Performance Evaluation of Double Action Q-Learning in Moving Obstacle Avoidance Problem

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

Laboratory for Intelligent Transportation Systems Research,
Department of Electrical & Electronic Engineering,
The University of Hong Kong, Pokfulam Road, Hong Kong SAR, China,
E-mail: ckngai@eee.hku.hk

Abstract - *This paper describes the performance evaluation of Double-Action Q-learning in solving the moving obstacle avoidance problem. The evaluation is focused on two aspects: 1) obstacle avoidance, and 2) goal seeking; where four parameters are considered, namely, sum of rewards, no. of collisions, steps per episode, and obstacle density. Comparison is made between the new method and the traditional Q-learning method. Preliminary results show that the new method has the sum of rewards (negative) 29.4% and 93.6% less than that of the traditional method in an environment of 10 obstacles and 50 obstacles respectively. The mean no. of steps used in one episode is up to 26.0% lower than that of the traditional method. The new method also fares better as the number of obstacles increases.*

Keywords: Q-learning, reinforcement learning, temporal differences, obstacle avoidance

1 Introduction

Reinforcement Learning (RL) defines a class of problem solving approaches by which the learner (agent) must learn through a series of exploratory searches and reinforcement in the form of delayed rewards from the environment [1], [2]. It maximizes the immediate reward, and the cumulative reward in the long run, from which the agent learns to approximate an optimal behavioral strategy by continuously interacting with the environment [3]. This enables the agent to learn to adapt gradually in unknown or dynamically changing environments.

Q-learning by Watkins [4] is one such solution to the RL problem. It is a simple model-free approach that allows the agent to learn to act optimally in the Markovian domain, and it is easy to implement and has a relatively low computation cost. In the problem of moving obstacle avoidance (OA), the environment is dynamically changing, regardless of the agent's action. This violates the assumption of Markov Decision Process (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 change gradually, it can only converge to the changed environment but not learn to act while the environment changes. For this reason, we are motivated to consider not only the action of the agent but also the predicted action taken by the environment (or objects in the environment). As such, the agent can learn to react with the most appropriate action whatever the environment may be.

In this paper, we will describe the principle of the Double-Action Q-learning (DAQ-learning) method and evaluate it with respect to its obstacle avoidance and goal seeking capabilities. Its performance is also compared with traditional Q-learning in terms of sum of rewards, no. of collisions, steps per episode and obstacle density. Preliminary results show that the new method has the sum of rewards (negative) 29.4% and 93.6% less than that of the Q-learning in an environment of 10 obstacles and 50 obstacles respectively. The mean no. of steps used in one episode is up to 26.0% lower than that of the traditional method. The new method also fares better as the number of obstacles increases. The rest of this paper is organized as follows. DAQ-learning is detailed in Section 2. Following that, the evaluation approach is presented in Section 3. Evaluation results and data analysis are depicted in Section 4, and the conclusion can be found in Section 5.

2 Double-Action Q-learning

According to Sutton [5], reinforcement learning is to learn an appropriate mapping from situations to actions (policy) in which the reward is maximized. To carry out the mapping and maximize the reward, an MDP model is often used and the value function is built to achieve such a purpose. Q-Learning is one of the methods derived from MDP, where 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 has 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] [6]. Its update rule is defined as below:

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

where s_t is the old state, s_{t+1} is the new state, a is the chosen action in state s_t, α is the learning rate, and γ is the discount rate. In a dynamically changing environment (DCE), the MDP model may not be sufficient in describing all its dynamics. This is because in a DCE, the next state is not solely affected by the agent but can also be affected by the change of the environment. As traditional Q-Learning is based on the MDP model, it basically does not apply in a

DCE. In such an environment, the update rule of the Q-Learning method may cause the Q-value to fluctuate. This effect could be visualized in the simulation results depicted in Section 4. The DAQ-learning was proposed to solve this problem [7]. It has two fundamental differences compared with MDP. First, a new state is defined over the relationship between the agent and an obstacle in the environment. As such, the environment can be considered as having different obstacles, static or not, with each obstacle having its own properties and its own state with respect to the agent. The states of all obstacles added together form the environment. Second, the environment can change by itself, disregard whether the agent acts or not. In this 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.

In DAQ-learning, the Q-value is updated iteratively and the agent learns to act in different states. The corresponding modified Q-learning update rule for DAQ-learning is:

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

where a^1 is the action taken by the agent and a^2 is the action taken by the environment. However, as $a^{2'}$ is the action that will be taken by the environment in the current time step, it is not controlled by the agent and therefore maximize the equations on $a^{2'}$ is not meaningful. Instead, the predicted $a^{2'}$ should be used to update the Q-values. If the agent knows the probability distribution of $a^{2'}$, the expected value can also be used as an alternative. However, 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 $\max_{a^{1'}} Q(s',a^{1'},a^{2'})$ over all $a^{2'}$. That is:

$$Q(s,a^1,a^2) \leftarrow Q(s,a^1,a^2) + \alpha\left[r + \gamma \frac{\sum_{a^{2'}} \max_{a^{1'}} Q(s',a^{1'},a^{2'})}{count(a^{2'})} - Q(s,a^1,a^2) \right], \quad (3)$$

where $count(a^{2'})$ is the number of $a^{2'}$ available in the system. The term mean value of $\max_{a^{1'}} Q(s',a^{1'},a^{2'})$ over all $a^{2'}$ is used 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. Fig. 1 shows the algorithm of the DAQ-learning method. In this algorithm, 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 be able to predict. The prediction module can be realized independently of the RL process, through using historical information of the environment, as discussed in Section IV.

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^1 from s using policy derived from Q
    Take action a^1
    Observe the new state s' and r
    Determine the action a^2 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. 1 The DAQ-learning algorithm

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 an action determination module. After knowing the action that has been taken by the environment, the agent can update it's Q-value according to (3). In the OA problem, we need to plan and control the agent's motion to ensure that it can avoid collisions with moving obstacles. By using DAQ-learning, states are interpreted as the relationship between an obstacle and the agent. Therefore, if there is more than one obstacle, multiple numbers of states will exist. By applied DAQ-learning on each pair of obstacle-agent relationship, the RL process can be made more efficient. Laurent and Piat [8], [9] 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, such as taking the maximum Q-value over all blocks, or taking the sum of all Q-values from all blocks. In our case, the latter is used in combining the Q-values from different obstacle-agent relationships, i.e.:

$$Q(s,a^1) = \sum_i q_i(s_i,a^1,a_i^2), \quad (4)$$

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. $\max_{a^1} Q(s,a^1)$ is then chosen as the final decision of the 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 obstacles which are of the same type in the environment, the Q-value belonging to that type of obstacles is updated multiple times in each time step. As such, this technique has greatly enhanced the efficiency of Q-learning. Fig. 2 depicts the architecture of the OA 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 GS 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 DAQ-learning method and thus assume that the prediction and action determination were accurate. The prediction accuracy issue will be further discussed at the end of section IV.

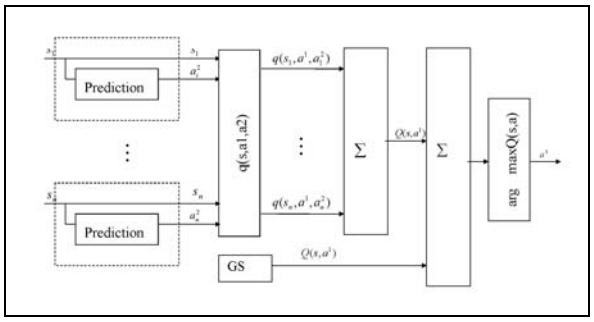


Fig. 2 Architecture of the obstacle avoidance system

3 Evaluation Approach

3.1 Evaluation focus

The evaluation is focused on comparing the difference between the proposed method and traditional Q-learning, in their obstacle avoidance and goal seeking abilities. With respect to the OA ability, the agent is evaluated on how punishment is minimized (or reward is maximized), how collision is minimized and how it copes with different number of obstacles. However, if an agent can avoid obstacles but can never reach its goal, it is useless. With respect to the GS ability, the agent is evaluated on how the number of steps required per episode to reach its goal is minimized. In general, there is always conflict between these two capabilities. In order to avoid collisions, one may need to travel for an extra distance before the goal can be reached. On the contrary, if one wants to reach the goal quickly, some collisions may not be avoidable. Therefore, a good approach would be the one that balances the two capabilities.

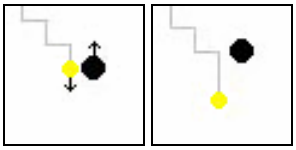
3.2 Evaluation parameters

According to the above, three parameters are used to evaluate the performance of the two OA methods. They are: 1) sum of rewards, 2) no. of collisions and 3) no. of steps per episode. Sum of rewards is the sum of all the rewards after at the end of 2000 episodes. Larger negative value means more collisions have occurred. A good learning algorithm should have the number of collisions decreases as the number of episode increases. In this paper, only one type of obstacles is used and therefore the sum of rewards is directly proportional to the number of collisions. The number of steps required in each episode corresponds to the steps taken by the agent to navigate from the origin to the goal. A smaller value means that the agent can reach the goal with a shorter path. It is bounded by the shortest path between the origin and goal. The first two parameters are directly related to the OA ability while the third one is a measure of the goal seeking ability. A sophisticated OA agent should be able to reduce its number of collisions, maximize the reward and take the shortest path to reach the goal. The agent that is able to fulfill all the above requirements is thus a good problem solver in the moving OA problem. To further evaluate the overall performance of DAQ-learning, environments with different number of obstacles are also used. All the obstacles are randomly placed in the same area so that the obstacle density increases with the number of obstacles linearly. As the probability of collision increases as the obstacle density increases, a high obstacle density environment can be used to distinguish easily between well performed OA method and other less well performed ones.

4 Evaluation Results and Data Analysis

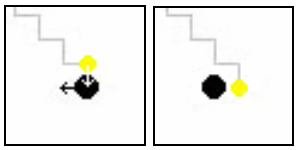
All the tests were carried out in a simulator built on the Linux platform. In our simulation, the state indicating the agent-object relationship is represented by the x and y coordinates of the object with the agent placing in the origin. The agent and the obstacles can choose one of the 5 actions: up, down, left, right, rest. A goal is given to the agent and obstacles are randomly placed in between so that the agent needs to navigate through the cluster of obstacles. When 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. Two environments of 10 and 50 obstacles respectively are used to test the two methods. In both environments, the obstacle are of the same type and will give the same amount of punishment (-10) to the agent upon collisions. They can only move within a certain boundary so that the obstacle density is relatively fixed. In our tests, the agent (grey circle) is originally placed at the upper left hand corner and its goal (not shown) at the lower right hand corner. The grey line indicates path that the agent has traversed. Initially, we present the behavior of the two methods in the environment of 10 obstacles. Fig. 3 depicts an agent using the traditional Q-learning method going through the crowd of obstacles towards the goal. In Fig. 3a, the agent encountered an obstacle on its right hand side. If the agents continue to move to the right and the obstacle remains static, collision will occur. As traditional Q-Learning does not consider the action of the obstacle, the Q-value for the action going to the right will have a relatively low value and thus the agent

select an action that have a higher value—move downward, even if the obstacle was moving upward, and the move-right action was possible.



(a) At time t (b) At time t+1

Fig. 3. Q-learning method



(a) At time t (b) At time t+1

Fig.4.DAQ-learning method

Fig. 4 showed a similar situation faced by the agent using the DAQ-learning method. In Fig. 4a, an obstacle was below the agent, which was moving to the left. As the DAQ-learning method has already considered the action that is taken by the obstacle, the agent moved downward following the suggestion of the GS module, continue its zigzag path. If the agent used Q-learning in the same situation, it would move to the right instead. The difference in responses of the two methods highlights the necessity of considering obstacles' action in a DCE. Fig. 5 depicts the sum of rewards versus episode. After 2000 episodes, the sum of reward of the new method is 29.4% less than that of the traditional method. Fig. 6 depicts the no. of steps taken in one episode of the two methods. 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). It can be seen that the no. of steps taken in one episode of the new method has a mean value of 3.11% less than that of the traditional method. This also showed that the new method can keep its route very near to the shortest path (80 steps) in most episodes. On the contrary, traditional Q-learning requires many more steps for the agent to accomplish the collision avoidance task. This can be explained by the fact that Q-learning does not consider obstacle actions, thus 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 in Fig. 3). As a result, the agent engages in avoidance action regardless of what actions the obstacles may take.

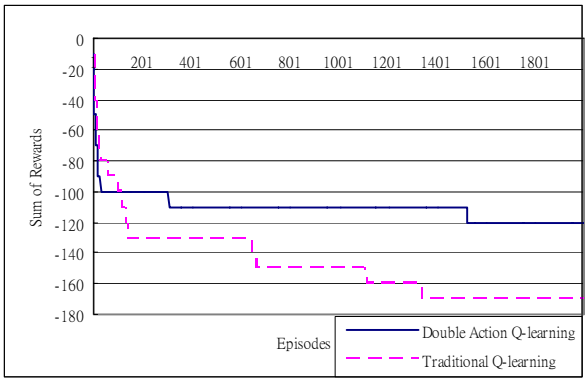
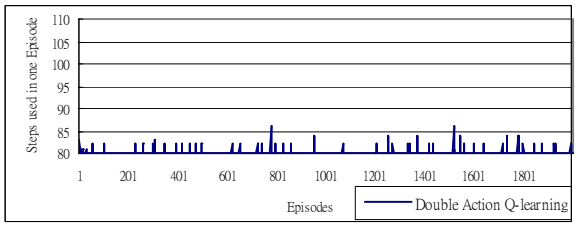
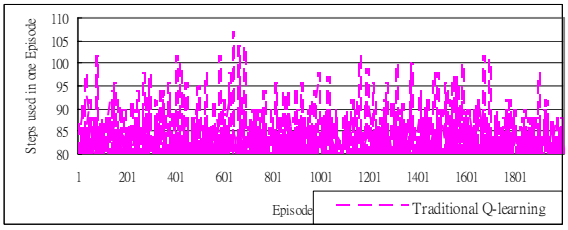


Fig. 5 Sum of Rewards (10 Obstacles).



(a) DAQ-learning



(b) Traditional Q-learning

Fig. 6. No. of steps taken/episode (10 Obstacles).

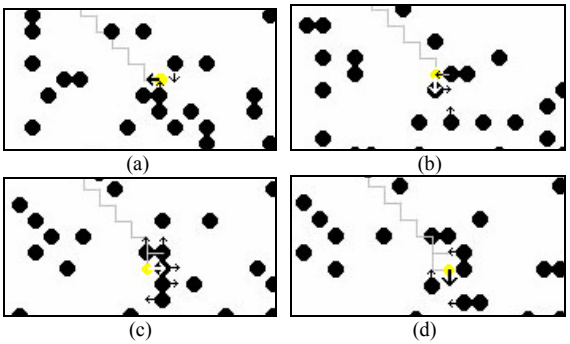


Fig. 7. Agent using DAQ-learning to avoid collision with multiple obstacles.

To investigate how DAQ-learning responds when dealing with multiple obstacles, the case of 10 and 50 obstacles in the environment were studied. In both cases, the chance of collision is high. Fig 7 depicts the agent using DAQ-learning to avoid collision with multiple obstacles. In Fig. 7a, the agent predicted that the obstacle on its upper right would block its path to the goal so it has chosen the action to move to the right. Moreover, it predicted that the obstacles in is bottom would move upward and cause collision. Therefore, it moved either up or left. Eventually, it has chosen to move left. In the next time step as shown in Fig. 7b, although an obstacle is beneath it, the agent moved downward in order to get closer to the goal and avoid collision as it predicted that the obstacle would move to the right. Such action also avoided collision with the obstacle on its right. Fig 7c and 7d depict the agent avoided collisions with other obstacles using similar approach. We can see that as obstacle actions are considered in the DAQ-learning method, the agent acted more intelligently in solving the OA problem when compared with the traditional Q-learning method. Fig. 8 depicts the sum of rewards versus episode of both methods for the environment with 50 obstacles. The new method has sum of reward 93.6% less than that of the traditional method after

2000 episodes. When comparing with the case of 10 obstacles, sum of rewards has increased (more negative) by 417% for DAQ-Learning and 5565% for traditional Q-Learning. The trend appears normal as the increase in the no. of obstacles also causes the no. of collisions to increase. On the other hand, DAQ-learning seems to avoid collision much better than Q-learning in a densely populated environment. Fig. 9 depicts the no. of steps taken in one episode of the two methods in the environment with 50 obstacles. The no. of steps taken in one episode by the new method has a mean value of 26.0% less than that of Q-learning. When comparing the environment of 50 obstacles with that of 10 obstacles, the values are 2.52% higher for the DAQ-Learning and 34.2% higher for traditional Q-Learning.

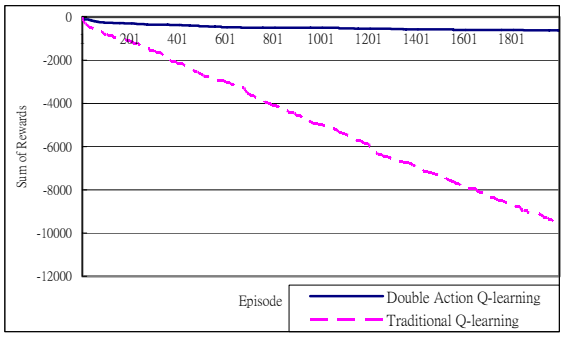
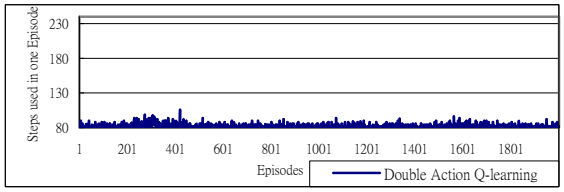
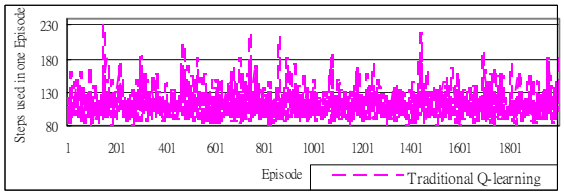


Fig. 8. Sum of Rewards (50 Obstacles).



(a) DAQ-learning



(b) Traditional Q-learning

Fig. 9. Steps used in one episode (50 Obstacles).

The result of the tests showed that when there is relatively smaller number of obstacles, the two methods behave similarly with DAQ-learning slightly better than the traditional one. But when the obstacle density increases, it is obvious that the new method is more capable in producing actions that balance collision avoidance and goal seeking. To further illustrate the effect of increasing obstacle density on the performance in the DAQ-learning method, Fig. 10-12 plots the three parameters versus the no. of obstacles up to 100. It shows that as the number of obstacles increases, the difference in performance between the two methods becomes greater. It appears that the new method highly suited for navigating in environments with high obstacle density.

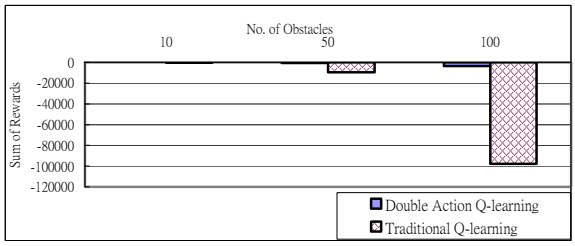


Fig. 10. Sum of rewards versus number of obstacles

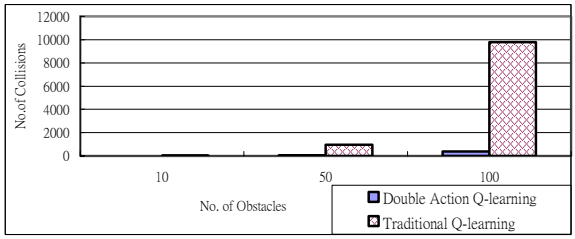


Fig. 11. Number of collisions versus number of obstacles

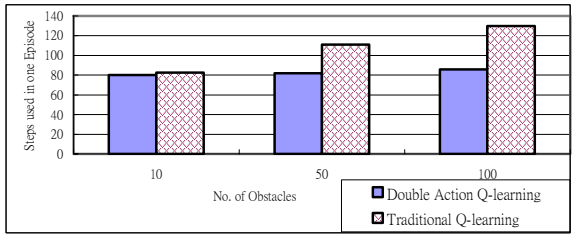
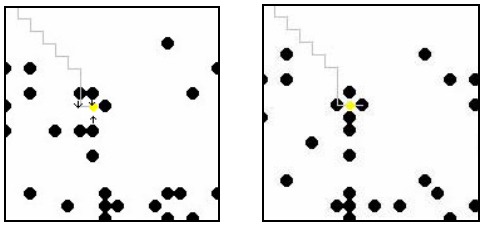


Fig. 12 Mean steps used versus number of obstacles

In order to better understand the DAQ-Learning method, we have taken a closer look at how collisions have occurred. We have identified two main causes for majority of collisions. The first cause is due to the conflicts between GS and OA. To solve this problem, a higher punishment can be given to the agent when collision occurs so that through enough learning, OA would dominate the decision process and thus collision would be minimized. The second cause is the presence of "traps". This particular situation is depicted in Fig. 13. It can be seen that the agent is 'boxed' by obstacles that no matter what actions that it takes, collisions are inevitable. One way to avoid it is to identify the "trap" some steps in advance before being boxed in. At present, the agent only predicts a step in advance, therefore is not able to handle it yet. In theory, if the prediction covers a few more steps, it is possible that such occurrence may be avoided.

Another issue concerning prediction accuracy is studied. Inaccurate prediction can be viewed as probability values on actions that the obstacles may take. For example, a prediction accuracy of 80% can be used to illustrate that in 80% of the time, the probability value of 1 is assigned to the action that the obstacle must take. For the remaining 20% of time, it assigns equal probability value to all the

actions that the obstacle may take, meaning that the prediction fails to predict correct actions of the obstacles. A summation of the product of the probability value and the corresponding Q-value is used to represent the actual Q-value for the current state for that particular obstacle. Using this definition, we have simulated prediction accuracy of 100% (perfect prediction) down to 0% (totally inaccurate prediction), with 50% and 80% in between. Table 1 depicts that the DAQL algorithm produces reasonable results even when the prediction is totally inaccurate, particularly when the obstacle density is high. Although inaccurate prediction has increased the number of collisions, which is to be expected, the performance of DAQL is still better than QL. This can be explained by the fact that even prediction is inaccurate in the agent's action determination cycle, it does not affect the learning cycle as it has the knowledge of what the agent has done after one time step. As a result, the correct Q-values generated help the agent to produce better values in representing the expected future reward then the QL algorithm. The contradictorily results show in the situation of 10 obstacles can be explained by the fact that the method of multiplying the probability measurement with the corresponding Q-value lowered the Q-value and caused conflicts between OA and DS. In this case, the agent is required to learn more before its Q-values are large enough to dominate DS. The DAQL method with inaccurate prediction may perform unsatisfactorily in the first several episodes, but with enough learning, the DAQL performs better than QL and finally outperform QL after 2000 episodes.



(a) "Trap" formed at t (b) Collision occurred at t+1

Fig. 13 A "trap" in the environment.

5 Conclusion

This paper has presented a solution and its detailed evaluation to the moving obstacle avoidance problem by using the RL approach. Evolved from the weaknesses of the traditional MDP model, a new method called double-action Q-learning has been proposed, which considers the action taken by both the agent and the environment [7]. The evaluation results showed that the DAQ-learning method is better than the traditional Q-learning method in the sum of rewards, no. of collisions per episode, no. of steps taken per episode, and is much more capable in navigating through an environment densely populated with obstacles. From our simulation results, the DAQ-learning method has the sum of rewards (negative) 29.4%, 93.6% and 96.3% less than that of the traditional method in an environment of 10 obstacles, 50 obstacles and 100 obstacles respectively. Apart from that, our new method also has the mean steps used in one episode 0.083%, 2.61% and, 7.37% more than the minimum no. of steps (80) in the three environments respectively. For the traditional Q-learning method, the three corresponding percentage are 3.30%, 38.6% and, 62.5% respectively. Results show that the new method works better than the traditional Q-learning method in solving the moving obstacle avoidance problem as the number of obstacles increases.

Table 1: Sum of rewards for different prediction accuracy

		No. of Obstacles		
		10	50	100
Sum of Rewards	DAQL 100%	-120	-620	-3610
	DAQL 80%	-260	-1480	-8430
	DAQL 50%	-400	-3120	-24690
	DAQL 0%	-490	-8020	-77690
	QL	-170	-9630	-97710

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] R. S. Sutton, *Reinforcement Learning*. Boston: Kluwer Academic Publishers, 1992.

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

[7] D. C. K. Ngai & N. H. C. Yung, "Double action Q-learning for obstacle avoidance in a dynamically changing environment", *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, Nevada, USA, pp.211-216, 2005.

[8] 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.

[9] 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.