Double Action Q-learning for Obstacle Avoidance in a Dynamically Changing Environment

Daniel C. K. Ngai, and Nelson H. C. Yung, *Senior Member, IEEE*

Abstract — **In this paper, we propose a new method for solving the reinforcement learning problem in a dynamically changing environment, as in vehicle navigation, in which the Markov Decision Process used in traditional reinforcement learning is modified so that the response of the environment is taken into consideration for determining the agent's next state. This is achieved by changing the action-value function to handle three parameters at a time, namely, the current state, action taken by the agent, and action taken by the environment. As it considers the actions by the agent and environment, it is termed "Double Action". Based on the Q-learning method, the proposed method is implemented and the update rule is modified to handle all of the three parameters. Preliminary results show that the proposed method has the sum of rewards (negative) 89.5% less than that of the traditional method. Apart form that, our new method also has the total number of collisions and mean steps used in one episode 89.5% and 15.5% lower than that of the traditional method respectively.**

Index Terms—Q-learning, reinforcement learning, temporal differences, obstacle avoidance

I. Introduction

Reinforcement Learning (RL) define a class of problem solving approaches by which the learner (agent) is not given the action to take but it must learn through a series of trial-and-error searches and the receipt of delayed rewards. [1, 2]. The purpose of the agent is to maximize not only the immediate reward, but also the cumulative reward in the long run [2]. In this way, the agent can learn to approximate an optimal behavioral strategy by continuously interacting with the environment [3]. This allows the agent to work in a previously unknown environment by learning to adapt to the rules in the new environment gradually, which is well-suited to dynamically changing environments.

There are two main strategies for the RL approach [1]. The first one is to search in the space of behaviors such as genetic programming in order to determine the behavior that performs well in the environment. The second is to consider the utility of taking actions such as dynamic programming (DP). In DP, the value functions are used to organize and structure the search for good policies so that it can learn to choose the best action in the corresponding state. In essence, DP is a collection of algorithms working on the Markov Decision Process (MDP) to give optimal policies [2]. By assuming that the decisions and values are functions only on the current state, the RL problem can be simplified into an MDP problem, which can be solved in a much easier way. Q-learning by Watkins [4] can be considered as an incremental method for DP as it determines the optimal policy in a way similar to DP but in a step-by-step manner instead. It is a simple model-free RL approach that allows the agent to learn to act optimally in the Markovian domain, and it is easy to implement and have a relatively lower computation cost than DP.

However, the ordinary RL approach based on the MDP model may face difficulties when handling dynamically changing environment. In a dynamically changing environment, the environment can change regardless of the agent's action. This violates the assumption of MDP that only the action of the agent can change the state of the environment. Although the learning feature of RL may enable it to adapt to the changing environment gradually, it can only converge to the changed environment but not learn to act regardless of the change of the environment. Therefore, more sophisticated approach is required to solve the problem. The world that we face every day is usually a dynamically changing environment. For instance, we perform obstacle avoidance when walking through a crowded street. In such an environment, the agent is faced with both moving and static objects and the agent should have the ability to deal with both of them. For this reason, we have decided to explore the possibility of improving the ordinary RL approach by considering not only the action of the agent but also the predicted action taken by the environment (or objects in the environment). By considering the action that may be taken by the environment, the agent can learn to react with the most appropriate action whatever the environment may change.

In this paper, a new method is proposed to solve the RL problem in a dynamically changing environment. The new method has taken into consideration the action taken by both the agent and the environment. The Q-learning technique is used to implement a new RL method called the double action Q-learning based on the new method. Simulation results show that the new method works better than the Q-learning method employed in the traditional

Manuscript received April 1, 2005. This work was supported by the research postgraduate studentship from University of Hong Kong.

D. C. K. Ngai is with the Department of Electronic and Electrical Engineering, University of Hong Kong, Hong Kong, Pokfulam Road, Hong Kong SAR, China (e-mail: ckngai@eee.hku.hk).

N. H. C. Yung is with the Department of Electronic and Electrical Engineering, University of Hong Kong, Hong Kong, Pokfulam Road, Hong Kong SAR, China (e-mail: nyung@eee.hku.hk).

0-7803-8961-1/05/$20.00 ©2005 IEEE.

MDP model, which makes it highly suitable for vehicle navigation where the environment consists of dynamically changing vehicles, and each one of them can take actions that may affect the agent's decisions.

The rest of this paper is organized as follow. Problem analysis is detailed in Section 2. Following that, the proposed method is presented in Section 3. Simulation results and discussion are depicted in Section 4. Suggestions for future development are given in Section 5, and the conclusion can be found in Section 6.

II. Problem Analysis

MDP is a model for sequential decision making under uncertainty. The term Markov stands for the fact that the transition probability and the reward function depend on the current state and the action selected by the agent only [5]. To describe the model, we first consider a specific instant of the system that consists of an agent and the environment. At this instant, the agent observes the state of the system. It then chooses an action based on the observed state and the policy used, which is implementation-dependent. The action taken by the agent causes the environment to change to a new state according to the transition probability. At the same time, the environment returns some feedback (reward or cost) to the agent according to a reward function. The agent then needs to face a similar problem again in the next time instant [5].

In order to determine the optimal policy in MDP, a value function is introduced to describe the value of a state. That is, the expected infinite discounted sum of reward that the agent will gain. The value function is given below [2]:

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\} \quad (1)$$

where $E_{\pi}\{\}$ represents the expected value when policy π is followed and R_t is the discounted sum of future rewards; γ is the discounting factor and r_t is the reward (or penalty) received at time t . The optimal value function can be expressed as follow [2]:

$$\begin{aligned} V^*(s) &= \max_a E_{\pi^*}\{R_t \mid s_t = s, a_t = a\} \\ &= \max_a E_{\pi^*}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\} \\ &= \max_a E_{\pi^*}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s, a_t = a\right\} \\ &= \max_a E\left\{r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a\right\} \\ &= \max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V^{\pi}(s')\right] \end{aligned} \quad (2)$$

where $P^a_{ss'}$ and $R^a_{ss'}$ are the transition probability from state s to s' and the expected reward for the transition respectively. The corresponding action-value function and the optimal action-value function are (3) and (4) respectively [2]:

$$\begin{aligned} Q^{\pi}(s,a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\} \end{aligned} \quad (3)$$

$$\begin{aligned} Q^*(s,a) &= E\left\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right\} \\ &= \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma \max_{a'} Q^*(s_{t+1}, a')\right] \end{aligned} \quad (4)$$

In Q-learning [4], the agent tries to take an action according to the policy used and the Q-value in the particular state. In the next time step, it evaluates the action by the reward or penalty it received and the expected value of the current state. The Q-learning method has been proven to converge to the optimal action-value with probability 1 as each action is executed in each state an infinite number of times [1] [5]. The Q-learning update rule is defined as below:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right] \quad (5)$$

When handling a dynamically changing environment, the ordinary Q-learning method may not be able to converge probably as the next state is not solely affected by the agent but can also be affected by the change of the environment. In this case, the update rule may cause the Q-value to fluctuate and the agent may fail to response appropriately. This effect could be visualized in the simulation results in Section 4. Thus, a new method which is able to handle the dynamics of the environment is proposed to solve the convergence problem.

III. Proposed Methodology

A. Philosophy and Concept

Traditional MDP states that when an agent takes an action according to the present observed state, the environment will change to a new state and the agent will receive a reward (or penalty). This has assumed implicitly that the environment is static and will change only according to the action taken by the agent. Although the transition probability function can in some way express changes in the environment, this factor is not considered in the Q-learning equation. Therefore, our proposal is to consider both the actions taken by the agent and the environment as factors that cause the change in state. We argue that the environment as a whole can be considered as the action taker. Multiple action takers can also exist if we assume that the environment contains objects that some of them can take avoidance action.

In this paper, the proposed method is implemented using the technique similar to Q-learning. The new method, called the double action Q-learning, has two fundamental differences compared with MDP. First of all, a state defines the relationship between two objects in the environment. A state has no meaning when there is no observer. That is, a state only exists when there is an observer standing on some point observing the environment. Besides, for two observers standing on different points observing the same environment, the observed states would be different. The

state information is only valid within the observer itself and so state should be interpreted as observer-dependent. As such, the environment can be considered as having different objects with each object having its own properties and its own state with respect to the agent. The states of all objects added together form the environment. Secondly, the environment can change by itself. It changes whether the agent acts or not. In the new method, both the observer and the environment can take actions and cause the change in state. As both parties are changing, what an observer observes is the net change after both parties have acted.

The new method can be described in more detail as depicted in Fig. 1b. It consists of 6 major parameters. They are: sampling time (T), the current state (s_t), the action selected by the observer (a^1), the action selected by the environment (a^2), the transition probability ($P_{ss'}^{a^1 a^2}$) and the reward function ($R_{ss'}^{a^1 a^2}$). The state will change when either (or both) the observer or the environmental object has taken its action. In the case of obstacle avoidance, one of the objects is the agent and the other one can be considered as the obstacle. The traditional MDP model is also shown in Fig.1a for reference. As it only considers the action a taken by the agent, all the parameters are related to a only.

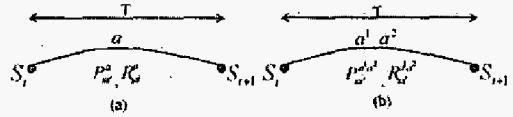

Fig. 1 State Diagrams: a) MDP model. b) new method

B. The new Value Function

In the new method, similar to the traditional MDP model, the value function is introduced to estimate the expected future rewards of a particular state.

$$\begin{aligned} V^{\pi}(s) &= E_{\pi,\pi'}\{R_t \mid s_t = s\} \\ &= E_{\pi,\pi'}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s\right\} \\ &= E_{\pi,\pi'}\left\{r_{t+1} + \sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s\right\} \\ &= \sum_{a^1}\pi(s,a^1)\sum_{a^2}\pi'(s,a^2)\sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma E_{\pi,\pi'}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_{t+1} = s'\right\}\right] \\ &= \sum_{a^1}\pi(s,a^1)\sum_{a^2}\pi'(s,a^2)\sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma V^{\pi}(s')\right] \end{aligned} \tag{6}$$

where $E_{\pi,\pi'}\{\}$ represents the expected value when policy π is followed by the agent and policy π' is followed by the environment. The optimal value function will then be:

$$\begin{aligned} V^{*}(s) &= \max_{a^1 a^2} E_{\pi^*,\pi'}\{R_t \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2\} \\ &= \max_{a^1 a^2} E_{\pi^*,\pi'}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2\right\} \\ &= \max_{a^1 a^2} E_{\pi^*,\pi'}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2\right\} \\ &= \max_{a^1 a^2} E\{r_{t+1} + \gamma V^{*}(s_{t+1}) \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2\} \\ &= \max_{a^1 a^2}\sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma V^{\pi}(s')\right] \end{aligned} \tag{7}$$

The corresponding optimal action-value function would then be:

$$\begin{aligned} Q^{*}(s,a^1,a^2) &= E\left\{r_{t+1} + \gamma\max_{a^{1'},a^{2'}}Q^{*}(s_{t+1},a^{1'},a^{2'}) \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2\right\} \\ &= \sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma\max_{a^{1'},a^{2'}}Q^{*}(s_{t+1},a^{1'},a^{2'})\right] \end{aligned} \tag{8}$$

It can be seen in equations (7) and (8) that the action a^2 is involved in determining the value function and the action-value function. However, as a^2 is the action taken by the environment, it is uncontrollable by the agent and therefore maximize the equations on a^2 is not meaningful. In this case, we may consider obtaining a more realistic version of the optimal value function by maximizing a^1 instead.

$$\begin{aligned} V^{*'}(s) &= \max_{a^1} E_{\pi^*,\pi'}\{R_t \mid s_t = s, a_t^1 = a^1\} \\ &= \max_{a^1} E_{\pi^*,\pi'}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} \mid s_t = s, a_t^1 = a^1\right\} \\ &= \max_{a^1} E_{\pi^*,\pi'}\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2} \mid s_t = s, a_t^1 = a^1\right\} \\ &= \max_{a^1} E_{\pi'}\{r_{t+1} + \gamma V^{*}(s_{t+1}) \mid s_t = s, a_t^1 = a^1\} \\ &= \max_{a^1}\sum_{a^2}\pi'(s,a^2)\sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma V^{\pi}(s')\right] \end{aligned} \tag{9}$$

The corresponding optimal action-value function would then be the expected value of Q* over the action a^2:

$$\begin{aligned} Q^{*'}(s,a^1) &= E_{\pi'}\{r_{t+1} + \gamma\max_{a^{1'}}Q^{*}(s_{t+1},a^{1'}) \mid s_t = s, a_t^1 = a^1\} \\ &= \sum_{a^2}\pi'(s,a^2)\sum_{s'}P_{ss'}^{a^1 a^2}\left[R_{ss'}^{a^1 a^2} + \gamma\max_{a^{1'}}Q^{*}(s_{t+1},a^{1'})\right] \end{aligned} \tag{10}$$

This is the action-value function for the agent to maximize its rewards regardless of the action taken by the environment. When comparing equations (2) and (4) with equations (9) and (10) respectively, we can find that the old model is in fact a special case of the new one where the old model simply assume that the environment does not act .

Fortunately, in the real world, it is possible to predict the motion of the environment through sensory inputs from which we can take appropriate action to maximize our reward. If we can predict the action that will be taken by the environment accurately, the function $Q^{*}(s,a^1,a^2)$ will be very useful to determine our best action in the next time step.

By using the same technique as Q-learning, the function $Q^{*}(s,a^1,a^2)$ can be updated continuously that fulfills the purpose of RL. The Q-learning type update rule for the new MDP model is given below:

$$\begin{aligned} Q(s,a^1,a^2) \leftarrow\ & Q(s,a^1,a^2) \\ & + \alpha\left[r + \gamma\max_{a^{1'},a^{2'}}Q^{*}(s',a^{1'},a^{2'}) - Q(s,a^1,a^2)\right] \end{aligned} \tag{11}$$

However, using $\max_{a^{1'},a^{2'}}Q^{*}(s',a^{1'},a^{2'})$ to update the Q-value is not appropriate as the maximum Q-value for a^2, may not be the actual action that is taken by the environment. Instead, the predicted $a^{2'}$ should be used to update the Q-values. If

the agent knows the probability distribution of the a^2, the expected value can also be used as an alternative.

This update rule may be used only if we can predict the action taken by the environment accurately. If the prediction is not accurate, the Q-value may not converge and thus the agent may fail to act appropriately. One less aggressive approach is to use the mean value of $\max_{a^1} Q(s',a^1)$ over all $a^{2'}$. That is:

$$Q(s,a^1,a^2) \leftarrow Q(s,a^1,a^2) + \alpha\left[r+\gamma\frac{\sum_{a^{2'}}\max_{a^1} Q(s',a^{1'},a^{2'})}{count(a^{2'})} - Q(s,a^1,a^2)\right] \quad (12)$$

where $count(a^{2'})$ is the number of $a^{2'}$ available in the system. The term mean value of $\max_{a^1} Q(s',a^1)$ over all $a^{2'}$ is used instead of $\max_{a^1} Q(s',a^1,a^2)$ because it is difficult at this stage for the agent to know exactly what action the environment will take in the next time step. As a result, it would be more appropriate for the agent to choose the mean value of the Q value as the discounted future state-action value in the update rule.

C. The Double Action Q-learning Algorithm

As depicted in Fig. 2, there are two importantly points to be noted in the new method.

Initialize $Q(s,a^1,a^2)$ arbitrarily
Repeat (for each episode)
  Initialize s
  Repeat (for each step of episode):
    Get a^2 by predicting the action that will be taken by the environment
    Choose a from s using policy derived from Q
    Take action a
    Observe the new state s' and r
    Determine the action a^1 that have been taken by the environment
    $Q(s,a^1,a^2) \leftarrow Q(s,a^1,a^2) + \alpha\left[r+\gamma\frac{\sum_{a^{2'}}\max_{a^1} Q(s',a^{1'},a^{2'})}{count(a^{2'})} - Q(s,a^1,a^2)\right]$
    $s \leftarrow s'$
  until s is terminal

Fig. 2 The double action Q-learning algorithm

First of all, in the new method, the agent needs to consider two actions (one from itself and the other from the environment) before it can choose the action with the highest expected future rewards. To know the action that is taken by the environment in the next time step, the agent must have the ability to predict. The prediction module can be realized independently of the RL process. That is, the prediction module can learn to predict disregarding the MDP model and predict by using historical information of the environment.

During the learning cycle, the agent needs to be aware of what actions have been taken by the environment before the Q-values can be updated. This is performed by the action determination module. After knowing the action that has been taken by the environment, the agent can update it's Q-value according to (12).

D. Obstacle Avoidance by using the new method

In the obstacle avoidance case, we need to plan and control the agent's motion to ensure that it can avoid collisions with moving obstacles simultaneously. In this paper, obstacles are objects or vehicles and the environment contains multiple numbers of obstacles. It is highly suitable for vehicle navigation in which there are a multiple number of moving vehicles and each is capable to avoid collision. Moreover, the interpretation of states as the relationship between one object and the agent allows us to implement the Q-learning algorithm in a more efficient way.

By treating each obstacle as an independent object, Q-learning can be applied on each pair of object-agent relationship. Laurent and Piat [6,7] have proposed a similar approach called the parallel Q-learning to solve the block-pushing problem. They have tried different methods to combine Q-value from different blocks. The first one is to take the maximum Q-value over all blocks. The second one is to take the sum of all Q-values from all blocks. In our case, the second approach is used in combining the Q-values from different object-agent relationships. That is:

$$Q(s,a^1) = \sum_i q_i(s_i,a^1,a_i^2)$$

where $Q(s,a)$ is the resultant Q-value for the entire environment and $q_i(s_i,a^1,a_i^2)$ is the Q-value of the particular type of object when the agent face that object along. The action a^2 is determined through the prediction module. $\max_a Q(s,a)$ is then chosen as the final decision of the obstacle avoidance (OA) module. For the same type of objects, the agent uses the same set of Q-value to represent them. Therefore, when there are multiple objects which are of the same type in the environment, the Q-value belonging to that type of objects is updated multiple times in each time step. As such, this technique has greatly enhanced the efficiency of Q-learning.

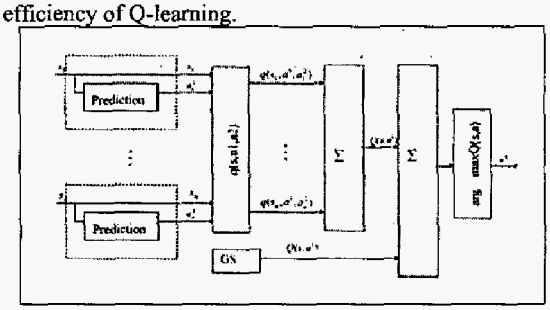

Fig. 3 Architecture of the obstacle avoidance system

In our case, a goal is established for the agent so that it must reach the goal even if it has to pass through a crowd of obstacles but not staying away from the obstacles. This ability is called the Goal Seeking (GS) ability. In this paper,

we assumed that the agent knows the position of the goal so that the GS module is able to provide indications on which action can get closer to the goal. Other actions have their Q-value gradually decreasing according to the distance that they can minimize. The Q-values from the GA and the OA modules are summed to form the final Q-value. The action with the maximum final Q-value is selected as the final decision. For evaluation purpose, we focus on the double action Q-learning method and thus assume that the prediction and action determination were accurately.

IV. Simulation Results And Discussion

The objective of the test is to compare the performance of the new method with that of the ordinary Q-learning method. The testing environment consists of large numbers of obstacles and the agent itself. The state indicating the agent-object relationship is represented by the x and y coordinates of the object with the agent placing in the origin. The agent and the obstacles can choose one of 5 actions: up, down, left, right, rest. In our test, all the obstacles are of the same type and thus all of them use only one set of Q-values. A goal is given to the agent and obstacles are randomly placed in between so that the agent needs to handle the problem of obstacle avoidance in its navigation. If collision occurs, both the obstacle and the agent are brought back to their last position before collision and a punishment of -10 is given to the agent. We have tested the system for 2000 episodes (origin to goal) in the two different methods.

Fig. 4 depicts a screen capture of the simulator when the agent (grey circle) was moving among 50 obstacles (black circles) in between the starting point (upper left hand corner) and goal (square at the lower right hand corner). The grey line shows the path that the agent has traveled. In the episode shown, there was no collision recorded. The obstacles lied on the grey lines only shows that they have moved to the area after the agent has moved away. The branched grey line shows that the agent has in some time applied an action that can avoid collision but contradicted with the GS module.

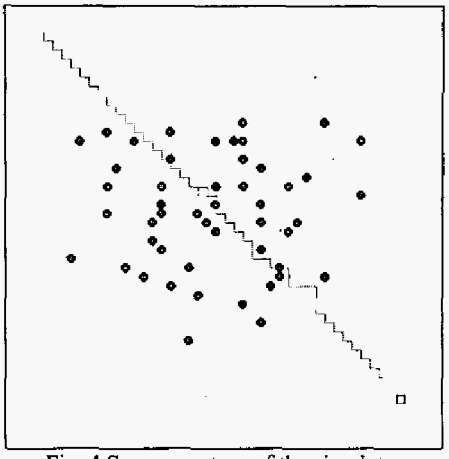

Fig. 4 Screen capture of the simulator

In Fig. 5, sum of rewards versus episode of both methods are shown. It is the cumulated sum of all the rewards over the total number of episodes. Larger negative value means more collisions have occurred. After 2000 episodes, the sum of reward (negative) of the new method is 89.5% less than that of the traditional method.

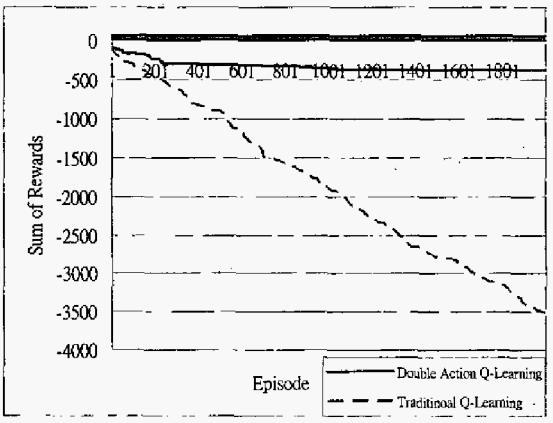

Fig. 5 Sum of Rewards (SR)

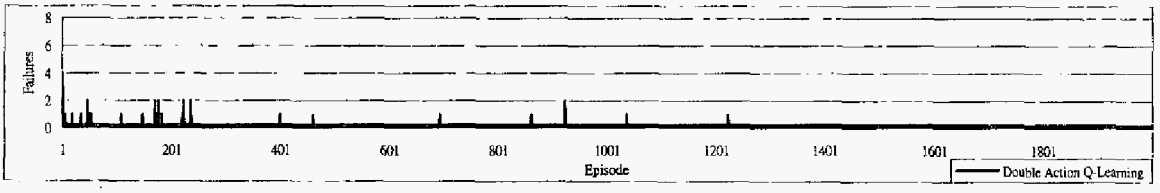

Fig. 6 Collisions for the Double action Q-learning Method (NoC)

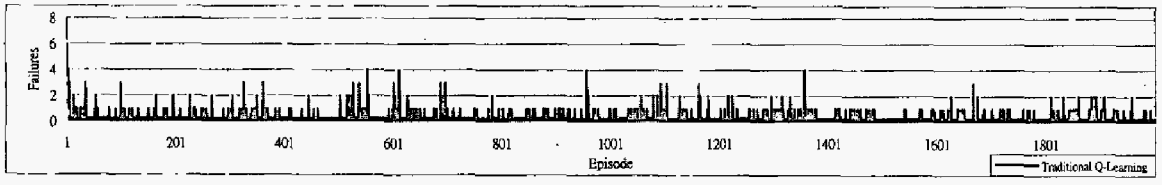

Fig. 7 Collisions for the traditional Q-learning Method

The number of collisions in an episode (NoC) represents how well collision avoidance is performed. A good learning algorithm should have the number of collision decreased as the number of episode increases. In Fig. 6 and 7, we can see that the total number of collisions for the new method is 37 compared with 351 collisions in the traditional method. The new method has its total number of collisions 89.5% less than that of the traditional method. In addition, the mean time between collisions (MTBC) for the new method is 678.6% longer than the traditional method after 2000 episodes.

According to Fig. 5, we can see that the sum of rewards (SR) for the new method is close to the traditional method in first 50 episodes. However, from episode 50 to 200, the sum of rewards for the new method slowed substantially. When considering Fig. 6, we can see that the number of collisions also starts to decrease at episode 50 and there are only a few collisions after episode 200. The reason is that the agent is learning in the first 200 episodes and it has nearly learned all the Q-values for all the cases that would cause collision after episode 200. Therefore, the number of collisions decreases naturally.

Fig. 8 shows the number of steps used (S/E) in one episode (from origin to goal) for the last 500 episodes. A smaller value means that the agent has chosen a shorter path. In this test, the minimum number of steps needed to reach the goal from the origin is 80 steps (shortest path). With reference to this figure, the new method requires only an average of 82 steps, or 15.5% less steps to complete the course than the traditional method. This showed that the new method can keep its route very near to the shortest path (80 steps), in most episode. On the contrary, the traditional Q-learning method requires a much larger number of steps for the agent to accomplish the collision avoidance task. This can be explained by the fact that the new method contains the prediction ability so that it can avoid collisions with obstacles properly according to the predicted action of the obstacles. In the traditional method, it is very likely that the agent needs to take actions to avoid obstacles more frequently although collision may not occur necessarily in the next step. As a result, the agent engages in avoidance action regardless of what action the obstacles may take. The results are summarized in Fig. 9.

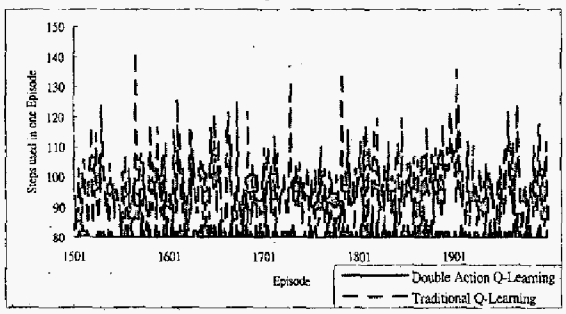

Fig. 8 Steps used in one Episode (S/E)

	Double action Q	Q-learning	% change
SR	-370	-3510	89.5% less (negative)
NoC	37	351	89.5% less
MTBC	4259	547	678.6% more
S/E	82	97	15.5% less

Fig. 9 Summary of Simulation results

V. SUGGESTIONS FOR FUTURE DEVELOPMENT

If the agent wants to maximize its future rewards from objects that may give rewards or punishments, it would need to compromise. In this paper, the summation method is used to simplify the problem. In the long run, weighted average according to their level of importance would have to be considered. Furthermore, more realistic prediction and action determination would need to be considered. At this stage, it can be seen that the prediction can work independently from the inference process. One possible direction is to integrate the two to form a complete solution.

VI. CONCLUSION

This paper has presented the double action reinforcement learning method. The Q-learning algorithm is used to implement the method and to solve the problem of dynamic obstacle avoidance. The new method has taken into account of the actions that are taken by the environment so that the system is now enabled to handle the dynamics of the environment. The concept of parallel Q-learning is also adopted so that the system can handle multiple numbers of obstacles at the same time. The test results show that the new method is better than the ordinary Q-learning method in handling the obstacle avoidance problem. From our simulation results, the new method has the sum of rewards (negative) 89.5% less than that of the traditional method. Apart form that, our new method also has the total number of collisions and mean steps used in one episode 89.5% and 15.5% lower than that of the traditional method respectively.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman & A. W. Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, vol. 4 pp237-285, 1996

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning-An Introduction*, The MIT Press, Cambridge, 1998

[3] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement", *Discrete Event Dynamic Systems: Theory and Application*, Vol. 13, pp 41-77, 2003

[4] C. J. C. H. Watkins and P. Dayan, "Technical Note: Q-learning", *Machine Learning*, Vol. 8, pp279-292, 1992

[5] M. L. Puterman, *Markov Decision Processes: discrete stochastic dynamic programming*, John Wiley & Sons, New York, 1994

[6] G. Laurent, E. Piat, "Parallel Q-Learning for a block-pushing problem", *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, USA, 2001

[7] G. Laurent, E. Piat, "Learning Mixed Behaviours with Parallel Q-learning", *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002