Neural Network Approaches to Dynamic Collision-Free Trajectory Generation

Simon X. Yang, *Member, IEEE,* and Max Meng, *Member, IEEE*

***Abstract*—In this paper, dynamic collision-free trajectory generation in a nonstationary environment is studied using biologically inspired neural network approaches. The proposed neural network is topologically organized, where the dynamics of each neuron is characterized by a shunting equation or an additive equation. The state space of the neural network can be either the Cartesian workspace or the joint space of multi-joint robot manipulators. There are only local lateral connections among neurons. The real-time optimal trajectory is generated through the dynamic activity landscape of the neural network without explicitly searching over the free space nor the collision paths, without explicitly optimizing any global cost functions, without any prior knowledge of the dynamic environment, and without any learning procedures. Therefore the model algorithm is computationally efficient. The stability of the neural network system is guaranteed by the existence of a Lyapunov function candidate. In addition, this model is not very sensitive to the model parameters. Several model variations are presented and the differences are discussed. As examples, the proposed models are applied to generate collision-free trajectories for a mobile robot to solve a maze-type of problem, to avoid concave U-shaped obstacles, to track a moving target and at the same to avoid varying obstacles, and to generate a trajectory for a two-link planar robot with two targets. The effectiveness and efficiency of the proposed approaches are demonstrated through simulation and comparison studies.**

***Index Terms*—Dynamic environment, mobile robot, neural dynamics, neural networks, obstacle avoidance, robot manipulators, trajectory generation.**

I. Introduction

TRAJECTORY generation with obstacle avoidance is a fundamentally important issue in robotics. Real-time collision-free trajectory generation becomes more difficult when robots are in a dynamic, unstructured environment. There are a lot of studies on trajectory generation for robots using various approaches (e.g., [1]–[14], [21]–[24], [26]–[30], [32]–[40], [43], [45], [47]–[54] and [59]–[61]). Some of the previous models (e.g., [1], [3], [5], [7], [9], [21], [22], [28], [29], [36], [50] and [60]) use global methods to search the possible paths in the workspace, which normally deal with static environment only and are computationally expensive when the environment is complex. Some searching based models (e.g., [12], [23], [50]) suffer from undesired local minima, i.e., the robots may be trapped in some cases such as with concave U-shaped obstacles. Seshadri and Ghosh [47] proposed a new path planning model using an iterative approach. However this model is computationally complicated, particularly in a complex environment. Li and Bui [28] proposed a fluid model for robot path planning in a static environment. Ong and Gilbert [38] proposed a new model for path planning with penetration growth distance, which searches over collision paths instead of searching over free space as most other models do. This model can generate optimal, continuous robot paths in a static environment only. Oriolo *et al.* [39], [40] proposed a model 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.

Several neural network models (e.g., [11], [27], [34], [46] and [59]) were proposed to generate real-time trajectories through learning. Ritter *et al.* [46] proposed a Kohonen's self-organizing mapping algorithm based neural network model to learn the transformation from Cartesian workspace to the robot manipulator joint space. Li and Öğmen [27] 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. However, both models deal with static environment only and assume that there are no obstacles in the workspace. Muñiz *et al.* [34] proposed a neural network model for the navigation of a mobile robot, which can generate dynamic trajectories with obstacle avoidance through unsupervised learning. However, this model is computationally complicated since it incorporates the vector associative map model and the direction-to-rotation effector control transform model [11], [59]. Fujii *et al.* [10] proposed a multilayer reinforcement learning based model for path planning with a complicated collision avoidance algorithm. However, the generated trajectories using learning based approaches are not optimal, particularly during the initial learning phase.

Glasius *et al.* [13] 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 [13]. They later proposed another model [14] by cascading two neural network layers where each layer has a similar architecture to the model in [13]. This is an unsupervised model which uses the second layer to find the next robot position. However, it doubles the computational burden as a result. All these

Manuscript received May 10, 1999; revised January 1, 2001. The work of S. X. Yang was supported by the Natural Sciences and Engineering Research Council (NSERC) under Grant RGPIN227684. The work of M. Meng was supported by the Natural Sciences and Engineering Research Council (NSERC) under Grant RGPIN170446. This paper was recommended by Associate Editor S. Lakshmivarahan.

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

M. Meng is with the Advanced Robotics and Teleoperation (ART) Lab, Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6G 2G7 Canada.

Publisher Item Identifier S 1083-4419(01)05218-9.

models [12]–[14] suffer from slow dynamics and cannot perform properly in a fast changing environment, since they unrealistically require that the robot dynamics must be faster than the target and obstacle dynamics [13], [14].

In this paper, inspired by Hodgkin and Huxley's membrane model [20] for a biological neural system and the later developed Grossberg's shunting model [15], [18], a novel neural network approach is proposed for dynamic collision-free trajectory generation in an arbitrarily varying environment (which first appeared in [33] and [58]). The state space of the topologically organized neural network is either the Cartesian workspace of mobile robots or the joint space of multijoint robot manipulators. The neural dynamics of each neuron is characterized by a shunting equation or a simple additive equation. There are only local, excitatory lateral connections among neurons. Thus the computational complexity linearly depends on the size of the neural network. In addition, the target *globally* attracts the robot in the whole state space through neural activity propagation, while the obstacles *locally* push the robot away to avoid collisions, since there is no inhibitory lateral connections among neurons. The real-time optimal robot trajectory is generated through the dynamic activity landscape of the neural network *without* explicitly searching over the free space nor the collision paths, *without* explicitly optimizing any global cost functions, *without* any prior knowledge of the dynamic environment, and *without* any learning procedures. Therefore, the model algorithm is computationally efficient. The generated solution to a maze-solving type of problem or the generated trajectory in a static environment is globally optimal in the sense of a shortest path from the starting position to the target if it exists. The optimality of the real-time trajectory generation in a nonstationary environment is in the sense that the robot travels a continuous, smooth path toward the target. The term "real-time" is in the sense that the robot trajectory generator responds immediately to the dynamic environment, including the robot, target, obstacles and sensor noise. There are several variations of the proposed model. Although a special case of the proposed simple additive model is similar to the Glasius *et al.* model [13], [14], there are significant differences between the proposed models and the Glasius *et al.* model (see Section IV).

In Section II, the originality, model algorithm and stability analysis of the proposed neural network approach to dynamic collision-free trajectory generation is presented. Simulation studies of the proposed model are presented in Section III, including an optimal solution to a maze-type of problem, trajectory generation of a mobile robot to avoiding concave U-shaped obstacles, to track a moving target, and to avoid moving obstacles, and trajectory generation of a two-link planar robot to catch the closest target. In Section IV, the parameter sensitivity and the model variations are discussed. Finally, a conclusion noticing several feature properties of the proposed neural network approach is addressed in Section V.

II. Model

The proposed novel neural network approach for dynamic collision-free trajectory generation is motivated by the neural dynamics and computation in biological neural systems. The philosophy of the proposed approach is to develop a topologically organized neural network architecture, whose dynamic neural activity landscape represents the dynamically varying environment. By properly defining the external inputs from the varying environment and internal neural connections, the target and obstacles are guaranteed to stay at the peak and the valley of the activity landscape of the neural network, respectively. The target globally attracts the robot in the whole state space through neural activity propagation, while the obstacles have only local effect to avoid collisions. The dynamic collision-free trajectory is generated through the dynamic activity landscape of the neural network.

In this section, the originality of the proposed neural network approach is introduced. Then, the computational algorithm is presented. Finally, the stability of the proposed model is proved by using both qualitative analysis and Lyapunov stability analysis.

A. Originality

Hodgkin and Huxley [20] proposed a model for a patch of membrane in a biological neural system using electrical circuit elements. This modeling work together with other experimental work led them to a Nobel Prize in 1963 for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell membrane. In their membrane model, the dynamics of voltage across the membrane V_m can be described using state equation technique as

$$C_m \frac{dV_m}{dt} = -(E_p + V_m)g_p + (E_{Na} - V_m)g_{Na} - (E_K + V_m)g_K \quad (1)$$

where

C_m	membrane capacitance;
E_K, E_{Na}, and E_p	Nernst potentials (saturation potentials) for potassium ions, sodium ions, and the passive leak current in the membrane, respectively;
g_K, g_{Na}, and g_p	conductance of potassium, sodium, and passive channels, respectively.

This model provided the foundation of the shunting model and led to a lot of model variations and applications [18], [19], [44].

By substituting $C_m = 1$, $x_i = E_p + V_m$, $A = g_p$, $B = E_{Na} + E_p$, $D = E_k - E_p$, $S_i^+ = g_{Na}$ and $S_i^- = g_K$ in (1), a shunting equation is obtained [42]

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)S_i^+(t) - (D + x_i)S_i^-(t) \quad (2)$$

where

x_i	neural activity (membrane potential) of the ith neuron;
A, B, and D	nonnegative constants representing the passive decay rate, the upper and lower bounds of the neural activity, respectively;
S_i^+ and S_i^-	excitatory and inhibitory inputs to the neuron [41], [42].

This shunting model was first proposed by Grossberg to understand the real-time adaptive behavior of individuals to complex and dynamic environmental contingencies [15]–[18], and has a lot of applications in biological and machine vision, sensory motor control and many other areas [18], [27], [41], [42], [56], [57].

B. Model Algorithm

The neural network architecture of the proposed model is a discrete topologically organized map that has been used in many neural network models such as [13], [25]. The proposed model is expressed in a finite (F-) dimensional (F-D) state space $\mathcal{S}$, which can be either the Cartesian workspace of the mobile robots or the joint space of multijoint robot manipulators. For example, a two–dimensional (2-D) workspace has $F = 2$ and a 6 degree-of-freedom (DOF) manipulation robot has $F = 6$. The location of the ith neuron at the grid in the F-D state space $\mathcal{S}$, denoted by a vector $q_i \in R^F$, represents a position in the workspace or a configuration in the joint space. Each neuron has only local lateral connections to its neighboring neurons that constitute a subset $\mathcal{R}_i$ in $\mathcal{S}$. The subset $\mathcal{R}_i$ is called the receptive field of the ith neuron in neurophysiology. The neuron responds only to the stimulus within its receptive field.

In the proposed model, the dynamics of the ith neuron in the neural network is characterized by a shunting equation derived from (2). The excitatory input S_i^+ results from the target and the lateral connections from its neighboring neurons, while the inhibitory input S_i^- results from the obstacles only. Thus the differential equation for the ith neuron is given by

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{N} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \tag{3}$$

where

N	total number of neurons in the neural network;
$[I_i]^+ + \sum_{j=1}^{N} w_{ij}[x_j]^+$ and $[I_i]^-$	excitatory and inhibitory inputs, respectively;
I_i	external input to ith neuron defined as

$$I_i = \begin{cases} E, & \text{if there is a target} \\ -E, & \text{if there is an obstacle} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $E \gg B$ is a very large positive constant. Function $[a]^+$ is a linear-above-threshold function defined as $[a]^+ = \max\{a, 0\}$, and the nonlinear function $[a]^-$ is defined as $[a]^- = \max\{-a, 0\}$. The connection weight w_{ij} from the jth neuron to the ith neuron is a function of distance defined as

$$w_{ij} = f(d_{ij}) \tag{5}$$

where $d_{ij} = |q_i - q_j|$ represents the Euclidean distance between positions q_i and q_j in the state space $\mathcal{S}$. The connection weight function $f(a)$ is a monotonically decreasing function, e.g., a function defined as

$$f(a) = \begin{cases} \mu/a, & \text{if } 0 < a < r_0 \\ 0, & \text{if } a \geq r_0 \end{cases} \tag{6}$$

where μ and r_0 are positive constants. It is obvious that the weight w_{ij} is symmetric $w_{ij} = w_{ji}$ and does not depend on the moving directions of the robot. In addition, the neuron has only local connections in a small region $(0, r_0)$, i.e., its receptive field $\mathcal{R}_i$ is the space whose distance to the ith neuron is less than r_0. All the neurons having lateral connection to the ith neuron is defined its *neighboring neurons*. Therefore, the dynamics of the ith neuron can be further written as

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)\left([I_i]^+ + \sum_{j=1}^{k} w_{ij}[x_j]^+\right) - (D + x_i)[I_i]^- \tag{7}$$

where k is the number of neighboring neurons of the ith neuron.

The proposed neural network characterized by (7) guarantees that the positive neural activity can propagate to the whole state space through lateral neural connections, while the negative activity stays locally only, since there is no inhibitory connections among neurons. Therefore, the target *globally* influences the whole state space to attract the robot, while the obstacles have only *local* effect to avoid collisions. In addition, the activity propagation from the target is blocked when it hits the obstacles by choosing $r_0 = 2$. Such a property is very important for maze-solving type of problems.

The positions of the target and obstacles may vary with time. The activity landscape of the neural network dynamically changes according the varying external inputs and the lateral excitatory connections. From (7), each neuron responds to only the real-time inputs from the target and obstacles. Thus no *prior* knowledge of the varying environment is needed in the proposed model. The real-time trajectory is generated from the dynamic activity landscape by a steepest gradient ascent rule. For a given present position in the state space of the neural network (i.e., a position in the Cartesian workspace or a configuration in the robot manipulator joint space), denoted by q_p, the next position q_n (also called "command position") is obtained by

$$q_n \Leftarrow x_{q_n} = \max\{x_j, j = 1, 2, \cdots, k\} \tag{8}$$

where k is the number of neighboring positions of the q_pth neuron, i.e., all the possible next positions of the present position. After the present position reaches its next position, the next position becomes a new present position (if the found next location is the same as the present location, i.e., the neural activity at all the neighboring positions is not larger than that of the current position, the robot stays there without any movement). The present position *adaptively* changes according to the varying environment.

The dynamic activity landscape of the topologically organized neural network is used to determine *where* the next robot position is. However, *when* to generate the next robot position is determined by the robot moving speed. In a static environment,

the activity landscape of the neural network will reach a steady state, which will later be proved using the Lyapunov stability theory. Mostly the robot reaches the target much earlier than the activity landscape reaches the steady state. When a robot is in a changing environment, the activity landscape will never reach a steady state. Due to the very large external input constant E, the target and the obstacles keep staying at the peak and the valley of the activity landscape of the neural network, respectively. The robot keeps moving toward the target with obstacle avoidance till the designated objective is achieved.

C. Stability Analysis

In the shunting model in (2), (3), or (7) the neural activity x_i increases at a rate of $(B-x_i)S_i^+$, which is not only proportional to the excitatory input S_i^+, but also proportional to an auto gain control term $B - x_i$. Thus, with an equal amount of input S_i^+, the closer the values of x_i and B are, the slower x_i increases. When the activity x_i is below B, the excitatory term is positive causing an increase in the neural activity. If x_i is equal to B, the excitatory term becomes zero and x_i will no longer increase no matter how strong the excitatory input is. In case the activity x_i exceeds B, $B - x_i$ becomes negative and the shunting term pulls x_i back to B. Therefore, x_i is forced to stay below B, the upper bound of the neural activity. Similarly, the inhibitory term forces the neural activity stay above the lower bound $-D$. Therefore, once the activity goes into the finite region $[-D, B]$, it is guaranteed that the neural activity will stay in this region for any value of the total excitatory and inhibitory inputs [56].

The stability and convergence of the proposed model can also be rigorously proved using a Lyapunov stability theory. Introducing the new variables, $y_i = x_i - B$, the proposed model in (3) or (7) can be written into the general form proposed by Grossberg [17], [18]

$$\frac{dy_i}{dt} = a_i(y_i)\left(b_i(y_i) - \sum_{j=1}^{N} c_{ij} d_j(y_j)\right) \tag{9}$$

by the following substitutions:

$$a_i(y_i) = -y_i \tag{10}$$

$$b_i(y_i) = \frac{1}{y_i}\left(AB + y_i(A + [I_i]^+ + [I_i]^-) + (B + D)[I_i]^-\right) \tag{11}$$

$$c_{ij} = -w_{ij} \tag{12}$$

and

$$d_j(y_j) = [y_j + B]^+. \tag{13}$$

Since $w_{ij} = w_{ji}$, then $c_{ij} = c_{ji}$ (symmetry). Since y_i varies within the finite interval $[-B - D, 0]$, where B and D are non-negative constants, then y_i is a nonpositive number. Hence the amplification function $a_i(y_i)$ is nonnegative, i.e., $a_i(y_i) \geq 0$ (positivity). From the definition of function $[a]^+$, have $d_j'(y_j) = 0$ at $y_j < -B$ and $d_j'(y_j) = 1$ at $y_j > -B$. Hence the signal function $d_j(y_j)$ has a nonnegative derivation, i.e., $d_j'(y_j) \geq 0$ (monotonicity). Therefore, (7) satisfies all the three stability conditions required by Grossberg's general form [18]. The Lyapunov function candidate for (9) can be chosen as

$$v = -\sum_{i=1}^{N} \int^{y_i} b_i(\eta_i) d_i'(\eta_i)\, d\eta_i + \tfrac{1}{2} \sum_{j,k=1}^{N} c_{jk} d_j(y_j) d_k(y_k). \tag{14}$$

The time derivative of v along all the trajectories is given by

$$\frac{dv}{dt} = -\sum_{i=1}^{N} a_i d_i' \left(b_i - \sum_{j=1}^{N} c_{ij} d_j\right)^2. \tag{15}$$

Since $a_i \geq 0$ and $d_i' \geq 0$, then $dv/dt \leq 0$ along all the trajectories. The rigorous proof of the stability and convergence of (9) can be found in [17]. Therefore, the proposed neural network system is stable. The dynamics of the network is guaranteed to converge to an equilibrium state of the system.

III. Simulation Studies

The proposed neural network model is capable of generating real-time optimal trajectory with obstacle avoidance for a robot in an arbitrarily varying environment. The state space can be the Cartesian workspace of a mobile robot or the joint space of a multi-joint robot manipulator. The generated solution to a maze-solving type of problem and the generated trajectory in a static environment is globally optimal in the sense of a shortest path from the starting position to the target, if it exists. Such a property results from the fact that the connection weight of the neural network is a function of the distance only, the neural activity propagation from the target to all directions are exactly in the same manners. In a static environment, the neural activity from the target always reaches the robot along a shortest path. For real-time trajectory generation in a nonstationary environment, the optimality is in the sense that the robot travels a continuous, smooth route toward the target.

In this section the proposed model is first applied to a point mobile robot in a 2-D workspace. A small and maneuverable mobile robot can be treated as a point robot, when comparing the size of the robot and its maneuvering possibilities to the size of the free workspace. For example, in practice, a car in the traffic planning in large cities or tanks in field military operations can be treated as point robots. Several cases for a point robot are studied here, including a maze-solving type of problem, trajectory generating to avoid concave U-shaped obstacles, a moving target tracking problem, and varying obstacles avoiding problem. Then, this model is applied to a two-link planar robot, where the state space of the neural network is the robot joint space and there are two targets in the state space.

A. Trajectory Generation in a Static Environment

The proposed model is first applied to the obstacle avoidance problem for a set of U-shaped obstacles. Potential field based methods and other strictly local avoidance schemes cannot deal

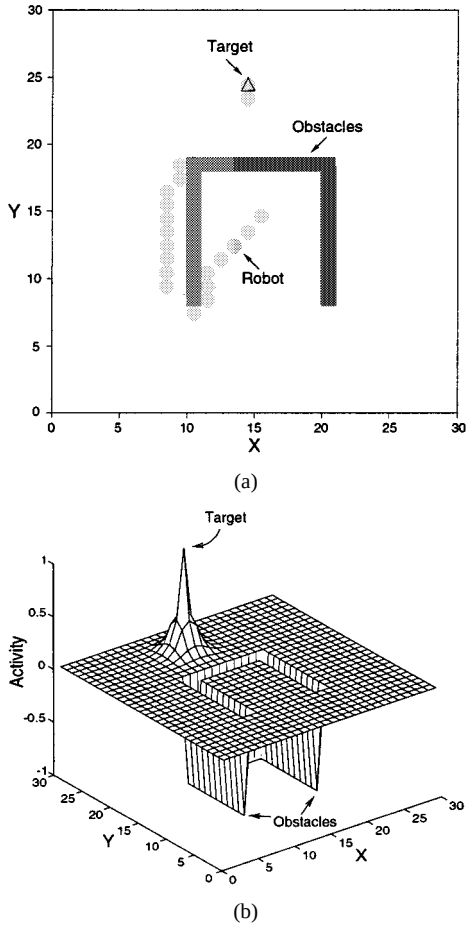


Fig. 1. Trajectory generation of a mobile robot to avoid a set of concave U-shaped obstacles. (a) Generated robot trajectory. (b) Stable activity landscape of the neural network.

with these type of problems [34]. The concave U-shaped obstacles are shown in Fig. 1(a) by solid squares. The neural network has 30 × 30 topologically organized neurons, where all the neural activities are initialized to zero. The model parameters are chosen as $A = 10$ and $B = D = 1$ for the shunting equation, $\mu = 1$ and $r_0 = 2$ for the lateral connections, and $E = 100$ for the external inputs. The generated trajectory is shown in Fig. 1(a) by solid circles. It shows that the generated trajectory is a continuous, smooth route from the starting position to the target with obstacle avoidance. The stable (the time is long enough) activity landscape of the neural network is shown in Fig. 1(b), where the peak is at the target location, while the valley is at the obstacle location.

The solution to a maze-solving type of problem can be treated as a special case of the trajectory generation problem in a 2-D workspace, along which a mobile robot can reach the target from a given starting position with obstacle avoidance. The example of the well-known beam robot competition micromouse maze (solid squares in Fig. 2 shows a typical quarter of the maze) is used. The proposed neural network model is applied to solve this maze-type of problem. The neural network has 17 × 17 neurons with the same the model parameters as in the previous case. The generated globally optimal solution is shown in Fig. 2, where the trajectory of the robot is represented by solid circles. Simulation studies demonstrate that the proposed model does

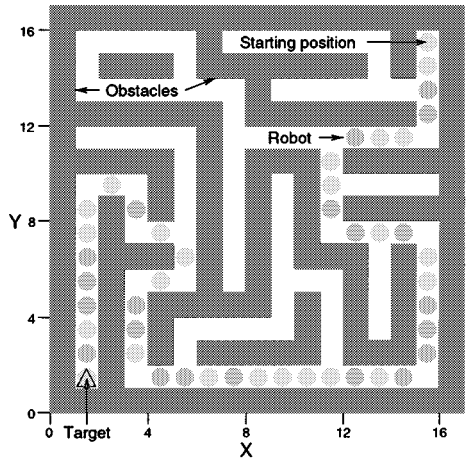


Fig. 2. Solution to a maze-solving type of problem.

not suffer from undesired local minima, i.e., the robot will not be trapped in the situation with concave U-shaped obstacles or with complex maze-solving type of problems.

B. Trajectory Generation to Track a Moving Target

The proposed model is then applied to a real-time trajectory generation problem for a robot to track a moving target. The neural network assumes 30 × 30 neuron structure with the same model parameters as in previous cases. In a 2-D workspace without any obstacles, the traveling route of the target is shown in Fig. 3(a) as indicated by hollow triangles, with an initial position at (X, Y) = (5, 5). The target moves at a speed of 25 *block/min* (it is convenient to assume that the space and time units are block and minute, respectively), and stops at (25, 25) after it arrives there. Note that the proposed neural network responds to the real-time location of the targets and obstacles. No prior knowledge of the varying environment is needed. The robot starts to move from position (0, 0) at a speed of 10 *block/min*. The generated trajectory of the robot is shown in Fig. 3(a) by solid circles. The activity landscapes of the neural network at two time instants during the motion are shown in Fig. 3(b) and (c).

When the robot tracks a moving target, the relative moving speed between the target and the robot is an important factor to influence the tacking trajectory. With the same model parameters as in Fig. 3, Fig. 4(a) shows the generated real-time trajectory of the robot that moves at a speed of 20 block/min, which is twice of that in Fig. 3. When the robot moves faster than the target at a speed of 30 block/min, the generated real-time robot trajectory is shown in Fig. 4(b). It shows that the target is caught before it reaches its final position. Comparing Figs. 3(a) and Fig. 4, it is shown that the robot with a slower moving speed takes less steps (not time) to reach the target, since the robot has more time to "wait and see" which position is to go next. However, the faster moving robot spends less time to reach the target. It takes 0.67, 2.04, and 3.06 min for the robot at speeds of 30 [Fig. 4(b)], 20 [Fig. 4(a)], and 10 [Fig. 3] block/min, respectively, to reach the target.

When there are obstacles in the workspace, the robot is able to track the moving target with collision avoidance. Choosing the

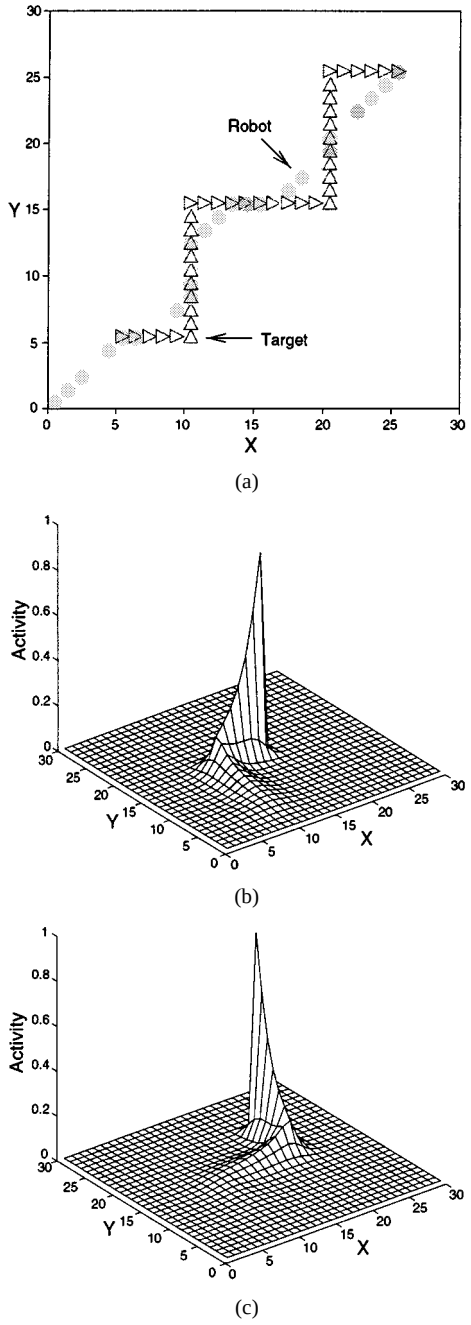


Fig. 3. Trajectory generation of a mobile robot to track a moving target. (a) Dynamic trajectories of the target (hollow triangles) and the robot (solid circles). Activity landscapes (b) and (c) when the target arrives at (16, 16) and (20, 21), respectively.

same model parameters as in Fig. 3. The generated trajectory with presence of obstacles is shown in Fig. 5. The robot takes more steps (and time) to reach the target due to the influence of the obstacles. Note that all the robot trajectories in Figs. 3–5 are continuous, smooth routes. The traveling path of the robot is generally shorter than that of the target.

C. Trajectory Generation to Catch a Moving Target with Varying Obstacles

Next the proposed model is applied to a more complex case, where both the target and the obstacles are moving. The neural network architecture and the model parameters are chosen as the same as in the previous cases, i.e., 30×30 neurons, zero initial neural activity, $A = 10$, $B = D = 1$, $\mu = 1$, $r_0 = 2$, and $E = 100$. The target starts at position (4, 25) and continuously moves back and forth along the line between (4, 25) and (24, 25) at a speed of 10 block/min (shown in Fig. 6 by hollow triangles). The static obstacles shown in Fig. 6 by solid squares form

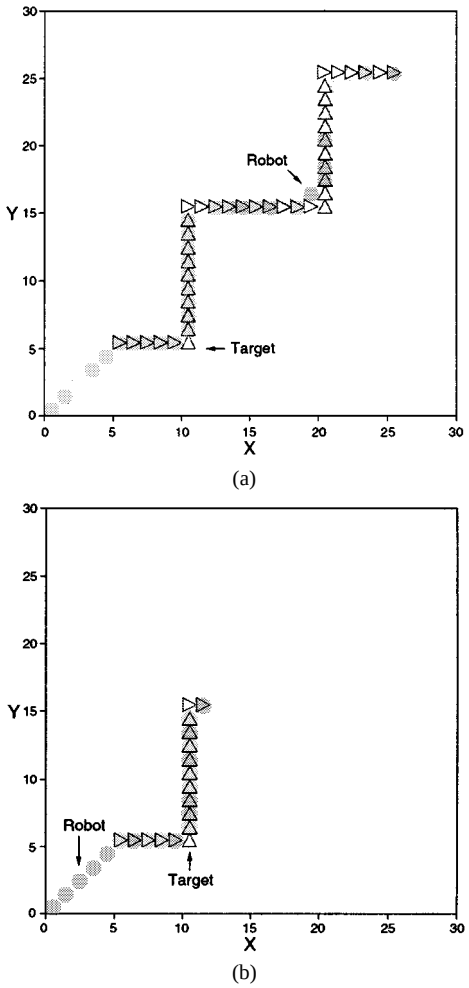


Fig. 4. Trajectory generation to track a moving target when the robot moves faster than that in Fig. 3. Target moves at 25 block/min. Robot speed in Fig. 3 is 10 block/min. (a) Dynamic trajectory when the robot moves at 20 block/min. (b) Trajectory when robot moves at 30 block/min.

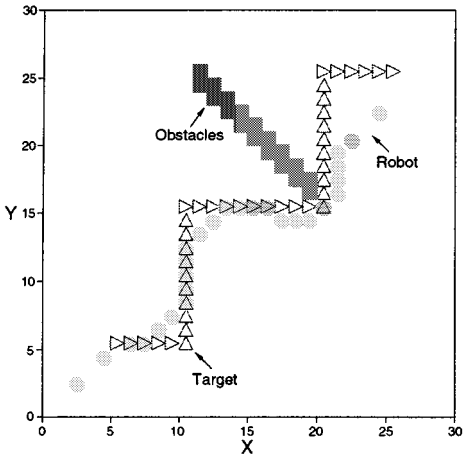


Fig. 5. Trajectory generation to track a moving target with static obstacles. Obstacles are represented by sold squares.

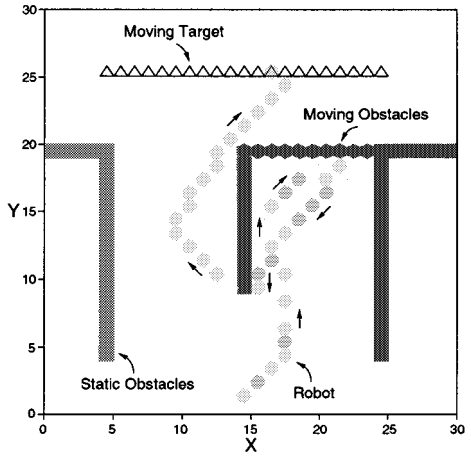


Fig. 6. Trajectory generation to catch a moving target with moving obstacles. Moving obstacles are represented by solid hexagons. Target moves back and forth along the hollow triangles.

two possible channels for the robot to reach the target. In addition, there are ten movable obstacles. They initially stop for 0.5 min at positions from (5, 19) to (14, 19) inside the left channel, where they completely block the left channel. Then the obstacles start to move toward the right at a speed of 20 block/min, and finally stop at positions from (14, 19) to (23, 19), where they completely block the right channel. Note that no *prior* knowledge of the varying environment is needed in the proposed model. The robot starts to move from position (14, 1) at a speed of 20 block/min. The generated trajectory is shown in Fig. 6 by solid circles. Initially the robot moves toward the right channel, since the left channel is completely blocked while the right channel is open. However, during the time the robot is moving toward the target through the right channel, the obstacles are gradually moving to close the right channel and open the left channel. Before the robot is able to pass through the right channel, the moving obstacles completely block the right channel and leave the left channel completely open. The robot has to move away from the target, passes around the middle static obstacles, and finally catch the moving target through the left channel. Here again the robot traveling route is continuous, smooth, and free of collisions.

D. Trajectory Generation of a Multijoint Robot Manipulator with Multiple Targets

The proposed model is capable of generating real-time trajectory with multiple targets as well, where the task can be designed as either catching the closest target or catching all the targets. In the latter case, a target should disappear from the state space once it is caught. The proposed model is applied to trajectory generation of a two-link planar robot with two targets in the state space of the neural network, which is the joint space of the robot manipulator. The robot link lengths are $l_1 = 1$ m and $l_2 = 1.1$ m, respectively. The base of the first link is fixed at the center (0, 0) of a $5\text{m} \times 5\text{m}$ 2-D Cartesian workspace [Fig. 7(a)]. Initially the tip of the second link is at position (1.366, 1.366) in the workspace [Fig. 7(a)], the initial configuration in the joint space is at $(\theta_1, \theta_2) = (30°, 30°)$ [Fig. 7(b)], where θ_1 and θ_2 are the joint angles of the first and second links, respectively. The task is to move the tip of the second link to position (1.366, -1.366) in the workspace [Fig. 7(a)]. Thus, there are two target configurations in the joint space, $(330°, 330°)$, and $(300°, 30°)$ (shown in Fig. 7(b) by hollow triangles). There are three obstacles in the workspace, which are located at positions (0.8,0), (0, 1.5) and (0, -1.5) [which is shown in Fig. 7(a) by solid circles], and the corresponding obstacles in the configuration joint space are shown in Fig. 7(b) by solid squares. The task in the robot joint space is to generate a trajectory from the initial robot configuration to the closest target configuration.

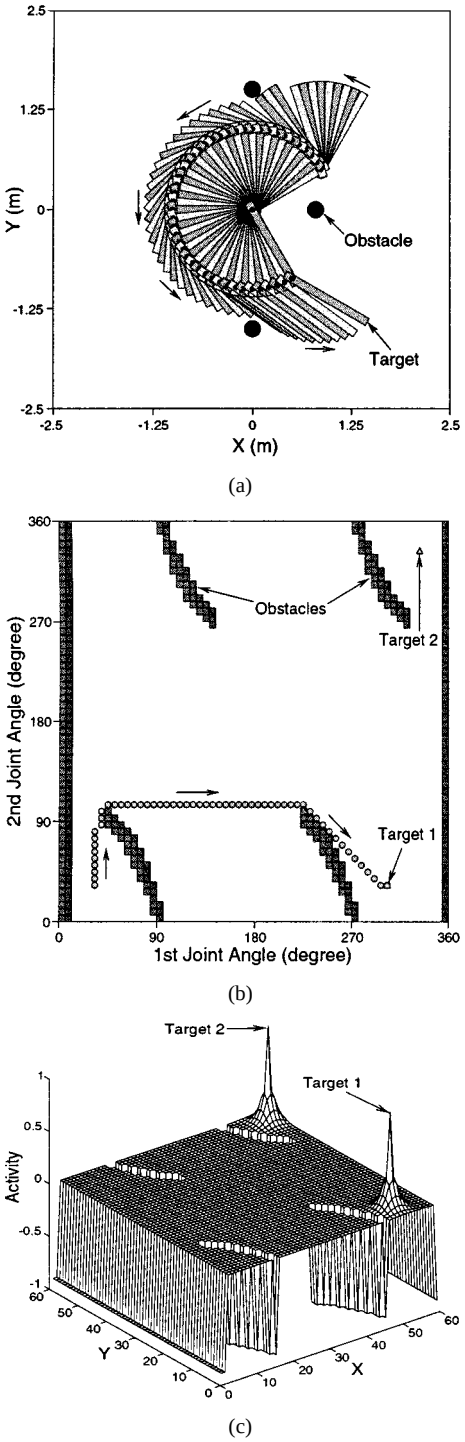
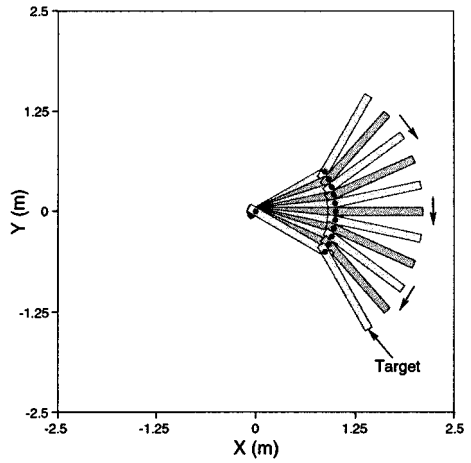
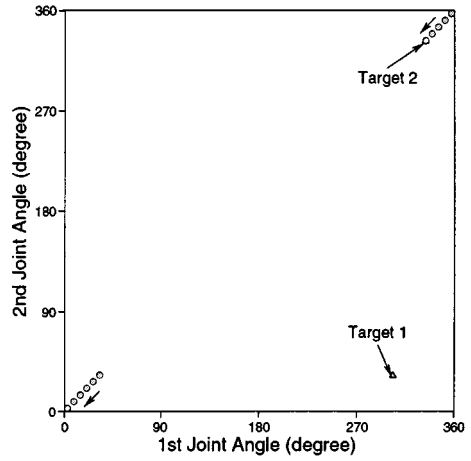


Fig. 7. Trajectory generation of a two-link planar robot with two target configurations. (a) Robot performance in the workspace. (b) Trajectory in the joint space. (c) Stable activity landscape of the neural network.



(a)



(b)

Fig. 8. Trajectory generation of a two-link planar robot without obstacles. (a) Robot performance in the workspace. (b) Trajectory in the joint space.

The neural network has 60×60 topologically organized neurons, which represents the joint angles from $0°$ to $354°$ with a step of $6°$. Since geometrically $360° = 0°$, the neuron at $(0, 0)$ is an immediate neighboring neuron of the neuron at (59, 59) or (0, 59) in the state space, and likewise. The model parameters are chosen as $A = 10, B = D = 1, \mu = 1, r_0 = 2$, and $E = 100$. The dynamic robot performance in Cartesian workspace are shown in Fig. 7(a) by the alternating light and dark two-link planar robots. The trajectory in joint space are shown in Fig. 7(b) by solid circles. It shows that the robot travels a smooth, continuous, collision-free route in both the workspace and the joint space, and reaches the closest target (Target 1). The stable (time is long enough) activity landscape of the neural network is shown in Fig. 7(c), where two peaks of the activity landscape are at the two target locations, while the valley is at the obstacle locations. When there is no obstacle, the robot performance in workspace and the trajectory in joint space are shown in Fig. 8(a) and (b), respectively. The robot reaches the closest target (Target 2).

IV. Parameter Sensitivity and Model Variations

In this section, the parameter sensitivity of the proposed model is discussed by descriptive analysis and simulation studies. Then three model variations of the proposed model are introduced, and a comparison study among these models is presented.

A. Parameter Sensitivity

The sensitivity of a system to parameter variations is a factor of prime importance to be considered when proposing or evaluating a model. An acceptable model should be robust to variations in its parameters. There are few parameters in the proposed model: parameters A, B, and D for the shunting equation, μ and r_0 for the lateral connections, and E for the external inputs.

1) Parameters for the Shunting Equation: In the shunting equation (2) or (7), parameters B and D are the upper and lower bounds of the neural activity, respectively. The transient response to an input signal does not depend on B and D. At steady state the neural activity *linearly* depends on B and D [56], [57]. In the proposed model, because only the relative value of the neural activity is concerned, B and D are not important factors in the proposed model. For example, B and D can be chosen as constants $B = D = 1$ for all cases.

Parameter A in (7) represents the passive decay rate, which solely determines the transient response to an input signal. In addition, the steady-state neural activity is *nonlinearly* dependent on the value of A [56]. Therefore, A plays the most important role in the model dynamics, which is essential when the targets and obstacles are varying. For a detailed qualitative and quantitative study of the parameter sensitivity of the shunting model, see [56]. To illustrate the importance of A, two simulation studies are carried out under the same condition of the case in Fig. 3, except choosing two different values of parameter A.

First a much smaller A value is chosen, $A = 2$ instead of $A = 10$ as in Fig. 3. Fig. 9(b) shows the activity landscape of the neural network when the target arrives at position (20, 21), which is at the same time point as in Fig. 3(c). Comparing Figs. 9(b) and 3(c), it shows that the activity landscape in Fig. 9(b) has a much longer and wider "tail." This is because that a smaller A value results in a slower passive decaying of the neural activity. The generated trajectory is shown in Fig. 9(a) by solid circles, where the robot follows the target for a few steps and then completely stops at (10, 10), failing to finally reach the target. The slow decaying of the activity causes a fast increase of the neural activity due to the lateral excitatory connections among neurons, yielding a quick saturation of the neural activity. When the robot moves to (10, 10), the neural activity over there is saturated. The robot cannot find its next position through the activity difference and has to stop there forever. The activity landscape of the neural network at time = 6.0 min is shown in Fig. 9(c), where the neural activity is saturated. Therefore, when the value of A is too small, this model cannot function properly due to the quick activity saturation.

Then a much larger A value is chosen, $A = 50$ instead of $A = 10$ as in Fig. 3. Fig. 10(b) shows the neural activity landscape when the target arrives at (20, 21), which is at the same time point as in Figs. 3(c) and 9(b). It shows that the activity landscape has a much shorter "tail" than those in Figs. 3(c) and 9(b). This is because that a larger A results in a faster passive decaying of the neural activity. The generated trajectory is shown in Fig. 10(a), where the robot takes much less steps to reach the target (the traveling route of the robot becomes a straight line). Since the robot moves at the same speed as in Fig. 3, it certainly

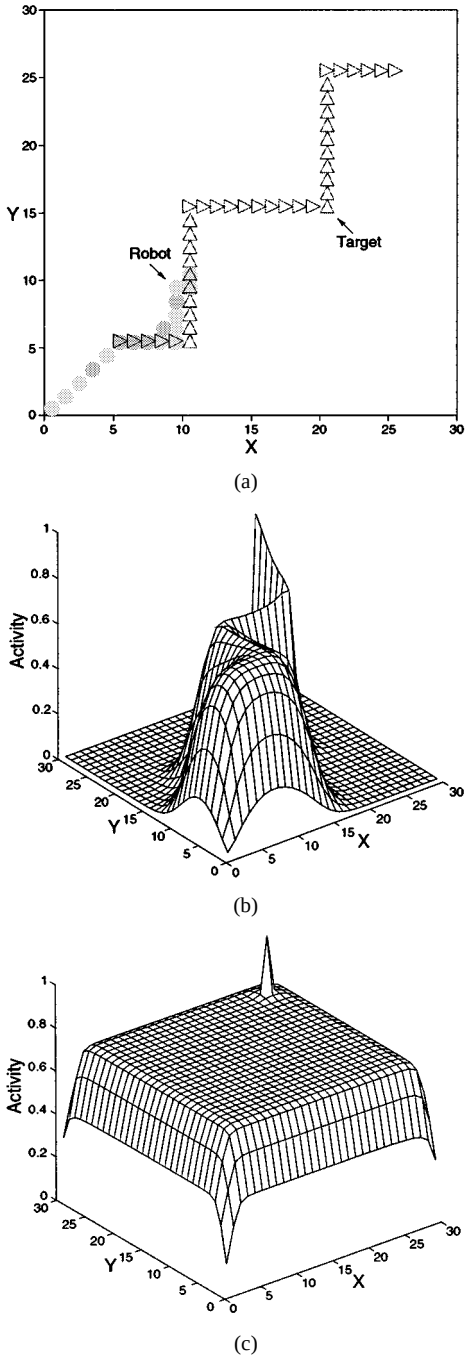


Fig. 9. Trajectory generation with a much smaller A value than in Fig. 3, $A = 2$ instead of $A = 10$ as in Fig. 3. (a) Generated trajectory. (b) Neural activity landscapes when the target arrives at (20, 21). (c) Activity landscape at time = 6.0 min.

takes less time (2.55 min, in comparison to 3.06 min in Fig. 3) to catch the target. The faster decaying of the remaining activity after the target passes away makes the travel "history" of the target disappear faster. The activity propagation from target becomes the domain contribution in forming the neuron activities. Hence, the trajectory of the robot highly aims at the current position of the moving target. Therefore, choosing a large enough A value is necessary for the robot to aim at the target. However, as shown later in Fig. 12(a) and Fig. 17, a smaller A value is necessary if the robot is required to tightly follow the traveling route of the target.

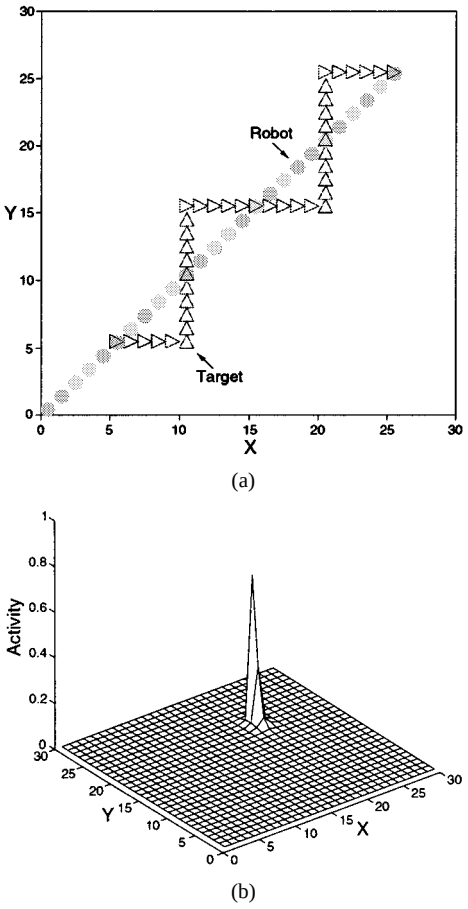


Fig. 10. Trajectory generation with a much larger A value than in Fig. 3, $A = 50$ instead of $A = 10$ as in Fig. 3. (a) Generated trajectory. (b) Activity landscapes when the target arrives at (20, 21).

2) Parameters for the Lateral Connections: Although each neuron has only local connections in a small region and the target is the only positive stimulus, the positive neural activity can propagate to the whole state space of the neural network. Therefore the lateral connections among neurons are essential in forming the dynamic neural activity landscape. It shall be pointed out that the proposed model is not sensitive to the connection weight function $f(a)$ in (6), which can be chosen as any monotonically decreasing function. The connection weight is *solely* determined by parameter μ. Therefore, μ is an important factor in the proposed model. To illustrate the role played by μ, a simulation is carried out under the same condition as in Fig. 3, except choosing a much smaller μ value, $\mu = 0.2$ instead of $\mu = 1$ over there. The activity landscape when the target arrives at (20, 21) is shown in Fig. 11(b). It shows that the activity landscape has a much narrower "tail" than in Fig. 3(c), since a small μ value results in weak lateral connections. The generated trajectory is shown in Fig. 11(a), which is very similar to that in Fig. 3(a), since they have the same A value and have the same system dynamics. However, there is a difference: The traveling route of the robot in Fig. 11(a) takes more steps to reach the target. This is because that a smaller μ value weakens the neural activity propagation from the target, and results in relatively stronger contribution from the remaining activity after the target leaves there. Therefore, to aim at the target, a large enough μ value is necessary.

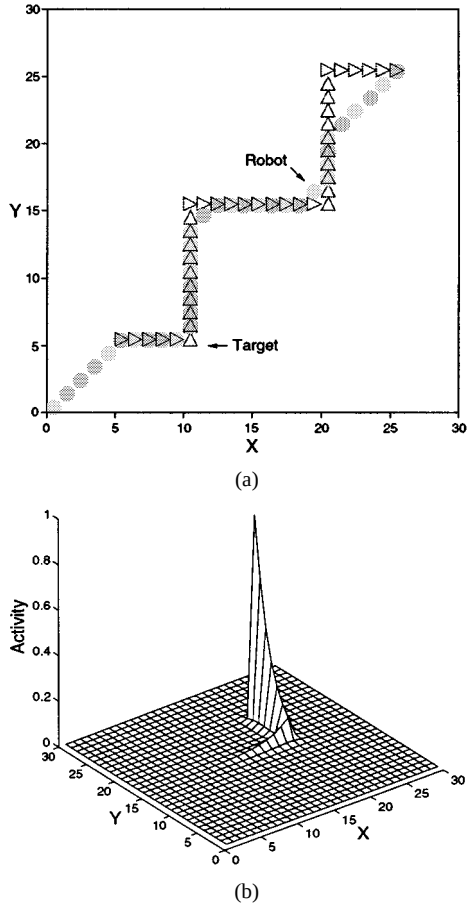


Fig. 11. Trajectory generation with a much smaller μ value than in Fig. 3, $\mu = 0.2$ instead of $\mu = 1$ as in Fig. 3. (a) Generated trajectory. (b) Activity landscapes when the target arrives at (20, 21).

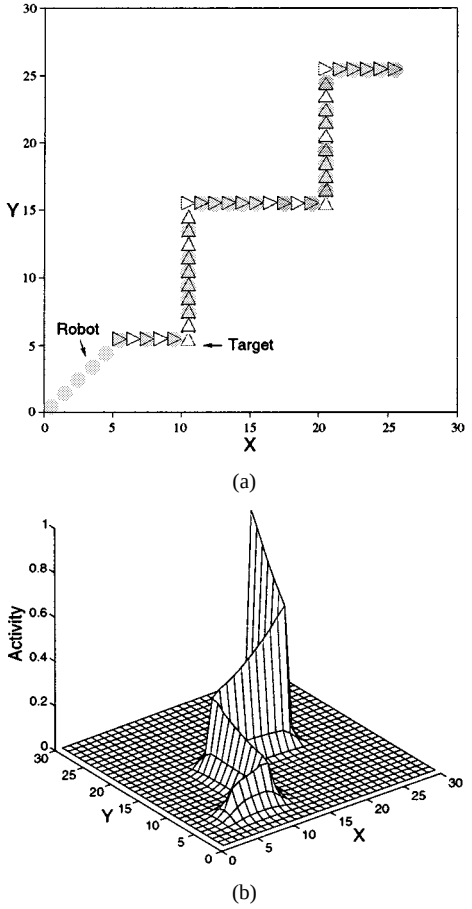


Fig. 12. Trajectory generation with a much smaller μ value than in Fig. 10, $\mu = 0.2$ instead of $\mu = 1$, as in Fig. 10. (a) Generated trajectory. (b) Activity landscapes when the target arrives at (20, 21).

To further illustrate the role played by μ, one more case study is carried out under the same condition as in Fig. 9, except choosing a much smaller μ value, $\mu = 0.2$ instead of $\mu = 1$. Fig. 12(b) shows the activity landscape when the target arrives at (20, 21). As expected, the "tail" is very narrow in comparison to the very wide "tail" in Fig. 9(b), where they are at the same time point. It also shows that both of them have a very long "tail" because they have the same very small A value $A = 2$, which results in a very slow passive decaying of the neural activity. The generated trajectory is shown in Fig. 12(a). It shows that the robot is able to catch the target, whereas the robot in Fig. 9 fails to do so due to the activity saturation. A small μ value results in weak lateral connections and can prevent the possible saturation in neural activity. In addition, Fig. 12(a) shows that the traveling route of the robot tightly follows the travel "history" of the target. It results from both the small A value *and* the small μ value: The small A slows down the passive decaying of the neural activity and increases the influence from remain activity; the small μ weakens the propagation from target activity and decreases the direct influence from the target.

When parameter $\mu > 1$, the propagated activity is amplified and the neural activity is very easy to saturate. A case under the same condition as in Fig. 3, except choosing a larger μ value, $\mu = 2$ instead of $\mu = 1$. Fig. 13 shows the activity landscape when the target arrives at (20, 21), which is at the same time point of Fig. 3(c). It shows that the neural activity landscape has a very long and wide "tail," which is similar to the case in Fig. 9. This results from the large μ value. The activity landscape at time = 6.0 min is shown in Fig. 13(b), where the neural activity is saturated. The generated trajectory is similar to Fig. 9(a) where the robot fails to reach the target due to the neural activity saturation that is caused by the very slow passive decay of neural activity. Therefore, to prevent possible neural activity saturation a smaller μ is necessary; to strengthen the influence from the target, a larger μ is needed. Therefore, parameter μ is usually chosen in the region $\mu \in (0, 1]$.

Parameter r_0 determines the size of the receptive field of the neuron, which is not an important factor in the proposed model. A larger value r_0 will increase the propagation of the neural activity. However, when applying the model to solve maze-type of problems, a small value is necessary, e.g., $r_0 = 2$, since it is required that the activity cannot pass through any obstacles ("wall"). Therefore $r_0 = 2$ is used for all cases.

3) Parameters for the External Inputs: Parameter E determines the amplitude of the external inputs from the target and the obstacles. To keep the target and obstacles staying at the peak and valley, respectively, the value E should be chosen as a very large value over the total input from the lateral connections. Since the neural activity is bounded at the interval $[-D, B]$, by choosing $B = D = 1$ and $r_0 = 2$, the maximum total input

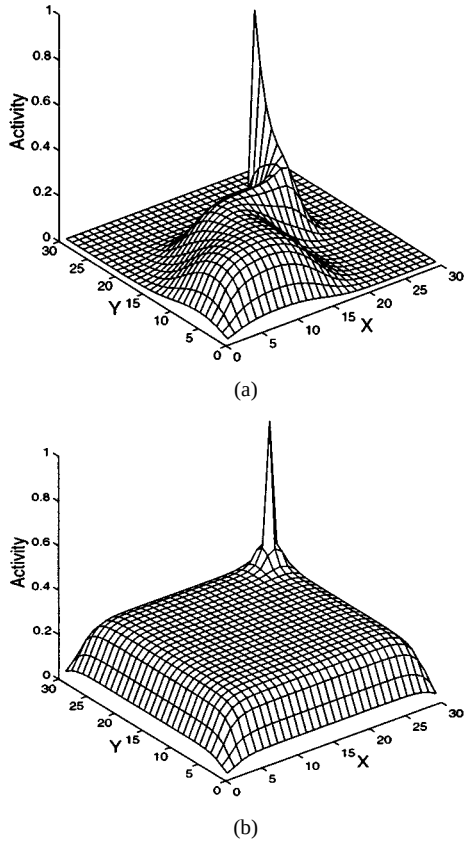


Fig. 13. Trajectory generation with a larger μ value than in Fig. 3, $\mu = 2$ instead of $\mu = 1$, as in Fig. 3. (a) Activity landscapes when the target arrives at (20, 21). (b) Activity landscape at time = 6.0 min.

from lateral connections is eight, then choosing any large E value, e.g., $E > 40 \gg 8$ is good enough. Therefore, parameter E is not an important factor in the proposed model.

In summary, only two parameters A and μ are fundamentally important in the proposed model. The model dynamics is determined by the value of A. Parameter μ determines the activity propagation among neurons. Note that the parameter values in the above simulation studies are chosen in a very wide range, e.g., $A \in [0.2, 50]$ and $\mu \in [0.2, 1]$, and all the simulations for different cases in Section III choose exactly the same model parameters. Therefore, it is obvious that the proposed neural network model is not very sensitive to the model parameters.

B. Model Variations

The neural network model characterized by (7) has only excitatory lateral connections. From the philosophy of this biologically inspired neural network approach for real-time dynamic trajectory generation, alternatively a shunting model with only inhibitory lateral connections is proposed. In addition, by lumping together the excitatory and inhibitory terms and removing the auto gain control terms in the shunting models, two simple models characterized by the additive equations are obtained. These simple additive models for dynamic collision-free trajectory generation also satisfy the philosophy of the proposed neural network approaches presented in Section II. Finally, a comparison study with simulations among these models and the Glasius *et al.* [13] model is presented.

1) Shunting Model with Only Inhibitory Lateral Connections: The neural network model in (7) is characterized by shunting equation with only excitatory lateral connections. In the dynamic neural activity landscape, the neural network design guarantees that the target is always at the peak and the obstacles are always at the valley [e.g., see Fig. 1(b)]. The procedure to generate the real-time robot trajectory can be viewed as that the robot is climbing up the dynamic activity landscape to reach the activity peak. Such a network function is guaranteed by the fact that there are only excitatory connections among neurons.

Alternatively, based on the same concept of the proposed neural network approach, a different neural network model is proposed, which is characterized by a shunting equation with only inhibitory connections among neurons. In this inhibitory model, the inhibitory input S_i^- in (2) results from the target and the lateral connections to its neighboring neurons, while the total excitatory input S_i^+ results from the obstacles only. Hence the dynamics of the ith neuron activity x_i is characterized by

$$\frac{dx_i}{dt} = -Ax_i + (B - x_i)[J_i]^+ - (D + x_i)\left([J_i]^- + \sum_{j=1}^{k} w_{ij}[x_j]^-\right). \quad (16)$$

The definitions of $[a]^+$, $[a]^-$, and w_{ij} are the same as those defined in (7). The external input J_i from the dynamic environment is defined as $J_i = E$ if there is an obstacle $J_i = -E$, if there is a target and $J_i = 0$ otherwise.

The shunting model characterized by (16) guarantees that only the negative neural activity can propagate to the other neurons. For a given present position q_p, the next position can be obtained by

$$q_n \Leftarrow x_{q_n} = \min\{x_j, j = 1, 2, \cdots, k\}. \quad (17)$$

The present position adaptively changes according to the varying environment. The procedure to generate the real-time robot trajectory can be viewed as that a ball (the robot) is naturally falling down to reach the valley of the dynamic activity landscape.

The inhibitory lateral connection shunting model is a stable system, because the neural activity is bounded in the finite region $[-D, B]$. In addition, the stability and convergence can also be rigorously proved using a Lyapunov stability analysis. Introducing the new variables, $y_i = x_i + D$, i.e., where y_i is a nonnegative number varying in the finite interval $[0, B + D]$, (16) can be rewritten into Grossberg general form in (9) via the following substitutions: $a_i(y_i) = y_i$, $b_i(y_i) = [AD - y_i(A + [J_i]^+ + [J_i]^-) + (B + D)[J_i]^+]/y_i$, $c_{ij} = -w_{ij}$, and $d_j(y_j) = -[y_j - D]^-$. Obviously, have $c_{ij} = c_{ji}$ (symmetry), and $a_i(x_i) \geq 0$ (positivity). From the definition of function $[a]^-$, have $d_j'(y_j) = 0$ at $y_j > D$ and $d_j'(y_j) = 1$ at $y_j < D$. Hence, the signal function $d_j(y_j)$ has $d_j'(y_j) \geq 0$ (monotonicity). Therefore, (16) satisfies all the three stability conditions required by the Grossberg's general form in (9) [18]. Choosing the same Lyapunov function candidate in (14), have

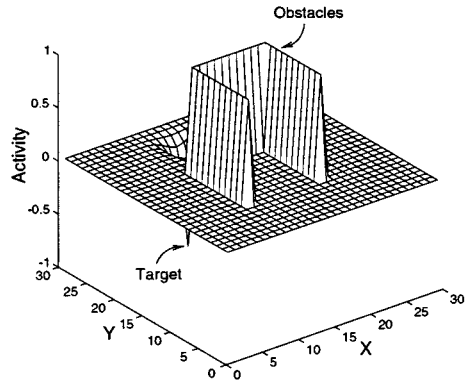


Fig. 14. Stable neural activity landscape using the inhibitory lateral connection shunting model.

$dv/dt \leq 0$ along all the trajectories. Therefore, this inhibitory-lateral-connection neural network system is stable.

A case under the same condition as in Fig. 1 is simulated. The neural network architecture and all the model parameters are chosen as the same as in Fig. 1. The generated trajectory is exactly the same as in Fig. 1(a). The stable (time is long enough) neural activity landscape is shown in Fig. 14. In contrast to Fig. 1(b), the activity peak is at the obstacle location, while the valley is at the target location.

2) Simple Additive Models: If the excitatory and inhibitory terms in the excitatory lateral connection shunting equation in (7) are lumped together and the auto gain control terms are removed, then (7) can be written into a simpler form

$$\frac{dx_i}{dt} = -Ax_i + I_i + \sum_{j=1}^{k} w_{ij}[x_j]^+. \tag{18}$$

This is an additive equation, which is widely applied to a lot of areas such as vision, associative pattern learning and pattern recognition [16], [18]. The term $I_i + \sum_{j=1}^{k} w_{ij}[x_j]^+$ represents the total input to the ith neuron from the external and lateral connections. The definitions of I_i, $[a]^+$, and w_{ij} are the same as those in (7). The nonlinear function $[a]^+$ guarantees that only the positive neural activity can propagate to the other neurons. The very large external input E guarantees that target and obstacles stay at the peak and valley of the neural activity landscape, respectively. Therefore this simple additive model satisfies the fundamental concept of the proposed neural network approach presented in Section II. This additive model is capable of generating dynamic collision-free robot trajectories in most situations.

Although the neural activity characterized by the additive equation in (18) is not bounded as in the shunting model, it is easy to prove that this additive model is a stable system using a Lyapunov stability theory. Equation (18) can be rewritten into the Grossberg's general form in (9) by substituting

$$a_i(x_i) = 1 \tag{19}$$

$$b_i(x_i) = -Ax_i + I_i \tag{20}$$

$$c_{ij} = -w_{ij} \tag{21}$$

and

$$d_j(x_j) = [x_j]^+. \tag{22}$$

Obviously, have $c_{ij} = c_{ji}$ (symmetry), $a_i(x_i) \geq 0$ (positivity), and $d_j'(x_j) \geq 0$ (monotonicity). Thus (18) satisfies all the three stability conditions required by the Grossberg's general form in (9) [18]. Therefore this additive neural network system is stable.

Similarly, a different additive model is obtained from the inhibitory lateral connection shunting model in (16). By lumping together the excitatory and inhibitory terms and removing the auto gain control terms in (16), the dynamics of the ith neuron is given by a simple additive equation

$$\frac{dx_i}{dt} = -Ax_i + J_i + \sum_{j=1}^{k} w_{ij}[x_j]^- \tag{23}$$

where $J_i + \sum_{j=1}^{k} w_{ij}[x_j]^-$ is the total input to the ith neuron from the external and lateral connections. The definitions of J_i, $[a]^-$, and w_{ij} are the same as those in (16). The nonlinear function $[a]^-$ guarantees that only the negative neural activity can propagate to the other neurons, and the large external input E guarantees that the target and obstacles are at the valley and peak of the neural activity landscape, respectively. Therefore, this additive model with only inhibitory lateral connections follows the philosophy of the proposed neural network approach. It is capable of generating dynamic collision-free trajectories.

The additive neural network in (23) can also be proved to be stable using a Lyapunov stability analysis. Equation (23) can be rewritten into Grossberg's [18] general form in (9) by the following substitutions: $a_i(x_i) = 1, b_i(x_i) = -A_i + J_i, c_{ij} = w_{ij}$, and $d_j(x_j) = -[x_j]^-$. Again, we have $c_{ij} = c_{ji}$ (symmetry), $a_i(x_i) \geq 0$ (positivity), and $d_j'(x_j) \geq 0$ (monotonicity). Therefore, all the three stability conditions required by the general form in (9) are satisfied by additive model in (23). This additive neural network system is stable.

There are a lot of important differences between the shunting models in (7) and (16) and the additive models in (18) and (23), although the additive models are computational simpler, and can also generate real-time trajectories with obstacle avoidance in most situations. First, by rewriting the shunting and additive models into the Grossberg general form in (9), unlike the additive model in (18) with a constant $a_i(x_i) = 1$ and a linear function $b_i(x_i)$, for the shunting model in (7) the amplification function $a_i(y_i)$ in (10) is not a constant, and the self-signal function $b_i(y_i)$ in (11) is nonlinear. Second, the shunting model in (7) has two *auto gain control* terms [$(B - x_i)$ and $(D + x_i)$], which result in that the neural dynamics of (7) remains sensitive to input fluctuations [18]. Such a property is important for the real-time trajectory generations when the target and obstacles are varying. In contrast, the dynamics of the additive equation may saturate in many situations [18]. Third, the activity of the shunting model is bounded in the finite interval $[-D, B]$, while the activity in the additive model does not have any bounds. A detailed analysis of the shunting model and the additive model can be found in [18], [41], [42], and [56].

Glasius *et al.* [13] proposed a similar neural network model for real-time trajectory generation, where the *output* z_i of the ith neuron is modeled by

$$\frac{dz_i}{dt} = -z_i + g\left(I_i + \sum_{j=1}^{k} w_{ij} z_j\right) \tag{24}$$

which is derived from a simple discrete model

$$z_i(t+1) = g\left(I_i + \sum_{j=1}^{k} w_{ij} z_j(t)\right) \tag{25}$$

where $g(a)$ is an input–output transfer function that can be any sigmoid function, e.g., a function defined as $g(a) = 0$ if $a < 0$, $g(a) = \beta a$ if $0 \leq a \leq 1$, $g(a) = 1$ if $a > 1$, where β is a constant, and $\beta \in [0, 1]$. The connection weight w_{ij} is defined as a function of the distance d_{ij} between ith and jth neurons, e.g., $w_{ij} = 1$ if $d_{ij} < 2$ and $w_{ij} = 1$ otherwise [14]. This model is a Hopfield-type neural network [13].

Comparing the Glasius *et al.* model in (24) and the proposed shunting model in (7) or the simple additive model in (18), the most important difference is that (24) has a constant passive decay rate at $A = 1$. As discussed in Section IV, parameter A plays an essential role in dynamic trajectory generation. Although parameters μ in (18) or β in (24) can prevent the possible saturation in the neural activity, as discussed in Section IV they do not have any effects to the transient response characteristics of the model, i.e., the system dynamics does not depend on the neural connection parameters μ or β. Therefore, the Glasius *et al.* model has limitations with fast dynamic systems. It cannot perform properly in a fast changing environment. For example, it requires that the robot dynamics must be faster than that of the target and the obstacle dynamics [13], [14].

Another major difference between the Glasius *et al.* model in (24) and the proposed shunting model in (7) or the simple additive model in (18) is that (24) describes the dynamics of the neuron *output* z_i, which is derived from a simple discrete input–output ($I_i + \sum w_{ij} z_j$ versus z_i) function; while (7) or (18) characterizes the dynamics of the neural *activity* x_i, which is derived from Hodgkin and Huxley's biological model [20], and describes the input (S_i^+ and S_i^-), output ($[x_i]^+$) and activity (x_i) of a neuron. In addition, unlike the Glasius *et al.* model in (24) that does not model the neural activity, the proposed shunting model has a continuous neural activity with both upper and lower bounds. Therefore, the proposed model is more biologically plausible. By doing a linear, invertible mathematical transformation $x_i = \sum w_{ij} z_j + I_i$ (note that biologically this is not true, since x_i is the neural activity, while $I_i + \sum w_{ij} z_j$ is the total input), a similar equation of (18) in a special case of $A = 1$ can be obtained from (24). This transformation from a nonlinear signal function of sum, $g(\sum z_j)$, to a sum of nonlinear signals, $\sum g(y_j)$, is usually called "S Σ exchange" in neural network analysis.

3) Comparison Study Among Models: A comparison study is carried out to illustrate the differences among these models. A target catching case is designed. In a 2-D Cartesian workspace, the target starts to move from position (10, 5) at a speed of 25 block/min. The traveling route of the target is shown in Fig. 15 by hollow triangles. After target reaches (20, 29) at time = 3.0 *min*, it disappears, i.e., the target leaves the workspace. The robot moves at a speed of 20 block/min starting from (0, 0). The task of the robot is to *catch the target before the target disappears*. Since the robot moves slower than the target, it can catch the target only if it can find a shorter traveling route than the target.

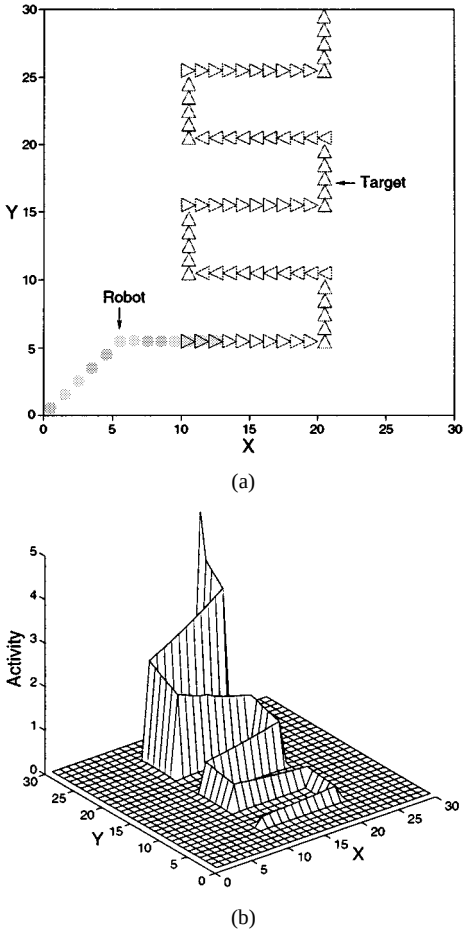


Fig. 15. Trajectory generation of a mobile robot to catch a moving target using the additive model in (18) with $A = 1$ and $\mu = 0.01$. (a) Generated trajectory. (b) Neural activity landscape when the target leaves the workspace.

First, the proposed additive model in (18) is used. The neural network has 30×30 neurons with zero initial neural activities, and the model parameters A and μ are chosen as $A = 1$ and $\mu = 0.01$. All the other model parameters are chosen as the same as in previous cases, i.e., $B = D = 1$, $r_0 = 2$ and $E = 100$, which will be also used in the following simulations. The activity landscape of the neural network when the target leaves the workspace is shown in Fig. 15(b). As expected, it has a very long and very narrow "tail," since a very small A value and a very small μ value are used. The generated trajectory of the robot is shown in Fig. 15(a) by solid circles. It shows that the robot ends at (12, 5) and fails to catch the target.

Second, the Glasius *et al.* model in (24) is used. Parameter β is chosen as $\beta = 0.1$ [14]. The generated trajectory is shown in Fig. 16(a), where the robot ends at (20, 6) and fails to catch the target. The activity landscape when the target leaves the workspace is shown in Fig. 16(b), which is qualitatively the same as that in Fig. 15 except some quantitative differences.

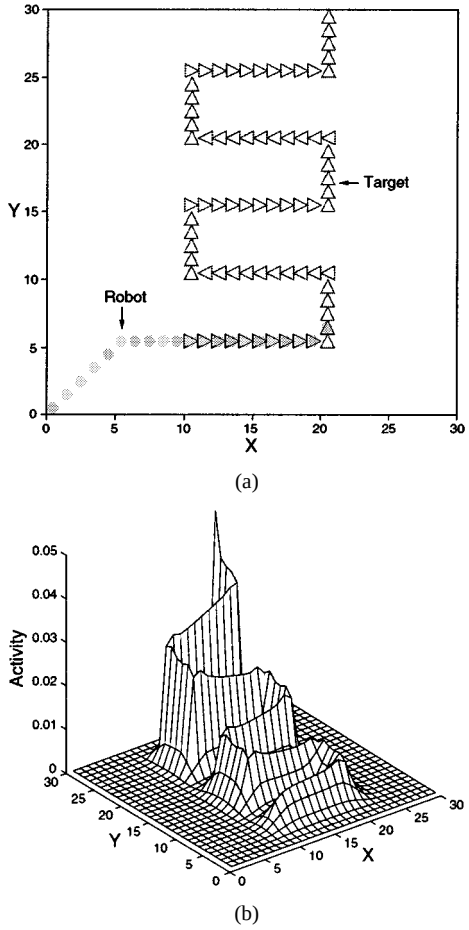


Fig. 16. Trajectory generation to catch a moving target using the additive model in (24). (a) Generated trajectory. (b) Activity landscape when the target disappears.

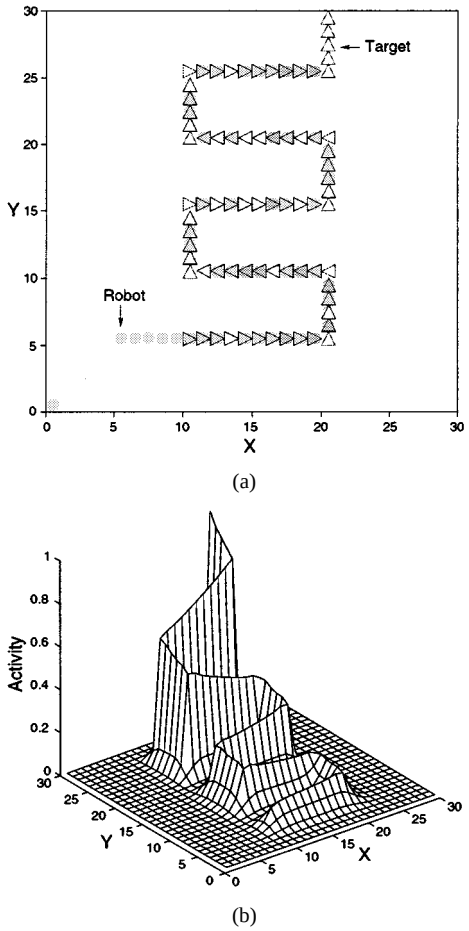


Fig. 17. Trajectory generation to catch a moving target using the shunting model in (7) with $A = 1$ and $\mu = 0.1$. (a) Generated trajectory. (b) Activity landscape when the target disappears.

Third, the proposed shunting model in (7) is used with $A = 1$ and $\mu = 0.1$. The neural activity landscape when the target leaves the workspace is shown in Fig. 17(b). Unlike those in Figs. 17(b) and 16(b), the neural activity in Fig. 17(b) is bounded in [0, 1], although their activity landscapes are qualitatively the same. The generated trajectory is shown in Fig. 17(a). It shows that the robot tightly follows the traveling route of the target. However, the robot also fails to catch the target before the target leaves the workspace because it does not travel a much shorter route than the target. If the task is to detect and follow the traveling route of the target, the robot does a very good job. Unlike the cases in Figs. 15(a) and 16(a) where the robot ends at a location forever, the robot in Fig. 17(a) can continuously follow the traveling route of the target.

Fourth, the proposed additive model in (18) is used with $A = 50$ and $\mu = 1$. The generated trajectory is shown in Fig. 18(a). It shows that the robot travels a *shorter* route than the target and catches the target at position (11, 15). The neural activity landscape when the robot catches the target is shown in Fig. 18. As expected, it has a very short "tail" due to the very large A value.

Finally, the proposed shunting model in (7) is used with the same parameters as in the case depicted in Fig. 18, i.e., $A = 50$ and $\mu = 1$. The generated trajectory is the same as that shown in Fig. 18(a). The neural activity landscape when the robot catches the target is qualitatively the same as that in Fig. 18(b).

By choosing a difference β value, the Glasius *et al.* model can generate a trajectory similar to Fig. 17, since the role of β is similar to μ in the proposed models. However, the Glasius *et al.* model cannot generate a trajectory similar to Fig. 18 that allows the robot to achieve the task, i.e., travel a much shorter trajectory than the target trajectory to catch the target before it disappears. The Glasius *et al.* model requires that the robot moves faster than the target [13], [14].

In summary, the above simulations demonstrate that parameter A is fundamental important in real-time trajectory generation, particularly when the environment is changing in a fast manner. The neural dynamics of the additive models may saturate in some situations, but the shunting models do not.

V. Conclusions

In this paper, a novel biologically inspired neural network approach is proposed for the dynamic collision-free trajectory generation in an arbitrarily dynamic environment. The state space of the neural network can be the Cartesian workspace of mobile robots or the joint space of multijoint robot manipulators. Several model variations are presented and the differences are compared by descriptive analysis and simulation studies. The proposed approach is applied to the real-time trajectory generation with obstacle avoidance for a mobile robot and a multi-joint

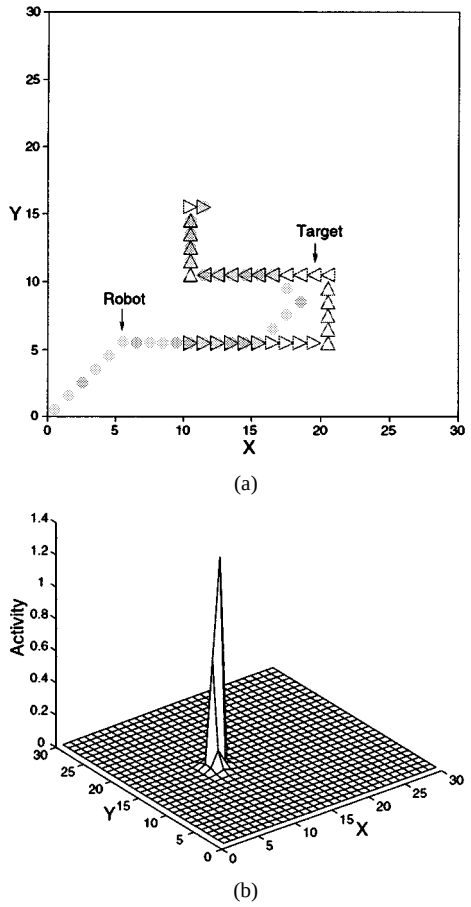


Fig. 18. Trajectory generation to catch a moving target using the additive model in (18) with $A = 50$ and $\mu = 1$. (a) Generated trajectory. (b) Activity landscape when the robot catches target.

robot manipulator. The optimal real-time trajectory is generated through the dynamic activity landscape that represents the varying environment. The stability and convergence of the proposed models are guaranteed by a qualitative analysis and a rigorous Lyapunov stability analysis.

Some points are worth to mention about the proposed neural network approach to dynamic collision-free trajectory generation.

- This model is biologically plausible. It is originally derived from Hodgkin and Huxley's biological membrane model [20]. The neural activity is a continuous analog signal and has both upper and lower bounds. In addition, the continuous activity prevents the possible oscillations related to parallel dynamics of discrete neurons [13], [31].
- The model algorithm is computationally *efficient*. The optimal robot trajectory is generated *without* explicitly searching over the free workspace or the collision paths, *without* explicitly optimizing any global cost functions, *without* any prior knowledge of the dynamic environment, and *without* any learning procedures.
- The computational complexity *linearly* depends on the state space size of the neural network. Each neuron in the neural network has only local lateral connections, which does not depend on the size of the overall neural network. However, if the state space is F-D, and each dimension is discretized with d neurons, then the total neurons will be d^F. Therefore, the proposed model will be computational expensive for trajectory generation of robot manipulators with many degrees of freedom.
- This model can perform properly in an arbitrarily dynamic environment, even with sudden environmental changes, such as suddenly adding or removing obstacles or targets (additional study of this property can be found in [53] and [54]). The neural network system is characterized by a continuous shunting model, it is stable and keeps sensitive to variations in the environment [18].
- The proposed models do not suffer from undesired local minima, i.e., the robot will not be trapped in deadlock situations such as with concave U-shaped obstacles and in a complex maze-solving type of problems. The target globally influences the whole workspace through neural activity propagation to all directions in the same manners. The trajectory is generated through the dynamic activity landscape of the neural network.
- This model is not very sensitive to the model parameters and the connection weight function. Only two model parameters A and μ are important factors. The model parameters can be chosen in a very wide range. The weight function can be any monotonically decreasing function.
- This model is not sensitive to any irrelevant obstacles. There are no inhibitory lateral connections in the neural network. The negative neural activity from the obstacle location stays locally only without propagating to any other neurons. Thus the obstacles have only local effect to push the robot away to avoid collisions. Therefore, unlike some previous models (e.g., [21]), the irrelevant obstacles do not influence the global trajectory generation.
- The proposed model is capable of generating real-time collision-free trajectories of a robot with multiple moving targets and the trajectories of multiple robots in a common workspace. This study is presented in [54].
- Based on the proposed model, an extended model is capable of generating a real-time path with safety consideration, i.e., by choosing suitable strength of obstacle clearance, the extended model can plan a "comfortable" path that does not clip the corners of obstacles nor runs down the edges of obstacles. Thus it does not suffer from either the "too close" (narrow safety margin) or "too far" (waste) problems [37], [60], [61]. This study is presented in [52] and [53].
- Based on the proposed model, another extended model can be applied to real-time collision-free navigation of holonomic and nonholonomic car-like mobile robots. In many situations, e.g., when the size of the robot is comparable to the free workspace, the robot should be considered with its shape and size. Unlike most other path planning methods for car-like robots where a local collision checking has to be taken at every step of robot movement, no local collision checking procedure is needed in the extended model. This study is presented in [54] and [55].

REFERENCES

[1] K. S. Al-Sultan and D. S. Aliyu, "A new potential field-based algorithm for path planning," *J. Intell. Robot. Syst.*, vol. 17, no. 3, pp. 265–282, 1996.

[2] J. C. Alexander *et al.*, "Shortest distance paths for wheeled mobile robots," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 657–662, Oct. 1998.

[3] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *Int. J. Robot. Res.*, vol. 10, no. 6, pp. 628–649, 1991.

[4] A. Bicchi *et al.*, "Planning shortest bounded-curvature paths for a class of nonholonomic vehicles among obstacles," *J. Intell. Robot. Syst.*, vol. 16, no. 4, pp. 387–405, 1998.

[5] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 224–233, Feb. 1985.

[6] A. Chohra and C. Benmehrez, "Neural navigation approach for intelligent autonomous vehicles (iav) in partially structured environments," *Appl. Intell.*, vol. 8, pp. 219–233, 1998.

[7] J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE J. Robot. Automat.*, vol. RA-1, pp. 31–41, Jan. 1985.

[8] J. G. De Lamadrid and M. L. Gini, "Path tracking through uncharted moving obstacles," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, pp. 1408–1422, Nov./Dec. 1990.

[9] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artif. Intell.*, vol. 31, no. 3, pp. 295–353, 1987.

[10] T. Fujii *et al.*, "Multilayered reinforcement learning for complicated collision avoidance problems," in *Proc. IEEE Int. Conf. Robot. Automat.*, Leuven, Belgium, May 1998, pp. 2186–2191.

[11] 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. Syst., Man, Cybern. B*, vol. 26, pp. 485–496, June 1996.

[12] R. Glasius *et al.*, "Population coding in a neural net for trajectory formation," *Network: Computat. Neural Syst.*, vol. 5, pp. 549–563, Aug. 1994.

[13] ——, "Neural network dynamics for path planning and obstacle avoidance," *Neural Networks*, vol. 8, no. 1, pp. 125–133, 1995.

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

[15] S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Stud. Appl. Math.*, vol. 52, pp. 217–257, 1973.

[16] ——, *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*. Boston, MA: Reidel, 1982.

[17] ——, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 815–926, May 1983.

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

[19] A. L. Hodgkin, *The Conduction of the Nervous Impulse*. Liverpool, U.K.: Liverpool Univ. Press, 1964.

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

[21] J. Ilari and C. Torras, "2d path planning: A configuration space heuristic approach," *Int. J. Robot. Res.*, vol. 9, no. 1, pp. 75–91, 1990.

[22] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, pp. 72–89, 1986.

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

[24] R. Kimmel *et al.*, "Multi-valued distance maps for motion planning on surfaces with moving obstacles," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 427–436, June 1998.

[25] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 73, pp. 49–60, 1982.

[26] J. C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.

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

[28] Z. X. Li and T. D. Bui, "Robot path planning using fluid model," *J. Intell. Robot. Syst.*, vol. 21, pp. 29–50, 1998.

[29] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Comput.*, vol. 32, pp. 108–320, 1983.

[30] V. J. Lumelsky and A. A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 1058–1063, Nov. 1986.

[31] C. M. Marcus *et al.*, "Associative memory in an analog iterated-map neural network," *Phys. Rev. A*, vol. 41, no. 6, pp. 3355–3364, 1990.

[32] K. Marti and S. Qu, "Path planning for robots by stochastic optimization methods," *J. Intell. Robot. Syst.*, vol. 22, pp. 117–127, 1998.

[33] M. Meng and X. Yang, "A neural network approach to real-time trajectory generation," in *IEEE Proc. Int. Conf. Robot. Automat.*, Leuven, Belgium, May 1998, pp. 1725–1730.

[34] F. Muñiz *et al.*, "Neural controller for a mobile robot in a nonstationary environment," in *Proc. Second IFAC Conf. Intell. Autonom. Veh.*, Helsinki, Finland, June 1995, pp. 279–284.

[35] P. Muraca *et al.*, "Cooperative neural field for the path planning of a robot arm," *J. Intell. Robot. Syst.*, vol. 15, no. 1, pp. 11–18, 1996.

[36] I. Namgung and J. Duffy, "Two dimensional collision-free path planning using linear parametric curve," *J. Robot. Syst.*, vol. 14, no. 9, pp. 659–673, 1997.

[37] H. Noborio *et al.*, "A feasible motion-planning algorithm for a mobile robot on a quadtree representation," *Proc. IEEE Int. Conf. Robot. Automat.*, pp. 327–332, 1989.

[38] 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.

[39] G. Oriolo, G. Ulivi, and M. Vendittelli, "Fuzzy maps: A new tool for mobile robot perception and planning," *J. Robot. Syst.*, vol. 14, no. 3, pp. 179–197, 1997.

[40] ——, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Trans. Syst., Man, Cybern. B*, vol. 28, pp. 316–333, June 1998.

[41] H. Öğmen and S. Gagné, "Neural models for sustained and on-off units of insect lamina," *Biol. Cybern.*, vol. 63, pp. 51–60, 1990.

[42] ——, "Neural network architecture for motion perception and elementary motion detection in the fly visual system," *Neural Networks*, vol. 3, pp. 487–505, 1990.

[43] S. C. Pei and J. H. Horng, "Finding the optimal driving path of a car using the modified constrained distance transformation," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 663–670, Oct. 1998.

[44] R. Plonsey and D. G. Fleming, *Bioelectric Phenomena*. New York: McGraw-Hill, 1969.

[45] A. Pruski and S. Rohmer, "Robust path planning for nonholonomic robots," *J. Intell. Robot. Syst.*, vol. 18, pp. 329–350, 1997.

[46] H. J. Ritter *et al.*, "Topology-conserving maps for learning visuo-motor-coordination," *Neural Networks*, vol. 2, pp. 159–189, 1989.

[47] C. Seshadri and A. Ghosh, "Optimum path planning for robot manipulators amid static and dynamic obstacles," *IEEE Trans. Syst., Man, Cybern*, vol. 23, pp. 576–584, Mar./Apr. 1993.

[48] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artif. Intell.*, vol. 99, no. 1, pp. 21–71, Feb. 1998.

[49] A. Tsoularis and C. Kambhampati, "On-line planning for collision avoidance on the nominal path," *J. Intell. Robot. Syst.*, vol. 21, pp. 327–371, Apr. 1998.

[50] L. Wyard-Scott and Q.-H. M. Meng, "A potential maze solving algorithm for a micromouse robot," in *Proc. IEEE Pacific Rim Conf. Commun., Comput., Signal Process.*, Victoria, BC, Canada, May 1995, pp. 614–618.

[51] 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.

[52] ——, "An efficient neural network method for real-time motion planning with safety consideration," *Robot. Auton. Syst.*, vol. 32, no. 2–3, pp. 115–128, 2000.

[53] ——, "Real-time collision-free path planning of robot manipulators using neural network approaches," *Auton. Robots*, vol. 9, no. 1, pp. 27–39, 2000.

[54] ——, "An efficient neural network model for path planning of car-like robots in dynamic environment," *Int. J. Adv. Comput. Intell.*, vol. 4, no. 3, 2001.

[55] S. X. Yang *et al.*, "A biological inspired neural network approach to real-time collision-free motion planning of a nonholonomic car-like robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Takamatsu, Japan, Oct. 2000, pp. 239–244.

[56] X. Yang, "A neural network architecture for visual information processing in vertebrate retina," M.S. thesis, Univ. Houston, Houston, TX, Dept. Elect. Comput. Eng., Dec. 1996.

[57] ——, "Neural network approaches to real-time motion planning and control of robotic systems," Ph.D. thesis, Univ. Alberta, Edmonton, AB, Dept. Elect. Comput. Eng., June 1999.

[58] X. Yang and M. Meng, "Dynamical trajectory generation with collision free using neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Victoria, BC, Canada, Oct. 1998, pp. 1634–1639.

[59] E. Zalama *et al.*, "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.

[60] 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.
[61] D. Zhu and J. C. Latombe, "New heuristic for efficient hierarchical path planning for mobile robot," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 9–20, Feb 1991.

Simon X. Yang (S'97–M'99) 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, 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.

Since August 1999, he has been an Assistant Professor of engineering systems and Computing in the School of Engineering at the University of Guelph, Guelph, ON, Canada. Currently, he is the Director of the Advanced Robotics and Intelligent Systems (ARIS) Lab, University of Guelph. His research interests include robotics, intelligent systems, control systems, and computational neuroscience. He has published over 60 journal papers, book chapters, and conference proceedings.

Max Meng (S'92–M'92) received the Ph.D. degree from the University of Victoria, Victoria, BC, Canada, in 1992 .

He is currently a Professor of electrical and computer engineering and the Director of the Advanced Robotics and Teleoperation (ART) Lab, University of Alberta, Edmonton, AB, Canada. His research interests include robotics, intelligent control systems, and human-machine interface. He has published approximately 100 journal papers and edited volumes, book chapters, and conference proceedings.

Dr. Meng is the General Chair of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'01), and the General Chair of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05). He is the Chair of IEEE Northern Canada Section. Among many of his awards, he is the recipient of the IEEE Third Millennium Medal award. He is the editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS.