Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

On-Line Safe Path Planning in Unknown Environments

Chen Weidong, Fan Changhong and Xi Yugeng
Institute of Automation
Shanghai Jiao Tong University
Shanghai, 200030, P.R. China
changhongfan@sjtu.edu.cn, wdchen@sjtu.edu.cn, ygxi@sjtu.edu.cn

Abstracts **- For the on-line safe path planning of a mobile robot in unknown environments, the paper proposes a simple Hopfield Neural Network (HNN) planner. Without learning process, the HNN plans a safe path with consideration of " too close" or " too far" . For obstacles of arbitrary shape, we prove that the HNN has no unexpected local attractive point and can find a steepest climbing path, if a feasible path(s) exists. To effectively simulate the HNN on sequential processor, we discuss algorithms with $O(N)$ time complexity, and propose the constrained distance transformation-based Gauss-Seidel iteration method to solve the HNN. Simulations and experiments demonstrate the method has high real-time ability and adaptability to complex environments.**

I. Introduction

Planning a collision-free path to navigate a mobile robot through an unknown environment has received considerable attentions. The path planning methods [5,6] includes the graph-search, the roadmap and the potential field methods *etc*. The A* algorithm [5] is a typical graph-search method for path re-planning, and its time complexity is $O(N\log N)$ for an $\sqrt{N}\times\sqrt{N}$ grid map. The roadmap method [6] has fewer nodes than the grid map and can plan path very quickly. But the online reconstruction of the roadmap is time-consuming. The artificial potential field (APF) method, proposed by Kathib [8], is computationally advantageous, but introduces unexpected local attractive point.

Recently, Glasius [4] and Yang [13-15] respectively used the numerical potential fields (NPF) of generalized HNN to plan collision-free path in a dynamic, unknown environment. These HNN take parallel time complexities $O(N)$ to find feasible paths. But there're some problems to be analyzed for their method. 1) The completeness: [4,13-15] paid more attention to the stability of their HNN, and deemed the stability of the HNN would insure the finding of a feasible path if it exists, but ignore the analysis of the completeness. Unfortunately, we found HNN of [13-15] existed unexpected attractive point sometimes. 2) The effective application on sequential processors: because the outputs of those HNN nodes are very small (such as 10^{-100} Voltage), those HNN are still hard to be precisely realized by nowadays hardware technology, and can only be simulated on computers. A single sequential processor can do but pseudo-parallel simulation of the HNN. So in simulations on a sequential processor, continuous HNN of [4,13-15] become discrete HNN similar to Kassims' WENN [7], and also need $O(N^2)$ time to form feasible paths. Lagoudakis [6] used the "Raster Scan" (RS), a sequential scan, to accelerate the propagation of the NPF of Glasius' HNN. But the NPFs' propagation with RS is constrained by the obstacle nodes, and there is still much useless computing.

The safety of the planned path is nontrivial for navigating a mobile robot. Some methods for path planning try to find the shortest path. But the shortest path always clips the corners of obstacles and runs along the edges of obstacles, and this forms the "too close" problem [16]. Some other methods, such as the Voronoi diagram (VD) [5], plan path along the middle axis of the free configuration space. But the VD method always produces path that is far away from the shortest path, and forms the "too far" problem . For a static and known environments, Zlinsky [16] increased the cost of area nearing obstacles and used a distance transformation for the A* algorithm to search for the safe path, but the time complexity is still $O(N\log N)$. Muniz [10] used a feed-forward neural network to learning the planning of safe path, but the learning process is time-consuming. Yang [14] extended his HNN method to find a safe path in unknown environments. But as above discussion, method in [14] needs $O(N^2)$ time on a sequential processor, and the stability of HNN in [14] can't guarantee the elimination of unexpected attractive points.

We propose a simple HNN for safe path planning. This model has the following characters: 1) The neural nodes are locally connected; 2) The safety of the collision-free path is considered in the weight design of the HNN, and then no learning process is needed to keep the robot suitable distance away from obstacles. We give the condition for the existing of the feasible path, and then we prove the HNN can find a steepest climbing path if there is feasible path(s) and the HNN don't have unexpected attractive points, that means the method is complete. For effective application on a sequential processor, we use the constrained distance transformation (CDT) [11] information to guide the asynchronous iteration of the HNN and can plan a safe path in $O(N)$ time.

Section II describes the HNN. The stability of HNN and the condition for the existing of feasible path(s) are analyzed in Section III; Section IV analyzes the safe path planning. The effective solving of the HNN is given in Section V, and Section VI provides simulations and experiment. We summarize conclusions in Section VII.

II. The HNN model

The 2-dimensional environment may be unknown initially, and obstacles are not assumed to be convex. The robot R has only local sense ability and can incrementally construct the map of the environment. The configuration space [9] of the mobile robot R is C; obstacles are represented by O_i, i=1,2,..,m, which are sets of the unreachable points in C; $R(q)$ represents the robot R at the point $q \in C$. So the unreachable set is $\cup O_i^{'} = \cup\{q \in C \mid R(q) \cap O_i \neq \Phi\}$, and the free configuration space is $C_{free} = C \setminus \cup O_i^{'}$.

Fig.1 illustrates Net, the locally connected neural network similar to that in [7]. Each neural node in Net represents a configuration point in the discrete configuration space C. According to the connectivity between the neighboring points in C, each node of Net at most is connected with $2d$ neighboring nodes. If d=2, the nodes are 4-connected; and if d=4, the nodes are 8-connected. We select d=4 in following simulations and experiments. The robot makes decisions at discrete time, T=1,2,.., and moves along the steepest climbing path, if it exists, to reach the target.

We assume the number of nodes in Net is N for a $\sqrt{N} \times \sqrt{N}$ grid map. The output of the ith node is $x_i(t)$ for $1 \le i \le N$, and the 8-connected neighboring set of the ith node is NE_i. The target node is $Tset$, $1 \le Tset \le N$. And only the $Tset$ node has external constant input I>0.

The dynamic equation of the ith node is

$$\dot{x}_i(t) = \begin{cases} -Ax_i(t) + D_i(T)y_i(t), & if\ i \neq Tset \\ -Ax_i(t) + D_i(T)y_i(t) + I, & otherwise \end{cases} \tag{1}$$

$$y_i(t) = m \sum_{j \in NE_i} \omega_{ij} x_j(t) \tag{2}$$

Equations (1) and (2) form a linear system. In (1), A>0 is the negative feedback gain. $D_i(T)$ represents the accumulated information about the environment at discrete time T. If the ith node is occupied by obstacle, then $D_i(T)$=0; otherwise $D_i(T)$=1.

In (2), m is a positive gain coefficient. The connected weight from the node j to node i is $\omega_{ij} = \omega_{ij0}\,\omega_{ijs} \in (0,1]$. ω_{ij0} represents the factor of the Euclidean distance between nodes i and j, and we select $\omega_{ij0} == (2md/A)^{0.414} < 1$ (in following theorems we asks $2md/A$<1) for the diagonal 8-connected neighboring nodes, while ω_{ij0}=1 for the 4-connected neighboring nodes. Additionally, $0 < \omega_{ijs} \le 1$ represents the safety factor from the jth node to the ith one, and is given in Section IV. Because the safety factor from j to i may be different that from i to j, so we don't assume the connected weights are symmetrical, and this is different from [4,13-15].

The system equation of Net is rewritten in

$$\dot{x}(t) = -Ax(t) + G(T)x(t) + U \tag{3}$$

where only the element of U corresponding to $Tset$ is I, and the other elements of U are zeros..

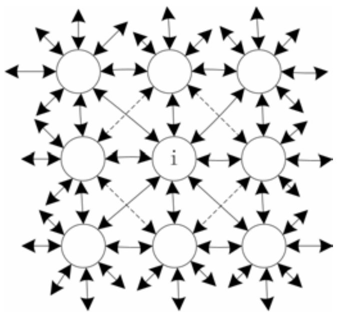


Fig.1 8-connected HNN of the 2-dimensional environment

III. Stability and non-negative of the HNN

Glasius [4] and Yang [13-15] respectively used the Liapunov method to prove the stabilities of their models. For the simple HNN model (3), The M-matrix [2] is introduced to elegantly prove the global exponential stability (GES) of (3) and the non-negative property of the HNN's equilibrium point which describes whether one node has feasible path or not.

Definition 1 [2]: The $n \times n$ matrix $Q=\{q_{ij}\}$ is a Z_n-matix, if q_{ii}>0 and q_{ij} is not larger than 0 for $i \neq j$.

Lemma 1 [2]: the Z_n-matrix Q is a nonsingular M-matrix, is equivalent to:

1) There exists positive diagonal matrix D, such that he matrix QD is a strictly diagonal dominant matrix and the diagonal elements of QD is larger than zero;
2) The real part of each eigenvalue of Q is positive;
3) Q is nonsingular and $Q^{-1} \ge 0$;

Theorem 1: If A>$2md$, then the system (3) has only one equilibrium point x_e and is GES.

Proof. The system matrix of the linear system (3) is $(-AE+G(T))$, where E is the $N \times N$ unit matrix. It can be easily known that the sum of elements in every row (column) of $G(T)$ is smaller than $2md$. By the definition 1, if D is an unit matrix and A>$2md$, $(AE-G(T))D=(AE-G(T))$ is a strictly diagonal dominant Z_n-matrix. By 1) of *Lemma* 1, $(AE-G(T))$ is nonsingular M-matrix, and then the real part of every eigenvalue of $(AE-G(T))$ is larger than 0 by 2) of *Lemma* 1, that is the real part of every eigenvalue of $(-AE+G(T))$ is smaller than 0. According to the linear system theory, system (3) has only one equilibrium point x_e and is GES. The proof is complete.

Theorem 2: If A>$2md$, then (a) the equilibrium point satisfies $x_e \ge 0$; (b) x_{ei}=0 for any node i that don't have feasible path to the target node $Tset$; and x_{ei}>0 for any node i that has feasible path to the target node $Tset$.

Proof. (a) Because the equilibrium point x_e of the system (3) satisfies $\dot{x}_e = 0$, we have $(AE-G(T))x_e=U$. By theorem 1, if A>$2md$, then $(AE-G(T))$ is nonsingular M-matrix of Z_n, and $x_e=[AE-G(T)]^{-1}U$. Then by 3) of *Lemma* 1 and $U \ge 0$, we have $[AE-G(T)]^{-1} \ge 0$ and $x_e=[AE-G(T)]^{-1}U \ge 0$.

(b). If there is no feasible path between the *i*th node and *Tset*, we separate all the nodes that have feasible paths to the *i*th node from the system (3), and a new linear system forms. By theorem 1, this new system is also GES. And more, it can be easily known that this new linear system has no external input, and so x_{ei}=0.

Otherwise, we suppose that *Tset* has a feasible path which passes the non-obstacle nodes i_1, i_2, i_3,... to reach the *i*th node. Firstly, from (a), *Tset* gives a positive excitation to the node i_1, and more all other excitations to i_1 are not smaller than 0, so it is immediate that the stable state of the node i_1 is larger than zero. Analogically, all nodes on the feasible path between *Tset* and the *i*th node have stable states that larger than zero. The proof is complete.

Form Theorem 2, we have the corollary:

Corollary 1: $x_{ei}>0$ is equivalent to that there is feasible path between the *i*th node and *Tset*.

In following, *Net*(*T*) represents the set of nodes that have feasible paths to *Tset* at time *T*. The stable state of the nodes in *Net*(*T*) are noted as $\bar{x}_e$, and $\bar{x}_e$ satisfies $\bar{x}_e > 0$.

IV. Path planning

This section firstly proves the NPF of the HNN doesn't have unexpected attractive point. So if there is feasible path(s), the NPF of the HNN must form a steepest climbing path from the starting node to the target.

1 The steepest climbing path

Similar to Glasius's and Yang's models, our model also uses the steepest climbing path of the NPF of the HNN to navigate the robot to the target, which needs that the NPF can't have local maximal attractive point at any node but the target node. In the following theorem, we analyze the local maximal property of the NPF formed by $\bar{x}_e > 0$, and show that the NPF has only one local maximal point that's just at the target point.

Theorem 3: If *A*>2*md*, the global NPF formed by $\bar{x}_e > 0$ has only one local maximal point, and this local maximal point must be at the target node.

Proof by contradiction. Assume that $\bar{x}_e$ has a local maximal point at the *i*th node, $i \in Net(T)$ and $i \neq Tset$. By (1), (2) and $\dot{x}_e = 0$, we know that

$$m \sum_{j \in NE_i} \omega_{ij} x_{e_j} = A\bar{x}_{e_i} > 0$$

$$0 < m \sum_{j \in NE_i} \omega_{ij} x_{e_j} \leq 2md \max_{j \in NE_i} \omega_{ij} x_{e_j}$$

then

$$2md \max_{j \in NE_i} \omega_{ij} x_{e_j} \geq A\bar{x}_{e_i} > 0$$

But by the precondition that *A*>2*md*, we know $\max_{j \in NE_i} x_{ej} \geq \max_{j \in NE_i} \omega_{ij} x_{e_j} > \bar{x}_{ei} > 0$,which means that there is at least one node in the neighborhood of the *i*th node has a larger output than the *i*th node. So this is in opposition to the assumption, and means that the NPF formed by $\bar{x}_e$ impossible has local maximal point at non-target node.

All elements of $\bar{x}_e$ are larger than 0 and the number of the elements of $\bar{x}_e$ is enumerable, so $\bar{x}_e$ has at least one global (and local at the same time) maximal point.

From the above two outcomes, it is immediately known that the global NPF formed by $\bar{x}_e > 0$ has only one local maximal point, and this local maximal point must be at the target node. The proof is complete.

Theorem 3 indicates that in the neighborhood of any non-target node *I* in *Net*(*T*), there is a node *j* having the largest output in node *i*'s neighborhood. We call the node *j* is the steepest climbing node of the node *i*. if $j \neq Tset$, node j also has a new steepest climbing node. If the steepest climbing nodes are selected one by one, a steepest climbing nodes sequence from the node i to the node *Tset* is formed. By contraries, the outputs of neighboring nodes of any node not in *Net*(*T*) are zeroes, so there doesn't exist the steepest climbing path for any node not in *Net*(*T*).

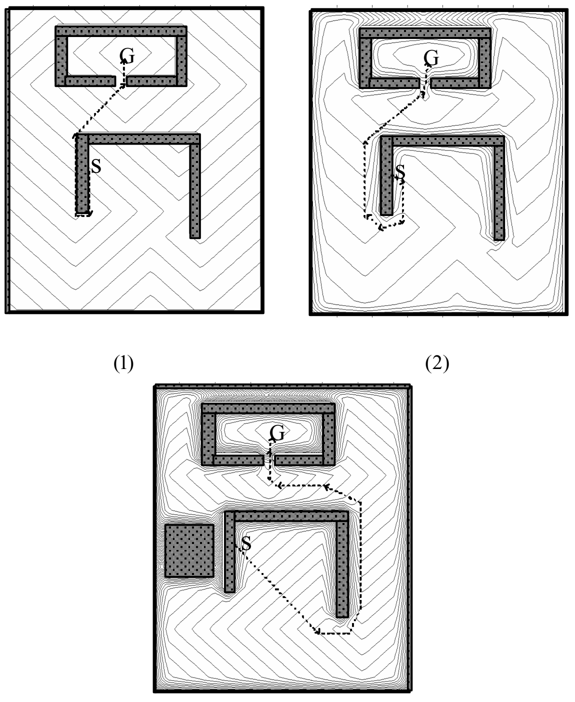


Fig.2 ▨ obstacle on; S: starting point; G: target point; (1) The contour of the NPF of the HNN without safety consideration and the shortest path; (2) The contour of the NPF of the HNN with safety consideration and the safe-path; (3) The new contour of the NPF of the HNN with safety consideration and the corresponding new safe-path

Notes: Model (3) can be extended to Yang's HNN [13] whose GES condition is A>0. But in simulations, if *A*>0 and

$A<2md$, the extended model has unexpected attractive points sometimes. So our analysis of theorem 3 is necessary.

2 Safe path planning

If $\omega_{ijs}=1$, the HNN (3) will also plan the shortest feasible path similar to [4,13,15]. This means that the HNN (3) doesn't consider the safety factor and implicitly optimize the length of the feasible path when $\omega_{ijs}=1$. Fig.2(1) illustrates the contour mapping of the equipotential surfaces of the NPF formed by $\bar{x}_e$ when $\omega_{ijs}=1$. The steepest climbing path search forms the shortest path from the starting node S to the target node G in Fig.2(1). Obviously the shortest path always clips the corners of obstacles and runs along the edges of obstacles.

From the theorem 3, if $\omega_{ijs}>0$, the NPF of $\bar{x}_e$ will forms the steepest climbing paths from any nodes that have feasible path to the target node. This means, that so long as we select $\omega_{ijs}>0$ then the NPF of $\bar{x}_e$ will not have unexpected maximal point. To consider the safe factor of the path, we decrease the weights of nodes that near obstacles to reduce the output of these nodes. A node is more near obstacles, the outputs of this node is smaller. When the robot is close to obstacles, the small weights ω_{ijs} will form repulsive potential filed to impulse the robot away from the obstacles. This kind of design of ω_{ijs} will not have unexpected local maximal point.

First, we compute the Euclidean distance (ED) of every no-obstacle nodes' to the nearest obstacle node by the Euclidean distance transformations (EDT) [12] in $O(N)$ time. For example, the ith nodes' ED is d_i. Then according to the sizes of the grid and the mobile robot, the safe distance limit D_{safe} is selected by the users: if $d_i<D_{safe}$, the safe factor is considered in the weight ω_{ijs}; otherwise, not considered. We select the following function to consider the safe factor:

$\omega_{ijs}=(2md/A)^{fs(d_i)}$, where the safe function $fs(d_i)$ satisfies that $fs(d_i)>0$ and $fs(d_i)$ is a non-increasing function with d_i, that is

$$fs(d_i)=\begin{cases} ks/d_i, if\ 0<d_i<D_{safe} \\ 0, otherwise \end{cases} \qquad (4)$$

where the parameter $ks>0$ can be adjusted.

By the above selection of ω_{ijs}, ω_{ij} will decrease very quickly while the node i is closer to obstacles, and as illustrated by Fig.2(2) the contour lines of the NPF of $\bar{x}_e$ near the obstacle is dense, so the robot will be repulsed away from obstacles while the robot still traces the gradient of the NPF to reach the target. Obviously, the safe-path in Fig.2(2) is longer than that in Fig2(1), but the safe-path is advantageous for the robot's safety and forwarding speed.

The safe-path in Fig.2(2) is still similar to that in Fig.2(1). In Fig.2(3), a new obstacle is added to the map of Fig.2(1), the shortest path is still feasible, but the shortest path must pass a strait channel, but the HNN plans a new safe-path in Fig.2(3), which is quite different from that in Fig.2(1) and Fig.2(2).

V. The effective solving of the HNN

Discussing the effective solving of the HNN is important when the HNN is simulated on a sequential processor. A single processor has only sequential computational ability, and can only do pseudo-parallel simulation of HNN. In simulations on a sequential processor, continuous HNN in [4,13-15] become discrete HNN, and need $O(N^2)$ time [7] to form feasible paths. Lagoudakis used the "Raster Scan" (RS), a sequential scan, to accelerate the simulation of HNN of [4], and this formed asynchronous HNN. But the NPF's propagation with RS is constrained by the obstacle nodes, and there are still many useless computing. We discuss how to effectively solve the equation $[AE-G(T)]x_e=U$ of HNN (3).

1) *The direct method*

Because each nodes of the HNN has only local connections to nodes in its neighborhood, the $N\times N$ matrix $AE-G(T)$ has at most $9N$ non-zero elements. So by the sparse matrix technology [17], $[AE-G(T)]x_e=U$ can be solved in $O(N)$ time. The direct method is advantageous for a known map. But the programming for solving sparse matrix is complicated, and the solving of the new equilibrium can't use the last equilibrium when new obstacles are added to the map.

2) *The indirect method——The iterative method*

Lagoudakis's "raster scan" [6] simulation turned the parallel HNN of [4] into a sequential or asynchronous HNN. The RS method could be considered a small integration step Gauss-Seidel iteration method (GSIM) [3], which is an efficient indirect method for solving linear system of equations. Compared the pseudo-parallel simulation on sequential processor, the GSIM accelerates the propagation of NPF quickly. But constrained by the obstacle nodes, after the GSIM is used once, the RS method can't propagate the NPF over all nodes that have feasible path to the target node, and there is still much useless computing.

We use the constrained distance transformation (CDT) [11] to decide the propagation order of the NPF, and then use the order to refresh the GSIM. The gist of this CDT-based GSIM (CDTGSIM) is that: firstly, the nodes are classified into different classes by their constrained connected distances (CCD) to the target node; and then the smaller CCD a class has, the priority the nodes in this class are calculated by the GSIM. For example, the node in the class with CCD=0 (that's just the target node itself) is firstly calculated by the GSIM; secondly the nodes in the class with CCD=1 are calculated by the GSIM; thirdly the nodes in the class with CCD=2 are calculated by the GSIM …, until all nodes that have feasible paths are calculated once. If the starting node can't be expanded by the CDT, then there is no feasible path ; otherwise repeating the CDTGSIM several times to approximate to x_e of the HNN (3).

The on-line CDTGSIM is detailed as following:

(1) The 4-connected constrained distances between all non-obstacle nodes and *Tset* are simply computed by the Breadth-First Search [18] in $O(N)$ time and then are classified into different classes by their distances to *Tset* in $O(N)$ time. Assume the largest 4-connected distance N_0, and the number of the nodes (include *Tset*) that have feasible paths to the target node is N_1. Set j_0=0, *repnum*=1.

If the starting node is not expanded by CDT, go to (5).

Merge all nodes in the N_0 number of classes into an increasing sequence $\tilde{x}$ according to their CDT values, and we notes the *i*th node in this sequence is $\tilde{x}^i$.

(2) GSIM is used for each node whose 4-connected distance to the target is j_0, e.g. for the *i*th node $\tilde{x}^i$ of $\tilde{x}$:

$$\tilde{x}^i(repnum+1)=\frac{m}{A}\Big(\sum_{j<i\cap j\in NE_i}\omega_{ij}\tilde{x}^j(repnum+1)+\sum_{j>i\cap j\in NE_i}\omega_{ij}\tilde{x}^j(repnum)\Big)$$

The above iteration formula means that the states of the nodes that have been iterated will be directly used by the successive nodes , and forms a asynchronous iteration.

(3) $j_0+1 \rightarrow j_0$. If $j_0>N_0$, go to (4); else go to (2)

(4) j_0=1, $repnum+1 \rightarrow repnum$. If *repnum* is smaller than pre-set iteration number, go to (2); else go to (6).

(5) There is no feasible path, so stop and go to (7).

(6) Search the steepest climbing path, and go to (7).

(7) In the next decision processing, if new obstacle information is obtained, then go to (1) to recalculate the iterative order of the CDTGSIM; else the iterative order of the CDTGSIM is unchanged and directly go to (2).

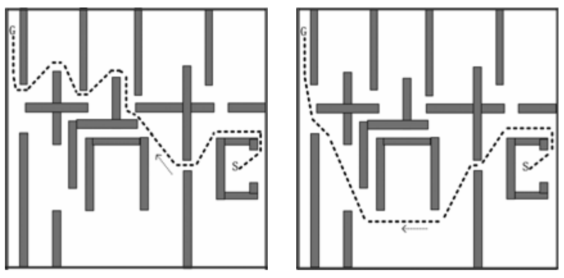
The time complexity from (2) to (4) is $O(N)$. So it is obviously that the whole process from (1) to (6) for a path re-planning in every decision period is just $O(N)$.

VI. Simulations and Experiments

In simulations, we assume: 1) the robots' perceptio n radius is 10 times of the length of the grid; 2) the HNN is computed twice by the CDTGSIM every discrete time with the new obstacle information.

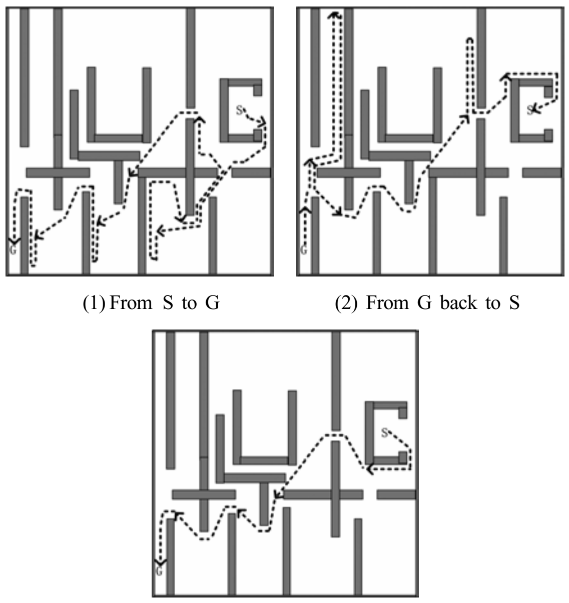
A Pioneer 2 mobile robot whose decision period is 0.1 second is used in experiments. The robot has 16 sonars and the effective detect ranges of these sonars are limited to 1.5 meters. The environment of 10× 10 square meters is divided into a grid map with 100× 100 grids. We use the histogamic in-motion mapping method [2] to decide whether a grid is occupied by obstacle or not. The obstacle distance transformation [12] is used to cut down the very narrow pass that can't be passed by the robot.

The processor for simulations and experiments is a 300M CPU with Linux OS. For a large grid map with 500×500 grids, the processor can do 30 times CDTGSIM per second.



(1) safe path with *ks*=5 (2) safe path with *ks*=10

Fig 3 The CDTGSIM for the HNN to plan safe-path



(1) From S to G (2) From G back to S

(3) From S to G again

Fig.4 The on-line path planning from S to G

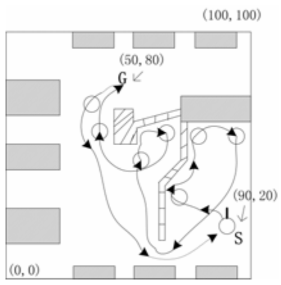


Fig.5 Experiment in the unknown environment: the real line gives the trajectories from S to G and from G back to S

1 The fast speed of the CDTGSIM

Within our experience, the CDTGSIM is very effective to solve the HNN for safe-path planning. In this simulation, the initial map is known and it has 500×500 grids. We select A=8 , m=1 , ks=5 and D_{safe}=5. After twice CDTGSIM, the

HNN plans a safe-path in Fig.3(1). The CPU time for this safe-path is less than 0.4s. A new safe-path is also planned for a larger *ks* =10 after twice CDTGSIM in Fig.3(2). The new safe-path is longer than that in the Fig.3(1), but the new safe-path is more smooth. The CPU time for the new safe-path is less than 0.4s.

The CPU time reported here is very faster than the results reported in [4,6,13-15], and shows that the CDTGSIM has high real-time ability for path planning.

2 Simulations under the unknown environment

Initially the robot doesn't know any obstacle information about the environment. After 283 times CDTGSIM, the robot reaches G the first time in Fig.4(1), and takes 312 times CDTGSIM back to S in Fig.4(2). And then the robot moves from S to G again with 133 times CDTGSIM in Fig.4(3). In all the round-trip, the robot doesn't clip the corners of obstacles or run along the edges of obstacles, and isn't trapped in unexpected local attractive point.

3 Experiment under the unknown environment

The forward speed of the real robot is proportional to the distances of forward obstacles, and is limited under 250 mm/s. Without initial information about the obstacles, the robots' spent much time to explore the enviro nment, and used 85 seconds on reaching G. Then with the grid map of the detected obstacles, the robot used less time, 30 seconds, to back to S.

VII. Conclusions

A HNN is used to plan safe path on-line. Without learning processing, the HNN can plan a safe-path that compromises between the "too close" and "too far" paths. For environments of arbitrary shape, the NPF of the HNN has no unexpected attractive points. Effective solving of the NPF of the HNN based on distance transformation is given. Simulations and experiments demonstrated the method has high real-time ability and adaptability to complex environments.

VIII. Acknowledgements

This work is supported by National high-tech research and development plan under grant 2001AA422140, by National Science Foundation under grant 60105005.

Reference

[1] Berman, A.*Nonegative matrices in the mathematical science.* 2nd ed. New York Academic, 1994

[2] Borenstein, J., Y. Koren, *The vector field Histogram --- fast obstacle avoidance for mobile robots.* IEEE RA, 1991,Vol.7(3):278-288

[3] Connolly, C.I. *et al*, *Path planning using Laplace's Equation*. ICRA, 1990,Vol.3:2102-2106

[4] Glasius, R., A. Komoda, *Neural network dynamics for path planning and obstacle avoidance*. Neural Networks, 1995, Vol.8(1):125-133

[5] Huang, Y., Ahuju, N., *Gross-motion planning*, ACM Computing Surveys, 1992, Vol.24(3): 220-291

[6] Lagoudakis, M.G., Maida, A.S. Neural maps for mobile robot navigation. 1999 Int. Joint Conf. on Neural Networks, 1999, Vol.3: 2011-2016

[7] Kassim, A., V. Kumar. *Path planners based on wave expansion neural network*. Robotics and Autonomous Systems, 1999, Vol.26:1-22

[8] Kathib, O., *Real-time obstacle avoidance for manipulators and mobile robots.* Int. J. Rob. Res. 1986,Vol.5:90-98

[9] Latombe, J.C., *Robot motion planning*. Kluwer Academic Publishers , 1991

[10] Muniz, F., *et al*, *Neural controller for a mobile robot in nonstationary environment*. the 2nd IFAC on Intelligent Autonomous Vehicles, Helsinki, Finland, 1995: 275-284

[11] J. Piper, E. Granum, *Computing distance transformations in convex and non-convex domains*. pattern Recognition, 1987, Vol.20, pp: 599-615

[12] Shih, F.Y., Liu, J.J., *Size-invariant four-scan Euclidean distance transformation*, Pattern Recognition, 1998,Vol.31(11):1761-1766

[13] Yang, S.X., M. Max, *Neural Network Approaches to Dynamic Collision-Free Trajectory Generation*. IEEE SMC Part B. 2001, Vol.31(3):302-318

[14] Yang, S.X., M. Max, *An efficient neural network method for real-time motion planning with safety consideration*, Robotics and Autonomous Systems, 2000, Vol.32: 115-128

[15] Yang, S.X., M. Max, *An efficient neural network approach to dynamic robot motion planning*, Neural Networks, 2000, Vol.13: 143-148

[16] Zelinsky, A., *Using path transforms to guide the search for findpath in 2D.* The Int J. of Robotics Research, 1994, Vol.13(4):315-325

[17] S. Pissanetzky, *Sparse matrix Technology.* New York: Academic, 1984

[18] Shaffer, C.A. *Data structures and algorithm analysis*. Prentice Hall Publishers, 1996