International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:4, No:2, 2010

Mobile Robot Path Planning in a 2-Dimentional Mesh

Doraid Dalalah

Abstract**—**A topologically oriented neural network is very efficient for real-time path planning for a mobile robot in changing environments. When using a recurrent neural network for this purpose and with the combination of the partial differential equation of heat transfer and the distributed potential concept of the network, the problem of obstacle avoidance of trajectory planning for a moving robot can be efficiently solved. The related dimensional network represents the state variables and the topology of the robot's working space. In this paper two approaches to problem solution are proposed. The first approach relies on the potential distribution of attraction distributed around the moving target, acting as a unique local extreme in the net, with the gradient of the state variables directing the *current flow* toward the source of the potential heat. The second approach considers two attractive and repulsive potential sources to decrease the time of potential distribution. Computer simulations have been carried out to interrogate the performance of the proposed approaches.

Keywords**—**Mobile robot, Path Planning, Mesh, Potential field.

I. Introduction

IN the past, numerous approaches has been done in solving the path-planning problem, mostly in static environments and unchangeable world space. In such planning some constraint conditions have to be met, such as the pass through via points, smoothness of the planned path when represented using spline, etc. There exists a number of global approaches, such as decomposition, road map, and retraction methods, randomized approaches, genetic algorithms as well as several local approaches, e.g., potential field methods. Recently, efforts have been directed toward, collision free, real-time trajectory planning in changing environments, [16, 17, 18]. Collision warning systems has been also implemented [6] and [9].

The configuration of the world space was based on the principle of splitting cubes in 3D to provide a compact index. When a primitive comes too close to other, a warning is issued. For the sake of real time operation, checks were done at discrete time intervals where it should be ensured that collisions cannot take place during these intervals and cycle time was taken as short as possible. The same idea of collision warning systems was presented by [5] et al. (2002). The authors tried to build an intelligent vehicle system to reduce collisions in transit buses. Although the system can predict early mutual collisions but it is still different from the problem of path finding and real-time sake.

Author is with Jordan University of Science and Technology, P. O. Box: 3030, Irbid 22110, Jordan

Other research interests were concerned in mobile manipulators and part orientation purposes with minimal sensing and manipulation, [10]. The study in [11] also presented a mobile manipulator that is used for manipulation as well as locomotion rather than motion panning. Such researches were dependent on geometrical relations and kinematics analysis.

Some of available approaches were concerned with determining the area of potential collision possibility, for instance using a set of all configurations causing a trajectory intersection of two cooperative robots, [2]. The task of finding collision free trajectories for a coordinated motion was defined as finding the successive configurations connecting the target point with the initial point configuration. However, although the approach turns out to be relatively simple, requiring a minimal number of calculations, the approach is not universal enough to include a large number of different robot geometries and orientations.

Movement synchronization, based on delaying actions, is a possible solution of mutual collision avoidance of two cooperative robots working in the same work place, as proposed by Roach (1987). However, later efforts have been focused on distributed potential approaches of attraction and repulsion, [1, 12, 13] and on hierarchical approaches using multi-pass dynamic programming, [13], or on genetic algorithms for real-time path planning, [1]. The approaches, however, still do not guarantee satisfactory collision-free real-time planning performances in changing environment. Other path planning approaches, mainly based on heuristic search methods, such as generate-and-test paradigm, [4], Puntryagin's Maximum Principle, [7], which is applied to the optimal control of differential drive mobile robots with velocity bounds. Such approaches considerably depend on heuristic or hierarchical methods and/or geometric relationships between the objects.

Topological maps for the robot workspace where efficient in determining successful routes. The study in [15] could use such maps accompanied with decision-theoretic approach in the robot navigation process. Topological and evaluation maps are the configuration system that will be used in the approaches presented in this paper.

With the application possibilities of neural networks, new trajectory planning approaches have been introduced, [8]. Using massively parallel, multi-neuron computing architectures, the topologically ordered maps for the robot configuration space have been built with occupied nodes as obstacles and with the node clamped to the unit value as the target. A full interconnection of neurons within the network was used as the initial configuration. For instance, [3]

proposed trajectory planning using a two- layer neural network. For internal information exchange between the neighboring nodes related to their potentials, the lateral interconnections between the neurons were implemented in the upper network layer, whereas the lower layer neurons were used as *"special memory"* in which the target and the obstacle positions are stored.

Construction of an evaluation map mainly involves the values assignment to the individual system states, so that the route planning then consists, at each step, in finding the neighboring state that has a higher value than the current state in the resistive grid in which the state space is divided into a set of small n-dimensional cubes. Each cube corresponds to a node of the resistive grid. The method consists of finding the potential distribution within the grid, in which the nodes are represented by neurons that receive input from their "*m*" neighbors, and from one neuron in a "spatial memory". The path found is not necessarily the shortest one, however, it is smooth and avoids the obstacles. The method is applicable to the higher dimensionality problems, but its hardware implementation is difficult, its resolution is poor, and the required computational time is relatively long, particularly when the workspace is large, [3].

This paper will present two methods for finding the feasible path for a mobile robot using neural networks: Dirichlet and Neumann based boundary conditions. Section II presents the neuronal and workspaces for the proposed path planning method. This is followed by the description of network dynamics in Section III, and by the considerations of relationship between the potential field and boundary conditions in Section IV. Thereafter, in Section V, an overview about some priority rules is given. In Section VI and VII the construction of the knowledge of the working world and the computer simulation results are described.

II. Neuronal Representation of Robot Workspace

In the following, the collision-free path planning of a mobile robot moving in a workspace cluttered with stationary and/or moving obstacles is considered. Such planning requires that the robot's workspace to be configured. This implies that each region in the space should have its own configuration. For instance, obstacles can be defined as the regions in the workspace that should not be crossed by the robot, while the trajectory of the robot is the set of regions that should not contact the restricted areas. Although the workspace can take any shape, a rectangular space is implemented here to simplify the simulation. The mobile robot can move in a determined area of a single horizontal plane, divided into small squares to form a matrix of blocks *(X, Y)*, each block representing a single point reachable by the robot. Further, each block will have its own neural representative state, i.e., "0" if occupied with an object and a positive value other wise.

That said, the problem of path planning can be considered as building an evaluation map that determines the priorities for the moves from a present block to a neighboring one. The collection of all the blocks will describe the real working environment. Associated with each block, a state that describes the current status (i.e. states of "0" value are obstacles while those of "≤ *k*" value are either a destination/start point or an empty block where *k* is any positive real value) as will be stated later.

A topology of neurons is established to represent the robot's workspace with the same dimensionality. The path finding approach taken here is based on the Laplacian method [3], where the gradient of different states will be used to simplify the search for a feasible robot path. The neural space consists of *"n"* neurons distributed in two-dimensional space topology. Each neuron is being connected to a set of neighbors $N = \{n_1, .., n_m\}$ connected by a synaptic resistor (link), Fig. 1.

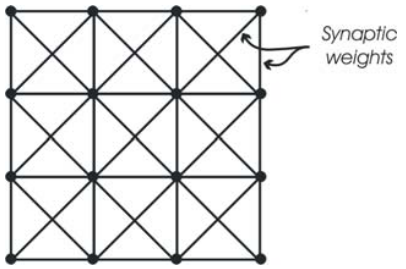


Fig. 1 4 by 4 network

For such topology, each neuron will accept a sum of inputs that will be processed by a transfer function resulting in some output. The outputs of individual neurons depend on:

- State value of the neighboring neurons.
- Number of neighbors included.
- Value of synaptic weight of each link.
- Transfer function of the neuron.
- External input bias *I*.

Each neuron receives inputs from its neighbors according to the value of the synaptic weight, and external input from the workspace topology, called the bias or threshold (*I*). The weighted sum of the inputs to the neuron is passed through a transfer function to generate its output. Accordingly, the network consists of two layers: the workspace nodes as external inputs from the workspace, and the neural space nodes, Fig. 2.

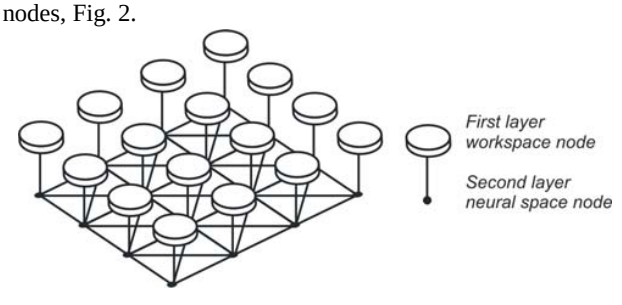


Fig. 2 Two-layer neural network

III. Dynamics of the Network

Denote the output of each neuron by the state variable σ_i, *(i=1... n)*, where $\sigma_i \in [0, k]$. The total input $v_i(t)$ for neuron *i* can be determined from the following relation:

$$v_i(t) = \sum_{j=1}^{m} w_{ij}\sigma_j(t) + I_i \quad (1)$$

where, m is the number of neighboring neurons and I is the external input bias or threshold to the neuron i.

The strength of the synaptic weight from the neighboring neuron j to neuron i is represented as w_{ij}. The w_{ij} values are not symmetric, i.e., $w_{ij} \neq w_{ji}$. Furthermore, the neighboring neurons participate in equal shares to the input of neuron i, that is:

$$w_{ij} = \frac{1}{m} \quad (2)$$

where, $j \in (1, ..., m)$.

This implies that if the neighbors of a neuron i are 8, then m=8 and $w_{ij} = 0.125$, while at the vertices of the robot workspace, these values will differ. For instance, Fig. 3 presents a corner where $w_{ij} = 1/3$. This applies for any marginal neuron in the network. This condition should be kept to avoid any local extreme that may occur when summing the inputs to any neuron as will be illustrated later.

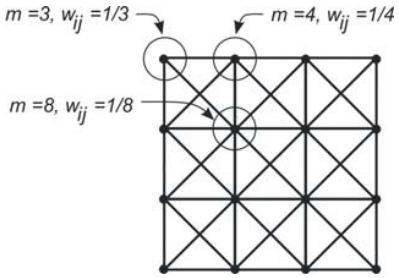


Fig. 3 Synaptic weight representation

For discrete time dynamics, the states of the neurons can change according to the relation.

$$\sigma_i(t+1) = g(\sum_{i=1}^{N} w_{ij}\sigma_j(t) + I_i) \quad (3)$$

where $g(.)$ is a threshold function and t is the time parameter, with a unit delay operator for each iteration, Fig. 4.

The total input $v_i(t)$ for any neuron generates the following output:

$$\sigma_i(t) = g(v_i(t)) \quad (4)$$

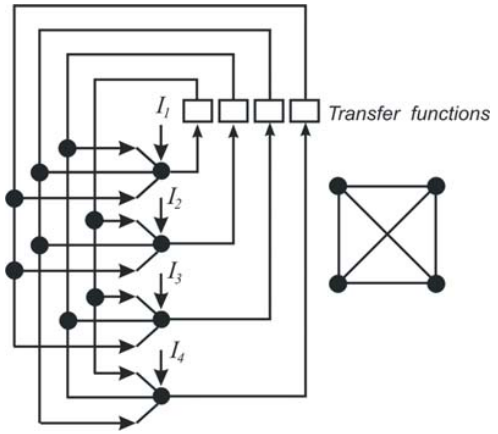


Fig. 4 Architectural graph for a network with 4 neurons

IV. Potential Field and Boundary Conditions

Assume that the robot destination point as a single and unique maximal source of attractive potential in the net. The robot, wherever located, should be guided by the attractive potential to the maxima (i.e. to the destination points). This potential of attraction is represented by the gradients of the field power that flows from the obstacles toward the target according to some boundary conditions. This is analogous to the heat transfer problem, where the destination is considered as a source that dissipates the heat within the net, the potential gradients will guide the robot safely to the destined location while avoiding any object that may hinder the robot. The representation of the state variables of a robot workspace is illustrated in section VI.

Hence, the above situation can be described using the two-dimensional heat equation:

$$\frac{\partial u}{\partial t} = c^2\nabla^2 u = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \quad (5)$$

where the temperature distribution is given by $u(x, y, t)$, $c^2$is the thermal diffusivity, and $\nabla^2 u$ is the Laplacian of u. Because of steady heat flow, i.e., no change in heat source over time, this implies that

$$\frac{\partial u}{\partial t} = 0, \quad \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad (6)$$

The above equations are now considered in a region **R** of XY-plane for a given boundary condition on the boundary curve C of **R**. Two cases will result according to the following boundary conditions, those are: the Dirichlet problem if u is prescribed on C, or the Neumann problem if the normal derivative $u_n = \frac{du}{dn}$ is prescribed on C.

The Dirichlet boundary condition states that σ_i, which represents the heat value u at node i, is known at specific points, where these points are expected to be the targets and obstacles, i.e.,

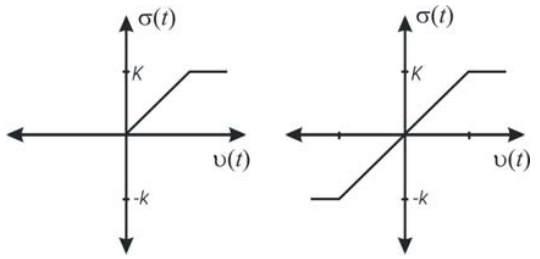
$\sigma_i(t) = 0$ if neuron *i* is associate with an obstacle.

$\sigma_i(t) = k$ if neuron *i* associated with a destination.

These two conditions should stay true the entire time to conform to the above differential equation, and to avoid the occurrence of any local heat extremes except that of the robot target. To keep the value of σ_i equal to zero if neuron *i* is associated with an obstacle, the external input *I* should be equal to (*-k*) while considering the transfer function shown in Fig. 5 (a), where *k* is any real positive value.

Now σ_i that refers to an obstacle existence will always have a value of zero. Correspondingly, to have σ_T (target state of the destination point) equal to *k*, the external input *I* should have the value of (*+k*). These assumptions will show a potential flow toward the target, starting from the obstacles.

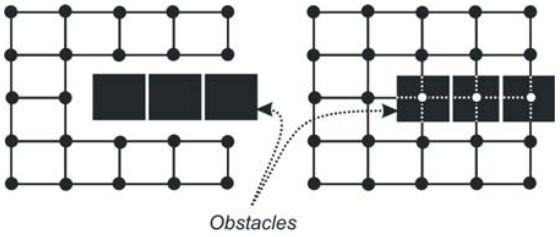
In contrast, Neumann boundary condition does not include the neurons that are associated with occupied obstacles. In other words, the synaptic weights that connect to the neighboring neurons have to be modified to conform to the previous assumption defined by equation (2). Besides, the destination point should have constant state variable with a value of (*+k*), while the neuron associated with the robot position of a state value of (*-k*), Fig. 6. To implement the above assumptions, the external input bias from the workspace to the destination should be equal to *+2k* the entire time and the threshold term for the robot position should be *-2k*. The activation function for the Neumann case is shown in Fig. 5 (b).



(a) (b)

Fig. 5 (a) Typical transfer function for Dirichlet boundary condition, (b) Typical transfer function for Neumann boundary condition.

Assigning a value of +k to the destination state variable and a value of –k to the current robot location state variable increases the margins the distributed attractive and repulsive potentials, that would considerably reduce the problem arising form small numerical gradients. Consequently, we should emphasize that the value of *k* has a great effect on the efficiency of the network. For instance, the larger the value of *k*, the more field diffusion attained over time. In other words, the value of *k* refers to the power of the potential source as well as the strength of the field absorption at the destination or obstacles. Finally, higher *k* values will help avoid the influence of small gradients that are far away from the potential source.



(a) (b)

Fig. 6 (a) Disconnection in synaptic weights takes place between the empty nodes and obstacle-occupied nodes for Neumann boundary condition, (b) No disconnection occurs in Dirichlet boundary condition while the state variables of the occupied nodes are set to equal zero by the addition of an input threshold (*I*) which equals to *-k*

V. Priority Rules of Step Selections

When solving a problem using any of the above two methods and starting from any initial point, some rule should be selected for determining the next robot jump (step). A traditional way, based on Dirichlet boundary condition, is to select the neighboring state having the highest value as the winning state. This rule could result in relatively hard jumps and sharp curves along the planed path, specially when square movements are allowed.

Below we present a step priority rule that calculate the proclivity at any point:

$$slope(x, y) = \frac{\sigma(x, y-1) - \sigma(x, y+1)}{\sigma(x-1, y) - \sigma(x+1, y)} \qquad (7)$$

Equation 7 depends on the values of the neighboring state variables, where *x* and *y* are used as indices for neuron *i* in a squared topology. For instance, a step priority rule of the following class can be used:

$$\begin{cases} if\ \sum_{j \in N} slope\ (n_j)/m = 0 \Rightarrow diagonal\ move \\ otherwise \qquad\qquad\qquad \Rightarrow orthognal\ move \end{cases}$$

In any case, the function *slope* refers to the field gradients where the priority rules are used to determine the winning states and moves. After determining a feasible path starting from an initial point P_0 and ending at a final point P_f, the length of the path can be calculated using the following summation:

$$L = \sum_{i=0}^{f} E(p_i, p_{i+1}) \qquad (8)$$

where *E* is the Euclidean distance between P_i and P_{i+1}.

VI. Modeling of Robot's Workspace Topology

The main objective of the workspace partitioning is to help construct a topology of the real space in which the robot is moving. The mobile robot will memorize its start position relative to the destination, while a scanning camera will feed

the robot with the required workspace patterns. These patterns should describe the workspace of the robot in a discrete manner as illustrated in earlier section.

The frequent update of the robot workspace will keep the robot's configuration known and will record the changes in the robot world. The patterns that describe the workspace will be fed to the robot processor. The processor in turn will build the evaluation map topology and based on the current destination point and obstacles, a path will be planned as stated in the network dynamics. The robot then is allowed to move a step or more ahead. After the movement is done another input pattern should be processed again taking in consideration the new changes in the workspace.

The above scheme is considered to be feasible if the following conditions are met the entire time:

- Obstacles speed < Robot speed.
- Initial nearest robot-to-obstacle distance > Robot speed× Pattern processing and computing time.

The first condition is a must in the presented approaches in this paper, because the robot will not be able to avoid obstacles that can move faster than the robot speed. The second condition is essential for the startup of the path planning. Moving obstacles should not have the chance of getting collided with the robot while the robot is computing and planning the next move. This condition does not add any constraints to the approaches presented here since information processing times are fairly short as illustrated in the experimental simulation.

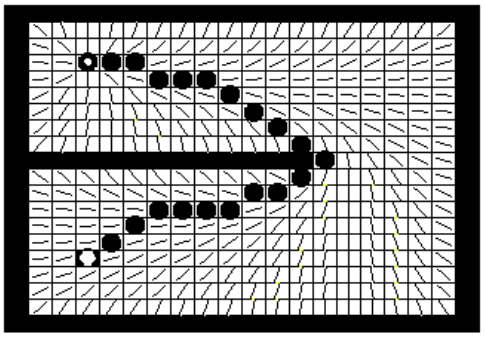
As an example, if a robot moves as fast as 0.2m/s in speed and if it takes 0.1 sec to find out a next move then, hypothetically, the robot should not be closer than 0.02m in distance from the moving obstacles upon startup. Our suggestion for solving this problem is to keep the time intervals as small as possible between the runs and pattern updating.

One of the significant contributions over such related work is the new transfer functions used which enhanced the ability to avoid small gradients of the state variables. In addition, the new approach used in deciding the robot future steps does not rely on hierarchical methods and therefore the path search span is small compared to others and the time required to find the path is relatively short. The presented approach is an intelligent decision making engine, which takes in consideration all the surrounding static and moving objects that are covered with the sensing zone. Addition to above, there are no restrictions on the shape of the obstacles nor the speed of the moving objects as long as their speed is lower than that of the robot.

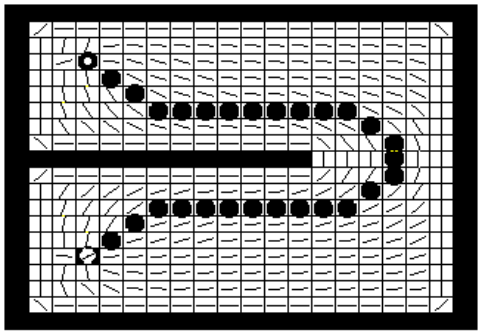
Target tracking is another achievement that can be held here, i.e. the target point is not necessarily static. The target is allowed to move with a speed less than that of the robot. Since the target is considered as a source of the distributed field within the net, it can be easily tracked if adequate and frequent patterns are fed to the robot processor.

VII. Simulation

To demonstrate the efficiency of the proposed methods, different simulation experiments have been conducted. The first experiment, shown in Fig. 7, the robot has to reach the target through a collision-free path within a room with a wall in the middle as an obstacle. It can be shown that for Neumann boundary condition the neural network has a higher convergence, i.e. it provides, for the same number of iterations, a higher performance than the Dirichlet boundary condition case. This is due to the existence of two attractive/repulsive sources that are diffusing at the same time in Neumann case. Besides, in Nuemann case, 20 iterations were enough for the robot to avoid collision with the existing obstacles, while in the Dirichlet case, the potential will not reach the start point by the same number of iterations since the minimum number of iterations required should be equal to at least the number of nodes passed by the path, as stated in relation (8). Furthermore, the simulation results show that for such space using Neumann boundary condition, fifty iterations are sufficient, whereas for Dirichlet condition the path obtained after fifty iterations is still not as smooth as in the Neumann case.



(a) The generated path obtained after 50 iterations with Dirichlet boundary condition



(b) Path generated with Neumann boundary condition after 50 iterations

Fig. 7 Path planning in a room of 20*20 blocks with a wall in the middle. Note the hollow square is the starting point

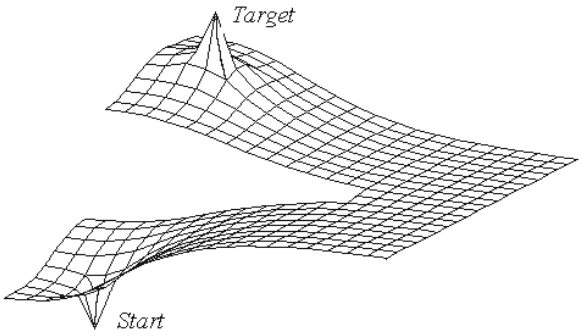
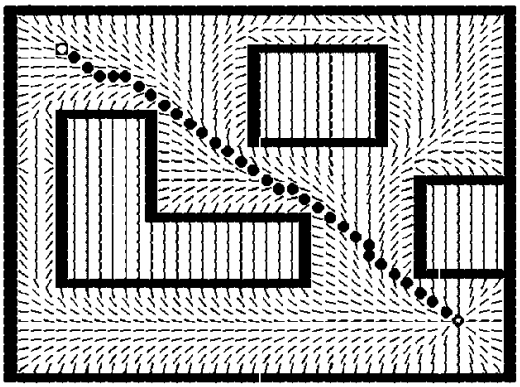


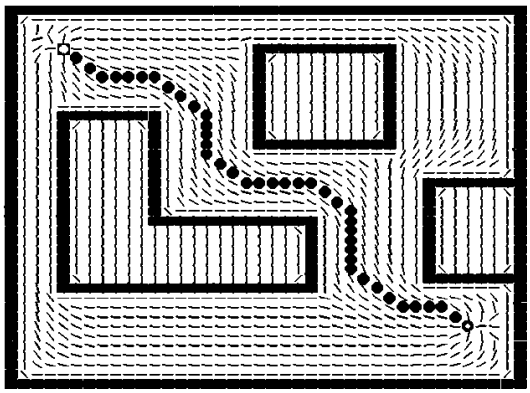
Fig. 8 State variables distribution in the network for the example shown in Fig. 7-b, (Neumann case). The two local extremes represent the target and the robot start points respectively

Regarding the potential flow, Dirichlet boundary condition shows a flow starting from the obstacles in a normal direction towards the unique maximum (target), therefore the robot will first try to move away from the obstacles. However, in the case of Neumann boundary condition, the flow has a parallel direction with respect to obstacles which forces the robot to move toward the wall until the flow direction is parallel to the wall , until then it will move in parallel to obstacles. This may increase the collision risk with the walls because of small tolerances maintained between the objects and the robot path.

A unique property of Neumann case is that at external corners, when the robot tries to turn around, the tolerance is high enough. Dirichlet boundary condition, however, forces the robot to turn tightly close around the obstacle if insufficient number of iterations is allowed. Finally, it was noted that the path followed by the two methods is not necessarily the shortest, rather, it can be considered as a smooth path.



(a) Path generated with Dirichlet boundary condition after 500 iterations



(b) Path generated with Neumann boundary condition after 500 iterations

Fig. 9 Path planning in a 40*40 room

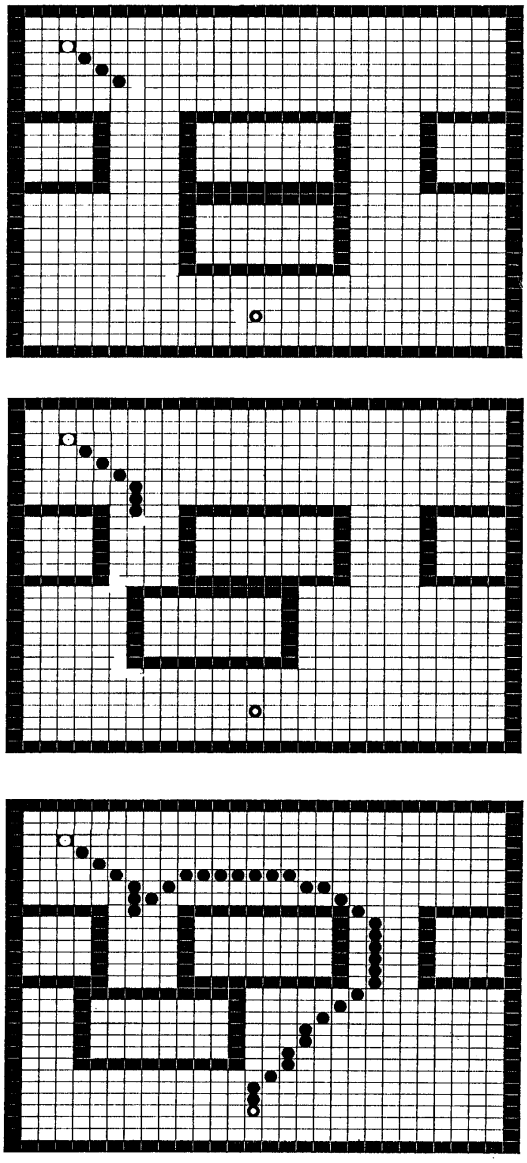


Fig. 10 Dirichlet case

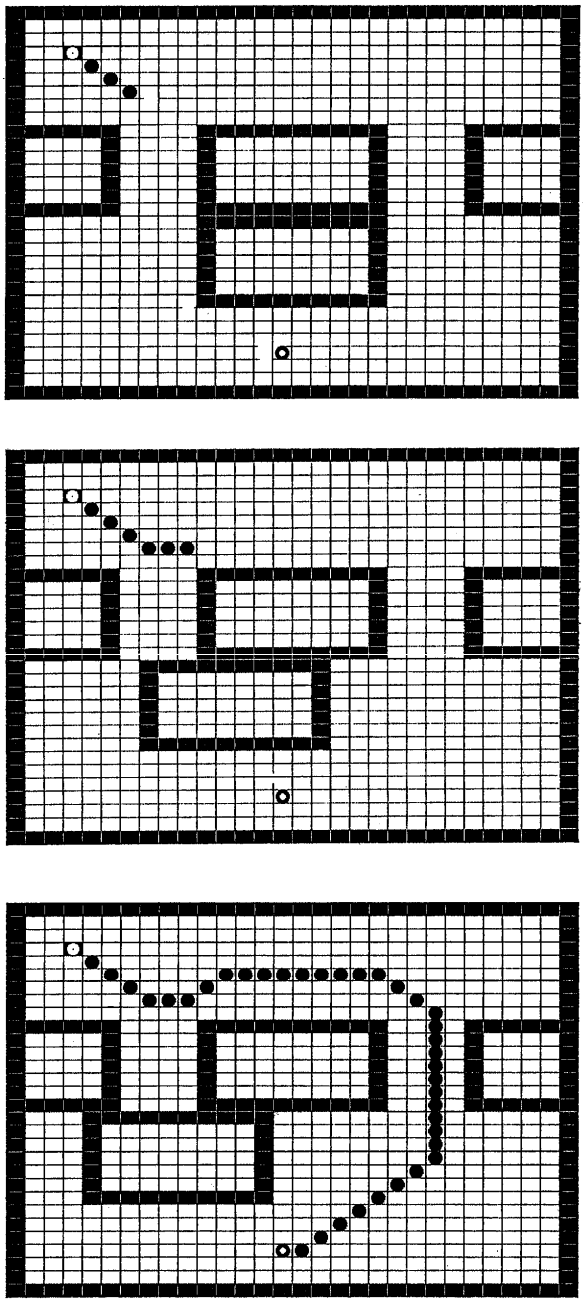


Fig. 11 Neumann case

Another example presented in Fig. 9. It shows the deference between the paths of the two boundary conditions. Here, Neumann boundary condition tries to force the robot to move ahead in parallel to the walls. In other words, the walls and the obstacles located in the working space are the guides for the robot path control.

Fig. 10 presents an example for a dynamic workspace. The bottom block is moving to the left with a speed of 3 steps per unite of time while the robot can update its information every three steps and can move as fast as the speed of the obstacles.

The first snapshot in Fig. 10 shows the first three steps that will be presented before the second update considering Dirichlet boundary condition. The second snapshot represents the next input pattern that includes three additional steps of the block towards the lift and three additional trajectory steps. Note since there is a small opening still found the robot would go down toward the moving bock. The third view shows the remaining path based on the third pattern update. In this configuration, the robot had to change its direction seeking for another path choice.

The same example was used for the Neumann boundary condition, as illustrated in Fig. 11. The result was the same for the first 3 jumps. However, the second pattern update shows a different path. In Neumann case the robot could guess the right path faster than Dirichlet case where it could predict the path more precisely as compared to Dirichlet.

VIII. Conclusion

This paper presented two different approaches to mobile robot path planning in an iterative manner. The two methods demonstrated successful trajectory planning of mobile robots in both stationary and dynamic environments. The ability to spread the heat potential around the destination point in Dirichlet boundary condition along with the new step priority rules helped find a collision free path. For the Neumann boundary condition case, the implementation of two source-to-destination field diffusion decreases the time and the required number of iterations to achieve convergence to a feasible collision free path. The planned paths stay well away from the obstacles as illustrated by the conducted experiments.

Dirichlet case is preferable when sufficient safety distance is required; however, it necessitates extra iterations to ensure good results, particularly at external edges and corners. In contrast, Neumann boundary condition provides safe turn-around at external edges with less computational times.

The most significant contribution of such work over the related effort is the use of a second order equation of the heat flow to model the robot attractiveness/repulsiveness to/from the destination/obstacles combined with topologically distributed neural network. In particular, the new transfer functions used to enhance the ability to evade small gradients of the state variables, the ability to plan the robot path in a dynamic environment, and the ability to track a moving target point.

The combination of the two boundary conditions can be a valuable future extension where more short and safe paths can be found. Solving the second order differential heat equation in a continuous form over a continuous boundary conditions, and obstacle geometries will be a important future extension of such problem.

References

[1] Ashiru I. and Czarnecki C. “Optimal Motion Planning for Mobile Robots Using Genetic Algorithms,” IEEE International Automation and Control Conference, pp. 297-300, 1995

[2] Brink Ten, C. and Popovic D. “A Collision-Space Approach to Trajectory Planning of Coordinated Robots,” 1995 IFAC World Congress, San Francisco, Vol. A, pp. 205-209.

[3] Bugmann G., Taylor G., and Michael, J. “Route Finding by Neural Nets, Application of Modern Heuristic Methods,” University of Plymouth PL4 8AA, United Kingdom, pp 1-11, 1994.
[4] Chen C. and Hwang. K. “Practical Path Planning Among Movable Obstacles,” 1991 IEEE International Conference on Robotics and Automation, pp. 444-449.
[5] Christoph M., McNeil S., and Thorpe C. “Side Collision Warning Systems for Transit Buses,” IV 2000- IEEE Intelligent Vehicle Symposium.
[6] Clifford, A. and Gregory, M. “A Real-Time Robot Arm Collision Avoidance System”, IEEE Transactions on Robotics and Automation, Vol. 8, No. 2, pp 149-160, 1992.
[7] Devin B. and Mason M. 2000. “External Trajectories for Bounded Velocity Differential Drive Robots,” (ICRA '00) IEEE International Conference on Robotics and Automation.
[8] Glasuis R. Komoda A., and Geilen S. “Neural Network Dynamics for Path Planning and Obstacle Avoidance,” Department of the medical physics and biophysics, University of Nejimegen, The Netherlands, pp 1-14, 1994.
[9] Gordon A., John P.H. and Rossmiller K. “Predicting Trajectories Using Recurrent Neural Networks,” ANNIE'91 Artificial Neural Networks in Engineering conference, pp 365-370.
[10] Mark. M. and Erdmann M. “Manipulation of Pose Distributions,” 2000 Tech Report, Computer Science Department, Carnegie Mellon University, CMU-CS-00-111.
[11] Matthew M., Pai D., Rus D., Taylor L.R., and Erdmann M. “A Mobile Manipulator,” (ICRA '99) IEEE International Conference on Robotics and Automation.
[12] Nagata S. Sekiguchi M. and Asakawa K. “Mobile Robot Control by a Structured Hierarchical Neural Network,” IEEE Contr. Syst. Mag. pp 69-76, 1990.
[13] Peterson K. “Path Planning in Analogue Valued Obstacle Array Using Hierarchical Dynamic Programming and Neural Networks,” ANNIE'91 Artificial Neural networks in Engineering Conference, pp 789-794.
[14] Roach W. and Michael N. “Coordinating the Motions of Robot Arm in a Common Workspace,” IEEE Journal, Vol. RA-3, No. 5, pp 30-37, 1987.
[15] Simon D. “Neural Networks-based Robot Trajectory Generation,” TRW Systems Integration Group, San Bernardino, CA 92402, pp 540-545, 1993.
[16] Simmons R., Fernandez J., Goodwin R., Koenig S., and O'Sullivan J. “Xavier: An Autonomous Mobile Robot on the Web,” IEEE Robotics and Automation Magazine, 1999.
[17] Popa, A. S. Popa, M. Silea, I. “Mobile robot navigation with obstacle avoidance capability,” 2008-IEEE 13th Power Electronics and Motion Control Conference, pp 1225-1232.
[18] Núñez P., Vázquez-Martín R., J. C. del Toro, A. Bandera, F. Sandoval. Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation. Robotics and Autonomous Systems archive, Vol. 56(3), pp247-264, 2008.