A Neural Network Approach to Dynamic Task Assignment of Multirobots

Anmin Zhu and Simon X. Yang, *Member, IEEE*

***Abstract*—In this paper, a neural network approach to task assignment, based on a self-organizing map (SOM), is proposed for a multirobot system in dynamic environments subject to uncertainties. It is capable of dynamically controlling a group of mobile robots to achieve multiple tasks at different locations, so that the desired number of robots will arrive at every target location from arbitrary initial locations. In the proposed approach, the robot motion planning is integrated with the task assignment, thus the robots start to move once the overall task is given. The robot navigation can be dynamically adjusted to guarantee that each target location has the desired number of robots, even under uncertainties such as when some robots break down. The proposed approach is capable of dealing with changing environments. The effectiveness and efficiency of the proposed approach are demonstrated by simulation studies.**

***Index Terms*—Multirobots, neural network, self-organizing map (SOM), task assignment.**

I. Introduction

MULTIROBOT systems have been the subject of much research since the 1970s in a wide variety of tasks. A mobile robot team can complete an assigned task rapidly and efficiently by dividing the overall task into several subtasks, assigning them to individual robots, and executing the subtasks simultaneously. Multirobot systems have obvious advantages such as faster operation, higher efficiency, and better reliability than a single robot system. The key challenge is the coordination and cooperation of these robots to achieve a satisfactory performance of the overall task. Task assignment of multirobot systems is to control a group of mobile robots so that they move to their designated target locations, with the coordination and cooperation of each robot.

There have been some studies on efficient control of a group of robots moving to target locations. Most algorithms are proposed for the task assignment problem in static environments, e.g., the graph matching algorithm [1], network simplex algorithm [2], distributed auction algorithm [3], genetic algorithm [4], agent based algorithm [5], and dynamic Tabu search algorithm [6]. These algorithms mainly focus on the target assignment problem without considering the motion planning of robots. The consequence is that the robots cannot move and have to wait until their destinations are finalized. In addition, these approaches cannot handle the situation with movable targets. Starke *et al.* [7] proposed an algorithm using pattern formation principles motivated by spontaneous pattern formation in fluids, where a fluid heated from below and cooled from above can generate rolls or hexagonal patterns. The key idea of pattern formation principles for obtaining the self-organizing behavior is to construct a suitable dynamic system where the dynamic variables can be identified with the configuration of the robot system. The algorithm carries out the assignment of autonomous mobile robots to targets in a two-dimensional (2-D) or 3-D space with an error resistance feature. However, it cannot be used in complicated situations, such as several robots being assigned to one target location. Furthermore, this approach is not able to handle situations with movable targets. Inspired by the biological system phenomena that many complex patterns appear to be the results of self-organization among homogeneous cells, Shen *et al.* [8], [9] developed a model called the "Digital Hormone Model" for multirobot self-organization to form a global pattern by assuming a robot as a cell. It is suitable for some tasks such as distributing and monitoring a given area or building; self-repairing damages to the global patterns; and avoiding pitfalls by detouring. For the searching and seizing target task, the algorithm cannot deal with multiple targets and dynamic target situations. If there are two targets and four robots, and all the robots are near one target and far away from another one, the result would be that one target would attract all of the four robots while the other target is not caught. The negotiation and cooperation between the robots is not considered in this algorithm.

The research on unmanned air vehicles (UAVs) [10]–[12] has a similar problem to the task assignment problem, which assigns a group of air vehicles to transit through several target locations while avoiding threats. UAV usually deals only with static environments. Beard *et al.* [10] proposed a solution to the cooperative control problem by decomposing the problem into subproblems, including target assignment, path planning, coordinated UAV intercept, and trajectory generation and following. First, a global map is created by Voronoi diagram approach, which describes the air vehicle positions, the target positions, the threat point positions, and the possible paths to minimize the threat. Based on the Voronoi diagram, the best paths, including the median length cost and median threat exposure cost for every air vehicle to every target, are calculated. Then each vehicle is assigned to a target by a satisfying paradigm that minimizes the group path length to the target, minimizes group threat exposure, maximizes the number of vehicles to each target, and maximizes

Manuscript received August 2, 2005; revised February 21, 2006. This work was supported by Natural Sciences and Engineering Research Council (NSERC).

A. Zhu is with the Advanced Robotics and Intelligent Systems (ARIS) Laboratory, the School of Engineering, University of Guelph, Guelph, ON N1G 2W1, Canada, and also with the College of Electrical and Information Engineering, Shenzhen University, Shenzhen 518060, China.

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

Digital Object Identifier 10.1109/TNN.2006.875994

the number of targets visited. After that, a trajectory for each vehicle to its respective target is planned while the timing coordination of target interception is considered. Finally, the trajectories are generated to control the velocities of each vehicle. When some other dynamic threats pop up, the vehicle then replans the path to avoid the dynamic threats. This is an end-to-end solution that focuses on several different aspects such as map building by Voronoi diagram approach, target management using satisfying game theory, intercept management, and trajectory generation. This solution does not focus on dynamic targets and dynamic threats, nor on dynamic vehicles, such as new vehicles are added or some vehicles break down suddenly. Because of the limitation of the Voronoi diagram approach, the solution is not suitable to dynamic environments.

Other studies have focused on priority control of a small group of robots that normally first break a task into several subtasks, with competition but little cooperation among the robots [13]–[17]. Miyata *et al.* [16] proposed a method to deal with the problem of transporting an object from one to another place by a group of robots in an unknown static environment. This method focuses on how to divide the transport task into many subtasks with priorities and how to assign the subtasks to different robots. The subtask may include searching around in the workspace, recognizing the object that needs to be moved, displacing movable obstacles, and handling an object. The definition of the task assignment focuses on different subtasks and the priorities of these subtasks. This method is suitable for a small group of robots and in a static environment only. Uchibe *et al.* [17] proposed a method for assigning tasks to a group of robots, which predesigns models for the task, then dynamically assigns the models to different robots. This method focuses on solving conflict between model selections. It is suitable for a small group of robots with a task that can be divided into several subtasks such as transporting an object by several robots. Brandt *et al.* [13] defined another task assignment problem in multiagent systems and proposed an algorithm to solve the problem. It focuses on different tasks created by the contractors, and different contractees who are interested in and will execute the tasks. Both the contractors and contractees want to get maximum benefits and should negotiate. Among the contractees there is more competition but less cooperation.

There are some methods available to deal with dynamic environments or robot failures after modifications [18], [19]. However, to modify for dynamic environments, stability problems or additional computational costs may arise. Although there are many neural network (NN) approaches proposed for robot systems [20]–[22], most of them deal with single robot or completely known environments.

In this paper, a novel self-organizing map (SOM) based NN approach is developed for multirobot systems to tackle the task assignment problem that focuses on the self-organization issue with a large number of robots and a large number of targets in static or dynamic environments. Because of the self-organizing property, the proposed new method is stable, robust, and suitable to dynamic environments with uncertainty. In the proposed approach, the robot motion planning is integrated with the task assignment, thus the robots start to move once the overall task is given. The group of mobile robots can automatically arrange the total task, and dynamically adjust their motion whenever the environment is changed, such as some robots break down, some robots and/or some tasks are added, or some tasks are modified.

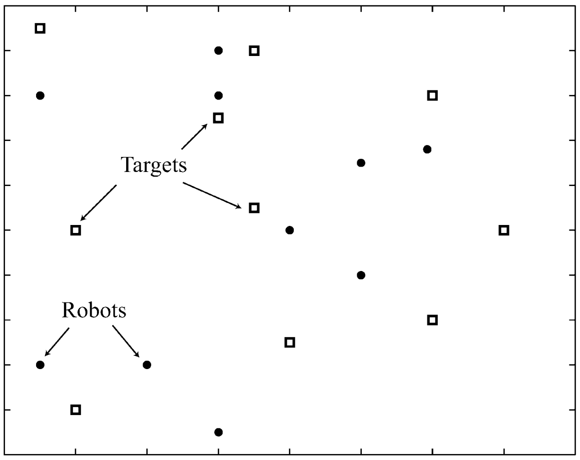

Fig. 1. Workspace with multiple robots and targets.

This paper is organized as follows. The problem of task assignment of multirobots is first defined in Section II. Section III presents the proposed SOM based NN algorithm for task assignment of a multirobot system. The simulation results on task assignments are provided in Section IV. A brief discussion and some concluding remarks of the proposed approach are given in Sections V and VI, respectively.

II. Problem Statement

In multirobot systems, the main challenge is the coordination and cooperation of the robots in performing a single task. In this paper, assume there are a group of autonomous mobile robots distributed randomly in a bounded area and there are a set of targets also randomly distributed in this workspace. Every target requires a specific number of robots to complete a task at that location. The goal is to dynamically assign a team of robots to every target location with a minimal or near-minimal total cost. For each robot, the cost is evaluated by the travel distance from its initial position to its final location. The total cost is defined as the sum of all of the costs from every robot. The task is completed when every target has the desired number of robots reached. An example workspace with the initial robot and target locations is shown in Fig. 1, where the dots represent initial mobile robot locations and the squares represent the target locations. In addition, it is assumed that the robots are homogeneous mobile robots with basic capabilities for navigation, obstacle avoidance, and location recognition. This paper only covers how to dynamically assign the robots to the targets and how to control the motion of mobile robots. The task assignment of multirobot systems in this paper emphasizes not only the assignment of the desired number of robots to every target location, but also the reduction of the total robot traveling distance from their initial locations to their target locations, where the targets can be either static or movable.

III. Proposed Method

Inspired by the ubiquity of cortical maps in the central nervous system, the SOM algorithm was first introduced by Kohonen in the 1980s [23], [24], extended later [25]. It is based on

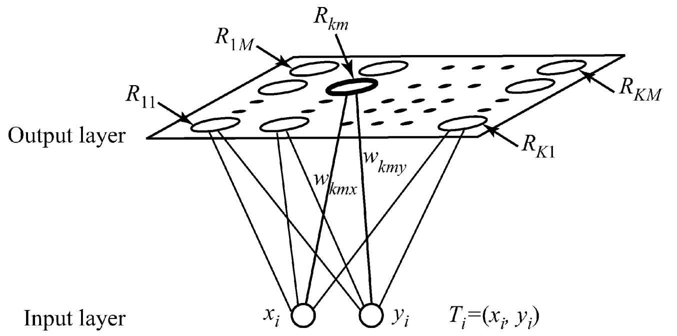

Fig. 2. SOM-based NN structure of the proposed approach, where input (x_i, y_i) denotes the ith target location, and output $(R_{11}, R_{12}, \ldots, R_{KM})$ denotes the positions of robots and the robot paths at a time instance.

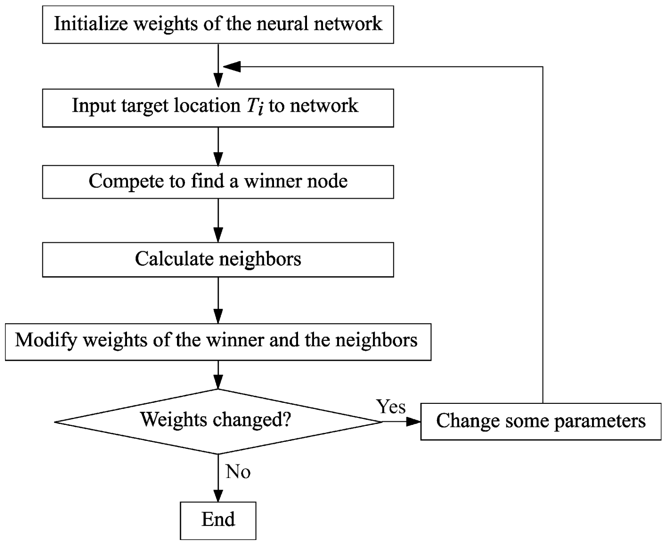
Initialize weights of the neural network
Input target location T_i to network
Compete to find a winner node
Calculate neighbors
Modify weights of the winner and the neighbors
Weights changed?
Yes
Change some parameters
No
End

Fig. 3. Flow diagram of the proposed algorithm.

the idea that there is a meaningful order of processing units in the mammalian brain, where each part is dedicated to a specific task and each group of neurons is sensitive to a particular type of input signals. The term "order" usually refers to spatial arrangement. The units are determined by parameters that can be changed in certain processes to produce meaningful organizations. Because of its universal applicability and easy handling, this algorithm soon became a valuable tool and was applied to many real-world problems [26]–[28]. The SOM based NN approaches combine a competitive learning principle with a topological structure of neurons, such that adjacent neurons tend to have similar weight vectors. They are capable of lacing the associated outputs of similar inputs close to each other; the more similarity in the input features, the closer the nodes in the output map.

The motivation of the proposed approach comes from the similar characteristics and phenomena between a multirobot system and a SOM algorithm. First, a multirobot system could be considered to be a self-organizing system that changes its basic structure as the system itself or the environment changes. Second, the SOM algorithm has the competitive, cooperative and self-organizing characteristics that are attractive for a multirobot system. Thus, the SOM based algorithm could be adapted to a multirobot system especially in a dynamic environment, controlling mobile robots automatically to achieve the task with cooperation and competition.

The proposed model focuses on the coordination of multirobots within a reasonable computation time, emphasizing the reduction of the total robot traveling cost and workload balance for every robot. Assume there are K robots and M targets randomly distributed in a workspace. The proposed SOM based NN model for a multirobot system is shown in Fig. 2. This model has two layers of neurons. The first layer is the input layer including two neurons (x_i, y_i), which represent the Cartesian coordinates of the ith target T_i in the 2-D workspace. All the coordinates of targets form the input data sets. The second layer is the output layer including KM neurons $(R_{11}, \ldots, R_{1M}, R_{21}, \ldots, R_{2M}, \ldots, R_{K1}, \ldots, R_{KM})$, which represent the coordinates of the K robots and the planed paths. Here, M neurons form a group for each robot. Every neuron of the output layer is fully connected to the neurons of the input layer. The connection strengths of the output neurons with the input neurons are given by a 2-D weight vector $R_{km} = \{w_{kmx}, w_{kmy}\}$, $k = 1, 2, \ldots, K$; $m = 1, 2, \ldots, M$, where the weight vectors of the M neurons for each robot are initialized as the coordinates of the initial robot position. The reason for introducing the K groups of output neurons is to record the dynamic trajectory of the K robots with workload balance for every robot. When the process of task assignment is completed, the M targets attract M neurons from KM output neurons to form K paths for the K robots. Every robot has its own path from its start position and passes through several targets. All of the M targets will be visited. The order of M neurons in K groups is the objective of the robot path planning.

The network is self-organizing, where the neurons tend to attain weight vectors that capture characteristics of the input vector space. At the beginning, the network is initialized with the weight vectors $\{w_{kmx}, w_{kmy}\}$, $k = 1, 2, \ldots, K$, $m = 1, 2, \ldots, M$, which are the initial locations of the K robots. The coordinates of targets as input data sets are given sequentially to the network in a random order during an iteration. In every iteration, all targets are put in a random order, then every target is input to the network one by one until the last target is input. This input strategy with the random order of the data sets results in the robustness of the algorithm and reduces its dependence on initial workspace configuration and the order of the input data sets.

For a given target as an input, the output neurons compete to be the winner according to a specified criterion described as

$$[N_k, N_m] \Leftarrow \min\{D_{ikm},\ i = 1, \ldots, M;\ k = 1, \ldots, K;\ m = 1, \ldots, M;\ \text{and } \{k, m\} \in \Omega\} \quad (1)$$

where $[N_k, N_m]$ denotes that the N_mth neuron from the N_kth group of the output neurons is the winner; and Ω is the set of neurons that have not been the winner yet in an iteration. The weighted distance D_{ikm} is given as

$$D_{ikm} = |T_i - R_{km}|(1 + P) \quad (2)$$

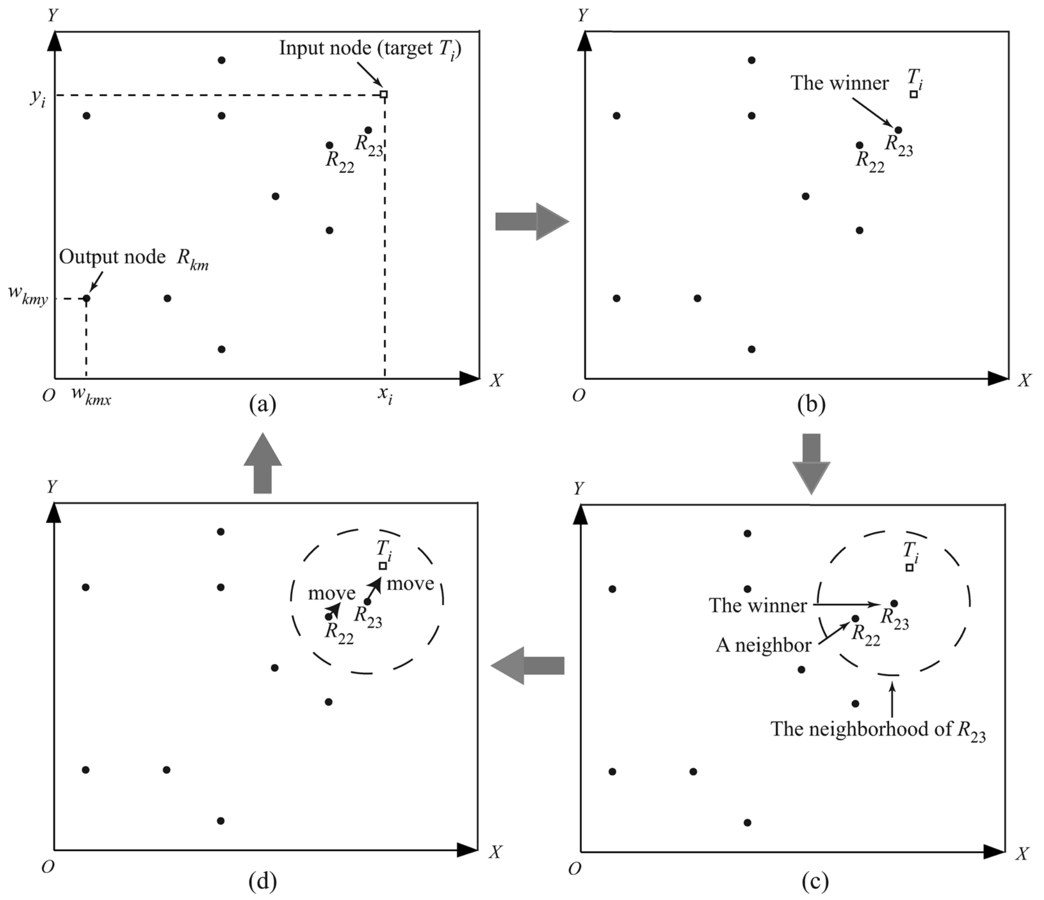

Fig. 4. Process of the proposed SOM-based approach. (a) Initial robot positions and a randomly selected target T_i (b) R_{23} is selected as the winner; (c) the neighborhood and the neighbors of R_{23}; and (d) the winner and its neighbors move toward to the target.

where

$$|T_i - R_{km}| = \sqrt{(x_i - w_{kmx})^2 + (y_i - w_{kmy})^2} \quad (3)$$

here $i = 1, 2, \ldots, M$; $|\cdot|$ denotes the Euclidean distance; and $R_{km} = (w_{kmx}, w_{kmy})$, $k = 1, 2, \ldots, K$, $m = 1, 2, \ldots, M$, is the coordinate of the mth neuron from the kth group of the output neurons. Parameter P in (2) controls the equitable distribution of workload for every robot, which is defined as

$$P = \frac{L_k - V}{1 + V} \quad (4)$$

where L_k is the path length of the kth robot, $k = 1, 2, \ldots, K$; and V is the average path length of the robots. The winner is considered to be not only the neuron with the minimum distance toward the input data, but also the neuron in the group of the output neurons with a lower workload. In order to restrict a neuron to be the winner more than once in each iteration, an inhibitory index in Ω is defined for each neuron. This strategy puts the winner neuron aside from the future competition, providing more opportunity for other neurons.

After the winner is selected, the next step is to design the neighborhood function, and decide the neighbors of the winner. The neighborhood function determines the influence (the attraction strength) of the input target location on the winner and the neighboring neurons. The attraction force of the winner should be the highest, diminishing as the proximity of the neuron to the winner decreases, and having no effect on the neurons outside the neighborhood. The neighborhood function $f(\cdot)$ is defined as

$$f(d_j, G) = \begin{cases} e^{-d_j^2/G^2(t)}, & \text{if } d_j < r \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $d_j = ||j - N_m||$, $j = 1, 2, \ldots, M$, which is the distance between the jth neuron and the winner N_m from the N_kth group of the output neurons, where the winner is; $||\cdot||$ denotes the absolute value; and r is a constant indicating the range of neighborhood. Function $G(t) = (1 - \alpha)^t G_0$ is a nonlinear function, where t is the number of iterations, and α is the change rate that decides the computation time. The smaller α is, the longer the computation time is; the smaller α is, the shorter the total path of the robots is.

After the winner neuron and its neighbors are selected, the next step is to move the winner neuron and its neighbors toward

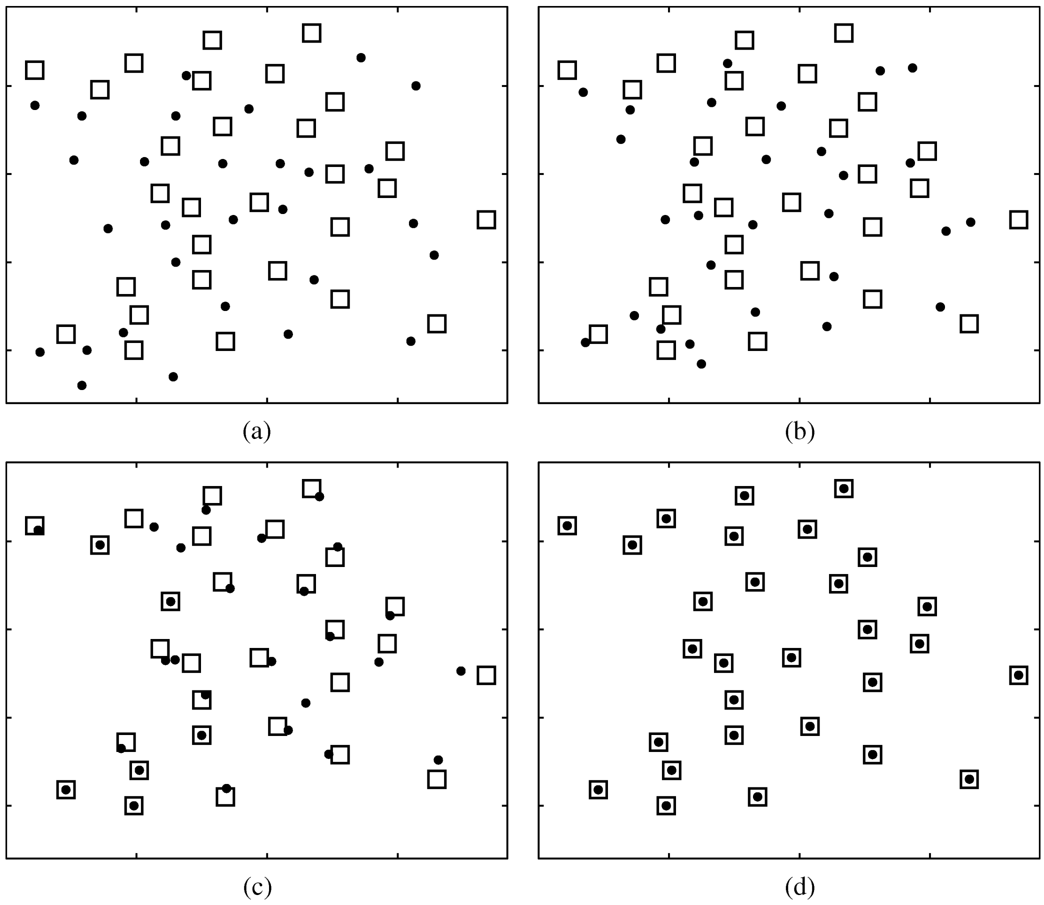

Fig. 5. Evolution of the solution at the (a) initial status, (b) after 5 iterations, (c) 20 iterations, and (d) the final status.

the position of the input target, while the other neurons stay no movement. The update rule is described as

$$R_{km}(t+1) = \begin{cases} T_i(t), & \text{if } D_{ikm} < \eta D_{\min} \\ R_{km}(t) + \beta f(d_j, G) \\ \quad \times \left(T_i(t) - R_{km}(t)\right), & \text{otherwise} \end{cases} \tag{6}$$

where β is the learning rate, η is a small constant, and $D_{\min}$ is the minimum distance of any two output neurons. The introduction of $D_{\min}$ can obviously reduce the computational time of the algorithm. It is evident that the modification of the weights depends not only on the original distance between the winner with its neighbors and the input target neuron, but also on the neighborhood function and the network learning rate.

The flow diagram of the proposed algorithm is summarized in Fig. 3. After the initialization of the NN, the positions of the targets are input to the network one by one. For a given target as an input in an iteration, three steps are involved. The first step is to find the winner; the second step is to decide the neighbors of the winner; and the third step is to modify the weights of the winner and its neighbors. These three steps are repeated until all of the weights do not change. Thus the robots move to targets step by step according to the changing weights. The task is completed when all the targets have been reached.

The detailed process of the proposed approach is shown in Fig. 4, where the dots represent the robot positions and the planned robot paths at a time instance, and the square represents the target location as input neuron. Fig. 4(a) shows the robot positions and a randomly selected target (e.g., T_i) as the input. Fig. 4(b) shows the winner according to the rule in (1) for winner selection, where the winner is R_{23} as it is the nearest one to the input T_i. Fig. 4(c) shows the neighbors of the winner according to the rule in (5) for neighbor selection, where R_{22} is the only neighbor in this example that it is located in the neighborhood of the winner R_{23}. Fig. 4(d) shows the movement of the winner and its neighbors according to the rule in (6). Both neurons R_{23} and R_{22} move a little toward Target T_i by changing the weight vectors $[w_{23x}, w_{23y}]$ and $[w_{22x}, w_{22y}]$, while the others do not move. The winner R_{23} moves a greater distance than its neighbor R_{22}. The closer the neuron is to the winner R_{23}, the greater distance the neuron will move. The loop then returns to Fig. 4(a) again with another randomly selected target as input, repeating from (a) to (d) until all of the targets are achieved by robots.

IV. Simulation Studies

In order to demonstrate the effectiveness of the proposed algorithm for task assignment of multirobot systems, five different cases are studied in this section, including arbitrary number of robots and targets, and static and dynamic environments. For every case, the 2-D workspace is normalized to 1 × 1. The algorithms are coded in MATLAB and implemented on a 2.5 GHz CPU computer with 376-MB RAM and Windows XP.

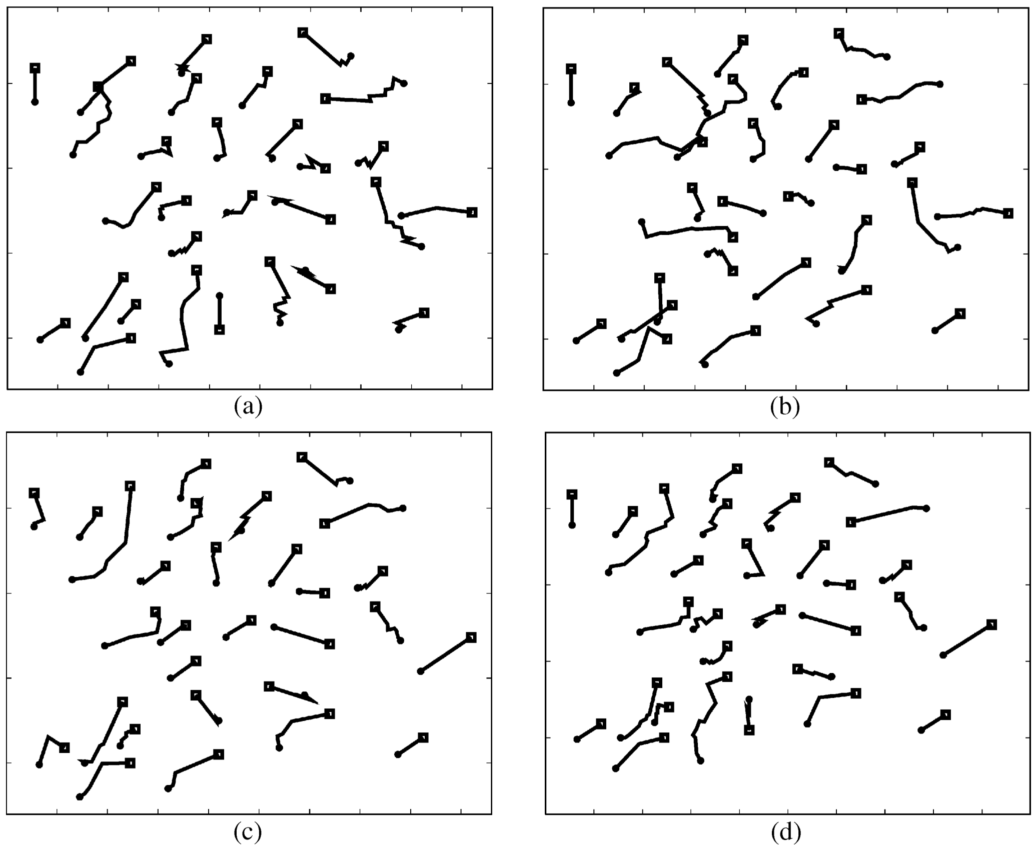

Fig. 6. Tracks of 29 robots to 29 targets with four different random sequences of target inputs.

A. In a Static Environment With the Same Number of Robots and Targets

The proposed algorithm is first applied to a simple case, where the number of robots and the number of targets are the same. The time-dependent process of the self-organizing behavior of a multirobot system is illustrated in Fig. 5 at four time instances. The initial state is shown in Fig. 5(a), where the targets are represented by squares and the robots by dots. There are 29 randomly distributed targets and 29 randomly distributed mobile robots in the workspace. The evolutions after 5 [Fig. 5(b)] and 20 [Fig. 5(c)] iterations show that the robots move to different targets gradually and individually. At the end, a stable state [Fig. 5(d)] emerges while every target is occupied by a robot.

Because the target locations as input data sets are input sequentially into the network in a random order during an iteration, it is obvious that the tracks of the robots are not the same for different runs, but all the results are reasonable. With exactly the same initial target and robot locations in the workspace, the results with four different random input orders of the target data sets are shown in Fig. 6, where squares represent targets, dots represent robots, and the solid lines represent the real-time tracks of robots to targets. These figures show the dynamic process of the self-organizing behavior in the multirobot system. Unlike the conventional approaches (e.g., [1]–[6]) to multirobot path planning for multiple tasks, where the task assignment and path planning are handled separately, the proposed approach integrates the task requirement of robots and robot motion planning, so that the robots can start to move before their destinations are finalized.

B. The Number of Robots and Targets are Randomly Given

The proposed algorithm is then applied to the case where the number of robots and targets are randomly given. The initial locations of robots and targets are also given randomly. To simplify the test, we assume that the number of robots or targets are selected from the range [1,100]. We have conducted the experiment 200 times. Every time, the algorithm can find a solution in 0.2 s to 10 min in about 160 iterations. For example, the results from 4 trails are shown in Fig. 7, where dots denote robots and squares denote targets with: (a) having 30 targets and 3 robots; (b) 3 targets and 30 robots; (c) 50 targets and 60 robots; and (d) 100 targets and 10 robots.

It is clear that the proposed approach is capable of dealing with the case where the number of robots is less than the number of targets. For example, if there are ten robots and 12 targets in a workspace, at first, the 12 targets would attract robots by the competition feature of the proposed method. Some of robots achieve the targets quickly, while others would be slow. When any robot reaches a target, it becomes free and can pursue another target. Fig. 8 shows the tracks of robots in this situation, where Robot R_1 continues to go for Target T_2 after it reached Target T_1; Robot R_2 continues to go for Target T_4 after it reached Target T_3; and Robot R_3 continues to go for Target T_2 after it reached Target T_5, but R_3 stops when T_2 is reached by Robot R_1. The competition and self-organizing features are

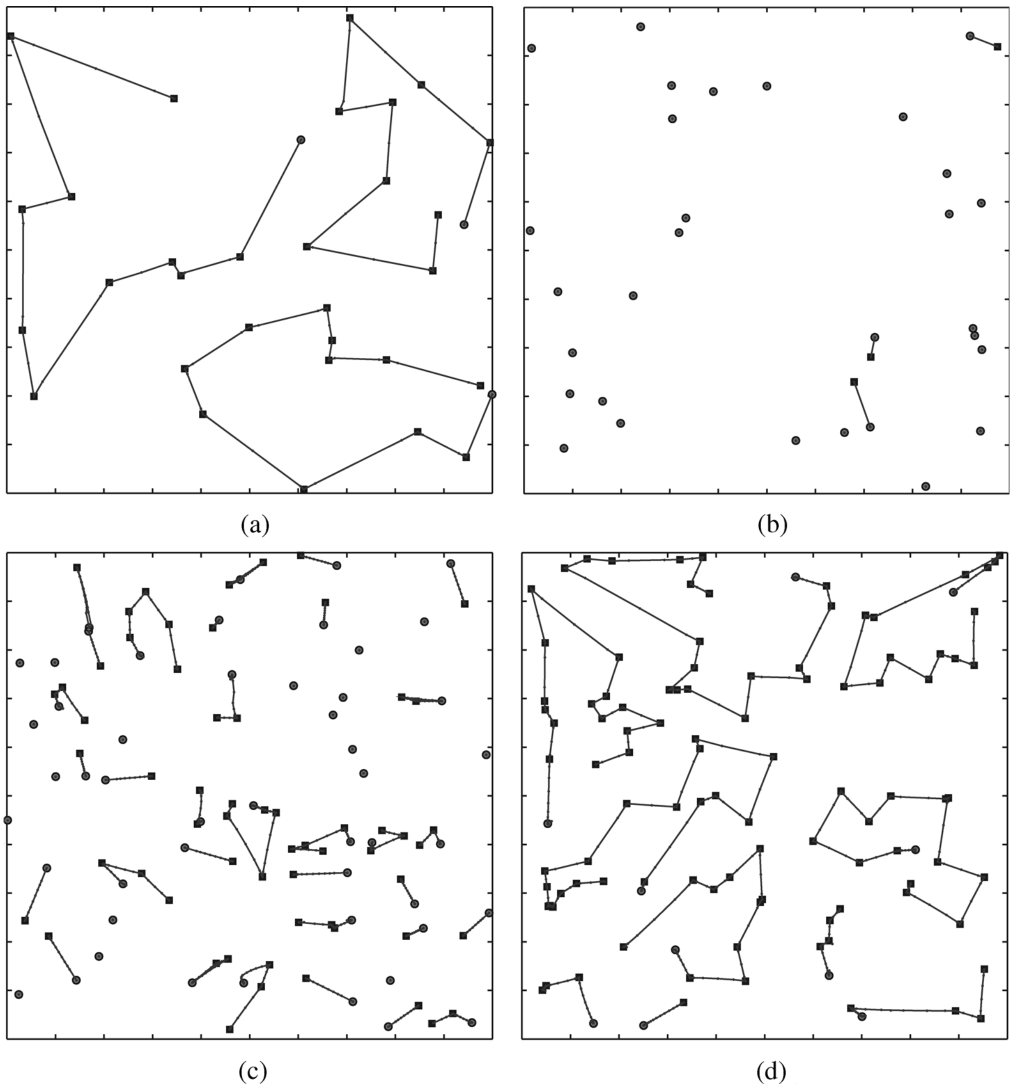

Fig. 7. Solutions to task assignment of multirobot systems with various numbers of robots and targets. (a) 30 targets and 3 robots; (b) 3 targets and 30 robots; (c) 50 targets and 60 robots; and (d) 100 targets and 10 robots.

interesting, allowing the multirobot system to complete the task assignment autonomously.

C. Some Targets Need More Than One Robot

The proposed algorithm can also deal with complicated cases that would not be handled by conventional approaches (e.g., [7]), where several robots have to be assigned to one target location and there are more robots than needed for the task. For example, in the case of Fig. 9, Target T_1 needs two robots, T_2 needs one robot, T_3 needs two robots, and T_4 needs three robots. At the beginning, Robots R_1 and R_9 move to Target T_1, while R_2 moves to T_2. After several steps, Robots R_1 and R_2 move to Target T_1 and Robot R_9 stops without moving further because of the self-organizing feature of the proposed approach. Robot R_{10} does not move at all, because it is relatively far away from any targets. It is clear that the proposed approach for multirobot systems can complete the task assignment autonomously with its self-organizing feature.

D. Some Robots Break Down

The proposed approach is robust. It can work satisfactorily in the case of unexpected events, which cannot be handled by conventional approaches (e.g., [1]–[6]). This error resistivity of the self-organizing process allows for a sudden change of the environment such as a breakdown of some robots during the movement. Fig. 10 shows this kind of situation. At the beginning period, Robot R_1 moves to T_1, Robot R_2 moves away from T_1 and to T_2; Robot R_4 moves to T_3, Robot R_5 moves away from T_3 and to T_4, and so on. Assume that Robots R_1 and R_4 break down after a certain time as shown by the multiplication signs in Fig. 10. In this situation, R_2 turns back and moves to T_1, and R_5 turns back and moves to T_3. At the end, every target location is occupied by a robot due to the self-organizing behavior of the proposed approach.

E. In a Dynamic Environment

The proposed SOM based approach can work properly in a dynamic environment. For example, in the case as shown in Fig. 11, Targets $T_1, \ldots, T_4$ move randomly in the workspace.

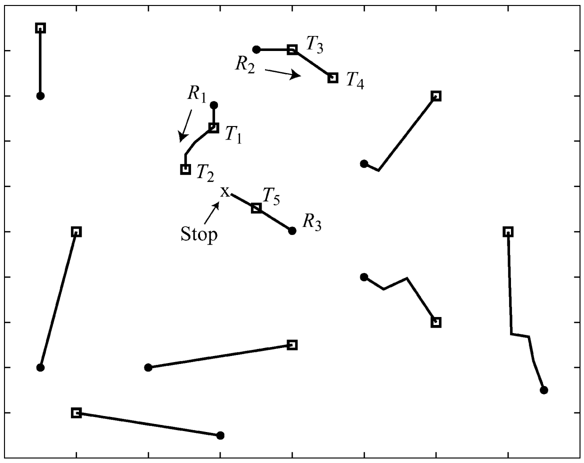

Fig. 8. Tracks of the robots to the targets, where the number of robots is less than the number of targets.

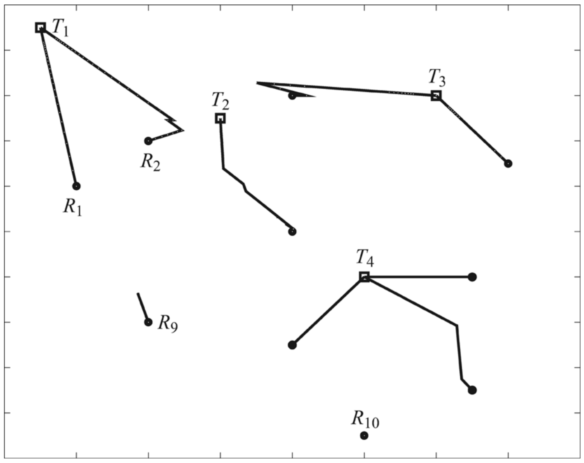

Fig. 9. Tracks of the robots to the targets, where Targets T_1, T_3, and T_4 need more than one robot.

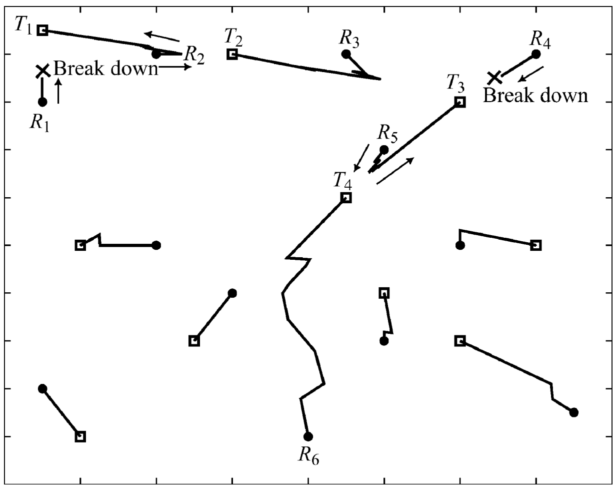

Fig. 10. Tracks of the robots to the targets, where Robots R_1 and R_4 are broken down on the way toward targets.

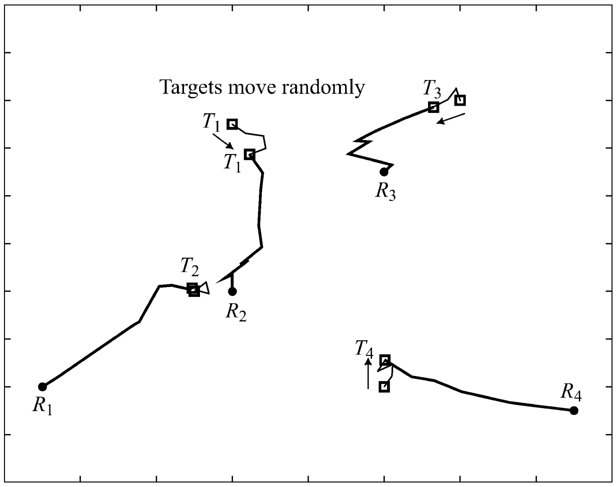

Fig. 11. Tracks of the robots to moving targets.

Robots $R_1, \ldots, R_4$ can still arrive at the target locations individually. Although the paths of the robots are not perfect, the proposed method can complete the task with reasonable paths and computational time, especially where dealing with a dynamic environment. Conventional approaches (e.g., [1]–[7]) are not able to deal with such environments.

V. Discussion

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. The proposed model is not very sensitive to model parameter variations, in the sense that the generated robot paths will not change sharply when one parameter varies slightly. To analyze the sensitivity of parameters, the proposed approach is tested with various parameter values. The results show that the learning rate β is not very sensitive when it is in the range [0.05, 0.1]. Parameter r, the range of the neighborhood, is not very sensitive when in the range [0.1, 0.4]. Parameter η is not very sensitive when in the range [0.01, 0.1]. Our experiments also show that the iteration number needed mainly depends on the change rate α. When α is initialized as 0.03, the number of iterations needed is only 160. In addition, the convergence speed of the proposed algorithm is much faster than other algorithms (for details, see our previous work in [27]).

The proposed approach is capable of controlling a group of mobile robots to achieve multiple tasks at several different locations. It combines the task requirement of robots and robot motion planning together; these are normally handled separately in conventional approaches (e.g., [1]–[6]). Thus, the robots can start to move once the total tasks are set. Robot navigation can be dynamically adjusted to guarantee each target location will have the desired number of robots.

Because of the self organizing feature, the proposed model for task assignment of multirobot systems treats the multirobots, the environment, and the targets as a self organizing system that can be changed while the robots are moving. It can be explained clearly from the structure of the model in Fig. 2. The robots are treated as the neurons in the output layer. The number of neurons can be changed during the procedure without influencing the whole system. Thus the proposed approach can handle uncertainties, such as when some robots break down or some robots are added. In addition, the targets in the environment are treated as the neurons in the input layer. The values of the input neurons can be changed during the process without influencing the whole system. As a result, the proposed approach is capable of dealing with a changing environment, e.g., the targets are movable or newly targets are added. This situation could not be handled by conventional approaches (e.g., [1]–[7]).

Furthermore, the proposed method can be extended to other problems, such as the traveling salesman problem (TSP), which is a typical task assignment problem with only one robot and more targets in a static workspace. One extended algorithm for TSP can be found in our previous work [27], which is much faster than conventional approaches.

VI. Conclusion

In this paper, a SOM based neural network approach is proposed for task assignment of multirobot systems in arbitrarily nonstationary environments. It involves special definition of the initial neural weights of the network, the rule to select the winner, the rule to update the weights, and the rule of the neighborhood function. With a self-organizing process, the proposed approach has several interesting features and advantages. This model combines the target assignment and motion planning for a multirobot system, allowing the robots to start moving before their destinations are finalized. The proposed algorithm can deal with sudden changes in a situation such as the breakdown of some mobile robots. In addition, it can deal with complicated cases such as assigning more than one robot to a target location, or robots are fewer than the targets. Furthermore, it is capable of dealing with changing environments such as targets being movable.

Acknowledgment

The authors would like to thank the Associate Editor and the anonymous reviewers for their valuable comments.

References

[1] K. S. Kwok, B. J. Driessen, C. A. Phillips, and C. A. Tovey, "Analyzing the multiple-target-multiple-agent scenario using optimal assignment algorithms," *J. Intell. Robot. Syst.*, vol. 35, pp. 111–122, 2002.

[2] J. Orlin, "A polynomial-time primal network simplex algorithm for minimum cost flows," in *Proc. 7th Annu. ACM/SIAM Symp. Discrete Algorithms*, 1996, pp. 474–481.

[3] J. Wein and S. A. Zenios, "Massively parallel auction algorithms for the assignment problem," in *Proc. 3rd Symp. Frontiers of Massively Parallel Computation*, Nov. 1990, pp. 90–99.

[4] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Comput. Oper. Res.*, vol. 24, no. 1, pp. 17–23, 1997.

[5] R. Akkiraju, P. Keskinocak, S. Murthy, and F. Wu, "An agent-based approach for scheduling multiple machines," *Appl. Intell.*, vol. 14, pp. 135–144, 2001.

[6] A. J. Higgins, "A dynamic tabu search for large-scale generalised assignment problems," *Comput. Oper. Res.*, vol. 28, pp. 1039–1048, 2001.

[7] J. Starke, M. Schanz, and H. Haken, "Self-organized behavior of distributed autonomous mobile robotic systems by pattern formation principles," in *Distributed Autonomous Robotic Systems 3*, T. Lueth, Ed. Berlin, Germany: Springer-Verlag, 1998, pp. 89–100.

[8] W.-M. Shen, C.-M. Chuong, and P. Will, "Simulating self-organization for multirobot systems," in *Proc. IEEE Int. Conf. Intelligent and Robotic Systems*, Lausanne, Switzerland, Oct. 2002, vol. 3, pp. 2776–2781.

[9] W.-M. Shen, P. Will, and A. Galstyan, "Hormone-inspired self-organization and distributed control of robotic swarms," *Auton. Robot.*, vol. 17, pp. 93–105, 2004.

[10] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Automat.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.

[11] R. W. Beard, T. W. McLain, and M. Goodrich, "Coordinated target assignment and intercept for unmanned air vehicles," in *Proc. IEEE Int. Conf. Robotics and Automation*, Washington, DC, May 2002, vol. 3, pp. 2581–2586.

[12] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter, "Cooperative control of UAV rendezvous," in *Proc. Amer. Control Conf.*, Arlington, VA, Jun. 2001, pp. 2309–2314.

[13] F. Brandt, W. Brauer, and G. Weiß, "Task assignment in multiagent systems based on vickrey-type auctioning and levelled commitment contracting," in *Proc. Cooperative Information Agents Workshop*, Boston, MA, 2000, pp. 95–106.

[14] T. Chadzelek, J. Eckstein, and E. Schör, "Multirobot path planning by predicting structure in a dynamic environment," in *Proc. 8th Canadian Conf. Computational Geometry*, Ottawa, Canada, Aug. 1996, pp. 131–136.

[15] Y. Endo, D. C. Mackenzie, and R. C. Arkin, "Usability evaluation of high-level user assistance for robot mission specification," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 34, no. 2, pp. 168–180, May 2004.

[16] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with realtime task assignment," *IEEE Trans. Robot. Automat.*, vol. 18, no. 5, pp. 769–780, Oct. 2002.

[17] A. Stoytchev and R. C. Arkin, "Dynamic task assignment in a multiagent/multitask environment based on module conflict resolution," in *Proc. IEEE Int. Conf. Robotics and Automation*, Seoul, Korea, May 2001, vol. 4, pp. 3987–3992.

[18] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991, ch. 8, pp. 356–402.

[19] K. M. Passino, *Biomimicry for Optimization, Control, and Automation*. London, U.K.: Springer-Verlag, 2004.

[20] S. X. Yang and Q. H. 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.

[21] H. D. Patino, R. Carelli, and B. R. Kuchen, "Neural networks for advanced control of robot manipulators," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 343–354, 2002.

[22] R. Reeve and J. Hallam, "An analysis of neural models for walking control," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 733–742, 2005.

[23] T. Kohonen, "Analysis of a simple self-organizing process," *Biol. Cybern.*, vol. 44, pp. 145–140, 1982.

[24] ——, "Self-organizing formation of topologically correct feature maps," *Biolog. Cybern.*, vol. 43, pp. 59–69, 1982.

[25] T. Kohonen, S. Kaski, K. Lagus, J. Salojaervi, J. Honkela, V. Paatero, and A. Saarela, "Self-organization of a massive document collection," *IEEE Trans. Neural Netw.*, vol. 11, pp. 574–585, 2000.

[26] H. Ritter, T. Martinetz, and K. Schulten, *Neural Computation and Self-Organizing Maps: An Introduction*. New York: Addision-Wesley, 1992.

[27] A. Zhu, S. X. Yang, and F. Wang, "A valuable algorithm on traveling salesman problem," in *Proc. IEEE Int. Conf. Robotics, Intelligent Systems and Signal Processing*, Changsha, China, Oct. 2003, pp. 674–679.

[28] X. Yu and S. X. Yang, "A study of motion recognition from video sequence," *Comput. Visual. Sci.*, vol. 8, no. 1, pp. 19–25, 2005.

Anmin Zhu received the B.Sc. and M.Sc. degrees from Tongji University, Shanghai, China, in 1987 and 1990, respectively, and the Ph.D. degree from the University of Guelph, Guelph, Canada, in 2005.

Currently he is a faculty member in the College of Electrical and Information Engineering, Shenzhen University, Shenzhen, China. His research interests include fuzzy systems, neural networks, robotics, self-organizing arrangement, machine intelligence, and multiagent systems.

Simon X. Yang (M'05) received the B.Sc. degree in engineering physics from Peking University, Beijing, China, in 1987, the first of two M.Sc. degrees in biophysics from Chinese Academy of Sciences, Beijing, China, in 1990, the second 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 University of Guelph, Guelph, ON, Canada, as an Assistant Professor in Systems and Computer Engineering in August 1999. Currently, he is an Associate Professor and the Director of Advanced Robotics and Intelligent Systems (ARIS) Laboratory at the University of Guelph, Guelph, Canada. His research interests are in the areas of robotics, intelligent systems, sensors, control systems, image and signal processing, neurocomputation, and bioinformatics. He has published over 200 refereed journal papers, book chapters, and conference papers.

Dr. Yang serves on the Editorial Boards of the journal *Dynamics of Continuous, Discrete and Impulse Systems*, the *International Journal of Information Acquisition*, and the *International Journal of Computational Intelligence and Applications*. He has involved in the organization of many international conferences. He is the General Chair of the 2006 International Conference on Sensing, Computing and Automation. He was a recipient of the 2004–2006 Presidential Distinguished Professor Awards at the University of Guelph.