An Efficient Dynamic System for Real-Time Robot Path Planning

Allan R. Willms, Simon X. Yang, *Member, IEEE,*

***Abstract*—This paper presents a simple yet efficient dynamic programming (DP) shortest path algorithm for real-time collision-free robot path planning applicable to situations where targets and barriers are permitted to move. The algorithm works in real time and requires no prior knowledge of target or barrier movements. In the case that the barriers are stationary, this paper proves that this algorithm always results in the robot catching the target provided it moves at greater speed than the target, and the dynamic system update frequency is sufficiently large. Like most robot path planning approaches, the environment is represented by a topologically organized map. Each grid point on the map has only local connections to its neighboring grid points from which it receives information in real-time. The information stored at each point is a current estimate of the distance to the nearest target and the neighbor from which this distance was determined. Updating the distance estimate at each grid point is done using only information gathered from the point's neighbours, that is, each point can be considered an independent processor, and the order in which grid points are updated is not determined based on global knowledge of the current distances at each point or the previous history of each point. The robot path is determined in real-time completely from information at the robot's current grid-point location. The computational effort to update each point is minimal allowing for rapid propagation of the distance information outward along the grid from target locations. In the static situation, where both targets and barriers do not move, this algorithm is a DP solution to the shortest path problem, but is restricted by lack of global knowledge. In this case, this paper proves that the dynamic system converges in a small number of iterations to a state where the minimal distance to a target is recorded at each grid point and shows that this robot path-planning algorithm can be made to always choose an optimal path. The effectiveness of the algorithm is demonstrated through a number of simulations.**

***Index Terms*—dynamic system, path planning, collision avoidance, mobile robot, real-time navigation, dynamic environment, dynamic programming.**

I. Introduction

REAL-TIME collision-free robot path planning is a fundamentally important issue in robotics. There are many studies on robot path planning using various approaches, such as the grid-based A* algorithm [1], [2], road maps (Voronoi diagrams and visibility graphs) [3], [4], cell decomposition [5]–[7], and artificial potential field [8]–[10]. Some of the previous approaches (e.g., [11]–[14]) use global methods to search the possible paths in the workspace, normally deal with static environments only, and are computationally expensive when the environment is complex. Some methods (e.g., [8], [15]) suffer from undesired local minima, i.e., the robots may be trapped in some cases such as with concave U-shaped barriers. Seshadri and Ghosh [16] proposed a new path planning method using an iterative approach, which is computationally complicated, particularly in a complex environment. Li and Bui [13] proposed a fluid model for robot path planning in a static environment. Ong and Gilbert [17] proposed a new method for path planning with penetration growth distance, which searches over collision paths instead of the free space as most methods do. This method can generate optimal, continuous robot paths in a static environment. Oriolo et al. [18] proposed a method for real-time map building and navigation for a mobile robot, where a global path planning plus a local graph search algorithm and several cost functions are used.

Some neural network models (e.g., [19]–[22]) were proposed to generate real-time robot trajectories through learning. Ritter et al. [19] proposed a Kohonen's self-organizing-map based model to learn the transformation from Cartesian workspace to the joint space of robot manipulators. Li and Öğmen [22] proposed a neural network model for real-time trajectory generation by combining an adaptive sensory-motor mapping model and an on-line visual-error-correction model. Both models deal with static environments only and assume that there are no obstacles in the workspace. Zalama et al. [21] proposed a neural network model for mobile robot navigation, which can generate dynamic collision-free trajectories through unsupervised learning. This model is computationally complicated since it incorporates the vector-associative-map model and the direction-to-rotation effector control transform model [20], [21]. Fujii et al. [23] proposed a multi-layer reinforcement learning based model for robot path planning with a complicated collision-avoidance algorithm. The generated trajectories using learning based approaches are not optimal, particularly during the initial learning phase.

To avoid the time-consuming learning process, Glasius et al. [24] proposed a Hopfield-type neural network model for real-time trajectory generation with obstacle avoidance in a nonstationary environment. It is rigorously proved that the generated trajectory does not suffer from undesired local minima and is globally optimal in a stationary environment [15], [24]. However, these models [15], [24], [25] require that the robot dynamics be faster than the target and obstacle dynamics, and have difficulty dealing with fast changing environments [25], [26].

Inspired by the dynamic properties of the Hodgkin and

Manuscript received ***; revised ***. This work was supported by Natural Sciences and Engineering Research Council (NSERC), Canada.

Allan Willms is with Department of Mathematics and Statistics, University of Guelph, Guelph, Ontario N1G 2W1, Canada. Email: awillms@uoguelph.ca

Simon X. Yang is with Advanced Robotics and Intelligent Systems (ARIS) Group, School of Engineering, University of Guelph, Guelph, Ontario, N1G 2W1, Canada. E-mail: syang@uoguelph.ca.

Huxley's membrane model [27] for a biological neural system and the later developed Grossberg's shunting model [28], [29], Yang and Meng [26], [30] proposed a new neural network approach to dynamic collision-free trajectory generation for robots in dynamic environments. It removes the restriction on slow robot dynamics and is more general and more powerful than Glasius et al.'s model [24], [25], [31]. Yang and Meng's approach can be extended to various robotic systems [32]–[34], and a special case of its simple additive model is equivalent to Glasius et al.'s model [26]. In Yang and Meng's model [26], the dynamics of each neuron in the topologically organized neural network are characterized by a shunting equation or an additive equation. With only local neural connections, the target globally attracts the robot in the entire workspace, while the obstacles have only local effect to push the robot away from possible collisions. The real-time optimal robot trajectory is generated through the dynamic activity landscape of the neural network.

In this paper we present a very simple yet efficient dynamic system algorithm for real-time collision-free path planning of robots in nonstationary environments, which is motivated by Yang and Meng's neural dynamic model and Zelinsky's distance transform model [14]. Similar to most robot path-planning approaches, the environment is discretized and represented by a topologically organized map. Each grid point has only local connections to its neighboring grid points. Unlike the neural network based path-planning model proposed by Yang and Meng [26], [30], our algorithm efficiently propagates the distance instead of neural activity from the target to the entire robot workspace. Unlike Zelinsky's distance model where the transformed distance at each grid point is a function of target and obstacle locations in the entire workspace, making it suitable for static environments only, our algorithm is capable of dealing with changing environments — moving targets and obstacles. The real-time optimal robot path is generated through the dynamic distance landscape without explicitly searching over the global free workspace nor the collision paths, without explicitly optimizing any global cost functions, and without any prior knowledge of the dynamic environment. Therefore the algorithm is computationally efficient. Our algorithm has all the advantages from Yang and Meng's model. In addition, its dynamic distance landscape has a better physical meaning than the neural activity landscape, and it offers faster and better path planning, which we detail in later sections.

We emphasize that our algorithm uses only local information. In contrast, most other path-planning algorithms use global information in some way. For example, the A^* algorithm [1] estimates the distance from a node n to the goal as the sum of a locally propagated distance $g(n)$ from the starting point to n, plus an estimate $h(n)$ of the distance from n to the goal (a piece of global information). Even if A^* is implemented with a non-informed heuristic, $h(n) \equiv 0$, the algorithm still sorts the nodes in its open list in ascending distance order, which implies the algorithm has global knowledge of the current distance for each node. Artificial potential field methods typically set up repulsive fields around barriers and an attractive field centered at the target, specifying them at a given node in the space depending on the (global) distance from that node to the various barriers and targets. Our algorithm is essentially a dynamic programming (DP) algorithm applied to the shortest path problem in a cyclic network [35]. However, typical solutions to the shortest path problem from a DP perspective also make use of global information by stepping through the nodes in an order determined by the current recorded distances at each node or the order in which node values most recently changed.

In the static case, where barriers and targets do not move, the fact that we restrict ourselves to using only local information means that our algorithm is not as fast as it otherwise could be. However, we are primarily concerned with real-time robot path planning in the dynamic case where both targets and barriers can move, appear, or disappear. In this situation, we demonstrate that our algorithm performs quite well. If barriers are stationary but targets are allowed to move, we prove, under mild conditions, that the robot will reach a target.

II. The Robot Path Algorithm

In this section, we present the distance-propagating dynamic system and the algorithm for the robot, target and barrier movement. We also discuss the types of grids on which our algorithm can be implemented.

A. The Distance-Propagating Dynamic System

Suppose the robot environment is discretized into a grid of M points, labeled by an index, i, each point being either a free space or a barrier location. The targets and the robot may occupy any free space. We define $d_{\min}$ and $d_{\max}$ to be the minimum and maximum distances between any two adjacent neighbors in the grid, and we define B_i to be the set of free spaces that are adjacent neighbors to point i. The distance, d_{ij}, between any two free spaces i and j is defined to be the minimum Euclidean length of all paths joining i and j through nonbarrier adjacent neighbors. For example, consider a regular square grid as illustrated in Figure 1, which has a barrier at point 6. The neighbor set for the point labeled 7 in this figure and the distances from it to all other points are

$$
\begin{aligned}
B_7 &= \{2,3,4,8,10,11,12\} \\
d_{7j} &= \begin{cases} 1, & \text{if } j \in \{3,8,11\}, \\ \sqrt{2}, & \text{if } j \in \{2,4,10,12\}, \\ 2, & \text{if } j = 15, \\ 1+\sqrt{2}, & \text{if } j \in \{1,9,14,16\}, \\ 2\sqrt{2}, & \text{if } j \in \{5,13\}. \end{cases}
\end{aligned}
$$

Each grid point has an associated variable $x_i(n)$, a real value that records the "distance" to the nearest target at time step n. The system is initialized by setting the variables, $x_i(0)$, for all target locations to 0 and all other locations to some large maximal distance D ($D > (M-1)d_{\max}$ is sufficiently large.) The dynamic system then evolves as follows:

$$
x_i(n+1) = \begin{cases} 0, & \text{if } i \text{ is a target location}, \\ D, & \text{if } i \text{ is a barrier location}, \\ \min\left(D, f(i,n)\right), & \text{otherwise}, \end{cases} \tag{1}
$$

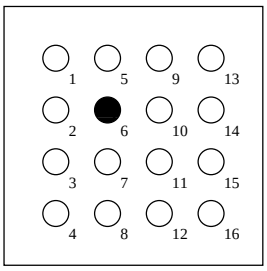

Fig. 1. A regular square grid with barrier at location 6.

where

$$f(i,n) = \min_{j \in B_i} \left(d_{ij} + x_j(n) \right).$$

Since the minimization is performed by searching over the neighbors of each point and each point has a finite number of neighbors, the computational burden for this distance-propagating dynamic system (the time required to update every grid point once) is proportional to the total number of points, M.

Consider first the static situation where the targets and barriers do not move. After one time step, the target locations will be zero and their adjacent neighbors will record the correct distance; after two time steps, the adjacent neighbors of the target neighbors will record the correct distance, and so on, so that the distance to each target location will spread outward by one grid step each time step. Note that in general a grid step is not always the same distance; for example, for a regular unit square grid it can be either 1 or $\sqrt{2}$. In consequence, the number of grid steps along two different paths between two points does not necessarily have the same ordering as the actual distance between these two points along these two paths. Each variable x_i will have the value D until some time step, say n_1, at which it will take on the value of the exact distance to the target that is n_1 grid steps away as measured along the route via which the distance information propagated. The variable x_i will remain at this value unless a target (possibly the same one) that is $n_2 > n_1$ grid steps away along some path, is actually a shorter distance away, in which case $x_i(n_2)$ will take on the value of the distance to this second target. In other words, the distance variable x_i can temporarily reflect the distance to the target that is the fewest grid steps away. See Figure 2 for an illustration. Once the number of time steps is greater than the number of grid steps from a point to each of the targets along the minimal distance paths, the value of x for that point will not change and is guaranteed to be the exact distance to the nearest target. In Section IV we give a more rigorous proof of this claim. For a regular square grid, no two points are more than $\sqrt{M}$ grid steps away from each other and hence the total number of times that f need be evaluated is no more than $M^{1.5}$. In the extreme case of a grid which is just a straight line of M points, the total number of evaluations of f would be no more than M^2.

In the situation where the targets are moving, the distance information continues to propagate from the targets outward at a speed of one grid step for each time step. And thus, in general, the value of x at any point is potentially N time steps out of date, where N is the number of grid steps from the point to the target the most grid steps away.

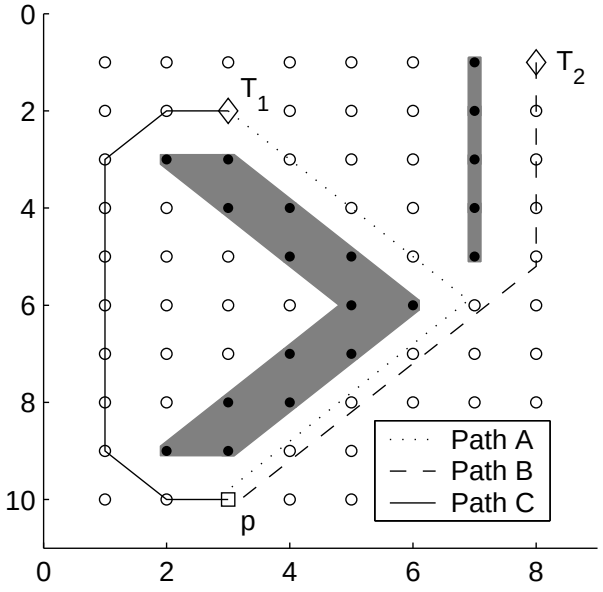

Fig. 2. Propagation of distance information. Barriers are shown as solid circles/shaded areas, free points as open circles, and targets as diamonds. For the point p (square): $x_p(n) = D$, for $n = 0, \ldots, 7$, $x_p(8) = 8\sqrt{2}$ giving the distance via path A to target T_1, $x_p(9) = 4 + 5\sqrt{2}$ giving the shorter distance to target T_2 via path B, and $x_p(n) = 8 + 2\sqrt{2}$, $n = 10, \ldots, \infty$, giving the shortest distance via path C to target T_1.

B. Robot Movement

We assume that the robot can move from any grid point to any neighboring grid point, that is, point robot dynamics. Constraints on robot movement could also be incorporated in this algorithm with a necessary increase in complexity.

The robot location, $r(t)$, is specified as an index of one of the points on the grid, and is a (discontinuous) function of real time $t \geq t_0$. Initially, $r(t_0) = i_0$. We assume that the robot's travel path is updated at a set of real time values $t_1 < t_2 < t_3 < \ldots$, and that the robot's actual location for $t \in (t_k, t_{k+1})$ is somewhere between the grid points $r(t_k)$ and $r(t_{k+1})$. At time t_k, the next update time, t_{k+1}, and next update location, $r(t_{k+1})$, are determined. The latter is defined as

$$r\left(t_{k+1}\right) = \mathrm{Ind}\left(r(t_k), n(r(t_k), t_k)\right), \tag{2}$$

where $n(i, t_k)$ means the highest time step n for which x_i has been computed up to time t_k, and $\mathrm{Ind}(i,n)$ is the index of the closest neighbor through which the value of $x_i(n)$ was calculated. Precisely, let

$$B_i^*(n) = \{ j \in B_i \mid f(i, n-1) = (d_{ij} + x_j(n-1)) \},$$

then

$$\mathrm{Ind}(i,n) = \begin{cases} i, & \text{if } x_i(n) = 0 \text{ or } x_i(n) = D, \\ j \in B_i^*(n) |\ d_{ij} = \min\limits_{k \in B_i^*(n)} d_{ik}, & \text{otherwise.} \end{cases}$$

(If more than one index j attains the minimum for $f(i, n-1)$ and is a minimal distance away from i then any one is

selected.) Note that the update interval, $\Delta t_k = t_{k+1} - t_k$, need not be constant nor pre-determined. For example, with a regular unit square grid, if the robot moves at constant speed v_r, then Δt_k will be either $1/v_r$ or $\sqrt{2}/v_r$ time units, depending on whether the distance from $r(t_k)$ to $r(t_{k+1})$ is 1 or $\sqrt{2}$.

In the case of a static environment, once the location at which the robot resides has settled to its final x value (which occurs when the number of time steps exceeds the number of grid steps between the target and the robot via the shortest distance path) evolution of the dynamic system can cease and the robot can simply follow the minimum distance path to the target using (2).

If targets and/or barriers are moving, then the robot location can be easily updated by simply keeping track of $\mathrm{Ind}(i, n)$ for the robot location i while updating x_i. Since the minimum of $f(i, n-1)$ must be found to update x_i, no extra work is involved in keeping track of the index j that achieves the minimum.

C. Target and Barrier Movement

The targets and barrier locations are also assumed to be updated in real time at a frequency much slower than the dynamic system update frequency. Specifically, if a target moves from location p_{i-1} and arrives at p_i at time τ, then x_{p_i} and $x_{p_{i-1}}$ are immediately (at time τ) updated to 0 and $d_{p_i p_{i-1}}$ respectively. Similarly, if a barrier moves from p_{i-1} to p_i at time τ then x_{p_i} is immediately updated to D (x_{pi-1} is not updated).

In general, the speed of the target must be slower than the speed of the robot to ensure the robot's ability to reach the target. Since barriers can move, collisions with robots or the target are possible. The simulations of the next section were designed so that barriers never collided with the target. We prevented collisions between barriers and robots by specifying that if at time τ a barrier appears at a location p_i to which a robot is currently moving, $p_i = r(t_k)$, $t_{k-1} < \tau < t_k$, then the robot returns from whence it came ($r(t_k)$ is set to $r(t_{k-1})$). In more complicated situations where the barrier is moving faster than the robot, one would also have to deal with the case of barriers running into the robot itself.

D. Non-Square and Irregular Grids

Although our illustrations and the simulations in Section III are for regular square grids, the algorithm we have outlined above works equally well on any grid. The only requirement is that each grid point has a pre-defined set of adjacent neighbors and that the Euclidean distance to each neighbor is known.

III. Simulation Studies

In this section we demonstrate the effectiveness and efficiency of the proposed algorithm by applying it to both static and dynamic environments. In addition, we compare it to Yang and Meng's model.

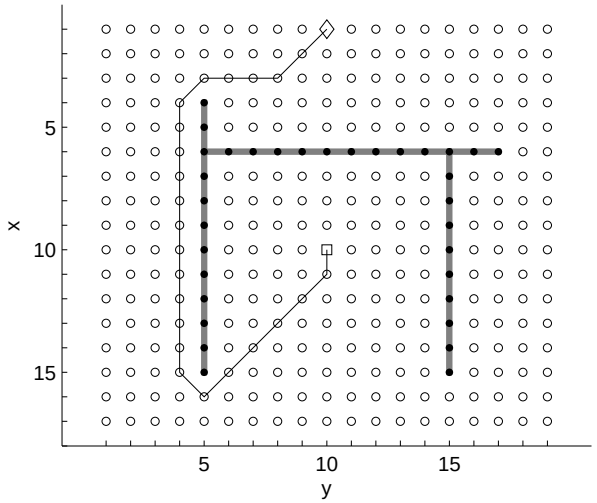

Fig. 3. Environment with a U-shaped obstacle and the robot trajectory around it. Barriers are shown as solid circles/shaded areas, free points as open circles, targets as diamonds, the initial robot location as a square, and the robot's path as a solid line.

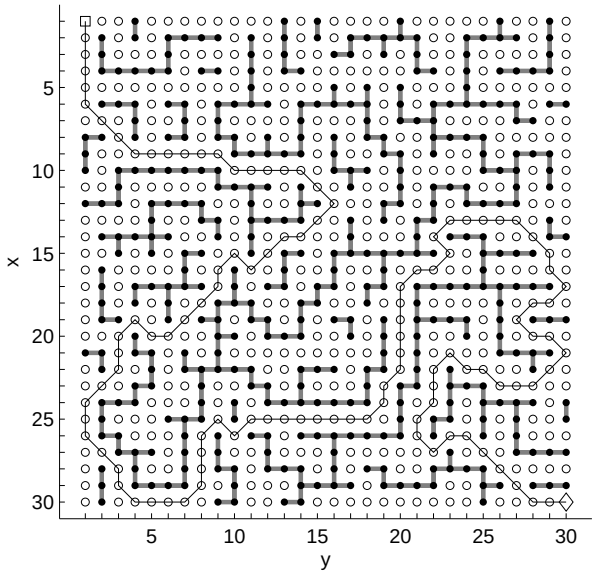

Fig. 4. A maze through which the robot finds the shortest path. Symbols as in Figure 3.

A. Static Environments

1) Obstacle Avoidance: In this simulation, a U-shaped barrier is placed between the robot and the target, Figure 3. The robot is not trapped behind the barrier since distance information cannot propagate through the barrier. Instead, the robot waits until distance information propagates around the barrier and then chooses the shortest path to the target.

2) Maze Traversal: This simulation shows that the robot can find its way through a complicated set of barriers to the target, Figure 4.

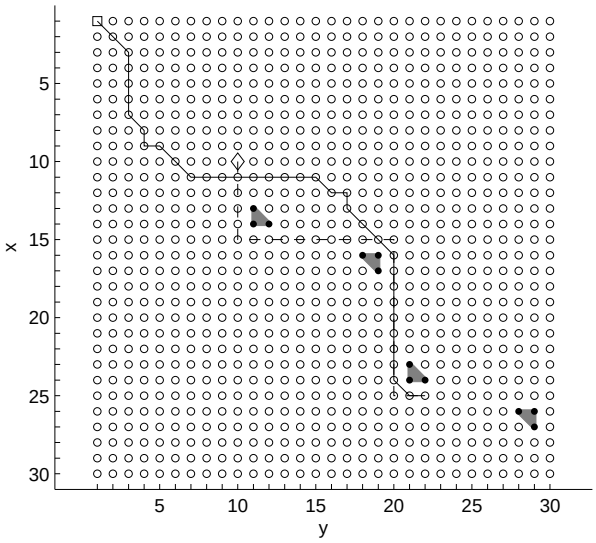

Fig. 5. The robot chases the zig-zagging target and catches it at (25,22). Symbols as in Figure 3; dashed line is the path of the target.

B. Dynamic Environments

1) Target Chasing: In this simulation, the target moves at a speed of 0.35 grid units per second from the location (10,10) in a zig-zag pattern around some non-moving barriers toward (30,30), see Figure 5. The robot starts at (1,1) and travels at a speed of 0.5 grid units per second. The robot catches the target at (25,22).

Note that there are two places where the robot's path is definitely sub-optimal. When the robot is at (8,4) and deciding where to move next, the target is at (15,11) and the shortest path is clearly to move diagonally 7 times in a south-easterly direction. However, the target just recently moved to (15,11) from (15,10) and there were only three time steps since that move. Consequently the distance information at (8,4) still reflected the situation when the target was at (15,10). The shortest path from (8,4) to (15,10) is not unique and involves 6 south-easterly steps and one southerly step in any order. Our implementation of the algorithm causes the robot to take the southerly step first since it is the shortest distance to the next neighbor along an optimal path. The reason behind this choice is that we want the robot to make its next choice as soon as possible to utilize up-to-date distance information as soon as it is available. Unfortunately, in this situation this choice leads to a longer eventual path to the target. A similar situation occurs when the robot is at (12,16) and the target has just recently moved from (15,20) to (16,20).

No matter what algorithm is used to choose which of several optimal paths to take, there are situations where the wrong choice will be made. In this simulation, the reason for the wrong choice was that up-to-date distance information had not yet propagated from the target to the robot. However, even if distance information propagated sufficiently fast, knowledge of where the target is going to move in the future would be required in order to always choose the best of several equal choices. Consider the situation shown in Figure 6. Here the target is at (3,2) when the robot is at (1,1). We assume that

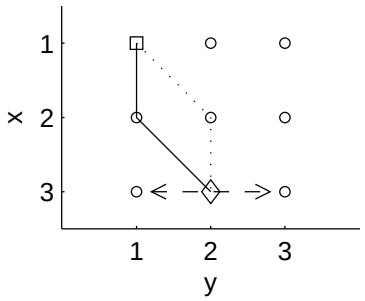

Fig. 6. The robot (square) has two optimal paths (solid and dotted line) to the target (diamond). However, the best choice of path will depend on whether the target moves east or west immediately after the robot moves.

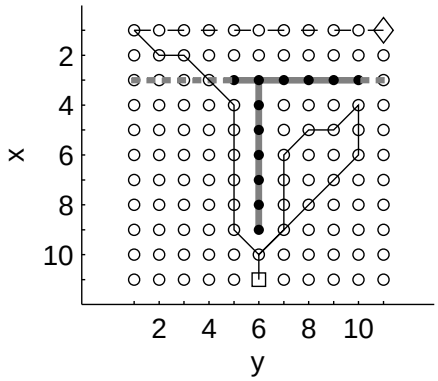

Fig. 7. Path occlusion. The robot initially starts toward the target moving along the east side of the central barrier. However, as it is moving northward, the target moves west and the upper barrier moves east blocking the robot's path. (The illustration shows the barrier just before it occludes the robot's path.) The robot then turns around and reaches the target via the west path. Symbols as in Figure 5.

the distance information is up-to-date, so that the robot has two optimal paths to the target $(1,1) \rightarrow (2,1) \rightarrow (3,2)$ or $(1,1) \rightarrow (2,2) \rightarrow (3,2)$. If the robot starts down the first path and the target moves east to (3,3), or if the robot starts down the second path and the target moves west to (3,1), the choice will have been sub-optimal.

2) Path Occlusion: In this simulation, Figure 7, the robot's path is blocked by a moving barrier. The target starts at (1,11) and moves westward at a speed of 0.2 grid units per second until it reaches (1,1). There is a stationary north-south barrier up the center of the grid, and an east-west barrier that starts by blocking columns 1 through 6 of row three. This barrier moves westward at a rate of 0.3 grid units per second until it blocks columns 6 through 11. The robot starts at (11,6) and moves at a speed of 0.5 grid units per second. Initially, the robot moves to the east of the central barrier toward the moving target. However, before it can manage to get north of row three, the moving barrier has completely blocked the robot's route causing it to turn around, go around the central barrier and approach the target via the west path, which is now open.

3) Circulating Barriers: In this final simulation there are two concentric U-shaped barriers initially lying on their sides with open ends in opposite directions. The robot starts at the

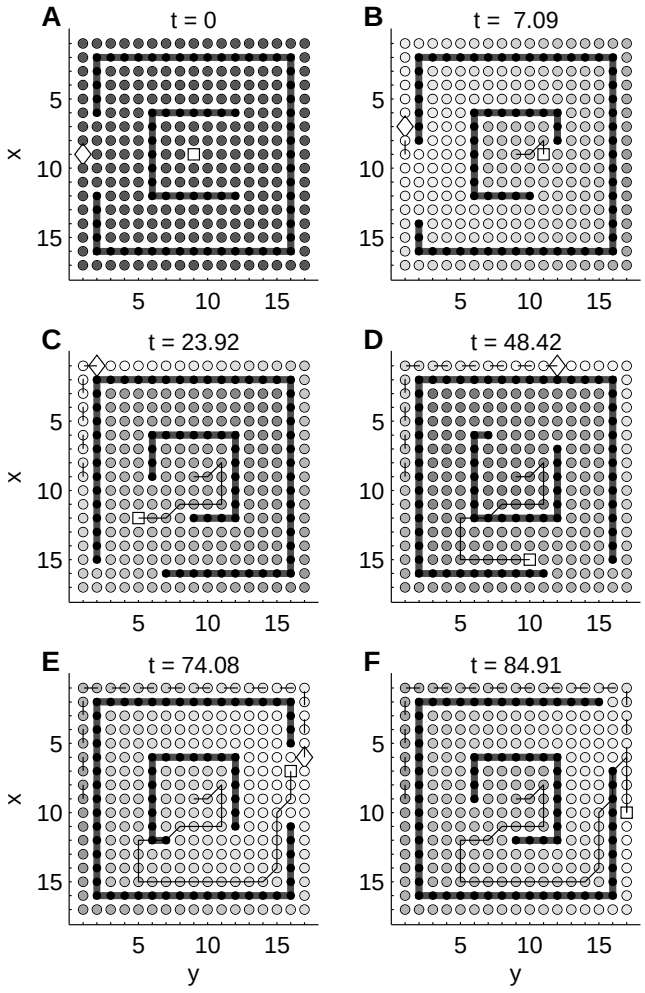

Fig. 8. Circulating barriers. The target (diamond) travels (dashed line) around the outside of the grid clockwise. The barriers (black circles/shaded areas) have openings that circulate: outer one counter-clockwise, inner one clockwise. The shading of all other grid points represents their current recorded distance to the target (light close, dark far). The robot (square) travels (solid line) from dark to light shaded points. The six panels show the state of the system at the given times.

middle of the grid and the target starts at the middle of the left side, Figure 8A. The target travels around the outside of the grid in a clockwise direction at speed 0.4 grid units per second, and the openings in the two barriers also circulate at the same speed with the outer one moving counter-clockwise and the inner one clockwise. The robot moves at 0.5 grid units per second. The robot first attempts to escape the inner barrier by traveling north-east, a decision made before the opening of the inner barrier had moved down one unit, but is cut off by the moving inner barrier and so heads south (Fig. 8B), eventually leaving the inner barrier in the south-west corner around $t = 23$ seconds (Fig. 8C). It then moves southward but cannot get through the outer barrier before the opening has moved eastward. The robot gives chase (Fig. 8D), eventually reaching the opening and passing through it near the middle of the west side. At $t = 74$ (Fig. 8E) the robot is at (7,16) and the target is at (6,15), directly to the north-east, hence the robot moves in that direction. However, by time it gets there,

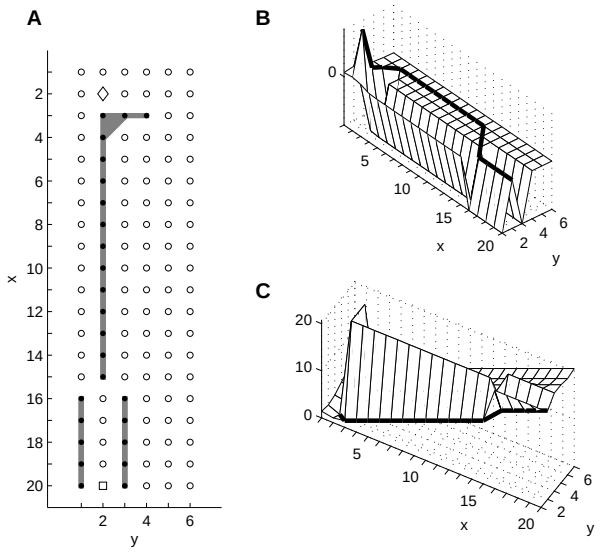

Fig. 9. A: The left path from the robot (square) to the target (diamond) is shorter but less connected than the right path. B: Neural activity landscape for the Yang and Meng model at $t = 3.5$ and the path chosen by the robot (bold line). C: The distance landscape and path chosen by the robot (bold line) using the distance-propagating model.

the target has moved south one unit and so the robot must chase it southward, reaching it at $t = 84.91$ (Fig. 8F).

C. Comparison with other Algorithms

The Yang and Meng model [26] uses a shunting equation to define a neural network where the "neural activity" of each grid point is a measure of the distance to the target. This network is governed by

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left(E\delta_{i,\text{target}} + \sum_{j \in B_i} \frac{\mu}{d_{ij}}[x_j]^+\right) - (D + x_i)\, E\delta_{i,\text{barrier}},$$

where A, B, D, μ, and E are positive constants (E large), $\delta_{i,\text{set}}$ is 1 if i is a member of set and zero otherwise, and $[x]^+$ is defined as $\max(x, 0)$. Since barrier locations receive a large negative input, these points have $x < 0$ and thus do not influence their neighbors. Initially all points start at zero. Points near a target end up with large (close to B) neural activity and, since each location receives excitatory input from its adjacent neighbors, generally speaking, locations near targets equilibrate to higher values than those far from the targets.

Although this model works well at generating dynamic collision-free trajectories [26] it is both more computationally intensive (depending on the numerical integration scheme used, but even via the Euler method it involves more computation), and less likely to find the optimal path than our distance-propagating model. Consider the grid shown in Figure 9A. Although the shortest distance to the target is via the left pathway, the right pathway has more neighbors for excitation and hence the equilibrium state of the Yang and Meng model has higher neural activity along the right path than the left,

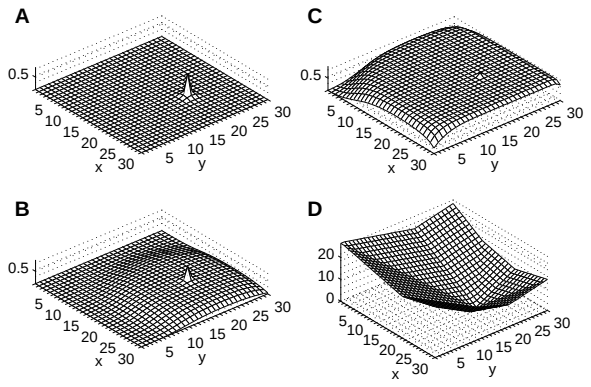

Fig. 10. A-C: Equilibrium neural activity landscape for the Yang and Meng model with no barriers and a target at (20,20). A: $\mu/A = 0.1$, B: $\mu/A = 0.25$, C: $\mu/A = 1$. D: Equilibrium values for the distance-propagating model.

Fig. 9B. Although the neural activity first reaches the robot's initial position via the left path, if the robot does not move sufficiently fast (and the assumption is that the robot moves slowly compared to the system update frequency) the neural activity on the right will exceed that on the left and entice the robot to choose the right path, Fig. 9B. Using the parameter values from Yang and Meng [26] ($A = 10$, $B = D = \mu = 1$, $E = 100$, and a robot update time of 0.1), the neural activity at position (15,3) already exceeds that at (15,1) by time $t = 0.4$. Since the robot does not ultimately decide which path to take until the fifth update time ($t = 0.5$), it chooses the more connected path rather than the shorter one. In comparison, our distance-propagating model will also quickly equilibrate but to a state where the values at each point are precisely the distance to the target and hence the robot will choose the left path, Fig. 9C.

A further difficulty with the Yang and Meng model is that it tends to equilibrate to landscapes with large plateau areas where the neural activity gradient is near zero making it difficult for the robot to determine which direction to advance. This happens when the ratio μ/A is too large or too small. This behavior is already evident in the previous example, Fig. 9B, where most of the non-barrier, non-target grid points have a neural activity near zero. Substantial gradients are confined to regions near the target. Figure 10 shows how the ratio of μ/A affects the equilibrium landscape. Here three plots of the equilibrium neural activity landscape of the Yang and Meng model for a 30×30 grid with no barriers and a target at (20,20) display the effect of varying μ/A. If the robot is somewhere on a flat plateau, there may be insufficient neural activity gradient for it to determine which direction to move. In comparison, our distance-propagating model has no parameters and quickly equilibrates to the surface shown in Fig. 10D where gradients remain large.

IV. Properties of the Dynamic System and Robot Movement Algorithm

In this section, we demonstrate the convergence of the distance-propagating dynamic system in the static case, and discuss the speed of convergence. We show that the generated robot path is optimal. For the dynamic situation where barriers are stationary but targets can move, we prove that our algorithm will result in the robot reaching a target provided the robot speed and system update frequency are sufficiently large.

A. Static Situation

When targets and barriers are stationary, our algorithm is a dynamic programming algorithm for finding the shortest path. However, since it only uses local information (every point is updated every iteration), it is inferior to the usual DP algorithms which sort the nodes in an ordered fashion. Proofs of convergence to the optimal solution for these DP algorithms typically rely on the sorted order of updating. For completeness, and to motivate our proofs in the dynamic case, we provide here a proof that in the static case our distance-propagating dynamic system converges and that our robot path planning algorithm results in an optimal shortest path.

1) Convergence of the Dynamic System: From the definition of the distance-propagating dynamic system given in Section II-A, since the distances d_{ij} are positive, it is clear that each x_i is bounded between 0 and D. The dynamic system constructively computes the distance from targets to nonbarrier points and is essentially itself the inductive proof that each x_i converges to the minimal distance to the nearest target.

More rigorously, recall that B_i is the set of nonbarrier adjacent neighbors to i and the distance d_{ij} between two free spaces i and j is defined to be the minimum Euclidean length of all paths joining i and j through nonbarrier adjacent neighbors. Let T be the set of target points and define y_i to be the minimum distance from a free space i to any target:

$$y_i = \min_{p\in T} d_{pi}.$$

We assume that every free space is connected by some nonbarrier path to every other free space, otherwise the grid subdivides into two or more isolated subgrids. Thus y_i is well defined for all free spaces i. We also assume that the constant D is chosen so that $D > \max_i y_i$. (If D is chosen smaller than this, then the algorithm will still work for all points that have minimal distance y_i less than D. In this way one could exclude points more than D away from any target if one so wished.) As stated earlier, $D > (M-1)d_{\max}$ is sufficiently large to include all points in the grid.

We wish to consider distances between two points in restricted neighborhoods. Let Γ_{ij}^n be the set of all paths joining i and j through n or less nonbarrier adjacent neighbors (that is, a path in Γ_{ij}^n includes at most n points), and let $|\gamma|$ be the Euclidean length of a path γ in this set. This set may be empty — in particular it is empty if i or j is a barrier. Also, $\Gamma_{ij}^n \subseteq \Gamma_{ij}^{n+1}$ for $n \geq 0$, and Γ_{ij}^M includes all paths joining i and j through nonbarriers points that do not intersect themselves. Define

$$d_{ij}^n = \begin{cases} D, & \text{if } \Gamma_{ij}^n = \emptyset, \\ \min\left(D, \min\limits_{\gamma\in\Gamma_{ij}^n} |\gamma|\right), & \text{otherwise.} \end{cases}$$

Notes: 1) $d_{ij}^1 = 0$ if and only if $i = j$ is a free space, 2) d_{ij}^n is a nonincreasing function of n, and 3) $d_{ij}^n = d_{ij}$ for $n \geq M$, if i and j are free spaces, since any path in Γ_{ij}^n, $n > M$, that is not in Γ_{ij}^M necessarily intersects itself and therefore could be shortened by removal of the redundant section.

The induction hypothesis is that for all points i in the grid,

$$x_i(n) = \min_{p \in T} d_{pi}^{n+1}, \qquad n \geq 0. \tag{3}$$

From the definition of the dynamic system we have $x_i(0) = 0$ if i is a target and $x_i(0) = D$ otherwise. Thus the induction hypothesis holds for $n = 0$. Clearly it also holds for all n if i is a target or barrier location. If i is a nontarget, nonbarrier location, then, assuming (3) holds at n we have

$$\begin{aligned} x_i(n+1) &= \min\left(D, \min_{j \in B_i}\left(d_{ij} + x_j(n)\right)\right) \\ &= \min\left(D, \min_{j \in B_i}\left(d_{ij} + \min_{p \in T} d_{pj}^{n+1}\right)\right) \\ &= \min_{p \in T}\left(\min\left(D, \min_{j \in B_i}\left(d_{ij} + d_{pj}^{n+1}\right)\right)\right) \\ &= \min_{p \in T} d_{pi}^{n+2}. \end{aligned}$$

Thus (3) holds for all n. For a free space i, since d_{pi}^n is nonincreasing in n and converges to d_{pi} it follows that each $x_i(n)$ is nonincreasing and converges to y_i, the minimal distance to any target.

a) Speed of Convergence: Although for many points i, x_i will drop completely from D to y_i at some time point N, the situation shown in Figure 2 illustrates that some points will record intermediate values before settling to y_i. The number of time steps required for x_i to converge to y_i depends on the number and distances between adjacent points on the paths connecting i to the various targets. Clearly, for a grid with M points, $M-1$ is the upper bound on the number of time steps required for convergence of all points. (This bound cannot be improved without some restriction on the grid itself; simply consider a linear grid with a target at one end and no barriers.)

Since the distance between adjacent neighbors is bounded between $d_{\min}$ and $d_{\max}$, the number of grid steps, n_{ip} between i and any target p is bounded by

$$\left\lceil \frac{d_{ip}}{d_{\max}} \right\rceil \leq n_{ip} \leq \left\lfloor \frac{d_{ip}}{d_{\min}} \right\rfloor.$$

(Here $\lceil z \rceil$ and $\lfloor z \rfloor$ represent z rounded upward or downward to the nearest integer respectively.) Information is propagated one grid step every time step so x_i will converge to y_i in N time steps, where

$$\left\lceil \frac{y_i}{d_{\max}} \right\rceil \leq N \leq \left\lfloor \frac{y_i}{d_{\min}} \right\rfloor.$$

Further, the number of time steps at which x_i records a value other than D or y_i is bounded above by

$$\max_{p \in T}(N - n_{ip}) \leq \left\lfloor \frac{y_i}{d_{\min}} \right\rfloor - \left\lceil \frac{y_i}{d_{\max}} \right\rceil.$$

We also have the following convergence test:

$$\mathrm{conv}_i = \left\lfloor \frac{x_i(n)}{d_{\min}} \right\rfloor - n \tag{4}$$

is the maximum number of time steps remaining until x_i converges to y_i; if conv_i is below one, convergence has already occurred. We emphasize that this convergence test uses only local information (plus the minimum adjacent grid length which is set before the algorithm commences) and that if a more typical DP algorithm was implemented which used global information then a better convergence test would be possible.

b) Arbitrary Initial Conditions: If the system starts from some arbitrary set of initial conditions for $x_i(0)$ rather than those given in (1), then essentially the same proof as above still applies. The induction hypothesis must simply be restricted to include just those points i connected to some $p \in T$ via some path in Γ_{ip}^n. In this case $x_i(n)$ is no longer nonincreasing for all n, but is nonincreasing for those n after the first time at which i is on some path in Γ_{ip}^n for some $p \in T$.

2) Optimality of the Robot Path: For any nonbarrier location i, define an optimal path for i as any path $\gamma \in \cup_{p \in T} \Gamma_{ip}^M$ such that $|\gamma| = y_i$. By the definition of y_i at least one such path exists, although it need not be unique. In other words, an optimal path for i is a path connecting i to a target whose length is the minimum distance from i to any target. Consider an adjacent neighbor, j, of i. If $y_i = y_j + d_{ij}$ then necessarily at least one optimal path for i will pass through j.

The robot path is determined by (2). If i is the current robot location and $x_i < D$ then the next location to which the robot moves is determined by choosing an adjacent neighbor of i that generated the value x_i. (If more than one adjacent neighbor generated equivalent values for x_i then any one of the nearest of these neighbors is selected.) Thus, if x_i has converged to y_i, an adjacent neighbor that generated the value of x_i will necessarily have value $x_j = y_i - d_{ij}$, which must be equal to y_j; otherwise x_j would eventually decrease to y_j and cause x_i to decrease below y_i. Therefore j will be on an optimal path for i, and the robot will move along an optimal path to the nearest target.

Thus, to assure the robot takes an optimal path to the nearest target, it is sufficient to delay the robot from moving until the location at which it currently resides has converged to y_i. This can be easily accomplished by using the convergence test given by (4). The proof that it is in general necessary for the robot to wait until convergence before moving to ensure selection of an optimal path is provided by the illustration in Figure 2. In this figure, the value of x at the robot location first drops below D at time step 8. But if the robot first moves sometime between time step 8 and before time step 10, it will choose to move north east rather than west and thus will not take the optimal path C. However, it is important to point out again that the simplicity of the dynamic system means a large system update frequency and consequently a large speed of information propagation compared to the speed of the robot. Thus waiting for convergence will only mildly delay the robot and will ensure an optimal path.

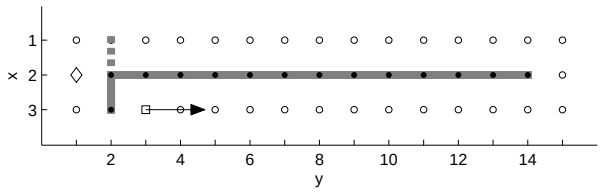

Fig. 11. A situation where the robot can never reach the target. Symbols as in Figure 3. The vertical barrier continuously moves north and south at just the right speed to block the robot from getting to the target causing the robot to turn around and attempt to reach the target by the newly opened path. See text.

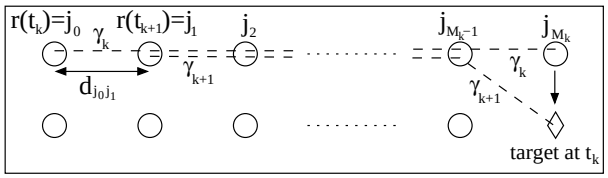

Fig. 12. Dynamic chase. The robot is at $r(t_k) = j_0$ and the target is at the diamond at time t_k, although, the distance information propagated along path γ_k to j_0 reflects the prior situation when the target was located at j_{M_k}. The robot moves to j_1 at time t_{k+1} by which point the distance information has propagated from the target along path γ_{k+1} to j_1.

B. Dynamic Situation

In the dynamic situation, where barriers and/or targets are moving, the dynamic system will in general not have a steady-state to which it converges. Further, the values of $x_i(n)$ are no longer nonincreasing since, in particular, if i was formerly a target location and becomes a free space or if it was formerly a free space and becomes a barrier, the value of x_i will increase at that time.

If barriers are permitted to move then, even if the distance-propagating dynamic system updates infinitely fast, and no matter how large the robot's speed, it is always possible to construct a situation where it is impossible for the robot to reach the target. Simply consider the environment shown in Figure 11. The robot begins moving east around the long horizontal barrier to reach the target via the open north path. However, just as the robot reaches $(1, 3)$, the vertical barrier moves northward occupying $(1, 2)$ and blocking the north path. At this point the robot must turn around and try to reach the target via the newly opened south path. The barrier then moves southward occluding the south path just as the robot reaches $(3, 3)$. No matter how large the robot's speed and how slow the barrier's speed, an environment similar to the one in Figure 11 can be constructed with a sufficiently long horizontal barrier so that the robot will never reach the target.

It is also necessary for the speed of the targets to be smaller than the speed of the robot in order for the robot to reach a target in general. Simply consider an annular grid of width one with target and robot moving at the same speed.

In the situation where barriers do not move, and the robot speed and system update frequency are sufficiently large, we can show that our robot moving algorithm will result in the robot reaching a moving target. We achieve this by showing that at each subsequent robot location, x is decreasing by a finite amount.

Since $D > d_{\max}(M-1)$, it follows that D is larger than the length of any non-intersecting path through free spaces in this grid. For any location i, once x_i takes on a value below D, it measures the length of some non-intersecting path between i and some other free space in the grid where a target once resided. Thus, once below D, x_i will remain below D forever. Define f_u to be the system update frequency (number of time steps, n, per unit real time), and v_r and v_t to be the robot and target speeds (distance/time) respectively. We assume that unless it is stopped, the robot moves with constant speed until it reaches a target. Targets are assumed to move with speed less than or equal to v_t.

Recall that the robot location is updated at times t_k according to (2). Let k^* be the smallest integer such that $r(t_{k^*+1}) \neq r(t_{k^*})$. In other words, t_{k^*} is the first time at which the robot starts to move. Update times prior to this occurred when the distance information had not yet propagated to the robot's location, i.e. $x_{r(t_k)} = D$ for $k < k^*$. The robot does not move until $x_{r(t_k)} < D$ and, by definition of the robot update function, $\mathrm{Ind}(i, n)$, it always moves to a location which must also have once been less than D. Since we have already noted that once x is below D it remains below D forever, it follows that the robot will continue to move until $x_{r(t_k)} = 0$.

For fixed $k > k^*$, define γ_k as a path joining $r(t_k)$ to the location where a target was when it generated the distance information that propagated to $r(t_k)$. That is, $\gamma_k = \{j_m\}_{m=0}^{M_k}$ where

$$\begin{aligned} j_0 &= r(t_k), \\ j_{m+1} &= \mathrm{Ind}(j_m, n(j_0, t_k) - m), \quad 0 \leq m < M_k, \end{aligned}$$

and M_k is the smallest integer such that

$$x_{j_{M_k}}(n(j_0, t_k) - M_k) = 0. \tag{5}$$

The integer M_k necessarily exists since along this path we have

$$x_{j_m}(n(j_0, t_k) - m) = x_{j_{m+1}}(n(j_0, t_k) - m - 1) + d_{j_m j_{m+1}}, \tag{6}$$

for $0 \leq m < M_k$, so that each step along the path x decreases by at least $d_{\min}$. In other words, a target resided at j_{M_k} at time step $n(j_0, t_k) - M_k$. See Figure 12.

For any nonnegative integer s, define

$$\Delta_s x_i(n) = x_i(n + s) - x_i(n).$$

By (5) we have

$$\begin{aligned} \Delta_s x_{j_{M_k}}(n(j_0, t_k) - M_k) &= x_{j_{M_k}}(n(j_0, t_k) - M_k + s) \\ &\leq v_t s / f_u, \end{aligned} \tag{7}$$

since the target which once resided at j_{M_k} can have moved at most this distance in s time steps. The inequality may be a result of the target not moving at full speed v_t, the update frequency, f_u, not being sufficiently large so that $x_{j_{M_k}}$ is reflecting the distance to the target when it was closer, or if a different path to a target becomes shorter. Note also that

by (6) for any point j_m, $m < M_k$, on γ_k we have

$$\begin{aligned}\Delta_s x_{j_m}(n) &= x_{j_m}(n+s) - x_{j_m}(n)\\ &= x_{j_{m+1}}(n-1+s) + d_{j_m j_{m+1}}\\ &\quad - \left(x_{j_{m+1}}(n-1) + d_{j_m j_{m+1}}\right)\\ &= \Delta_s x_{j_{m+1}}(n-1).\end{aligned}$$

The above combined with (7) implies

$$\Delta_s x_{j_1}(n(j_0,t_k)-1) \le v_t s/f_u. \tag{8}$$

Therefore, for a fixed $k \ge k^*$, setting $s = n(j_1,t_{k+1}) - n(j_0,t_k) + 1$ we have

$$\begin{aligned}&x_{j_1}(n(j_1,t_{k+1}))\\ &\quad = \Delta_s x_{j_1}(n(j_0,t_k)-1) + x_{j_1}(n(j_0,t_k)-1)\\ &\quad \le v_t s/f_u + x_{j_1}(n(j_0,t_k)-1)\end{aligned}$$

by (8). Using (6) we get

$$x_{j_1}(n(j_1,t_{k+1})) \le v_t s/f_u + x_{j_0}(n(j_0,t_k)) - d_{j_0 j_1}.$$

Now $\lfloor f_u t \rfloor \le n(j,t) \le \lceil f_u t \rceil$, therefore

$$\begin{aligned}s &\le \lceil f_u t_{k+1} \rceil - \lfloor f_u t_k \rfloor + 1\\ &\le f_u t_{k+1} - f_u t_k + 3.\end{aligned}$$

Consequently,

$$\begin{aligned}x_{j_1}(n(j_1,t_{k+1})) &\le x_{j_0}(n(j_0,t_k)) + 3v_t/f_u\\ &\quad + v_t\left(t_{k+1} - t_k\right) - d_{j_0 j_1}.\end{aligned}$$

Since the robot moves at constant speed $v_r > v_t$,

$$v_r\left(t_{k+1} - t_k\right) = d_{j_0 j_1},$$

and therefore

$$\begin{aligned}&x_{j_1}(n(j_1,t_{k+1}))\\ &\quad \le x_{j_0}(n(j_0,t_k)) + 3v_t/f_u + \left(\frac{v_t}{v_r} - 1\right) d_{j_0 j_1}\\ &\quad \le x_{j_0}(n(j_0,t_k)) + 3v_t/f_u + \left(\frac{v_t}{v_r} - 1\right) d_{\min}\end{aligned}$$

Therefore, recalling that $r(t_k) = j_0$ and $r(t_{k+1}) = j_1$, provided f_u is sufficiently large, and v_r is sufficiently larger than v_t such that

$$\frac{3v_t}{f_u} < d_{\min}\left(1 - \frac{v_t}{v_r}\right)$$

we will have

$$x_{r(t_{k+1})}(n(r(t_{k+1}),t_{k+1})) < x_{r(t_k)}(n(r(t_k),t_k))$$

so that at each robot update time, x at the robot location is decreasing by a nonzero amount. Since the robot update times are at most $d_{\max}/v_r$ apart it follows that $x_{r(t_M)}(n(r(t_M),t_M)) = 0$ for some finite M, in other words, the robot reaches a target. Technically, the robot may be somewhere between $r(t_M)$ and one of its adjacent neighbors at time t_M, but we assume that either being adjacent to a target is sufficiently close, or some other mechanism is available for telling the robot which way to move when the target is between it and an adjacent neighboring free space. The fact that $v_r > v_t$ will then assure capture of the robot within some further finite time.

C. Computational Issues

If the grid is uniform so that each point has neighbors in the same orientation, then it may be advantageous to use this feature to optimize the computation of the minimum of $f(i,n)$. For example, for a regular square grid with unit spacing, instead of computing $d_{ij} + x_j(n)$ for each j and then finding the minimum, it is less work to find $m_1 = \min\{x_j(n) \,|d_{ij} = 1\}$ and $m_2 = \min\{x_j(n) \,|d_{ij} = \sqrt{2}\}$, and then take the minimum of $m_1 + 1$ and $m_2 + \sqrt{2}$.

V. Conclusion

We have presented a simple yet efficient dynamic system algorithm for real-time collision-free robot path planning applicable to situations where targets and barriers are permitted to move. The algorithm works in real time and requires no prior knowledge of target or barrier movements. Updating the distance estimate at each grid point is done without any global knowledge, neither of distances at non-neighboring points nor the update history of any point. Thus each point could be implemented as an independent processor with only local connections to its neighbors. The robot path is determined in real-time completely from information obtained from the robot's current grid point location. The computational effort to update each point is minimal allowing for rapid propagation of the distance information outward along the grid from target locations. In the static situation, where both targets and barriers do not move, our algorithm is a dynamic programming solution to the shortest path problem, and as such generates optimal robot paths after a small number of iterations of the dynamic system. In the case where only barriers are stationary, our algorithm always results in the robot catching a target provided it moves at greater speed than the target, and the dynamic system update frequency is sufficiently large. Both of these requirements are mild and likely to be met in practice.

The local nature of our algorithm distinguishes it from most algorithms. For example A^* uses global information to estimate the distance to the robot, artificial potential field methods define a potential function over the entire workspace dependent on global distances from the targets and barriers, cell decomposition methods split up the workspace based on the global geometry of the barriers, and visibility graph methods require global information about distances between control points on various barriers. In addition, most of these algorithms do not easily extend to the dynamic situation and become computationally expensive when the environment is complex. Our algorithm in contrast is designed for the dynamic situation and is not at all affected by the complexity of the environment as far as computational speed is concerned.

Our algorithm could be sped up if global information were utilized. For example, the algorithm could update the neighbors of the most recently altered points first, or update them in an order depending on the distance to the nearest target. In the dynamic situation, where barriers and targets are moving, the latter option would need to be modified since there is a trade-off between propagating new information from the recently moved target or continuing to propagate older information to the robot's location.

Our algorithm has all the advantages of Yang and Meng's (2000) neural network model and is faster and more efficient. Unlike Yang and Meng's model, the proposed algorithm efficiently propagates the distance from the target to the entire robot workspace. Thus it has a better physical meaning and is easier for visualization than the neural activity landscape of Yang and Meng's model. In addition, it is not subject to saturation effects that can prevent the robot reaching its target with the Yang and Meng model.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. on Systems, Science and Cybernetics*, vol. SSC-4, pp. 100–107, 1968.

[2] C. W. Warren, "Fast path planning using modified A* method," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 5 1993, pp. 662–667.

[3] J. C. Latombe, *Robot Motion Planning*. Boston: Kluwer Academic Publisher, 1991.

[4] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Trans. on Robotics and Automation*, vol. 5, no. 2, pp. 143–150, 4 1989.

[5] E. Hou and D. Zheng, "Mobile robot path planning based on hierarching hexagonal decomposition and artificial potential fields," *J. of Robotic Systems*, vol. 11, no. 7, pp. 605–614, 1994.

[6] J. T. Schwartz and M. Sharir, "On the piano movers' problem: I. the case if a two-dimensional rigid polygonal body moving amidst polygonal barriers," *IEEE Trans. on Robotics and Automation*, vol. 36, pp. 345–398, 1983.

[7] D. Leven and M. Sharir, "An efficient and simple motion planning algorithms for a ladder moving in two-dimensional space amidst polygonal barriers," in *Proc. of the First ACM Symp. on Computational Geometry*, 1997, pp. 1208–1213.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Research*, vol. 5, pp. 90–98, 1986.

[9] J. Chuang and N. Ahuja, "An analytically tractable potential field model of free space and its application in obstacle avoidance," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 28, no. 5, pp. 729–736, 10 1998.

[10] K. P. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-D dynamic environments using an electrostatic potential field," *IEEE Trans. on Systems, Man, and Cybernetics, Part A*, vol. 30, no. 2, pp. 187–196, 3 2000.

[11] J. Barraquand and J. C. Latombe, "Robot motion planning: A distributed representation approach," *Intl. J. Robotics Research*, vol. 10, no. 6, pp. 628–649, 1991.

[12] K. Kant and S. W. Zucker, "Towards efficient trajectory planning: The path-velocity decomposition," *Int. J. Robotics Research*, vol. 5, pp. 72–89, 1986.

[13] Z. X. Li and T. D. Bui, "Robot path planning using fluid model," *J. of Intelligent and Robotic Systems*, vol. 21, pp. 29–50, 1998.

[14] A. Zelinsky, "Using path transforms to guide the search for findpath in 2D," *Intl. J. of Robotics Research*, vol. 13, no. 4, pp. 315–325, 1994.

[15] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Population coding in a neural net for trajectory formation," *Network: Computation in Neural Systems*, vol. 5, pp. 549–563, 8 1994.

[16] C. Seshadri and A. Ghosh, "Optimum path planning for robot manipulators amid static and dynamic obstacles," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 23, pp. 576–584, 1993.

[17] C. J. Ong and E. G. Gilbert, "Robot path planning with penetration growth distance," *J. of Robotic Systems*, vol. 15, no. 2, pp. 57–74, 1998.

[18] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 28, no. 3, pp. 316–333, 6 1998.

[19] H. J. Ritter, T. M. Martinetz, and K. J. Schulten, "Topology-conserving maps for learning visuo-motor-coordination," *Neural Networks*, vol. 2, pp. 159–189, 1989.

[20] P. Gaudiano, E. Zalama, and J. L. Coronado, "An unsupervised neural network for low-level control of a mobile robot: Noise resistance, stability, and hardware implementation," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 26, no. 3, pp. 485–496, 1996.

[21] E. Zalama, P. Gaudiano, and J. L. Coronado, "A real-time, unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment," *Neural Networks*, vol. 8, pp. 103–123, 1995.

[22] L. Li and H. Öğmen, "Visually guided motor control: Adaptive sensorimotor mapping with on-line visual-error correction," in *Proc. of the World Congress on Neural Networks*, 1994, pp. 127–134.

[23] T. Fujii, Y. Arai, H. Asama, and I. Endo, "Multilayered reinforcement learning for complicated collision avoidance problems," in *Proc. of IEEE Intl. Conf. on Robotics and Automation, Leuven, Belgium*, 5 1998, pp. 2186–2191.

[24] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Networks*, vol. 8, no. 1, pp. 125–133, 1995.

[25] ——, "A biologically inspired neural net for trajectory formation and obstacle avoidance," *Biological Cybernetics*, vol. 74, pp. 511–520, 1996.

[26] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free robot trajectory generation," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 31, no. 3, pp. 302–318, 6 2001.

[27] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol. Lond.*, vol. 117, pp. 500–544, 1952.

[28] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies in Applied Mathematics*, vol. 52, pp. 217–257, 1973.

[29] ——, "Nonlinear neural networks: Principles, mechanisms, and architecture," *Neural Networks*, vol. 1, pp. 17–61, 1988.

[30] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural Networks*, vol. 13, no. 2, pp. 143–148, 2000.

[31] G. D. Barreto, A. F. R. Araujo, and H. J. Ritter, "Self-organizing feature maps for modeling and control of robotic manipulators," *J. of Intelligent and Robotic Systems*, vol. 36, no. 4, pp. 407–450, 4 2003.

[32] S. X. Yang and M. Q.-H. Meng, "Real-time collision-free motion planning of mobile robots using neural dynamics based approaches," *IEEE Trans. on Neural Networks*, vol. 14, no. 6, pp. 1541–1552, 11 2003.

[33] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 1, pp. 718–725, 2 2004.

[34] S. X. Yang and M. Meng, "An efficient neural network method for real-time motion planning with safety consideration," *Robotics and Autonomous Systems*, vol. 32, no. 2–3, pp. 115–128, 2000.

[35] E. V. Denardo, *Dynamic Programming: Models and Applications*. Englewood Cliffs, NJ: Prentice Hall Inc., 1982.