

Real-Time Robot Path Planning Based on a Modified Pulse-Coupled Neural Network Model

Hong Qu, Simon X. Yang, *Senior Member, IEEE*, Allan R. Willms, and Zhang Yi

Abstract—This paper presents a modified pulse-coupled neural network (MPCNN) model for real-time collision-free path planning of mobile robots in nonstationary environments. The proposed neural network for robots is topologically organized with only local lateral connections among neurons. It works in dynamic environments and requires no prior knowledge of target or barrier movements. The target neuron fires first, and then the firing event spreads out, through the lateral connections among the neurons, like the propagation of a wave. Obstacles have no connections to their neighbors. Each neuron records its parent, that is, the neighbor that caused it to fire. The real-time optimal path is then the sequence of parents from the robot to the target. In a static case where the barriers and targets are stationary, this paper proves that the generated wave in the network spreads outward with travel times proportional to the linking strength among neurons. Thus, the generated path is always the global shortest path from the robot to the target. In addition, each neuron in the proposed model can propagate a firing event to its neighboring neuron without any comparing computations. The proposed model is applied to generate collision-free paths for a mobile robot to solve a maze-type problem, to circumvent concave U-shaped obstacles, and to track a moving target in an environment with varying obstacles. The effectiveness and efficiency of the proposed approach is demonstrated through simulation and comparison studies.

Index Terms—Collision avoidance, mobile robot, neural dynamics, path planning, pulse-coupled neural networks (PCNNs), spiking, wave.

I. INTRODUCTION

REAL-TIME planning of a collision-free path for a mobile robot to move from an initial position to a target position in a nonstationary environment is a well-known problem in robotics. Since the earliest work by Lozano-Perez and Wesley

in 1979 [1], which presented an algorithm to avoid polyhedral obstacles based on a visibility graph, various methods have been proposed to solve the robot path-planning problem. Global methods such as road map [2], cell decomposition [3]–[6], and distance transform [7] were able to search possible paths in the whole workspace, although these methods are mainly suitable to static environments, and are computationally expensive in complex environments. Many heuristic approaches, such as potential field methods [8]–[13] and the grid-based A^* algorithm [14]–[16], have been proposed to provide effective path searching. Some of those methods suffer from undesired local minima, i.e., the robot may be trapped in some cases such as with concave U-shaped barriers. In general, the global approaches would suffer from slow computation for implementation in complex environments, while the local planners have difficulty to find a reasonable solution to some situations with local minima.

Some new approaches have also been reported in recent years. Li and Bui [17] proposed a fluid model for robot path planning in a static environment. Ong and Gilbert [18] proposed a new model for path planning with penetration growth distance, which can generate optimal continuous robot paths in static environments. Oriolo *et al.* [19] proposed a model for real-time map building and navigation for a mobile robot, where a global path-planning algorithm, a local graph search algorithm, and several cost functions are used. Hu and Yang [20] proposed an efficient knowledge-based genetic algorithm for real-time path planning of mobile robots, where five genetic operators were designed particularly for robot path planning. Lebedev *et al.* [21] presented a dynamic wave expansion model for robot motion planning in time-varying environments, which is claimed to be parameter free, and computationally efficient in dynamic environments. More recently, Willms and Yang [22], [23] proposed an efficient dynamic programming algorithm for real-time robot path planning, which is reported to be suitable to situations where targets and barriers are permitted to move.

Since the original work of Hopfield [24] in 1985, much research has been done using neural networks to solve combinatorial optimization problems [25]–[27]. Some neural network models have been proposed to generate real-time robot trajectories. Zalama *et al.* [28] proposed a neural network model for mobile robot navigation. Kohonen's self-organizing map (SOM)-based model that learns the transformation from the Cartesian workspace to the joint space of robot manipulators was studied by Ritter *et al.* [29]. Zhu and Yang [30] proposed a SOM-based model for dynamic task assignment and path planning of multirobots. Glasius *et al.* [31]–[33] proposed a Hopfield-type neural network for real-time trajectory generation with obstacle avoidance in a nonstationary environment.

Manuscript received December 01, 2007; revised April 23, 2009 and July 27, 2009; accepted July 30, 2009. First published September 22, 2009; current version published November 04, 2009. This work was supported by the National Science and Engineering Research Council (NSERC) of Canada, the National Science Foundation of China under Grants 60905037 and 60802064, and the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 200806141049.

H. Qu is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China and also with the School of Engineering, University of Guelph, Guelph, ON N1G 2W1 Canada (e-mail: HongQu@uestc.edu.cn).

S. X. Yang is with the Advanced Robotics and Intelligent System (ARIS) Laboratory, School of Engineering, University of Guelph, Guelph, ON N1G 2W1 Canada (e-mail: syang@uoguelph.ca).

A. R. Willms is with the Department of Mathematics and Statistics, University of Guelph, Guelph, ON N1G 2W1 Canada (e-mail: awillms@uoguelph.ca).

Z. Yi is with the School of Computer Science, Sichuan University, Chengdu 610054, China.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TNN.2009.2029858

It is rigorously proved that the generated trajectory does not suffer from undesired local minima and is globally optimal in a stationary environment. However, these models require that the robot dynamic is faster than the target and obstacle dynamics, and have difficulty in dealing with fast changing environments [34]. Inspired by the dynamic properties of Hodgkin and Huxley's membrane model [35] for a biological neural system and the later developed Grossberg's shunting model [36], [37], Yang and Meng [38] proposed a new neural network approach for dynamic environments. This neurodynamics approach is later extended to various robot systems [39]–[45], and further inspires the development of a distance-propagating system for robot path planning [22], [23]. Low *et al.* [47] presented a cooperative extended Kohonen maps (EKMs) to achieve complex robot motion, and later they [48] described a distributed layered architecture for resource-constrained multirobot cooperation. Those two methods are able to avoid local minima such as concave obstacles as well as dynamic environments.

A pulse-coupled neural network (PCNN) is a biologically plausible computational algorithm, which is similar to the locally excitatory globally inhibitory oscillator network (LEGION) model proposed by Wang *et al.* for real-time image segmentation, figure/ground segregation, and many other applications [49], [50]. PCNNs are capable of emulating the behavior of cortical neurons as observed in the visual cortices of cats [51]. Since Johnson's work [52]–[54], there has been increased interest in using PCNNs for various applications, such as target recognition [55], image processing [56], motion matching [57], pattern recognition [58], and optimization [59]. Some recent research shows that the spatio-temporal dynamics of PCNNs provide good computational capability for solving some optimization problems. In 1999, Caulfield and Kinser [59] presented the idea of utilizing the autowave in PCNNs to find the solution to the maze problem. Their model can find the shortest path quickly and with minimum effort, where the solution is related to the length of the shortest path and independent to the path graph complexity. However, many neurons are needed to find the shortest path in large mazes or graphs, since one pulse of the coupled neuron corresponds to a unit length of path [60].

In this paper, a novel real-time collision-free robot-path-planning approach is proposed based on a modified pulse-coupled neural network (MPCNN). The proposed model is topologically organized with only local lateral connections among neurons. It needs fewer neurons than Caulfield and Kinser's approach [59], since the pulse propagation does not correspond to a unit length of path. Instead, the generated spiking wave in the proposed network spreads at a constant speed so that the time of travel between two neurons is proportional to the path length between them. Whenever the sensory systems (either onboard or installed in the workspace) detect a change in the dynamic environment, a new path is calculated in the same way. The computational complexity of the algorithm is only related to the length of the shortest path, and independent of the workspace complexity and the number of existing paths in the map. Each neuron in the proposed model propagates a firing event to its neighboring neuron without any comparison computations. The proposed model also works in real time and requires no prior

knowledge of target or barrier movement, no explicit optimization of any cost functions, and no explicit searching over the free workspace or collision paths. It is, therefore, useful for real-time robot path planning in dynamic environments, without the need of any learning procedures.

Similar to many existing path-planning methods [21], [22], [38], in the proposed approach to real-time robot path planning, the complete knowledge of the current environment is required. In particular, it is assumed that the locations of the robot, obstacles, and targets are completely and accurately known through various sensor measurements, multisensor fusion, and signal processing systems. The autonomous navigation of mobile robots is very complicated and involves many issues, such as real-time path planning, multisensor fusion, signal processing, and motion control. The sensor placement and multisensor fusion needed to obtain complete environment knowledge, the signal processing required to remove sensor measurement and communication noise for accurate environmental information, and the motion control systems needed to drive the robot wheels are beyond the scope of this paper. In indoor and closed environments, the complete knowledge of the workspace can be obtained through several charge-coupled device (CCD) cameras mounted over the ceiling and various onboard robot sensors. In outdoor and open environments, knowing the environment is a challenging task, which could possibly be achieved through global positioning system (GPS), land marks, laser sensors, ultrasonic sensors, and various other sensors [61]. Related work on real-time path planning of mobile robots with limited onboard sensory information in completely or partially unknown environments is found in [41], [42], and [62]. Knowing the accurate locations of the obstacles and robot in the environment at every instant can be addressed with simultaneous map building/updates and localization methods [41], [63], [64]. Some approaches for real-time navigation of mobile robots have no explicit global path-planning algorithms. These employ human-like or biologically inspired strategies, such as neurofuzzy and behavior-based methods [61], [65], where the robot movement is determined by the real-time sensor measurements and only simple local path planning is available.

The remainder of this paper is organized as follows. The proposed modified PCNN is described and studied theoretically in Section II. Algorithms for robot path planning, based on the MPCNN, are presented in Section III. Simulation results are given in Section IV. Section V discusses parameter setting for the proposed model. Finally, conclusions are drawn in Section VI.

II. MPCNN MODEL FOR ROBOT PATH PLANNING

In this section, an MPCNN model is proposed for real-time robot path planning. A simple description of the original PCNN is described in the Appendix. The performance of the proposed model is analyzed mathematically. In addition, many variables are defined and used in this section. The main symbols and definitions used in this paper are summarized in Table I.

A. Network Architecture

The neural network architecture of the proposed model is a discrete topologically organized map and can be expressed in a

TABLE I
MAIN SYMBOLS AND DEFINITIONS

Symbol	Definition
t_{fire}^i	Firing time of neuron i
R_i^F	The first fired neuron in neuron i 's neighboring set
R_i^P	The parent of neuron i
w_{ij}	The connection weight from i to j
R_i^l, R_i^r	Two subsets of i -th neuron's neighboring set
$U_i(t)$	The internal activity of neuron i at time t

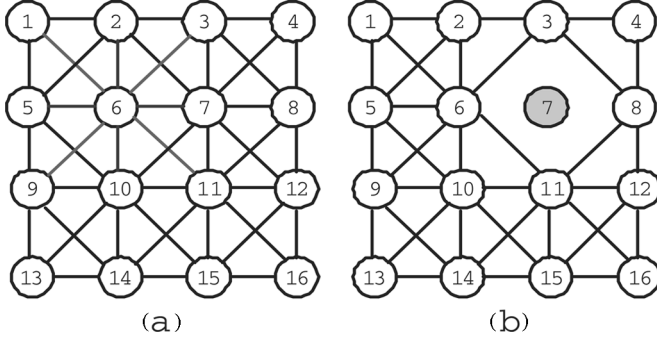


Fig. 1. (a) Network architecture without barrier. (b) Architecture with barrier at location 7.

finite-dimensional state space S . Each neuron has only local lateral connections with its neighboring neurons, which constitute a subset R_i in S , as shown in Fig. 1(a).

The neighbors of i are denoted by r^1, \dots, r^k . Each neuron is either a free space or a barrier location; the targets and the robot may occupy any free space. The connection weight w_{ij} from the j th neuron to the i th neuron is defined to be the Euclidean distance between the positions i and j in the space S . If the j th neuron is not in the neighboring set of the i th neuron, then $w_{ij} = \infty$. If the location of the j th neuron represents a barrier location, the connection weight is set as $w_{ij} = \infty$ [Fig. 1(b)]. The weights w_{ij} are symmetric: $w_{ij} = w_{ji}$, and are independent of the robot's motion.

For example, consider two networks, as illustrated in Fig. 1. There are no barriers in Fig. 1(a). The neighbor set for the neuron, labeled 6 in this figure, and the connection weights from it to all other neurons are

$$R_6 = \{1, 2, 3, 5, 7, 9, 10, 11\}$$

$$w_{6j} = \begin{cases} 1, & \text{if } j \in \{2, 5, 7, 10\} \\ \sqrt{2}, & \text{if } j \in \{1, 3, 9, 11\} \\ \infty, & \text{if } j \in \{4, 8, 12, 13, 14, 15, 16\}. \end{cases}$$

In comparison, with a barrier at location 7, as shown in Fig. 1(b), the neighbor set for the neuron labeled 6 and the connection weights from it to all other neurons are

$$R_6 = \{1, 2, 3, 5, 9, 10, 11\}$$

$$w_{6j} = \begin{cases} 1, & \text{if } j \in \{2, 5, 10\} \\ \sqrt{2}, & \text{if } j \in \{1, 3, 9, 11\} \\ \infty, & \text{if } j \in \{4, 7, 8, 12, 13, 14, 15, 16\}. \end{cases}$$

Note that, in general, the connection weight from the i th neuron to different neighboring neurons is not always the same.

For example, for a regular unit square grid, it is either 1 or $\sqrt{2}$. In this case, the neighboring set of the i th neuron can be divided into two subsets R_i^l and R_i^r , defined by

$$R_i^l = \{j \in R_i \mid w_{ij} = 1\}$$

$$R_i^r = \{j \in R_i \mid w_{ij} = \sqrt{2}\}$$

$$R_i = R_i^l \cup R_i^r.$$

For example, as illustrated in Fig. 1(a), R_6 , R_6^l , and R_6^r are

$$R_6^l = \{2, 5, 7, 10\}$$

$$R_6^r = \{1, 3, 9, 11\}$$

$$R_6 = R_6^l \cup R_6^r = \{1, 2, 3, 5, 7, 9, 10, 11\}.$$

B. Notions and Definitions

Before discussing the model itself, some notations and definitions about neuron firing and the firing time are defined as follows.

In the model, each neuron has a single output $Y(t)$.

Definition 1: A neuron is said to fire at time $T \geq 0$, if $\exists \epsilon \geq 0$, such that

$$Y_i(t) = \begin{cases} 0, & \text{when } T - \epsilon \leq t < T \\ 1, & \text{when } t = T \\ 0, & \text{when } T < t \leq T + \epsilon. \end{cases}$$

This time is denoted as t_{fire}^i .

Except for the case where a target is located at the neuron's location, the neuron will not fire until after one of its neighbors has fired. For any given neuron i , the first neuron in its neighbor set R_i to fire is denoted as R_i^F , and the firing time of this neuron is denoted as $t_{\text{fire}}^{R_i^F}$. If $j \in R_i$ and if t_{fire}^i is determined directly by t_{fire}^j , neuron i is said to fire on the stimulation of neuron j , and neuron j is called the parent of i . In this case, j is denoted as R_i^P and t_{fire}^i is denoted as $t_{\text{fire}}^{R_i^P}$.

Sometimes the parent of neuron i can be changed by the firing of some other neuron in R_i . For example, at time $t_{\text{fire}}^{R_i^F}$, R_i^F fires and becomes the parent of neuron i , preparing i to fire at some time in the future, say τ . If at some time before τ a neuron j in R_i other than R_i^F fires, and if, in the absence of neuron R_i^F , the firing of neuron j would stimulate neuron i to fire before τ , then j becomes the new parent of neuron i . In this situation, R_i^P is R_i^F until time t_{fire}^j at which point R_i^P becomes j .

Definition 2: For any neuron i , all neurons $j \in R_i$ that can potentially become the parent of neuron i are said to be in the *changing set* of neuron i , denoted as $\xi(i, t)$.

Note that $\xi(i, t)$ is time dependent. Initially, all neurons in R_i are in $\xi(i, t)$, but after time $t_{\text{fire}}^{R_i^F}$ only those neurons that could stimulate i to fire sooner are in the set. As t approaches the firing time for i , the set $\xi(i, t)$ becomes empty. Particularly, for a square grid with connections as shown in Fig. 1, $\xi(i, t)$ can be nonempty after time $t_{\text{fire}}^{R_i^F}$ only if $R_i^F \in R_i^r$, and further, $\xi(i, t) = R_i^l$ for $t_{\text{fire}}^{R_i^F} < t < t^*$, where t^* is some time prior to t_{fire}^i determined by the parameters of the system.

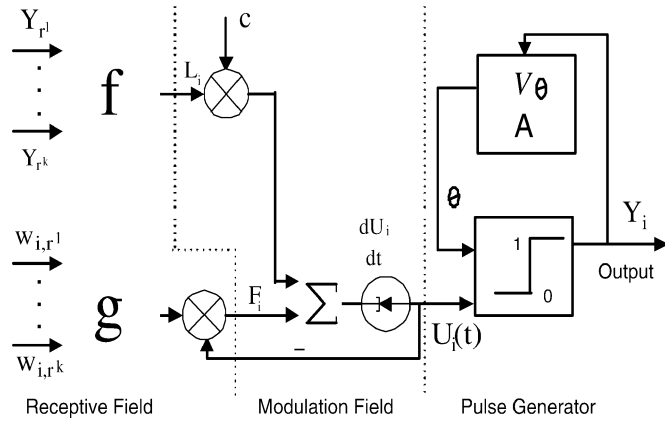


Fig. 2. MPCNN neuron model.

C. The Proposed Model for Robot Path Planning

The proposed MPCNN model is shown in Fig. 2. Each neuron i , $i = 1, 2, \dots, N$, has one output Y_i

$$Y_i(t) = \text{Step}(U_i(t) - \theta_i(t)) = \begin{cases} 1, & \text{if } U_i(t) \geq \theta_i \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where $U_i(t)$ and $\theta_i(t)$ are the internal activity and threshold function, respectively, t is the time, and N is the total number of neurons.

The major difference between the proposed threshold function and the typical PCNN is that the proposed function can receive simulated inputs from neighboring neurons, while the threshold function for typical PCNNs cannot. The threshold function of i th neuron can be expressed as

$$\theta_i(t) = \begin{cases} A_{\text{Init}}, & \text{when } t < t_{\text{fire}}^{R_i^F} \\ A_{ij}, & \text{when } t_{\text{fire}}^j \leq t < t_{\text{fire}}^i, j \in \xi(i, t) \\ V_\theta, & \text{when } t \geq t_{\text{fire}}^i \end{cases} \quad (2)$$

for all $i = 1, 2, 3 \dots N$, where A_{Init} and V_θ are positive constants. V_θ is set to be a very large value. The value A_{ij} is determined by

$$A_{ij} = \begin{cases} A^r, & \text{if } j \in R_i^r \\ A^l, & \text{if } j \in R_i^l \end{cases} \quad (3)$$

where A^r and A^l are two positive constants.

The linking field $L_i(t)$ and the feeding field $F_i(t)$ of each neuron are expressed as

$$L_i(t) = f(Y_{r^1}, \dots, Y_{r^k}, t) = \begin{cases} 0, & \text{when } t < t_{\text{fire}}^{R_i^F} \\ 1, & \text{else} \end{cases} \quad (4)$$

$$F_i(t) = -g(w_{ir^1}, \dots, w_{ir^k}, t)U_i(t) \quad (5)$$

where $w_{ir^1}, w_{ir^2}, \dots, w_{ir^k}$ are linking strengths from neuron i to its k neighboring neurons.

The internal activity $U_i(t)$ of each neuron determines the firing events and is determined by the initial value problem

$$\begin{cases} \frac{dU_i(t)}{dt} = F_i + CL_i = -g_i(w_{ir^1}, \dots, w_{ir^k}, t)U_i(t) + CL_i, & t \geq t_{\text{fire}}^i \\ U(t_{\text{fire}}^i) = 0 \end{cases}$$

where C is a positive constant and $g(\cdot)$ is a function of the outputs and connection weights from neighboring neurons and the time t . The function g is expressed as

$$g_i(w_{ir^1}, \dots, w_{ir^k}, t) = \begin{cases} 0, & \text{when } t < t_{\text{fire}}^{R_i^F} \\ \mu(w_{ij}), & \text{when } t \geq t_{\text{fire}}^j, \text{ for any } j \in \xi(i, t) \end{cases} \quad (6)$$

where $\mu(w_{ij})$ is given by

$$\mu(w_{ij}) = \frac{B}{w_{ij}} = \begin{cases} B, & \text{if } j \in R^l \\ \frac{B}{\sqrt{2}}, & \text{if } j \in R^r \end{cases} \quad (7)$$

and B is also a positive constant. Note, in particular, that if the parent of i changes, U_i is reset to zero, and we have (8) shown at the bottom of the page.

In general, the proposed neuron's model can be expressed by (1)–(8). But there is still one special instance. If more than one neuron fires at time $t_{\text{fire}}^{R_i^P}$, then that neuron that has the minimal linking strength to neuron i will be selected as R_i^P .

D. Theoretical Analysis of the Proposed Model

In this section, some theoretical results are mathematically deduced to show the performance of the proposed network when it is used to solve robot path-planning problems.

$$\begin{cases} U_i(t) = 0, & \text{when } 0 \leq t < t_{\text{fire}}^{R_i^F} \\ U_i(t) = 0, & \text{when } t = t_{\text{fire}}^j \text{ for any } j \in \xi(i, t) \\ \frac{dU_i(t)}{dt} = -\mu(w_{iR_i^P})U_i(t) + C, & \text{when } t \geq t_{\text{fire}}^{R_i^P}. \end{cases} \quad (8)$$

From the definitions above, it is clear that for any given neuron i , R_i^F and R_i^P are all in the neighboring set of i . Moreover, $t_{\text{fire}}^{R_i^F}$, $t_{\text{fire}}^{R_i^P}$, and t_{fire}^i satisfy the following inequality:

$$t_{\text{fire}}^{R_i^F} \leq t_{\text{fire}}^{R_i^P} < t_{\text{fire}}^i.$$

Theorem 1: For any given neuron i , if $R_i^P \in R_i^r$, then the changing set $\xi(i, t)$ is R_i^l for t satisfying

$$t_{\text{fire}}^{R_i^P} \leq t < t_{\text{fire}}^{R_i^P} + \frac{w_{ij}}{B} \ln \left(1 - \frac{A^l B}{w_{ij} C} \right) - \frac{w_{iR_i^P}}{B} \ln \left(1 - \frac{A^r B}{w_{iR_i^P} C} \right) \quad (9)$$

where $j \in R_i^l$ and $w_{i,j}$ is the linking strength from i to j .

Proof: It is clear that $U_i(t) = 0$ at time $t = t_{\text{fire}}^{R_i^P}$. Under the stimulation of neuron R_i^P , it follows from (2), (3), and (8) that

$$\frac{dU_i(t)}{dt} = -\frac{B}{w_{iR_i^P}} U_i(t) + C \quad (10)$$

$$\theta_i(t) = A_{iR_i^P} \quad (11)$$

for $t_{\text{fire}}^{R_i^P} < t < t_{\text{fire}}^i$. Solving the differential equation for U_i gives

$$U_i(t) = \frac{w_{iR_i^P} C}{B} \left(1 - \exp \left[-\frac{B}{w_{iR_i^P}} (t - t_{\text{fire}}^{R_i^P}) \right] \right) \quad (12)$$

for $t_{\text{fire}}^{R_i^P} < t < t_{\text{fire}}^i$. Thus, from Definition 1, we have

$$t_{\text{fire}}^i = t_{\text{fire}}^{R_i^P} - \frac{w_{iR_i^P}}{B} \ln \left(1 - \frac{A_{iR_i^P} B}{w_{iR_i^P} C} \right). \quad (13)$$

If $R_i^P \in R_i^r$, then

$$t_{\text{fire}}^i = t_{\text{fire}}^{R_i^P} - \frac{w_{iR_i^P}}{B} \ln \left(1 - \frac{A^r B}{w_{iR_i^P} C} \right).$$

However, if $j \in R_i^l$ and j fires after R_i^P , then, in the absence of neuron R_i^P , j would stimulate i to fire at time

$$\widetilde{t}_{\text{fire}}^i = t_{\text{fire}}^j - \frac{w_{ij}}{B} \ln \left(1 - \frac{A^l B}{w_{ij} C} \right).$$

Subtracting the above two expressions, it follows that time $\widetilde{t}_{\text{fire}}^i$ will be smaller than time t_{fire}^i provided

$$\left(t_{\text{fire}}^j - t_{\text{fire}}^{R_i^P} \right) < \frac{w_{ij}}{B} \ln \left(1 - \frac{A^l B}{w_{ij} C} \right) - \frac{w_{iR_i^P}}{B} \ln \left(1 - \frac{A^r B}{w_{iR_i^P} C} \right). \quad (14)$$

This concludes the proof. \blacksquare

From (2), (6), and (8), it is difficult to determine directly whether a neighbor j of i is in the changing set of i . However, Theorem 1 gives a straightforward and implementable test to determine this; namely, if (14) holds, then j becomes the new parent of i .

Theorem 2: Consider neurons i, j , and k , where $j \in R_i^l$ and $k \in R_i^r$. Suppose neuron i fires at time t_{fire}^i and that j and k

have not yet fired. If i remains the parent of both j and k , that is, $R_j^P = i$ and $R_k^P = i \forall t \geq t_{\text{fire}}^i$, and if

$$\text{Condition (I): } 0 < A_{ij} < \frac{w_{ij}}{B} C < V_\theta \quad (15)$$

$$\text{Condition (II): } 0 < A_{ik} < \frac{w_{ik}}{B} C < V_\theta \quad (16)$$

$$\text{Condition (III): } A_{ik} = \frac{w_{ik}}{w_{ij}} A_{ij} \quad (17)$$

all hold, then each neuron j and neuron k will each fire exactly once at some time after time t_{fire}^i , and the firing time of these two neurons satisfies

$$t_{\text{fire}}^k - t_{\text{fire}}^j = \frac{w_{ik}}{w_{ij}} \left(t_{\text{fire}}^j - t_{\text{fire}}^i \right). \quad (18)$$

Proof: For any neuron $m \subseteq R_i$ that has not fired, U_m is determined by $U_m(t_{\text{fire}}^i) = 0$ and

$$\frac{dU_m(t)}{dt} = -\frac{B}{w_{im}} U_m(t) + C \quad (19)$$

for all $t \geq t_{\text{fire}}^i$. Thus

$$U_m(t) = \frac{w_{im} C}{B} \left(1 - \exp \left[-\frac{B}{w_{im}} (t - t_{\text{fire}}^i) \right] \right) \quad (20)$$

for all $t \geq t_{\text{fire}}^i$. It follows that U_m is strictly increasing and

$$\lim_{t \rightarrow \infty} U_m(t) = \frac{w_{im} C}{B} < V_\theta. \quad (21)$$

By Conditions (I) and (II) and the continuity of U_m , there must exist $t_{\text{fire}}^m > t_{\text{fire}}^i$ satisfying

$$\begin{cases} U_m(t) < A_{im}, & \text{if } t < t_{\text{fire}}^m \\ U_m(t) = A_{im}, & \text{if } t = t_{\text{fire}}^m \\ A_{im} < U_m(t) < V_\theta, & \text{if } t > t_{\text{fire}}^m. \end{cases} \quad (22)$$

The output of neuron m can then be expressed as

$$Y_m(t) = \text{Step} (U_m(t) - \theta_m(t)) = \begin{cases} \text{Step} (U_m(t) - A_{im}) = 0, & \text{if } t < t_{\text{fire}}^m \\ \text{Step} (U_m(t) - A_{im}) = 1, & \text{if } t = t_{\text{fire}}^m \\ \text{Step} (U_m(t) - V_\theta) = 0, & \text{if } t > t_{\text{fire}}^m. \end{cases}$$

This shows that neuron m fires just once, regardless of whether $m \in R_i^r$ or $m \in R_i^l$.

For neurons k and j , it follows from (20) and (22) that

$$U_j(t_{\text{fire}}^j) = \frac{w_{ij}}{B} C \left(1 - e^{-\frac{B}{w_{ij}} (t_{\text{fire}}^j - t_{\text{fire}}^i)} \right) = A_{ij}$$

$$U_k(t_{\text{fire}}^k) = \frac{w_{ik}}{B} C \left(1 - e^{-\frac{B}{w_{ik}} (t_{\text{fire}}^k - t_{\text{fire}}^i)} \right) = A_{ik}.$$

Dividing the above expressions and using the fact that Condition (III) holds, we get

$$\frac{1 - e^{-\frac{B}{w_{ik}} (t_{\text{fire}}^k - t_{\text{fire}}^i)}}}{1 - e^{-\frac{B}{w_{ij}} (t_{\text{fire}}^j - t_{\text{fire}}^i)}} = 1.$$

Therefore

$$t_{\text{fire}}^k - t_{\text{fire}}^i = \frac{w_{ik}}{w_{ij}} (t_{\text{fire}}^j - t_{\text{fire}}^i).$$

This completes the proof of Theorem 2. \blacksquare

In fact, Theorem 2 shows that each neuron in the system fires just once upon the stimulation of a previously fired neighbor. This behavior is similar to the propagation of a wave. Smaller connection weights lead to earlier firing times. Thus, the proposed network can be used to solve some shortest path problems, such as robot path planning.

Theorem 3: Suppose *target* is the neuron that fired first in the system, and PATH_i and PATH_j are any two paths from *target* to neurons i and j (i can be the same as j). Assume that the firing wave requires time T_i to reach i along PATH_i , and it needs time T_j to reach j along PATH_j . If Conditions (I)–(III) in Theorem 2 hold, then

$$T_i \geq T_j \Leftrightarrow L(\text{PATH}_i) \geq L(\text{PATH}_j) \quad (23)$$

where $L(\text{PATH}_i)$ and $L(\text{PATH}_j)$ are the path lengths of PATH_i and PATH_j , which can be expressed as

$$\begin{aligned} L(\text{PATH}_i) &= \sum_{p \rightarrow q \in \text{PATH}_i} w_{pq} \\ L(\text{PATH}_j) &= \sum_{p \rightarrow q \in \text{PATH}_j} w_{pq}. \end{aligned}$$

Proof: If the wave propagates along PATH_i , then

$$\begin{aligned} T_i &= t_{\text{fire}}^i - t_{\text{fire}}^{\text{Target}} \\ &= (t_{\text{fire}}^i - t_{\text{fire}}^{R_i^P}) + (t_{\text{fire}}^{R_i^P} - t_{\text{fire}}^{R_{R_i^P}^P}) + \dots \\ &= \sum_{p \rightarrow q \in \text{PATH}_i} -\frac{w_{pq}}{B} \ln \left(1 - \frac{\alpha B}{C} \right) \\ &= -\frac{L(\text{PATH}_i)}{B} \ln \left(1 - \frac{\alpha B}{C} \right) \end{aligned}$$

where $\alpha = A_{ij}/w_{ij}$, which by Condition (III) is a constant. Similarly, if the wave propagates along PATH_j

$$T_j = -\frac{L(\text{PATH}_j)}{B} \ln \left(1 - \frac{\alpha B}{C} \right).$$

Since Conditions (I) and (II) hold

$$-\ln \left(1 - \frac{\alpha B}{C} \right) > 0. \quad (24)$$

Thus:

- (I) if $T_i \geq T_j$, then $L(\text{PATH}_i) \geq L(\text{PATH}_j)$;
- (II) if $L(\text{PATH}_i) \geq L(\text{PATH}_j)$, then $T_i \geq T_j$.

This completes the proof of Theorem 3. \blacksquare

As a consequence of the above theorem, since the parent of a neuron is the neighbor through whom the firing wave propagated, the path defined by the sequence of parents from any neuron i to *target* is the shortest path available.

III. ALGORITHMS FOR ROBOT PATH PLANNING

In this section, the algorithms for the robot, target, and barrier movement are presented, which are based on the proposed MPCNN model.

A. Static Environment

In the static case, where barriers and targets do not move, the proposed MPCNN guarantees that the firing event of the target neuron can be propagated to the whole state space through local connectivity of neurons, while the state of the obstacle neurons remains unfired. For each neuron i , except the target neuron, if the firing wave reaches it from neuron j at a certain time, then j is recorded as the parent of neuron i , marked as R_i^P . When the firing wave reaches the position of the robot, the robot neuron will fire and a path from the robot to the target is determined by successively moving to the parent of each neuron. This path is the shortest path from the robot to the target.

For clarity, the following algorithm is presented to show the steps of the proposed network when it is used for robot path planning. The innovative idea about this algorithm is that it can be done in parallel, resulting in efficient computation of the whole system, especially in dynamic environments.

- 1) Initialize the network: Set A , B , C , and V_θ according to Conditions (I)–(III) in Theorem 2. Set $\theta_i(0) = A_{\text{Init}}$. Set $U_i(0) = 0$ for all $i = 1, 2, \dots, N$, and $i \neq \text{Start}$, and $U_{\text{Start}}(0) = A_{\text{Init}}$; this makes neuron *Start* fire at time 0.
- 2) Start the network: For each neuron i in the network:
 - a) calculate $U_i(t)$ according to (8);
 - b) calculate $Y_i(t)$ according to (1);
 - c) when $Y_i(t) = 1$:
 - i) set $\theta_i(t) = V_\theta$;
 - ii) for $j \in R_i$, if $i \in \xi(j, t)$ set $\theta_j(t) = A_{ji}$;
 - iii) save the parent neurons that have stimulated neuron i in R_i^P .
- 3) If $Y_{\text{Robot}} = 1$, stop the network.

When the fire state propagates to the robot's position, there exists a sequence of parents starting from target neuron and ending at the robot neuron. Through this parent information, the robot's movement is easily determined. For example, suppose the robot's location $\phi(t)$ is specified as an index of one of the points on the grid, and is a function real time $t \geq t_0$ (initially, $\phi(t_0) = \text{Robot}$). Assume that the robot's travel path is updated at a set of real-time values $t_1 < t_2 < t_3 < \dots$, and the robot's actual location for $t \in (t_k, t_{k+1})$ is somewhere between the grid points $\phi(t_k)$ and $\phi(t_{k+1})$. At time t_k , the robot is located at $\phi(t_k)$, and the next location $\phi(t_{k+1})$ is determined by

$$\phi(t_{k+1}) = R_{\phi(t_k)}^P. \quad (25)$$

Now let us consider a real example as illustrated in Fig. 3, where 7 is the location of the robot, 25 is the location of the target, and 13 and 14 are the locations of barriers. In this example, set $B = 1$, $A^r = \sqrt{2}C(1 - e^{-1})$, $A^l = C(1 - e^{-1})$,

$C = 10$, and $V_\theta = 20$, which satisfy the conditions in Theorem 2. Consider the dynamics of each neuron step by step.

1) At time 0, neuron 25 fires, thus $t_{\text{fire}}^{25} = 0$. Then

$$\begin{aligned} \frac{dU_i(t)}{dt} &= -U_i(t) + 10, & i = 20, 24 \\ \frac{dU_{19}(t)}{dt} &= -\frac{1}{\sqrt{2}}U_{19}(t) + 10 \end{aligned}$$

and

$$\begin{aligned} U_i(t) &= 0, & i = 1, 2, \dots, 18, 21, 22, 23 \\ R_{19}^P &= 25 & R_{20}^P = 25 & R_{24}^P = 25. \end{aligned}$$

2) At time $t = 1$, for $i = 20$ and 24, we have

$$U_i(1) = C(1 - e^{-1}) = A_{i,25} = A^l$$

and thus

$$Y_{20}(1) = 1 \text{ and } Y_{24}(1) = 1.$$

Then

$$\begin{aligned} \frac{dU_i(t)}{dt} &= -U_i(t) + 10, & i = 15, 23, & t \geq 1 \\ \frac{dU_i(t)}{dt} &= -\frac{1}{\sqrt{2}}U_i(t) + 10, & i = 18, & t \geq 1. \end{aligned}$$

Also

$$R_{15}^P = 20 \quad R_{18}^P = 24 \quad R_{23}^P = 24.$$

Here, 20 and 24 are in the neighboring set of 19 but (14) does not hold, so neither of them is in the *changing set* of 19; the parent of 19 will not be changed.

3) When $t = \sqrt{2}$

$$U_i(t) = \sqrt{2}C(1 - e^{-1}) = A_{i,25} = A^r$$

for $i = 19$. Thus

$$Y_{19}(t) = 1.$$

4) When $t = 2$

$$U_i(t) = C(1 - e^{-1}) = A_{15,20} = A_{23,24} = A^l$$

for $i = 15, 23$, thus

$$Y_{15}(t) = 1 \quad Y_{23}(t) = 1$$

$$\frac{dU_i(t)}{dt} = -U_i(t) + 10, \quad \text{for } i = 10, 22 \text{ and } t \geq 2$$

$$\frac{dU_i(t)}{dt} = -\frac{1}{\sqrt{2}}U_i(t) + 10, \quad \text{for } i = 9, 17 \text{ and } t \geq 2$$

$$R_{17}^P = 23 \quad R_{22}^P = 23 \quad R_9^P = 15 \quad R_{10}^P = 15.$$

5) When $t = 1 + \sqrt{2}$

$$U_{18}(t) = \sqrt{2}C(1 - e^{-1}) = A_{18,24} = A^r$$

thus

$$Y_{18}(t) = 1$$

$$\frac{dU_{12}(t)}{dt} = -\frac{1}{\sqrt{2}}U_{12}(t) + 10, \quad \text{for } t \geq 1 + \sqrt{2}$$

$$R_{12}^P = 18.$$

6) When $t = 3$

$$U_i(t) = C(1 - e^{-1}) = A_{10,15} = A_{22,23} = A^l$$

for $i = 10, 22$, thus

$$Y_{10}(t) = 1 \quad Y_{22}(t) = 1$$

$$\frac{dU_i(t)}{dt} = -U_i(t) + 10, \quad \text{for } i = 5, 21 \text{ and } t \geq 3$$

$$\frac{dU_i(t)}{dt} = -\frac{1}{\sqrt{2}}U_i(t) + 10, \quad \text{for } i = 4, 16 \text{ and } t \geq 3$$

$$R_5^P = 10 \quad R_4^P = 10 \quad R_{16}^P = 22 \quad R_{21}^P = 22.$$

7) When $t = 2 + \sqrt{2}$

$$U_i(t) = \sqrt{2}C(1 - e^{-1}) = A_{9,15} = A_{17,23} = A^r$$

for $i = 9, 17$, thus

$$Y_9(t) = 1 \quad Y_{17}(t) = 1$$

$$\frac{dU_8(t)}{dt} = -U_8(t) + 10, \quad \text{for } t \geq 2 + \sqrt{2}$$

$$\frac{dU_i(t)}{dt} = -\frac{1}{\sqrt{2}}U_i(t) + 10,$$

for $i = 3, 11$ and $t \geq 2 + \sqrt{2}$

$$R_8^P = 9 \quad R_3^P = 9 \quad R_{11}^P = 17.$$

8) When $t = 1 + 2\sqrt{2}$

$$U_{12}(t) = \sqrt{2}C(1 - e^{-1}) = A_{12,18} = A^r$$

thus

$$Y_{12}(t) = 1$$

$$\frac{dU_7(t)}{dt} = -\frac{1}{\sqrt{2}}U_7(t) + 10 \text{ and } t \geq 1 + 2\sqrt{2}$$

$$R_7^P = 12$$

9) When $t = 4$

$$U_i(t) = C(1 - e^{-1}) = A_{5,10} = A_{21,22} = A^l$$

for $i = 5, 21$, thus

$$Y_5(t) = 1 \quad Y_{21}(t) = 1.$$

10) When $t = 3 + \sqrt{2}$

$$U_i(t) = \sqrt{2}C(1 - e^{-1}) = A_{4,10} = A_{16,22} = A^r$$

for $i = 4, 16$, thus

$$Y_4(t) = 1 \quad Y_{16}(t) = 1.$$

11) When $t = 2 + 2\sqrt{2}$

$$U_7(t) = C(1 - e^{-1}) = A_{7,12} = A^l, \text{ thus } Y_7(t) = 1$$

12) Stop. Then, robot's movement is

$$7 \rightarrow R_7^P \rightarrow R_{R_7^P}^P \rightarrow \dots$$

that is $7 \rightarrow 12 \rightarrow 18 \rightarrow 24 \rightarrow 25$.

This example is just to show the steps of the proposed method. The spreading speed of the wave can be modulated by the values of A^r and A^l . Small values of A^r and A^l lead to a faster speed. For example, if $A^r = \sqrt{2}C(1 - e^{-0.01})$ and $A^l = C(1 - e^{-0.01})$

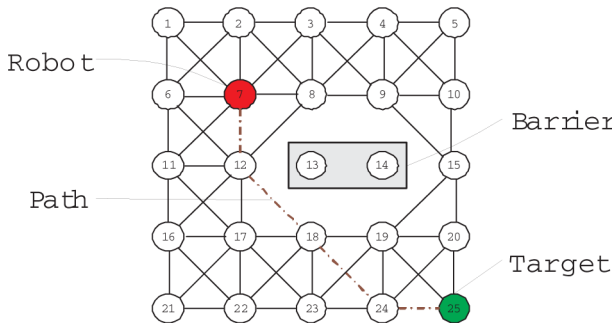


Fig. 3. Example network. Numbers 7 and 25 are the locations of the robot and the target, respectively. Numbers 13 and 14 are the locations of barriers. The dotted line is the resulting path for the robot.

in the above example, it only needs $(2 + 2\sqrt{2})/100$ seconds to find the path from 7 to 25.

B. Dynamic Environments

In the situation where targets and/or barriers are moving, our proposed algorithm can also perform well with few changes.

1) *Target Moving*: In the proposed model, the firing wave always propagates from the target to the robot. A new wave would be spread out whenever the target location is changed. Each wave can reach the robot in a very small time. Thus, the robot would acquire the updated path to the target in a short time determined by the multisensor system update frequency. If this frequency is sufficiently fast, and the robot moves at a sufficiently faster speed relative to the target, the robot will be able to track and catch the moving target.

2) *Obstacle Moving*: As mentioned above, neurons in barrier locations have no direct connections to other neurons, and never fire in the whole process of the algorithm. When a barrier moves from a point to a new one, some connections in the network change as well. The neuron at the old point will connect with its neighboring neurons, while the neuron in the new point will be isolated from its neighboring neurons. The new optimal path for the robot is achieved by a new running of the network.

IV. SIMULATION STUDIES

In this section, experiments are carried out to illustrate the effectiveness and efficiency of the proposed network. In static environments, a maze-solving type of problem and avoidance of a concave U-shaped obstacle are studied to examine the global optimization property of the proposed model. In dynamic environments, the moving target tracking problem and the varying obstacles avoidance problem are examined. In addition, comparison with other models are made.

A. In Static Environments

The first experiment is carried out to test the propagation of the firing event from the starting neuron to others in a static environment without any obstacles. A 20×20 network is considered with target location (10, 10) as shown in Fig. 4(a). The parameters of the network are set as: $A_{Init} = 1$, $A^r = 1$, $A^l = \sqrt{2}$, $B = 10$, and $C = 100$, which satisfy the conditions in Theorem 2. The target neuron (10, 10) fires first, with the firing

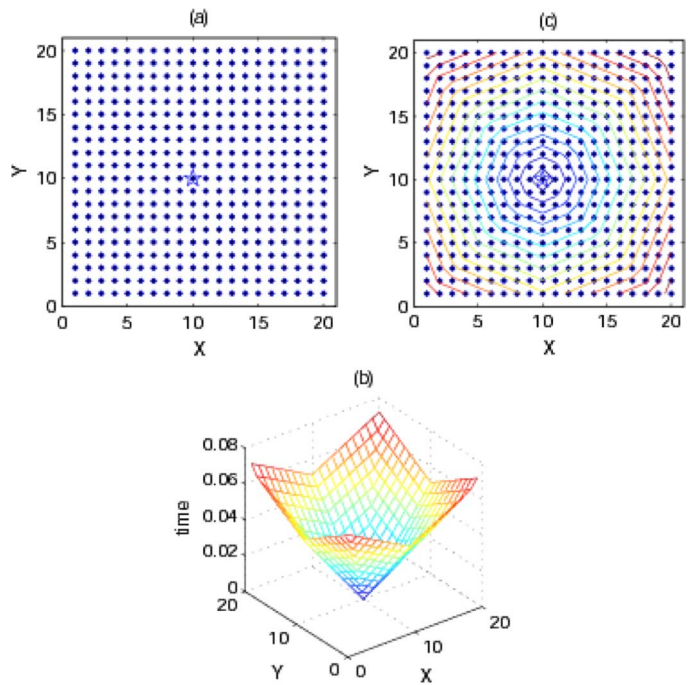


Fig. 4. Propagation of the firing wave on a given network. (a) Topology of the network. (b) Fire time of each neuron in the network. (c) Propagation of the firing wave.

event propagating from it to other neurons in the network. The firing time of each neuron is directly proportional to the distance between it and the target neuron as shown in Fig. 4(b) and (c).

It is clear that potential-field-based methods and other strictly local avoidance schemes cannot deal with U-shaped obstacle problems [66]. While under the proposed model, the robot is not trapped behind the barrier since barriers have no connections to neighbors. In this simulation, two U-shaped barriers are placed in the environment, and the robot and the target are located behind these two U-shaped barriers, as shown in Fig. 5. The neural network has 30×30 topologically organized neurons. The model parameters are chosen as $A_{Init} = 1$, $A^r = 1$, and $A^l = \sqrt{2}$ for the threshold function, and $B = 10$ and $C = 100$ for the dynamic of internal activities. A generated trajectory is shown in Fig. 5.

The proposed model is also applied to solve a maze-type problem. A maze on a 30×30 grid is tested. The neural network has the same model parameters as in the previous simulation. The generated globally optimal solution is shown in Fig. 6.

B. To Catch a Moving Target

In this simulation, the proposed network is applied to a real-time tracking of a moving target without any obstacles. The neural network has 30×30 neurons with the same parameters as in previous cases. In this simulation, the target moves at a speed of 0.5 grid units/s from the initial location at (5, 1), and stops at (25, 25). The route of the target is indicated by circles as shown in Fig. 7. The robot starts from the point (1, 1). The generated robot path is displayed by stars. It is clear that, when a robot tracks a moving target, the relative moving speed between the target and the robot has an important influence on the

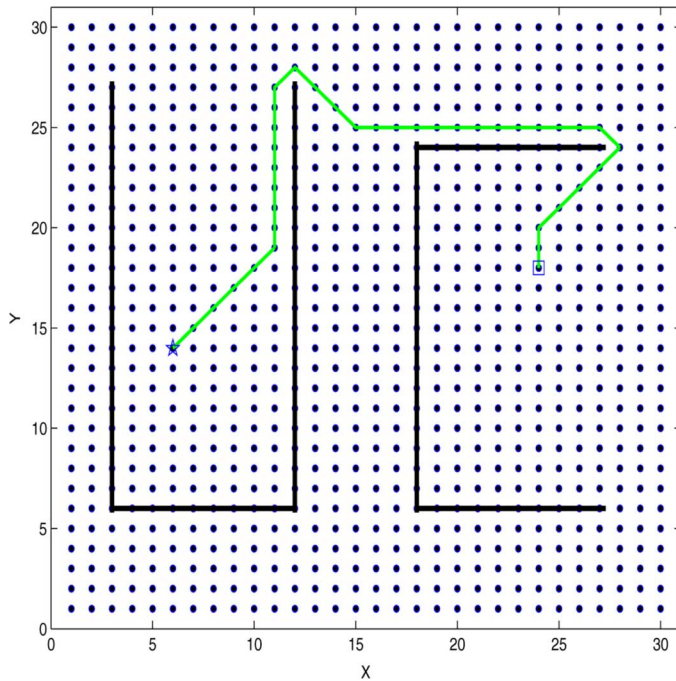


Fig. 5. Generated path trajectory with two U-shaped obstacles (dark lines). The initial robot location is a star, the target is a square, and the robot's path between them is shown.

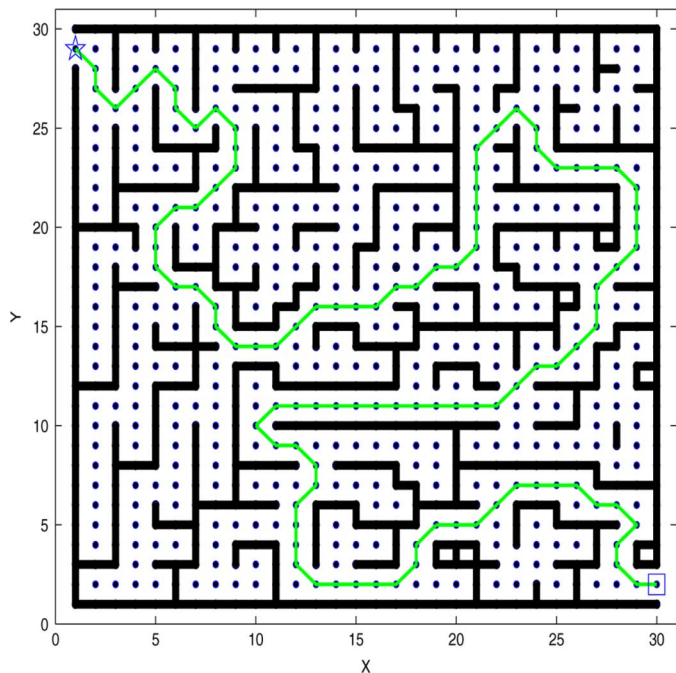


Fig. 6. Solution to a maze-type problem. The initial robot location is represented by a star and the target by a square.

generation of the tracking path. In this simulation, four different moving speeds for the robot are tested and compared.

If the robot moves at a speed of 0.25 grid units/s, which is half the target speed, the robot takes 28 steps and 112 s to reach the moving target. The generated robot path is shown in Fig. 7(a). If the robot moves at 0.35 grid units/s, the robot takes 33 steps and only 94 s to catch the moving target; the generated real-time path

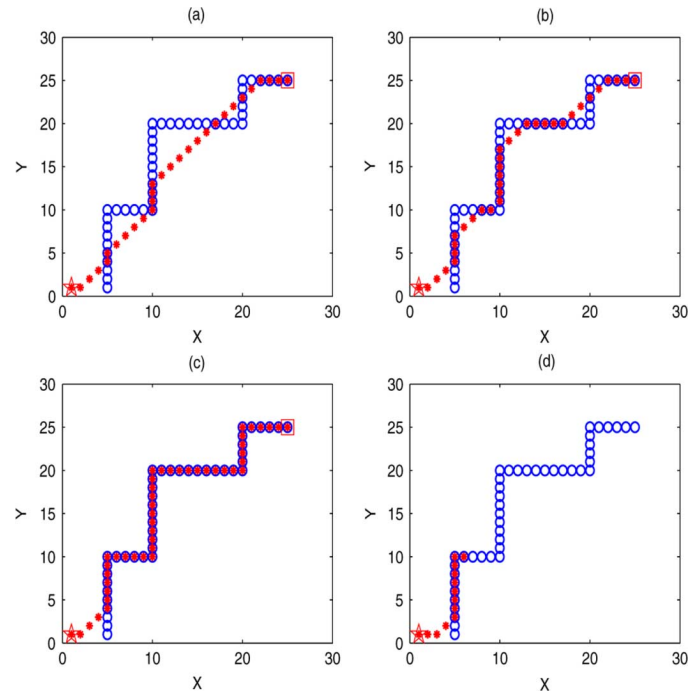


Fig. 7. Generated real-time paths of the robot to catch the target; the moving route of the target is indicated by open circles, and the robot paths are indicated by stars.

is shown in Fig. 7(b). Fig. 7(c) displays the generated path when the robot moves at a speed of 0.5 grid units/s, which is the same speed as the target. In this case, the robot takes 45 steps and 90 s to catch the moving target. When the robot moves faster than the target, at a speed of 0.55 grid units/s, the robot only takes 12 steps and 22 s to catch it. The generated real-time path is shown in Fig. 7(d). This simulation shows that when the robot moves at a slower speed than the target [Fig. 7(a) and (b)], the path of the robot is substantially different and shorter than the target's path. This is because the system is updating the shortest path very rapidly and, as the target moves, the shortest path to the robot changes. In the first three cases, the robot captures the target after the target stops. Clearly, if the target were to continue moving and the robot were too slow, it would never catch the target even though it would move in an almost straight line toward the moving target. As the robot's speed increases, it tends to track the target's path more closely, thus, in this simulation, increasing the robot's traveling distance. In the case where the robot's speed is the same as the target [Fig. 7(c)], the robot quickly gets close to the target and then tracks directly behind it until the target stops and the robot captures it. When the speed of the robot is faster than the target [Fig. 7(d)], the robot is able to gain ground on the target and capture it well before it completes its trip to (25, 25).

A more complex situation is shown by the simulation in Fig. 8. In this simulation, the network has the same topology and model parameters as before. The target moves from location (15, 13) with speed 0.65 grid unit/s, along a spiraling path shown in Fig. 8, and stops at location (2, 29). The robot tracks the moving target starting from (18, 18). When the robot moves at a speed of 0.35 grid unit/s, it takes 105 steps and 300 s to catch target; the generated path is shown in Fig. 8(a). When the

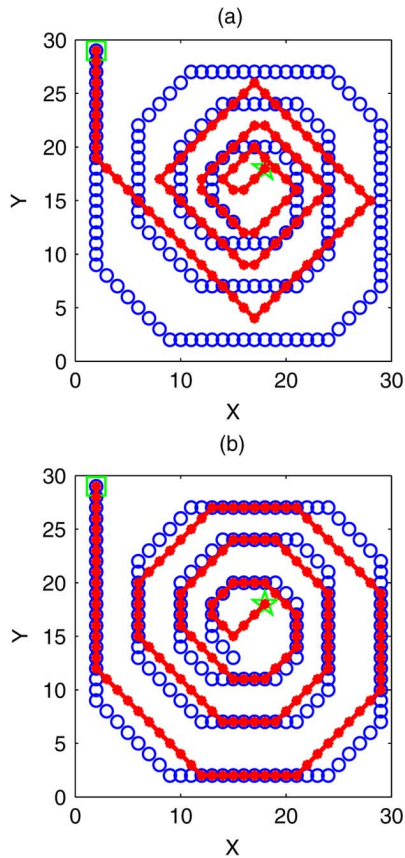


Fig. 8. Generated real-time paths of the robot to catch the outward spiraling target. The route of the target is indicated by open circles, and the robot paths are indicated by small stars. (a) Robot speed 0.35 unit/s. (b) Robot speed 0.55 unit/s.

robot moves at 0.55 unit/s, it take 151 steps and 274 s to catch the target; the generated path is shown in Fig. 8(b).

The robot is able to track the moving target without any collisions in an environment that has some static obstacles. Choosing the same network architecture and parameters as in Fig. 8, a line obstacle and a U-Shaped obstacle are placed in the workspace, as shown in Fig. 9, indicated by the black lines. The generated catching path of robot is shown in Fig. 9(a) and (b), where the speed of the robot is 0.35 and 0.55 grid unit/s, respectively. The robot takes more steps and time to catch the target, compared to that in Fig. 8.

C. Path Generation to Catch a Moving Target With Varying Barrier

In this section, the proposed model is applied to a more complex case, where both the target and the barrier are moving. The neural network is chosen with the same architecture and the parameters are as those in the previous cases: a 30×30 grid of neurons, and parameters $A_{\text{Init}} = 1, A^r = 1, A^l = \sqrt{2}, B = 10,$ and $C = 100$.

In this simulation, a static U-shaped obstacle, a static linear obstacle, and a moving U-shaped obstacle are laid in the workspace together, as shown in Fig. 10. The outer U-shape is static, while the inner U-shape circulates with a counterclockwise speed of 0.5 grid unit/s. The linear obstacle is

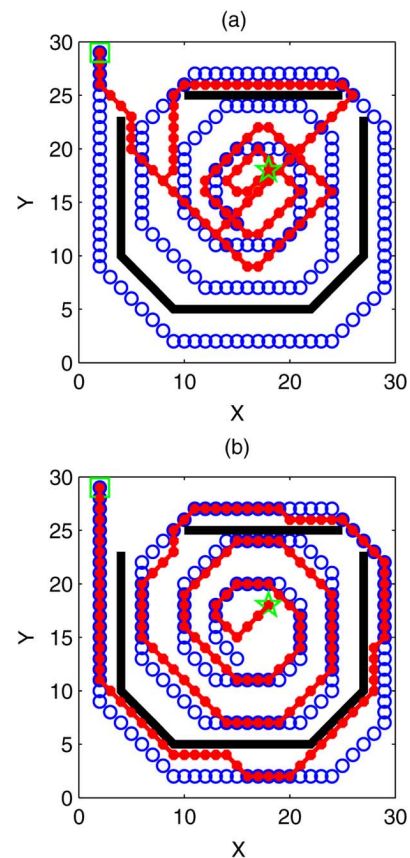


Fig. 9. Generated real-time paths of the robot to catch the outward spiraling target in the presence of obstacles (dark lines). The route of the target is indicated by the open circles, and the robot paths are indicated by stars. (a) Robot speed 0.35 unit/s. (b) Robot speed 0.55 unit/s.

initialized at the position (2,4) to (15,4), and continuously moves back and forth along the line between (2,4) and (29,4), at a speed of 0.7 grid unit/s. The target itself moves around the outside of the grid in a clockwise direction, at a speed of 0.5 grid unit/s, and with an initial starting location (15,30).

The robot moves at a speed of 0.6 grid unit/s, and starts at the location (15,15), as shown in Fig. 10(a). The robot begins moving along the current shortest path to catch the target. As the barriers move, the topology of the space changes and consequently so does the shortest path. At first, the robot moves north-east, but it turns to the north-west some time before $t = 12$ s [Fig. 10(b)] due to the movement of the inner U-shaped barrier. At $t = 16$ s, the linear barrier reaches the right bottom of the outer U-shaped barrier, cutting the old shortest path. The new shortest path now goes around the left-hand side of the grid [Fig. 10(c)]. Sometime before $t = 32$ s [Fig. 10(d)], the shortest path has again shifted to the right-hand side of the grid since the target is moving further toward the right. The robot exits out of the inside of the inner U-shaped barrier and travels east along the linear barrier [Fig. 10(e) and (f)]. At $t = 58$ s, the linear barrier again reaches the east side of the stationary U-shaped barrier and cuts the shortest path [Fig. 10(g)]. The new shortest path now skirts to the left of the linear barrier and the robot moves around it at $t = 62$ s [Fig. 10(h)]. Finally, the robot catches the target at time $t = 92$ s [Fig. 10(i)].

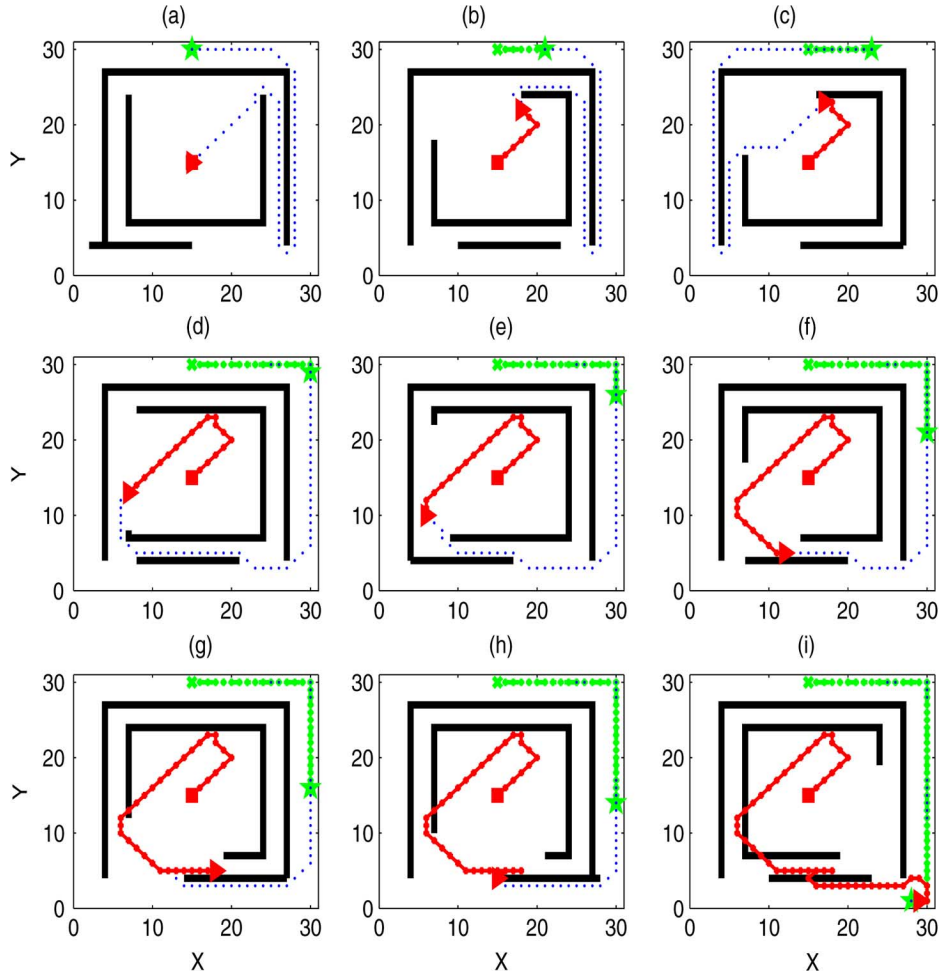


Fig. 10. Generated real-time paths of the robot to catch a moving target with varying barriers (bold lines). The robot's initial and current locations are a square and a triangle, respectively. The current target location is a star. The dotted line is the current optimal path to the target.

D. Comparison to Other Models

The proposed model is compared with the shunting model of Yang and Meng [38] and the dynamic model of Willms and Yang [22]. As stated in [38], the dynamic of each neuron in Yang and Meng's model can be described by a shunting equation

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i) \left(E\delta_{i,\text{target}} + \sum_{j \in B_i} \frac{\mu}{d_{ij}} [x_j]^+ \right) - (D + x_i)E\delta_{i,\text{barrier}} \quad (26)$$

where A , B , D , μ , and E are positive constants, $\delta_{i,\text{set}}$ is 1 if i is a member of set and zero otherwise, and $[x]^+$ is defined as $\max\{x, 0\}$. The barrier neurons do not influence their neighbors since they receive a large negative input. The positive neural activity can propagate from the target's location to the whole space through lateral neural connections, while the negative activity remains local only.

The Yang and Meng model can generate collision-free trajectories in both static and dynamic environments. However, in this model, the network dynamics and the quality of a generated path significantly depend on the choice of parameters.

This model is also less likely to find the optimal path than the proposed MPCNN model. Consider the same example as in [22]; the grid is shown in Fig. 11(a). Willms and Yang noted that, 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 Yang and Meng's model ($A = 10, B = D = \mu = 1, E = 100$) has a higher neural activity along the right path than the left [22]. Consequently, under Yang and Meng's model, the robot chooses the right path. In contrast, under the MPCNN model, the robot always moves along the shortest path from it to the target. Fig. 11(b) shows the neural activity and robot path for Yang and Meng's model. The firing time landscape and the path chosen by the robot under the MPCNN model is shown in Fig. 11(c).

Furthermore, Yang and Meng's model is greatly dependent on the relationships between model parameters. In some cases, it cannot even find a path to reach the target. In contrast, the proposed MPCNN model constrains the parameters through Conditions (I)–(III) that guarantees that the robot always proceeds to the target along the shortest path.

Concerning computational burdens, Willms and Yang's distance propagating model [22] more quickly calculates the optimal path from a robot to a target than Yang's model. In static

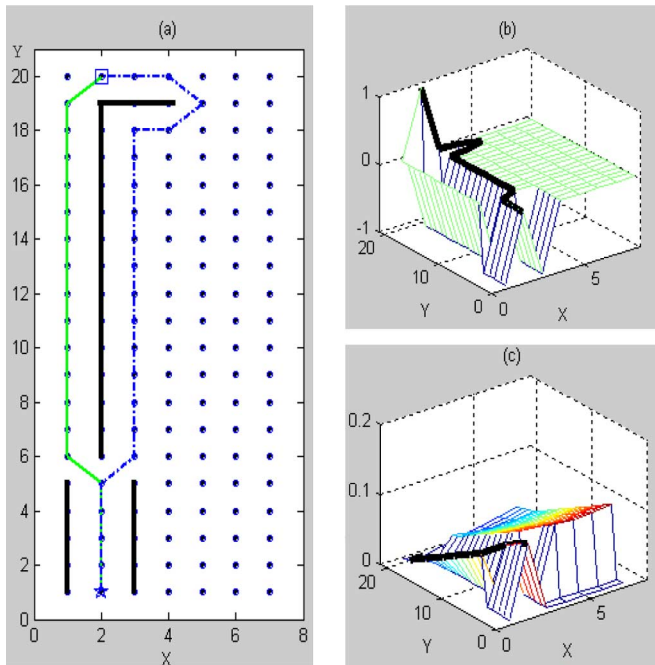


Fig. 11. Comparison to Yang and Meng's model. (a) Robot is represented by star and target is represented by square. The left path is the shortest path achieved by MPCNN model, while the right is the result path by Yang and Meng's model. (b) Neural activity landscape and the path chosen by the robot in the Yang and Meng model. (c) Firing time landscape and the path chosen by the robot using MPCNN.

environments, the proposed MPCNN model needs less computational time to propagate the firing event from the target to the robot compared with Willms and Yang's model, especially when the network scale is larger. In the distance propagating model, the distance information is updated in each step for each neuron, which leads to many comparing computations, while in our MPCNN model, the firing event propagates to its neighbors without any comparing computations. For example, on a 100×100 grid with the robot initially located at (20, 20) and a target at (80, 80) the central processing unit (CPU) time required to propagate the distance information from the target to the robot via Willms and Yang's model was 2.3741 s, while for the MPCNN, it was 1.9211 s. It must be pointed out that the Willms and Yang model is intended for highly dynamic environments whereas this simple test is in a static environment. These programs were coded serially in MATLAB 6.5 and were run on an IBM compatible Pentium 4 PC, 2.66-GHz and 512-MB RAM.

The proposed MPCNN model is also compared to the dynamic wave expansion neural network (DWENN) proposed by Lebedev *et al.* [21], which resembles other biologically motivated neural network models [38] in many aspects. Both the MPCNN and DWENN models are topologically organized with only local lateral connections among neurons and both propagate waves of information outward from the target. However, in the DWENN model, the waves propagate to their neighbors at a constant speed regardless of the distance to the neighbor, thus the DWENN actually finds paths that are a minimal number of grid points away rather than a minimal distance. This wave

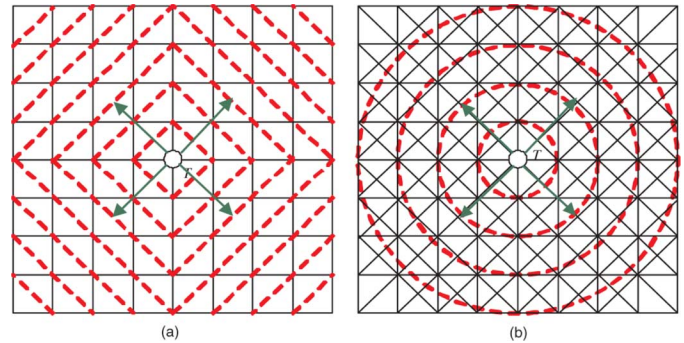


Fig. 12. Wave propagations: (a) in DWENN and (b) in MPCNN.

spreading of the DWENN model and our MPCNN model is illustrated in Fig. 12. In this figure, the DWENN model has only nearest neighbor connections (four neighbors for each interior grid point rather than eight) since this is the format used by Lebedev *et al.* [21]. If eight neighbors were used, then the waves for DWENN would look like squares aligned with the coordinate axes rather than diagonally. In either case, the wave fronts for the DWENN model are not a constant distance from the source whereas they are for the MPCNN model. The update rule at each time step of the DWENN model is given in a discrete manner, so it can be viewed as a discrete-time dynamic system. Thus, it is more convenient to implement using a soft computing module. The proposed MPCNN model is inspired by the dynamic properties of the PCNN, which is capable of emulating the behavior of a biological neural system. Each neuron in the proposed MPCNN model works as a leaky integrator, which is more convenient to implement using a digital very large scale integration (VLSI) [67].

The DWENN model's computational complexity grows linearly as a function of the number of neurons in the network field [21]. This is also true for Yang and Meng's model [38] and Willms and Yang's model [22]. The computational complexity of the proposed model is related to the length of the shortest path, and is independent of the workspace complexity and the number of existing paths in the map. All of these models expect immediate knowledge of current environment in order to give accurate results. The DWENN model and Willms and Yang's model [22] are parameter free and do not require a learning process. In contrast, the proposed model is not parameter free, but Theorem 2 gives an exact condition for the real value of the parameters, so it also does not need any learning process. As stated earlier, Yang and Meng's model [38] gives results that are dependent on its system parameters.

V. PARAMETERS

The sensitivity of a system to parameter variations is a factor of prime importance to be considered when proposing a model. There are few parameters in the proposed model: A^l , A^r , B , C , and V_θ . This section analyzes the parameter sensitivity of the proposed model.

First, consider the effect of the parameters B and C on the internal activity $U_i(t)$. From Section II, we can see that, for any neuron i , the value of $U_i(t)$ remains at 0 until the firing wave in

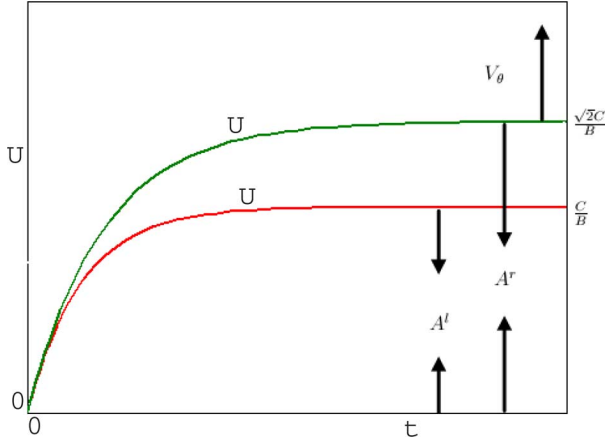


Fig. 13. Reasonable range of values for parameters A^l , A^r , V_θ , and C/B , that is, $0 < A^l < C/B$, $0 < A^r < (\sqrt{2}C)/B$, and $V_\theta > (\sqrt{2}C)/B$.

the network first reaches one of its neighboring neurons. After the neighbor R_i^F fires, the value of $U_i(t)$ is determined by

$$\frac{dU_i(t)}{dt} = -\mu(\omega_{iR_i^F})U_i(t) + C.$$

Clearly, unless the parent of i changes, $U_i(t)$ increases monotonically toward an upper limiting value of either C/B or $(\sqrt{2}C)/B$ as $t \rightarrow +\infty$ depending on whether R_i^F is in R_i^l or R_i^r , respectively. If R_i^F changes at time τ , then U_i is reset to zero at τ and monotonically increases as described above for $t > \tau$.

From Theorem 2, it follows that A^l and A^r determine the firing events of the neurons in the network. If $A^l > (C/B)$ and $A^r > (\sqrt{2}C)/B$, no neuron in the network would fire, while if $V_\theta < (\sqrt{2}C)/B$, some neuron would fire more than once. Fig. 13 shows the reasonable range of values for parameters A^l , A^r , V_θ , and C/B , which guarantee that the firing wave progresses through the whole network.

The spreading speed of the firing wave in the network (relative to the internal time variable in the model equations as opposed to the CPU time or wall clock) is determined by the values of A^l , A^r , B , and C . In general

$$t_{\text{fire}}^i = \begin{cases} t_{\text{fire}}^{R_i^P} - \frac{1}{B} \ln \left(1 - \frac{A^l B}{C} \right), & \text{if } R_i^P \in R_i^l \\ t_{\text{fire}}^{R_i^P} - \frac{\sqrt{2}}{B} \ln \left(1 - \frac{A^r B}{\sqrt{2}C} \right), & \text{if } R_i^P \in R_i^r \end{cases}$$

but by Condition (III), $A^l/1 = A^r/\sqrt{2} = \alpha$, so that the speed of propagation of the wave between neighboring neurons is given by

$$\frac{B}{\ln \left(\frac{1}{1 - \alpha B/C} \right)}.$$

By Conditions (I) and (II), $1/(1 - \alpha B/C) > 1$ so that this speed is positive. This inequality constrains the parameters A , B , and C , but, in general, small values of A^l and A^r (that is, small values of α) lead to faster wave speeds.

TABLE II
DEPENDENCE OF THE INTERNAL PROPAGATION TIME
OF THE MPCNN ON PARAMETERS

B	C	Internal Propagation Time
10	50	1.8960 s
10	100	0.8940 s
10	200	0.4380 s
1	100	0.8520 s
50	100	1.1760 s

To illustrate the wave speed for various values of B and C , a network with 10 000 (100×100) neurons is again considered where the target is fixed in the location (80, 80) and the location of the robot is set to be (20, 20). The other parameters are: $A_{\text{Init}} = 1$, $U_{\text{target}} = A_{\text{Init}} = 1$, $A^r = 1$, and $A^l = \sqrt{2}$. Five hundred runs are executed for each set of parameter values B and C . Table II shows the comparison results. The simulation results show that the MPCNN internal propagation time decreases as the value of C increases, and, to a lesser extent, as the value of B decreases.

VI. CONCLUSION

This paper presents an MPCNN model for real-time collision-free path planning of mobile robots in dynamic environments. The proposed model is topologically organized with only local lateral connections among neurons. It proves that the generated spiking wave in the network spreads at constant speed, which guarantees that the wave propagates along the shortest path from the target to the robot. The proposed approach uses the parallel pulse transmission characteristic of PCNNs to find the shortest path quickly. The computational complexity is only related to the length of the shortest path, and is independent of the workspace complexity and the number of existing paths in the map. Each neuron in the model propagates the firing event to its neighboring neuron without any comparing computations.

The main weakness of the proposed model is that the global knowledge of the current environment is required, which is not always available in some real-world applications. Similar to many existing path-planning approaches [21], [22], [38], it is assumed that the current environment is completely and accurately known. In indoor environments, the environment knowledge would be obtained through sensor measurement and multisensor fusion. In outdoor or open environments, real-time map building is needed to explore the workspace. Thus, it will be challenging to apply the proposed path-planning model to situations where the environment knowledge is partially or completely unknown.

APPENDIX

A typical neuron of PCNNs consists of three parts: the receptive fields, the modulation fields, and the pulse generator (as shown in Fig. 14). The neuron receives input signals from other neurons and external sources through the receptive fields. The receptive fields can be divided into two channels: one is the feeding inputs and the other is the linking inputs. The modulation fields generate the internal activity of the neuron. The pulse generator receives the result of total internal activity U_{ij} and determines the firing events.

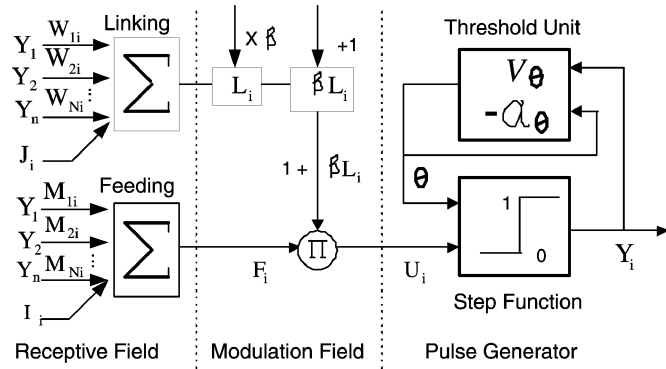


Fig. 14. Network structure of MPCNNs.

Let K be the total number of iterations and k ($k = 0, 1, \dots, K - 1$) be the current iteration, then the PCNNs can be described by the following equations:

$$\begin{aligned}
 F_i(k) &= e^{-\alpha_F F_i(k-1)} + \sum_{j=1}^N M_{ij} Y_j(k-1) + I_i \\
 L_i(k) &= e^{-\alpha_L L_i(k-1)} + \sum_{j=1}^N W_{ij} Y_j(k-1) + J_i \\
 U_i(k) &= F_i(k)(1 + \beta L_i(k)) \\
 \theta_i(k) &= e^{-\alpha_\theta \theta_i(k-1)} + V_\theta Y_i(k-1) \\
 Y_i(k) &= \text{Step}(U_i(k) - \theta_i(k-1)) \\
 &= \begin{cases} 1, & \text{if } U_i(k) > \theta_i(k-1) \\ 0, & \text{otherwise} \end{cases}
 \end{aligned}$$

where i stands for the position of neuron in the map, F and L are feeding input and linking input, respectively, U is the internal activity generated by the modulation fields, Y is the pulse output, α_F and α_L are time constants for feeding and linking, β is the strength of the linking, M and W represent the constant synaptic weights, I_i and J_i are constant input, $\theta_i(n)$ represents the dynamic threshold of the neuron i , and α_θ and V_θ are the time constant and the normalization constant, respectively.

If U_i is greater than the threshold, the output of neuron i turns into 1 and neuron i fires. Then, Y_i feeds back to make θ_i rise over U_i immediately and the output of neuron i turns into 0. So, a pulse output is produced. It is clear that the pulse generator is responsible for the modeling of the refractory period.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and anonymous reviewers for their valuable comments.

REFERENCES

[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, Oct. 1979.
 [2] O. Takahashi and R. J. Schilling, "Motion planing in a plane using generalized Voronoi diagrams," *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 143–150, Apr. 1989.
 [3] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[4] J. T. Schwartz and M. Sharir, "On the 'piano movers' problems: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Commun. Pure Appl. Math.*, vol. 36, no. 3, pp. 345–398, 1983.
 [5] E. Hou and D. Zheng, "Mobile robot path planning based on hierarching hexagonal decomposition and artificial potential fields," *J. Robot. Syst.*, vol. 11, no. 7, pp. 605–614, 1994.
 [6] 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. ACM Symp. Comput. Geometry*, Nice, France, 1983, pp. 1208–1213.
 [7] A. Zelinsky, "Using path transforms to guide the search for findpath in 2D," *Int. J. Robot. Res.*, vol. 13, no. 4, pp. 315–325, 1994.
 [8] K. S. Al-Sultan and D. S. Aliyu, "A new potential field-based algorithm for path planning," *Int. J. Intell. Robot. Syst.*, vol. 17, no. 3, pp. 265–282, 1996.
 [9] J. Chuang and N. Ahuja, "An analytically tractable potential field model of free space and its application in obstacle avoidance," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 28, no. 5, pp. 729–736, Oct. 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. Syst. Man Cybern. A, Syst. Humans*, vol. 30, no. 2, pp. 187–196, Mar. 2000.
 [11] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
 [12] L. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst. Man Cybern.*, vol. 22, no. 2, pp. 224–241, Mar./Apr. 1992.
 [13] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Trans. Robot. Autom.*, vol. 8, no. 1, pp. 23–32, Jan. 1992.
 [14] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.
 [15] C. W. Warren, "Fast path planning using modified a* method paths," in *Proc. IEEE Int. Conf. Robot. Autom.*, Atlanta, GA, 1993, pp. 662–667.
 [16] A. Yahja, S. Singh, and A. Stentz, "Recent results in path planning for mobile robots operating in vast outdoor environments," in *Proc. Symp. Image Speech Signal Process. Robot.*, Hong Kong, Sep. 1998, pp. 181–186.
 [17] Z. X. Li and T. D. Bui, "Robot path planning using fluid model," *J. Intell. Robot. Syst.*, vol. 21, pp. 29–50, 1998.
 [18] C. J. Ong and E. G. Gilbert, "Robot path planning with penetration growth distance," *J. Robot. Syst.*, vol. 15, no. 2, pp. 57–74, 1998.
 [19] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 28, no. 3, pp. 316–333, Jun. 1998.
 [20] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, Apr. 26–May 1 2004, pp. 4350–4355.
 [21] V. Lebedev, J. Steil, and J. Ritter, "The dynamic wave expansion neural network model for robot motion planning in time-varying environments," *Neural Netw.*, vol. 18, pp. 267–285, 2005.
 [22] A. R. Willms and S. X. Yang, "An efficient dynamic system for real-time robot-path planning," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 36, no. 4, pp. 755–766, Aug. 2006.
 [23] A. R. Willms and S. X. Yang, "Real-time robot path planning via a distance-propagating dynamic system with obstacle clearance," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 38, no. 3, pp. 884–893, Jun. 2008.
 [24] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.
 [25] D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 5, pp. 533–541, May 1986.
 [26] H. J. Tang, K. C. Tang, and Z. Yi, "A columnar competitive model for solving combinatorial optimization problems," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1568–1574, Nov. 2004.
 [27] H. Qu, Z. Yi, and H. J. Tang, "Improving local minima of columnar competitive model for TSPs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 6, pp. 1353–1362, Jun. 2006.
 [28] 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 Netw.*, vol. 8, no. 1, pp. 103–123, 1995.

- [29] H. J. Ritter, T. M. Martinez, and K. J. Schulten, "Topology-conserving maps for learning visuo-motor coordination," *Neural Netw.*, vol. 2, no. 3, pp. 159–168, 1989.
- [30] A. Zhu and S. X. Yang, "A neural network approach to task assignment of multirobots," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1278–1287, Sep. 2006.
- [31] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Population coding in a neural net for trajectory formation," *Network: Comput. Neural Syst.*, vol. 5, pp. 549–563, Aug. 1994.
- [32] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural network dynamics for path planning and obstacle avoidance," *Neural Netw.*, vol. 8, no. 1, pp. 125–133, 1995.
- [33] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "A biologically inspired neural net for trajectory formation and obstacle avoidance," *Biol. Cybern.*, vol. 74, pp. 511–520, 1996.
- [34] S. X. Yang and M. Meng, "Neural network approaches to dynamic collision-free robot trajectory generation," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 31, no. 3, pp. 302–318, Jun. 2001.
- [35] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, pp. 500–544, 1952.
- [36] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies Appl. Math.*, vol. 52, no. 3, pp. 217–257, 1973.
- [37] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architecture," *Neural Netw.*, vol. 1, no. 1, pp. 17–61, 1988.
- [38] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural Netw.*, vol. 13, no. 2, pp. 143–148, 2000.
- [39] S. X. Yang and M. Meng, "Real-time collision-free motion planning of mobile robots using neural dynamics based approaches," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1541–1552, Nov. 2003.
- [40] S. X. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, no. 1, pp. 718–725, Feb. 2004.
- [41] C. Luo and S. X. Yang, "A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1279–1298, Jul. 2008.
- [42] S. X. Yang, "Neural dynamics and computation for real-time map building and path planning of mobile robots," *Dyn. Continuous Discrete Impulse Syst. Ser. B*, vol. 10, no. 1, pp. 1–17, Jan. 2003.
- [43] S. X. Yang and M. Meng, "An efficient neural network method for real time motion planning with safety consideration," *Robot. Autonom. Syst.*, vol. 32, no. 2/3, pp. 115–128, Aug. 2000.
- [44] X. Yuan and S. X. Yang, "Virtual assembly with biologically inspired intelligence," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 33, no. 2, pp. 159–167, May 2003.
- [45] H. Li, S. X. Yang, and M. L. Seto, "Neural network based path planning for a multi-robot system with moving obstacles," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 39, no. 4, pp. 410–419, Jul. 2009.
- [46] X. Yuan and S. X. Yang, "Multi-robot-based nanoassembly planning with automated path generation," *IEEE Trans. Mechatron.*, vol. 12, no. 3, pp. 352–356, Jun. 2007.
- [47] K. H. Low, W. K. Leow, and M. H. Ang, Jr., "An ensemble of cooperative extended Kohonen maps for complex robot motion tasks," *Neural Comput.*, vol. 17, no. 6, pp. 1411–1445, Aug. 2005.
- [48] K. H. Low, W. K. Leow, and M. H. Ang, Jr., "Autonomic mobile sensor network with self-coordinated task allocation and execution," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 36, no. 3, pp. 315–327, May 2006.
- [49] D. L. Wang and D. Terman, "Locally excitatory globally inhibitory oscillator networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 283–286, Jan. 1995.
- [50] D. L. Wang and D. Terman, "Image segmentation based on oscillatory correlation," *Neural Comput.*, vol. 9, pp. 805–836, 1997.
- [51] R. Eckhorn, H. J. Reitboeck, M. Arndt, and P. W. Dicke, "Feature linking via synchronous among distributed assemblies: Simulations of results from cat visual cortex," *Neural Comput.*, vol. 2, pp. 293–307, 1990.
- [52] J. L. Johnson and D. Ritter, "Observation of periodic waves in a pulse-coupled neural network," *Opt. Lett.*, vol. 18, no. 15, pp. 1253–1255, 1993.
- [53] J. L. Johnson, "Waves in pulse coupled neural networks," in *Proc. World Congr. Neural Netw.*, 1993, vol. 4, pp. 299–302.
- [54] J. L. Johnson, "Pulse-coupled neural nets: Translation, rotation, scale, distortion, and intensity signal invariance for images," *Appl. Opt.*, vol. 33, no. 26, pp. 6239–6253, 1994.
- [55] S. H. Ranganath and G. Kuntimad, "Object detection using pulse coupled neural networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 615–620, May 1999.
- [56] X. Gu, D. Yu, and L. Zhang, "Image shadow removal using pulse coupled neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 692–698, May 2005.
- [57] X. Zhang and A. Minai, "Temporally sequenced intelligent block-matching and motion-segmentation using locally coupled networks," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1202–1214, Sep. 2004.
- [58] R. C. Muresan, "Pattern recognition using pulse-coupled neural networks and discrete Fourier transforms," *Neurocomputing*, vol. 51, pp. 487–493, 2003.
- [59] H. J. Caulfield and M. Kinsler, "Finding the shortest path in the shortest time using PCNNs," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 604–606, May 1999.
- [60] X. Gu, L. Zhang, and D. Yu, "Delay PCNN and its application for optimization," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2004, vol. 3173, pp. 413–418.
- [61] H. Li and S. X. Yang, "A behavior-based mobile robot with a visual landmark-recognition system," *IEEE Trans. Mechatron.*, vol. 8, no. 3, pp. 390–400, Sep. 2003.
- [62] X. Wang, S. X. Yang, and M. Q.-H. Meng, "A co-evolution approach to sensor placement and control design for robot obstacle avoidance," *Int. J. Inf. Acquisition*, vol. 2, no. 2, pp. 77–91, Jun. 2005.
- [63] G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csobor, "A solution to the simultaneous localisation and mapping (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.
- [64] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 242–257, Jun. 2001.
- [65] A. Zhu and S. X. Yang, "A neuro-fuzzy-based approach to mobile robot navigation in unknown environments," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 37, no. 4, pp. 610–621, Jul. 2007.
- [66] F. Muñoz, E. Zalama, P. Gaudiano, and J. López-Coronado, "Neural controller for a mobile robot in a nonstationary environment," in *Proc. 2nd IFAC Conf. Intell. Autonom. Veh.*, Helsinki, Finland, Jun. 1995, pp. 279–284.
- [67] A. F. Murray, D. D. Corso, and L. Tarrasenko, "Pulse stream VLSI neural networks mixing analog and digital techniques," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 193–204, Mar. 1991.



Hong Qu received the B.S. degree and the M.S. and Ph.D. degrees in computer science and engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2000 and 2003, respectively.

From July 2007 to February 2008, he was a Post-doctoral Fellow at the Advanced Robotics and Intelligent Systems Lab, School of Engineering, University of Guelph, Guelph, ON, Canada. Currently, he is a Lecturer at the School of Computer Science and Engineering, University of Electronic Science and Tech-

nology of China. His current research interests include neural networks, neurodynamics, intelligent computation, and optimization.



Simon X. Yang (S'97–M'99–SM'08) received the B.Sc. degree in engineering physics from Beijing University, Beijing, China, in 1987, the M.Sc. degree in biophysics from the Chinese Academy of Sciences, Beijing, China, in 1990, the M.Sc. degree in electrical engineering from the University of Houston, Houston, TX, in 1996, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1999.

He joined the School of Engineering, University of Guelph, Guelph, ON, Canada, in 1999. Currently, he is the Professor and the Head of the Advanced Robotics and Intelligent Systems (ARIS) Laboratory, University of Guelph. His research interests include intelligent systems,

robotics, sensors and multisensor fusion, wireless sensor networks, control systems, soft computing, and computational neuroscience.

Prof. Yang serves as an Associate Editor of the IEEE TRANSACTIONS OF NEURAL NETWORKS, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, the *International Journal of Robotics and Automation*, and several other journals. He has been involved in the organization of many conferences. He was the General Chair of the 2006 International Conference on Sensing, Computing, and Automation. Among many of his awards, he is a recipient of the Distinguished Professor Award for 2004–2006 at the University of Guelph.



Allan R. Willms received the B.Math. and M.Math. degrees in applied mathematics from the University of Waterloo, Waterloo, ON, Canada, in 1992 and 1993, respectively, and the Ph.D. degree from Cornell University, Ithaca, NY, in 1997.

He subsequently was a faculty member at the Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand, for five years. Since 2003, he has been a faculty member at the Department of Mathematics and Statistics, University of Guelph, Guelph, ON, Canada. His research interests include dynamical systems, mathematical biology, parameter identification, bifurcation theory, and robot path planning.



Zhang Yi received the B.S. degree in mathematics from Sichuan Normal University, Chengdu, China, in 1983, the M.S. degree in mathematics from Hebei Normal University, Shijiazhuang, China, in 1986, and the Ph.D. degree in mathematics from the Institute of Mathematics, The Chinese Academy of Science, Beijing, China, in 1994.

From 1989 to 1990, he was a Senior Visiting Scholar at the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, U.K. From February 1999 to August

2001, he was a Research Associate at the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Kowloon, Kong Kong. From August 2001 to December 2002, he was a Research Fellow at the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He is currently a Professor at the School of Computer Science, Sichuan University. His current research interests include neural networks and data mining.