "Autonomous Mobile Robots", 1999

[7] A. C. Victorino, P. Rives, J.J. Borrelly, "Safe Navigation for Indoor Mobile Robots. Part II: Exploration, Self-Localization and Map Building", 2004

[8] J. Pineau, S. Thrun, D. Fox, "Concurrent Mapping and Navigation with Mobile Robots",1998

[9] J. Gomes-Mota, M. I.Ribeiro, "LOCALISATION OF A MOBILE ROBOT USING A LASER SCANNER ON RECONSTRUCTED 3D MODELS1", 2000

[10] D. Burschka, G. Hager "Laser-Based Position Tracking and Map Generation", IASTED, International Conference, 2000

[11] D. Fox, W. Burgard and S. Thrun, "Markov Localization for Reliable Robot Navigation and People Detection"

[12] J. Howell, B. R. Donald, "Practical Mobile Robot Self-Localization", 2000

[13] D. Fox, W. Burgardy, F. Dellaert and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation forMobile Robots", AAAI, 1999

[14] A. Milstein, "Dynamic Maps in Monte Carlo Localization", 2002

[15] V. Varveropoulos, "Robot Localization and Map Construction Using Sonar Data", 1999

[16] L. Kleeman, R. Kuc, "Mobile Robot Sonar for Target Localization and Classification", 1999

[17] T. Duckett, "A Genetic Algorithm for Simultaneous Localization and Mapping", 2003

[18] M. Montemerlo, S. Thrun, D. Koller and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem", 2002

[19] E. T. Baumgartner, H. Aghazarian, and A. Trebi-Ollennu, "Rover Localization Results for the FIDO Rover", 2002

[20] K. O. Arras, N. Tomatis, "Improving Robustness and Precision in Mobile Robot Localization by Using Laser Range Finding and Monocular Vision", 1999

[21] J. Nygards, A. Wernersson and R. Karlsson, "Sensor Fusion with Coordinated Mobile Robots", 2003

[22] C. F. Olson, "Mobile Robot Self-Localization by Iconic Matching of Range Maps", International Conference on Advanced Robotics, 1997

[23] P. Goel, S. I. Roumeliotis and G. S. Sukhatme, "Robot Localization Using Relative and Absolute Position Estimates", 1999