Avoidance of Multiple Dynamic Obstacles

Areolino de Almeida Neto
Universidade Federal do Maranhão
areolinoneto@yahoo.com.br

Bodo Heimann
Mechatronik-Zentrum Hannover
heimann@mzh.uni-hannover.de

Cairo L. Nascimento Jr.
Instituto Tecnológico de Aeronáutica
cairo@ele.ita.br

Luiz Carlos S. Góes
Instituto Tecnológico de Aeronáutica
goes@mec.ita.br

Abstract: this article is a continuation of the previous article called "Obstacle Avoidance in Dynamic Environment: a Hierarchical Solution", which presented the concept for obstacle avoidance in dynamic environment suitable for mobile robot. The task of obstacle avoidance is divided in three principal groups: local, global and for emergencies. The global avoidance is here approached, in which the concept used is based on reinforcement learning and on an algorithm for planning the avoidance, in such a way that the situations are divided into four states and two kinds of actions are possible. The states define in what situation the movement relationship between the robot and the dynamic obstacles is present, and the actions decide in which direction the robot must follow, in order to avoid a possible collision. The algorithm plans in order to decide an action, which certainly is not the best for each obstacle, but normally is the best considering the group.

Keywords: obstacle avoidance, dynamic environment, reinforcement learning, Monte Carlo method

1. Introduction

In recent years many researchers have their attention pointed to the solution of obstacle avoidance, principally obstacles which change their position. For static obstacles, many strategies were presented with excellent results. But for dynamic obstacle, there is too much to do, although many proposals have interesting results.

Different strategies were and are presented in order to solve this problem. It is possible to divide these strategies in two classes: model-based and learning-based. The model-based concept utilizes mathematical models to describe the robot and obstacles movement and to describe the possibility of a collision, and therefore how to avoid them. On the other hand, learning-based methods use the knowledge obtained in/from real situations and "learn" the way to avoid obstacles.

Regarding the model-based methods, interesting work is presented by Freund et all. (2001), which deals with model and conditions for safety path planning. Shiller et all. (2001) present another excellent work, in which mathematical model of free places is developed.

An interesting learning-based method in obstacle avoidance is presented by Tang et all. (2001). In this work rules for the possibility of collision are detailed and implemented in a fuzzy system. Another interesting work is presented by Tsourveloudis et all. (2001). In this work a fuzzy system is combined with the electrostatic potential field method for path planning. In the work of Kawano and Ura (2001), it is presented a path planning done by multi reinforcement learning (RL) modules, which can also accept external command. Yen and Hickey (2002) propose improvements in the performance of traditional RL methods used for robotic navigation. Gachet et all. (1994) propose a special Neural Network representation of a RL system.

Some complex tasks need more than one method in order to solve it. The autonomous navigation problem seems to be one of that. Therefore, for the case where several obstacles are present, the RL technique and an algorithm for planning the best escape are used. The RL technique provides a group of actions, which can avoid one obstacle with different performance. And the algorithm chooses one action, which presents the best escape considering all obstacles.

This article has the following organization: in section 2 the desired characteristic of learning methods are presented, followed by a discussion about the RL technique, as well as the states and actions used here are detailed. Then, the architecture of the obstacle avoidance system is shown. The results obtained in simulations are presented and the conclusions and future works are depicted.

2. Desired Characteristic of Learning Methods

Diverse techniques can be used in order to solve this kind of problem. Neural Network, Fuzzy Logic and

Reinforcement Learning are some of artificial intelligence techniques, which can be used. Each of these has particular characteristic, such that one can be more suitable for a type of problem. For obstacle avoidance, it is difficult to say which strategy is the best. However, it is possible to say the characteristics, which are desirable in a technique, such that it is more eligible for the obstacle avoidance problem:

- Intuitive data;
- Cumulative learning;
- Constructive solution;
- Direct knowledge acquisition.

Intuitive data means that the data used by a strategy should be "visible", like distance between the robot and one obstacle, time for occurrence of impact and turns in robot movement. Quantities strongly modified by complex mathematical equations are good for precise calculations, but they become a value more artificial, less intuitive and, therefore, tend the interpretation of them to be difficult.

The problem of dynamic obstacle avoidance consists of several situations, which are different among them, but are interconnected too. Then learning acquired for a given situation must be kept, while another knowledge is learned for another situation. That means cumulative learning. More, if learning already acquired should be modified, the modification must not interfere in other knowledge, which doesn't have relation with it.

Another characteristic of dynamic obstacle avoidance is that a unique action normally is not sufficient for a collision free movement. Many times it is necessary to have a series of actions. This complex task can be performed by a set of primitive actions (Gachet et all. 1994). Therefore a solution can be well founded when hierarchical decisions are taken in a well-defined sequence. For example, for a certain obstacle, turn right and after go ahead makes a robot to avoid it, but the same cannot be true if it first goes ahead and then turns right.

An important characteristic of a learning strategy is the possibility of acquiring a knowledge at the first time it is presented to this strategy, or in some cases after few times. It is not good, when knowledge must be presented many times repeatedly, until the learning strategy can acquire this knowledge. If an obstacle is coming in front of the robot, turn right or left should be learned at the first time, or in the worst case after three or four times.

Based on these four characteristics, it seems to be a good option, the reinforcement learning (RL) technique. Moreover, there are other conditions for this choice: no previous knowledge of the correct solution must be known by the designer, as Fuzzy Logic needs; and the environment also must not present the correct solution to the learner (Yen and Hickey 2002). RL techniques can learn a solution without this information. They can learn by trial-and-error method, but with directions to the correct way of the solution.

3. The RL techniques, States and Actions

Diverse RL techniques exist nowadays. They can be classified as Direct Reinforcement (DR), value function methods and Actor-Critic (Moody and Saffell 2001). Each one has advantages and characteristic suitable for a particular class of problem. An excellent reference on this topic is presented by Sutton and Barto (1998).

Considering the characteristics presented in section 2, principally the last three, the value function methods seem to be a good option for obstacle avoidance. More precisely, the Monte Carlo technique offers good reasons to be selected: it has cumulative learning, whereas it is possible, a solution for a given situation can be learnt today, and tomorrow another knowledge can be learnt for another situation, with no interference from previous knowledge acquired, except if an interference with relationship aspect is desired. The constructive solution can be performed defining primitive actions on robot's movement, such that a set of these basic actions can perform all complex movements. And the most important characteristic is achieved because Monte Carlo updates the Q-value direct into Q-value matrix. But for excellent results, a proper function of the performance in obstacle avoidance must be defined.

The evaluation function used in this work during training, in order to classify the decisions taken in an obstacle avoidance situation, is the time spent by the robot from the beginning of the robot-obstacle transaction until the end of this transaction. Here, the most important thing, when an obstacle avoidance path is taken, is not the absolute time value, but the order of these values, that means, the qualitative evaluation is more important than the quantitative value. And this classification will be used wherever another similar situation is presented to the robot and, therefore, is necessary to decide which action to take. In fact, the evaluation function *f* used is shown in equation below.

$$f = \frac{-100}{t + a/10 + n/600} \tag{1}$$

where t is the time spent during the avoidance, a is the number of actions taken and n is the number of trials performed. The terms a and n are used, because it is possible, two actions for the same robot-obstacle situation receive equal evaluation, because they spent the same time, thus, in order to have different values for all actions ever, these components are added to the time spent. The reason of the inversion and the constant –100 is due to the initialization of the matrix used for saving the states-actions values. This initialization was made with zero for all elements of this matrix, except to those, which correspond to the action "go ahead". This way permits the decision with the best evaluation to have a minimal value and to be inferior to zero.

Considering the constructive solution of complex tasks based on primitive behaviors, an obstacle avoidance solution is obtained with a series of actions in directions and velocities. Thus, it is necessary first to know if a sequence of actions was able to avoid a collision. If so, then the function f above is calculated; if not, then it is not possible to say about the actions taken and no action is updated. And more, for consistency reasons, the evaluation values are saved into matrix only for the last action taken in one state. That means, in a sequence of decisions, it is possible for a given state S1, the action A1 to be taken and after, but yet in the same problem, the robot can be in the same state S1. In this case another action A2 is taken, then only this action is evaluated, because it conducted the robot to safety place. This procedure is made for all states visited by the robot for specific obstacle avoidance.

Moreover, an action chosen for updating for a given state, according to the explanation above, is updated only if the present evaluation is better than the previous one already saved in state-action matrix.

The most problem (and even so not easy to solve it) in designing with RL is to determine the representation of the situations (states) and the attitudes to be made by the agent (actions), in order to solve the problem.

For robot navigation, the states and actions chosen were based on probable human representation and decision. The actions are simple: bounded velocities of the robot in two directions, in such a way that the robot can move in 2-dimensional space, or stop.

The states, which represent a situation in obstacle avoidance, are more complex. However, when we humans do obstacles avoidance, we look into the environment, detect them and extract important information based on their position and velocity with respect to us. The objective is to avoid a collision (two bodies in the same point and at the same moment), then the following information can be inferred from the environmental data: 1) if there is a possibility of collision; 2) where is the possible collision; 3) when is the possible collision. Based on these inferences, a proper decision is taken.

For the decision using the RL, the four states below are used in this work:

- di: distance to the possible point of impact;
- beta: direction of an obstacle;
- y_dro: shortest distance between obstacle and the robot's path;
- ti: condition of arriving: if the robot will arrive before, at the same time or later related to the obstacle at the impact point.

More details about these states can be found in the work wrote by Almeida Neto et all. (2002).

However, one problem with RL representation is the explosion of the number of states and actions, that means, too much number of states and actions are necessaries for a good representation. In this work, in order to decrease the amount of memory used to save the state-action matrix, a neural representation is used. More about Neural Networks can be found in Zurada (1992).

Moreover, it is used also a scheme of neural networks (NN) in parallel that can make a cooperative training. The following figure shows this scheme with two NNs., but if necessary, more NNs can be added to it. In this scheme Y is the output of the set of NNs, which is equal to sum of each NN. D is the desired output for the scheme and E is the output error of the set of NNs.

In this scheme, the training is a little bit special: in order to avoid the NNs make a dirty competition, in which one tries to destroy the other and so they can't construct a solution together, only one NN is trained, while the other NNs produce their results with their weights unaltered. This procedure has two great advantages: avoids dirty competition among the NNs and, therefore, no necessity of an element for coordination. In many schemes, in which more than one NN is used, a coordinator is necessary for the organization. The coordinator can be another NN or other element that can apply a linear or non-linear combination among the NNs' output. This scheme was already utilized in control of a flexible link (Almeida Neto et all. 2001). An important reference on use of more than one NN can be found in Sharkley (2000).

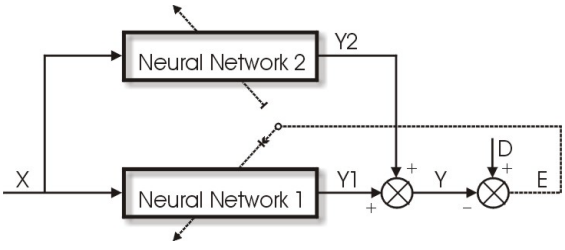

Figure 1: A scheme of two NNs in parallel.

The scheme begins with only one NN and this is trained alone with a partial set of the training data. When the training can't advance, another NN is added to the scheme, whose outputs are zero for all inputs. This second NN is trained, while the first produces its output. If the training of the second NN can't advance too, it is possible to continue the training of the first NN, while the weights of the second NN become invariant, or another NN is added and the process repeats. This procedure causes another advantage of this scheme: when a NN is trained, it utilizes the knowledge acquired by the others NNs, that means, the training of each NN doesn't reject the knowledge already

learned by the others NNs because it is used in the present training. The equations below show this. When the first NN was training, the output error was E = D – Y1 and the teaching signal for this NN was Y1 = D.

With two NNs, the output error is E = D – Y1 – Y2 and the teaching signal for this NN is

$$Y2 = D - Y1 \tag{2}$$

If the first NN is trained again, the second NN is not trained and the teaching signal for the first NN is

$$Y1 = D - Y2 \tag{3}$$

Or if another NN is added, and the teaching signal for the third NN is

$$Y3 = D - Y1 - Y2 \tag{4}$$

As it can be seen, the desired output of each NN is different from each one. Even when a NN is trained again, its teaching signal is different from that used in previous training.

4. Obstacle Avoidance Architecture

There are three different obstacle avoidance approaches: local (decision), global (planning) and for emergencies (failure). The local avoidance tries to avoid the most imminent collision. The global avoidance chooses an action that can be used for the local avoidance (not necessarily the best one), but that also avoids the others obstacles or has a greater chance to avoid them. However, if the action chosen, or the sequence of actions, drives to a collision, because the action was not properly chosen or the environment has another context, another type of actions is made considering the emergency situation. The action chosen has many times no relationship with the destination point. The goal in this moment is to avoid an immediate collision.

The avoidance above must combine with the path defined for the static environment. That means, there is a path defined *a priori*, which dictates the robot's movement in a well-known and static environment. And, in complement of this path, avoidance is made based on the pre-defined path and the dynamic obstacles condition. The following explanation clarifies how both systems can work together.

First, a path for the static environment is made. Normally, the algorithm for this case uses the initial (present) and final positions and velocities of the robot, as well as a map with description of free and occupied positions.

Then, the path produced feeds the dynamic avoidance system. For each time step, this system uses the present robot's and environment's states and the trajectory to be done by the robot at this time. Based on this information, it classifies the situation in one of the three possibilities: no collision, possible collision or collision. In this work, this classification is made by the following rule: if the robot's semi-line and the obstacles' semi-line have no intersection point, then the situation is classified as no collision. If there is at least one intersection point and the distance between the robot and the obstacle is inferior to 4 m, then there is a possible collision; and if the distance is less than 0.8 m, then the classification is collision. The figure below shows an example of possible collision. Here the semi-lines are drawn as dashed lines and the arrows indicate the direction of the objects.

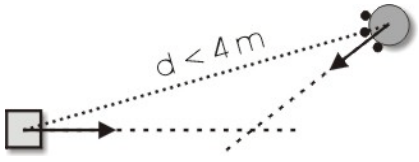

Figure 2: A classification of "possible collision".

For the first classification, no problem is detected and the robot can follow the movement defined by the pre-defined path. If the situation is classified as collision, then an escape path must be done. This escape tries to avoid an immediate collision, even if it follows an opposite direction relative to the desired destination.

However, if the classification is as possible collision, then avoidance must be done. The avoidance must consider the destination of the robot's path, the movement of the dynamic obstacles and the position of the static obstacles. Based on this information, a sequence of actions is taken, which can drive the robot to the destination through a safe path. After the collision is surpassed, generally another pre-defined path to the destination is necessary.

The sequence of actions is defined by the RL technique. During the training, the robot is put in many situations with only one punctual dynamic obstacle. After several trials, a classification of the actions for each set of states is available.

For the avoidance in a real situation, first three tracking points should be defined: left, central and right. These points define a non-punctual obstacle. In Fig. (2), these points are the three black circles around the obstacle. The point to be avoided is the central point, while the others define the boundaries of the obstacle. The central point moves along the obstacle's surface, according to the avoidance taken by the robot. The central point movement continues until no

more impact condition is found or until it is equal to one of the other two points, when the tracking points are redefined.

However, the RL technique alone is used for the avoidance of one obstacle only. To find the avoidance when several obstacles are in the situation, it is necessary to use the RL technique with a heuristic defined below:

1. One obstacle is defined as the principal obstacle and based on this obstacle, actions are indicated by the RL technique;
2. The situation is divided in sectors, as shown in figure below

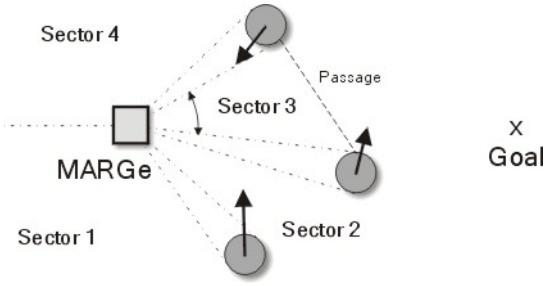

Figure 3: The situation in sectors.

3. If the last action decided has a chance to avoid the obstacles, that means, if it can drive the robot by a free and sufficient large sector, than it is maintained;
4. If not, then the RL technique indicates 10 actions for the present situation. For a new situation or in the beginning of one, the actions are the 10 best actions, otherwise they are the 10 best actions belong to the same quadrant of the last action;
5. For the 10 actions indicated, if at least one can drive the robot by a safe trajectory, then the action with the fewest changing in lateral velocity is chosen;
6. If none, so the 10 actions indicated are reflected to the other side (left or right side) and again a safe trajectory is searched using the same criteria pointed in item 4. The figure below shows a case

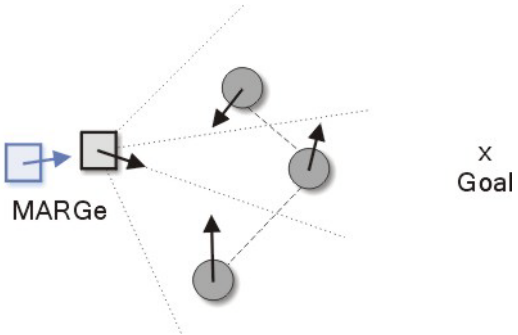

Figure 4: The choice of a new action.

7. If no actions could be chosen, an action, considering the 10 best actions for all quadrants, that presents the possibility of no collision is immediately chosen, as shown below

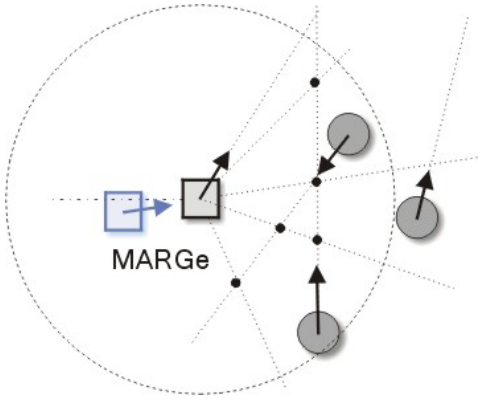

Figure 5: The choice of an action that presents no possibility of collision.

8. If no action could still be found, so an action, considering the 10 best actions for all quadrants, that presents the possibility of arrival at the collision point before or after the obstacle is immediately chosen;
9. Finally, if none was found, then the robot should stop.

5. Simulations

The following figures present the avoidance tested in simulation. For each robot-obstacles transaction, 4 sequences (a to d) are shown for better understanding of the avoidance performed. In these figures, the different faced colors indicate the different positions in time and a drawing of the robot (square) and the obstacles with the same color indicates their position for the same instant. Hence, for each sequence the movements go from white to black color, and the sequences b, c and d begin from the last point of the previous sequence. From one position, the time spent to the next position was 2 s for the Fig. (6) and (7), and 1 s for the Fig. (8). The arrows mean that after this position, they probably can follow the indicated direction. The goal in these simulations is to achieve the X point. The obstacle's movements had some changing in direction and velocity, in these cases the initial velocity was 0.3 m/s in modulus.

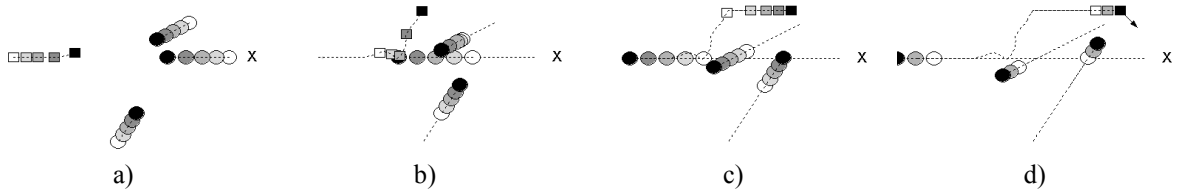

Figure 6: Obstacles avoidance – case 1.

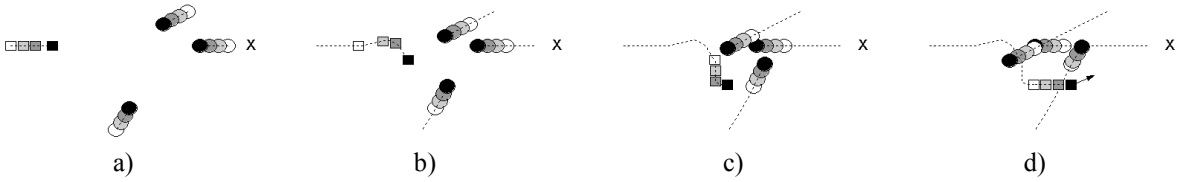

Figure 7: Obstacles avoidance – case 2.

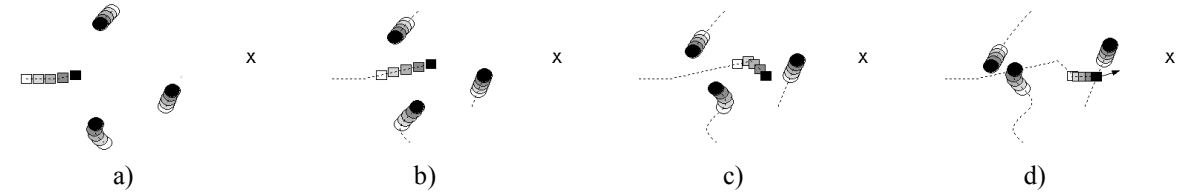

Figure 8: Obstacles avoidance – case 3.

6. Conclusions and Future Works

This work showed learning-based methods for dynamic obstacles avoidance, which can avoid more complex obstacles based on primitive actions. These primitive actions were obtained with RL technique trained with Monte Carlo algorithm. A combination of primitive actions and a well-structured logic permitted the robot avoid obstacles not used during the training, like ones with different trajectories or/and velocities.

However, more is still necessary. The concept presented here must be proved in situation with dynamic and static obstacles. The insistence of performing a solution for more complex problem utilizing primitive actions combined with some procedure will be tested. The success of avoidance of bigger obstacle, while the training was performed using only punctual obstacles, is a motivation of this strategy.

7. Acknowledgement

Sincere acknowledgements to CAPES, DAAD, ITA and UFMA for the execution and financial support of this research.

8. References

Almeida Neto, A., Góes, L. C. S. and Nascimento Jr., C. L., 2001, "Multiple Neural Networks in Flexible Link Control Using Feedback-Error-Learning", Proceedings of the 16th Brazilian Conference on Mechanical Engineering, Vol.1, Uberlândia, Brazil, pp. 307-315.

Almeida Neto, A., Heimann, B., Góes, L. C. S. and Nascimento Jr., C. L., 2002, "Obstacle Avoidance in Dynamic Environment: a Hierarchical Solution", Proceedings of the 6th Brazilian Conference on Neural Network, Vol.1, São Paulo, Brazil, pp. 289-294.

Freund, E., Schlusse, M. and Rossmann, J., 2001, "Dynamic Collision Avoidance for Redundant Multi-Robot Systems", Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System, Vol. 3, pp. 1201-1206.

Gachet, D., Salichs, M. A., Moreno, L. and Pimentel, J. R., 1994, "Learning Emergent Tasks for an Autonomous Mobile Robot", Proceedings of the IEEE/RSJ/GI Int. Conference on Intelligent Robots and Systems '94, Vol. 1, pp. 290-297.

Kawano, H. and Ura, T., 2001, "Dynamics Control Algorithm of Autonomous Underwater Vehicle by Reinforcement Learning and Teaching Method Considering Thruster Failure under Severe Disturbance", Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System, Vol. 2, pp. 974-979.

Moody, J. and Saffell, M., 2001, "Learning to Trade via Direct Reinforcement", IEEE Transactions on Neural Networks, Vol. 12, pp. 875-889.

Sharkley, A., 2000, "Combining Artificial Neural Networks", Proceedings of the 6th Brazilian Symposium on Neural Networks.

Shiller, Z., Large, F. and Sekhavat, S., 2001, "Motion Planning in Dynamic Environments: Obstacles Moving Along Arbitrary Trajectories", Proceedings of the 2001 IEEE/RSJ Int. Conference on Intelligent Robots and System, Vol. 4, pp. 3716-3721.

Sutton, R. S. and Barto, A. G., 1998, "Reinforcement Learning: An Introduction", Cambridge-USA, The MIT Press.

Tang, P., Yang, Y. and Li, X., 2001, "Dynamic Obstacle Avoidance Based on Fuzzy Inference and Transposition Principle for Soccer Robots", Proceedings of the 10th IEEE Int. Conference on Fuzzy Systems, Vol. 2, pp. 1062-1064.

Tsourveloudis, N. C., Valavanis, K. P. and Hebert, T., 2001, "Autonomous Vehicle Navigation Utilizing Electrostatic Potential Fields and Fuzzy Logic", IEEE Transactions on Robotics and Automation, Vol. 17, no. 4, pp. 490-497.

Yen, G. and Hickey, T., 2002, "Reinforcement Learning Algorithms for Robotic Navigation in Dynamic Environments", Proceedings of the 2002 Int. Joint Conference on Neural Networks, Vol. 2, pp. 1444-1449.

Zurada, J. M., 1992, "Introduction to Artificial Neural Networks", New York, West Pub. Co.