Autonomous Robot Navigation with Self-learning for Collision Avoidance with Randomly Moving Obstacles

Yunfei Zhang and Clarence W. de Silva
Department of Mechanical Engineering
The University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
yfzhang@mech.ubc.ca and desilva@mech.ubc.ca

Dijia Su and Youtai Xue
Department of Electrical and Computer Engineering
The University of British Columbia
Vancouver, BC, V6T 1Z4, Canada

***Abstract*—This paper develops a hierarchical controller to avoid randomly moving obstacles in autonomous navigation of a robot. The developed method consists of two parts: a high-level Q-learning controller for choosing an optimal plan for navigation and a low-level, appearance-based visual servo (ABVS) controller for motion execution. The use of robot learning ability in collision avoidance is a novel feature, in a combined system framework of planning and visual servo control. The developed approach takes advantage of the on-board camera of robot whose finite field of view is naturally suitable for the Q-learning algorithm. Because of the Q-learning controller, knowledge of obstacle movement and a control law for the ABVS controller are not needed. This is a significant computational advantage. The method is implemented in a simulation system of robot navigation. The results show that Q-learning, which is a method of reinforcement learning, successfully converges to an optimal strategy for the robot to establish a proper motion plan.**

***Index Terms*—Robotic obstacle avoidance, Q-learning, Appearance-based visual servoing, Autonomous robot navigation.**

I. INTRODUCTION

Integrating learning ability into obstacle avoidance for a robot autonomously navigating in an unknown and dynamic environment has attracted considerable interest in the robotics community in the past decade. The schemes of obstacle avoidance in autonomous navigation may be categorized as model-based and model-free depending whether a model of the work environment is incorporated into the scheme. Integrating obstacle avoidance into a model-based navigation scheme would be desirable if the knowledge of the environment and the obstacles can be obtained in advance. Common methods [1]-[3] assume that some information about the environment is already known in advance. For example, a common characteristic of all these methods is the use of a three-dimensional (3D) model of the environment including such aspects as walls and doors, or the geometry of the path. In the present paper, instead, a model-free scheme is proposed for obstacle avoidance, which is suitable for real-time autonomous navigation. In particular, the method does not require a model of the environment or obstacle behavior.

In visual servo control (or visual servoing), the motion of a robot is guided to a target object through feedback control based on the information obtained from a vision system [4]. The method has great promise in sensor-based robotic tasks. Reference [5] executed trajectory following by merging differential flatness and predictive control. Epipoles was used in [6] to drive a nonholonomic robot to a desired configuration.

The visual servo method that is developed in the present paper relies on appearance-based navigation. The method was inspired by the work of [7][8]. It addresses navigation in a sensor space without any geometrical knowledge of the environment, and by incorporating obstacle avoidance and visual navigation directly into the control level, thereby avoiding complicated planning. Unlike the previous method, however, the present paper uses the concept of appearance-based visual servoing and proposes a new system framework that is suitable for both indoor and outdoor scenarios. Particularly, advantage is taken of the field of view from the visual system, and reinforcement learning is used to train the robot for avoiding obstacles. Once training is completed, the robot is able to adjust its movement depending on the real-time visual information, without any knowledge of the obstacles. The control model for visual servoing may be established in advance, as long as the target position is given. In [8], however, a path stored in the database of key images has to be followed, using a tentacle-based technique for obstacle avoidance, and a model-based approach for visual servo control. The associated computational burden can be prohibitive in practice. The method developed in the present paper is rather general, and it can be extended to many other sensors, such as 2D camera, 3D camera, range sensor and so on, assuming they can provide an adequate field of view or range. In the present paper, a 3D camera mounted on a mobile robot is used instead of an actuated camera and a rang scanner, which provides practical and economic advantages, in indoor environments.

The main contributions of the present paper are summarized now. The field of view of the visual system is used in obstacle avoidance where the obstacles are treated as parts of the real-time image. In previous work, obstacle

information is obtained by extracting useful features from the camera image and then comparing it with previous images in the image memory or in the database [8]. Also, what is proposed here is a generalized framework for obstacle avoidance in autonomous robot navigation, through the use of vision information, which is suitable for both local and global trajectory planning. Furthermore, in view of reinforcement learning, it is not necessary to develop a control law for visual servoing and a deterministic motion model for the obstacles.

The remainder of this paper is organized as follows. In Section 2 the problem definition is presented. In Section 3, the system model and frame projection model are developed. Reinforcement learning and its application in the present problem are introduced in Section 4. Simulation results are presented and discussed in Section 5. Conclusions are given in Section 6.

II. PROBLEM DEFINATION

A. General Description

This paper addresses the problem of obstacle avoidance in robotic navigation in a scenario where an initial local trajectory or global trajectory is given (e.g., using a global camera in an indoor environment). While following this initial trajectory, the robot might encounter an obstacle that was unknown (or did not exist) when establishing the initial path. For example, an obstacle might be occluded (a dead zone of the global camera). Under such scenarios, it is important for the robot to possess the capability of avoiding initially unknown obstacles that might suddenly appear during navigation. A technique is needed to recomputed the trajectory, which deviates from the current trajectory, in order for the robot to successfully reach the goal location. It is assumed that the goal location (local or global) is always available to the robot.

B. Obstacle Representation

The method developed in the present paper involves direct representation of an obstacle that appears in the image frame of the on-board camera, and then controlling the robot using that image information to avoid the obstacle. The navigation trajectory is then regenerated for reaching the goal location. As shown in Fig.1, first the safe area and the dangerous area have to be specified in the real field of robot navigation.

The robot will occupy a certain area when following a trajectory. The dangerous area may be specified on this basis, as shown in shadow, in Fig. 1. If an obstacle enters this area, collision may occur. The area outside this is the safe area. The degree of safety, with regard to collision, may be improved by expanding the dangerous area. These areas in the physical environment have to be mapped onto the camera image. This mapping will be discussed in section 3. It is sufficient to only map the dangerous area onto the image, because the remaining area of the image frame may be considered as the safe area. When an obstacle appears in the dangerous area of the field of view of the on-board camera, the robot has to be moved so that the complete obstacle is in the safe area.

There are three advantages in the present approach of obstacle avoidance. First, the safe and dangerous areas can be shaped according to the specific requirements of safety. Second, although camera images are used to present the method, it can be easily extended to other types of sensors (e.g., ultrasound). Third, the present appearance-based approach eliminates the need for robot pose or an obstacle model.

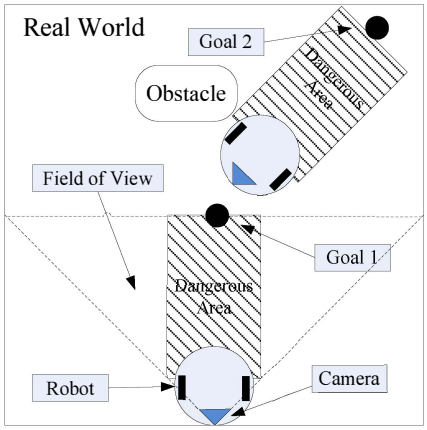

Fig. 1. Specification of the safe and dangerous areas based on newly detected obstacles.

III. SYSTEM MODELING AND OBJECT MAPPING

A. Mapping Model for Control System

Four sets of coordinate frames are defined in the overall system: the robot frame $F_R(O_R, X_R, Y_R, Z_R)$ (O_R = robot center of rotation; similarly for the other frames), camera frame $F_C(O_C, X_C, Y_C, Z_C)$, image frame $F_I(O_I, x_I, y_I)$ and pixel frame $F_P(O_P, x_P, y_P)$. Fig.2 shows the first two frames, where the robot frame is fixed to the mobile robot and the camera frame is fixed to the camera.

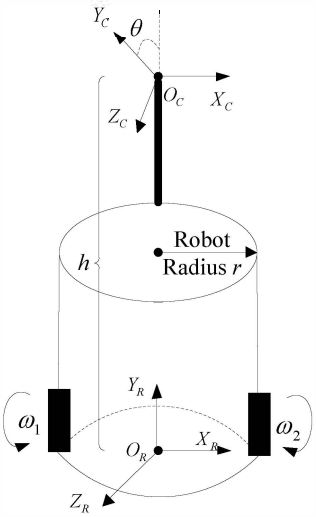

Fig. 2. Robot frame and camera frame.

The coordinate transformation between the two frames is given by the homogenous transformation [9]:

$$\boldsymbol{H}_c^r = \begin{bmatrix} \boldsymbol{R}_c^r & \boldsymbol{d}_c^r \\ 0 & 1 \end{bmatrix} \quad (1)$$

Here $\boldsymbol{H}_c^r$ represents the homogeneous transformation from the camera frame to the robot frame, $\boldsymbol{R}_c^r$ (3×3) represents the rotational matrix, and $\boldsymbol{d}_c^r = [d_x, d_y, d_z]^T$ is the position vector of the origin of the camera frame with respect to the robot frame.

A point $\boldsymbol{P}[X_C, Y_C, Z_C]^T$ in the camera frame can be transformed into the robot frame as,

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_c^r & \boldsymbol{d}_c^r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_c^r & \boldsymbol{d}_c^r \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} \quad (2)$$

In equation (2), the point vector $\boldsymbol{P}[X, Y, Z]^T$ is extended as $\boldsymbol{P}[X, Y, Z, 1]^T$ for the purpose of using the homogenous transformation.

The relationship between the camera frame and the image frame is modeled as a modified pinhole camera [10]. The camera plane is located at a distance f (focal length) behind the pinhole. The intersection of the z axis and the image plane is called the principal point. A clear advantage of this model is the absence of the image reversal problem.

Given the camera focal length f and the coordinates of the point object (for example, as obtained directly using the depth information from a 3D camera) with respect to the camera frame, the corresponding coordinates of the point the in image plane may be determined by

$$k \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = \begin{bmatrix} x_I \\ y_I \\ f \end{bmatrix} \quad (3)$$

where, $k = \dfrac{f}{z}$.

B. State definition and mapping

This section presents the method of dividing different areas in the robot frame and mapping them onto the image plane. Intuitively, the closer the obstacle to the robot, the more dangerous it is. The dangerous area should surround the robot and provide enough space for the robot to move without colliding with obstacles. When the system detects an obstacle that approaches the dangerous area, the robot is controlled to avoid it. Fig.3 illustrates this idea.

In Fig.3 (a), the dangerous area, shown as a shadow rectangle, is defined according to the size of the physical robot operating in the real environment. The width w of the red rectangle is equal to the diameter d of the robot, and the length l is equal to $2d$. Within the dangerous area, an extremely dangerous area may be defined where the robot will completely stop its movement when an obstacle appears. The empty area represents the safe area where obstacles will not appear. The dotted shadow area represents the goal area.

The dangerous area corresponding to the image plane can be obtained by substituting equation (3) into (2):

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} = \frac{1}{k} \begin{bmatrix} \boldsymbol{R}_c^r & \boldsymbol{d}_c^r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_I \\ y_I \\ f \\ 1 \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_I \\ y_I \\ f \\ 1 \end{bmatrix} = k \begin{bmatrix} \boldsymbol{R}_c^r & \boldsymbol{d}_c^r \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} X_R \\ Y_R \\ Z_R \\ 1 \end{bmatrix} \quad (4)$$

Clearly there is no need to map the entire dangerous area, point by point. Only eight points:

$(A_C, B_C, C_C, D_C, E_C, F_C, G_C, H_C)$ have to be mapped, which represent the corners of the areas.

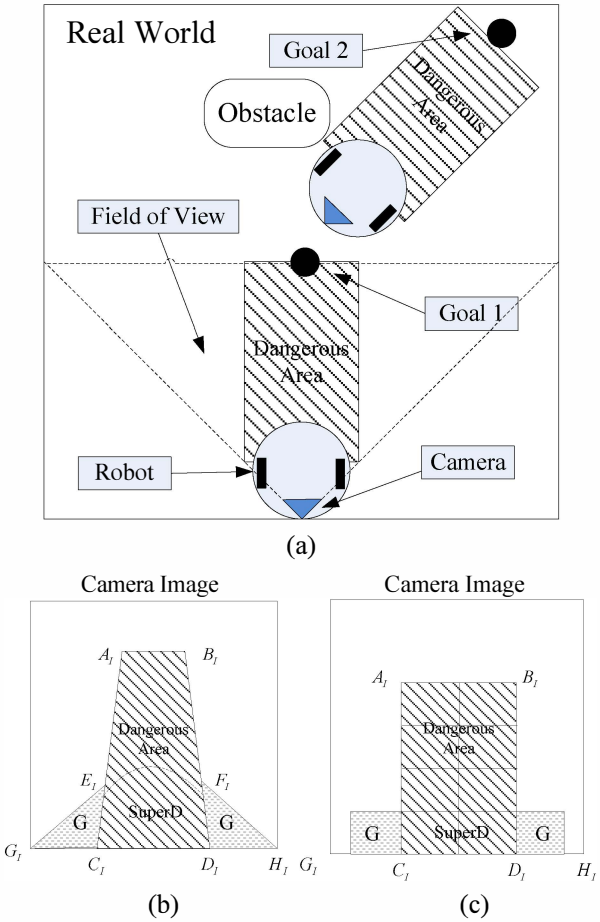

Fig. 3. State area definition and mapping: (a) In robot frame; (b) In image frame; (c) Adjusted image frame for safety.

These corner points are then linked using the linear relationship between the camera plane and the image plane. An example of a mapped dangerous area and a goal area (the real shape depends on the camera specification and the frame relationships) is shown in Fig.3 (b). Once the dangerous area is

determined, it may be expanded for added safety and for the convenience of defining states, as shown in Fig.3 (c).

The dangerous area with respect to the robot frame is used to define the state space of the environment and the action space. The dangerous area with respect to the image frame is used to check for obstacles and obtain their information. Details are given in Section 4.

IV. DESIGN OF Q-LEARNING CONTROLLER

A. Q-Learning

In reinforcement learning (RL) an agent learns the needed behavior through trial-and-error interactions with its dynamic environment. In this process, the agent performs an action a_t in state s_t and receives a real-valued reward $r_t = r(s_t, a_t)$ from the environment depending on the outcome of the action. Through this process, the agent learns a control policy $\pi: S \rightarrow A$, which maps state set S into an action set A, and arrives at its next state $s_{t+1} = \delta(s_t, a_t)$. The policy should maximize the cumulative reward. The agent's learning purposes is to find such π, as indicated in Fig. 4.

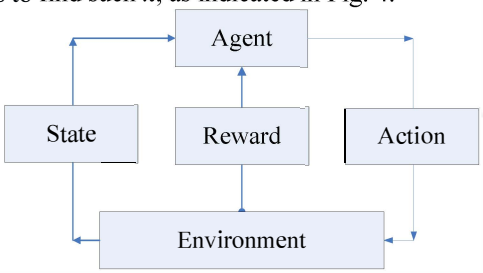

Fig. 4. The process of reinforcement learning.

Key issues of RL are the method of precisely specifying a policy π, which the agent should learn, and evaluating the π by calculating the maximum cumulative reward. To calculate the cumulate reward in choosing π, we use:

$$V^{\pi}(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots\ldots = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \qquad (5)$$

Here $0 \leqslant \gamma \leqslant 1$ is a constant that determines the relative value between the delayed and immediate rewards. It is reasonable to discount future rewards achieved by policy π because, in many cases, we prefer to obtain the reward sooner rather than later. In the domain of reinforcement learning, $V^{\pi}(s_t)$ is called the value function of state $r = -1$. This is the expected return by starting in s_t and following π thereafter. The value of $V^{\pi}(s_t)$ indicates how good the policy π is, in s_t.

The exact value of $V^{\pi}(s_t)$ cannot been known in advance if the robot had not actually interacted with its environment. A natural way is to use the notion of expected value of $V^{\pi}(s_t)$, $E_{\pi}(V^{\pi}(s_t))$, to express the process of reinforcement learning, as given by:

$$\begin{aligned} E_{\pi}(V^{\pi}(s_t)) &= E_{\pi}(\sum_{i=0}^{\infty} \gamma^i r_{t+i} \,|\, s_t) \\ &= E_{\pi}(r_t + \gamma \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \,|\, s_t) = E_{\pi}(r_t + V^{\pi}(s_{t+1})) \end{aligned} \qquad (6)$$

At this point it is appropriate to introduce the Temporal Difference (TD) learning structure. The basic TD algorithm that is used in the present work is TD(0), as given by:

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha[r_t + V^{\pi}(s_{t+1}) - V^{\pi}(s_t)] \qquad (7)$$

Similarly, the value function of selecting action a_t in state s_t under a policy π, denoted by $Q^{\pi}(a_t, s_t)$, is defined. The corresponding TD(0) algorithm is given by:

$$Q^{\pi}(a_t, s_t) \leftarrow Q^{\pi}(a_t, s_t) + \alpha[r_t + Q^{\pi}(a_{t+1}, s_{t+1}) - Q^{\pi}(a_t, s_t)] \qquad (8)$$

In the present work Q-learning is used, which is a popular version of reinforcement learning. It provides a significant off-policy TD method with many advantages. The underlying core algorithm is given by:

$$\begin{aligned} &Q(s_t, a_t) \leftarrow Q(s_t, a_t) \\ &+\alpha[r_t(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \end{aligned} \qquad (9)$$

The corresponding action policy π is given by:

$$\pi(s) = \arg\max_a Q(s, a) \qquad (10)$$

The advantages of Q-learning compared with other reinforcement learning approaches include the following: First, it does not require a model of the environment, which is an advantage when dealing with unknown environments. For example, in an unknown environment, $r_t = r(s_t, a_t)$ and $s_{t+1} = \delta(s_t, a_t)$ are nondeterministic functions. Here, r and s are initiated arbitrarily, and the process will converge to the optimal Q value after a certain number of iterations. Second, it is able to update estimates by using partially learned other estimates without waiting for completing the entire policy; this means it bootstraps. Although the details of the exploration strategy will not affect the convergence of the learning algorithm assuming that every state is visited infinitely according to its instinct convergence characteristic, it may converge quite slowly if the state space is extensive. (Sutton and Barto, 1998). However, in the present application, the image plane has a finite area, which is divided into finite and relatively few regions. This is a benefit when applying the Q-learning method.

B. Definition of States, Actions and Rewards

The definition of the states in the present Q-learning controller incorporates a discrete-grid world of the image plane, as shown in Fig.3(c). Here, the 640×480 image plane is divided into a world state of few discrete regions where each cell of the grid frame has a differently defined area. When an image is obtained from the camera, the position of the obstacle (e.g., represented by a key visual feature point associated with the

obstacle) on the image plane can be considered as a signal that determines whether a certain region is occupied by an obstacle. Accordingly, the world state at time t in the Q-learning controller is defined as $s_t = (g_1, g_2, ..., g_i, ..., g_{n-1}, g_n)$, where $g_i \in (0,1)$ is the state of each discrete region (number 1 represents that the state is occupied by an obstacle, 0 otherwise), and n is the total number of regions in the world state. Then, the total number of world states in the present scenario is

$$N = \sum_{i=1}^{n} \left(C_n^i = \frac{n(n-1)...(n-i+1)}{i!} \right).$$

In each world state, it is assumed that the robot is able to select four actions within the robot coordinate frame as follows:

Action 1: Move forward through 20 cm.

Action 2: Move back through 20 cm.

Action 3: Rotate left through 22.5º.

Action 4: Rotate right through 22.5º.

After the robot takes an action, the position of the visual point on the image plane will change. Then the Q-learning controller will receive the reward r according to the state s_{t+1} achieved by the action, as given by the following rules:

$$r = \begin{cases} +10, \text{if } s_{t+1} \in \text{the goal area} \\ +0, \text{ if } s_{t+1} \in \text{the safe area} \\ -20, \text{if } s_{t+1} \in \text{the super dangerous area} \\ -10, \text{if } s_{t+1} \in \text{the dangerous area} \\ -5, \text{ if } s_{t+1} \in \text{out of the view} \end{cases}$$

It is clear that the visual feature point of the obstacle is pushed toward the goal area without losing the view of the obstacle.

C. *Training of the Q-Learning Controller*

Now a Q-learning controller is developed to determine an optimal control strategy, which will maximize the rewards. The controller will continuously read images from the 3D camera, and check if obstacles are present in the dangerous area of the image plane. Then the Q-learning method will learn and select an optimal action for the robot in the current state using the pre-learned experience from the ABVS simulator. It follows that the simulator is a critical part of the Q-learning controller in executing the learning method. It first computes the Q value for each of the possible states through simulation, by considering each available action in each state until the corresponding Q values converge. These Q values are saved as the experience, in the Q table. When choosing an action, the ε-greedy exploration method is used. Here, the Q-learning controller chooses the action according to the maximum Q value with probability (1- ε), and chooses the action randomly with probability ε . Then, the current image state obtained from 3D camera is compared with the pre-learning states in the Q table, to find the matching state. Finally, the corresponding action is chosen according to the Q table and sent to the low-level actuator of the robot. If the chosen action is a "failure" with regard to obstacle avoidance, the corresponding feedback from the 3D camera will enable the robot to update the current state and trigger a re-learning process. Choosing an action according to the experience will considerably improve the response speed since the computational load is significantly smaller.

V. SIMULATION

Now the obstacle avoidance approach developed in this paper is evaluated in a scenario where a certain local trajectory or global trajectory is given. Suppose that a background image as in Fig. 3(c) has been obtained by using the image processing functions in OpenCV. Let us consider how the robot learns the Q table using the simulated environment built by Visual Studio 2010. The Q-learning algorithm is run for 800 iterations. Fig.5 shows the learning history of the Q-learning controller when it detects an obstacle at the pixel coordinates (320, 225), with the corresponding state $s(0,0,0,0,0,1,0,0,0,0,0,0,0)$.

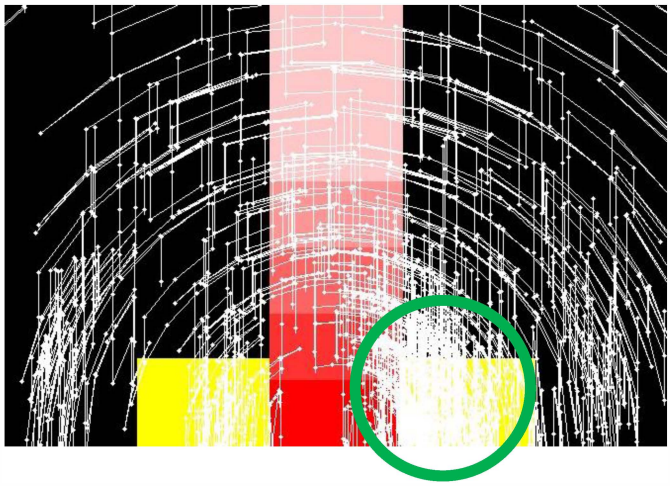

Fig. 5. Learning history of the simulated trajectory generated by the Q-learning controller.

Here the exploration policy is set as $\varepsilon = 0.2$ in order that the robot has a 20% probability in choosing its actions instead of just choosing the action with the maximum Q value. In this manner the robot is encouraged to explore the possibility of finding better trajectories than those it has learned. It is clear that the trajectory density in the goal region (marked by a circle) is the highest. This means that the goal region successfully attracts the robot to itself by setting the reward as 10. Conversely, the dangerous region, which has only a few trajectories compared with other regions, successfully drives the robot away to safer regions by setting the reward as -10 in the dangerous area, and as -50 in the extremely dangerous area.

Fig. 6 shows the convergence history of the Q value of the state-action pair (s,a). It is seen that the Q value of each state-action pair successfully converges.

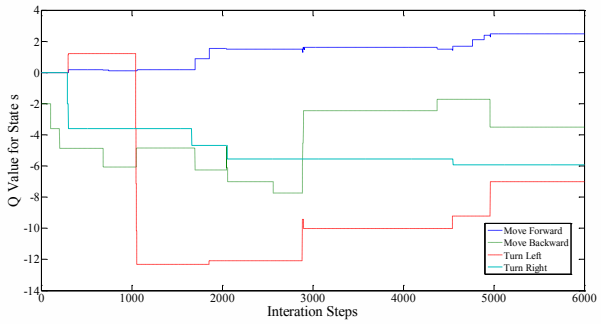

Fig. 6. Convergence history of the Q values.

The Q value converges at several stages (each jump between different convergence states is a result of its ε-greedy exploiting policy). Also, it is seen that the action "move forward" has a higher Q value than the others. This means, given the experience stored in the Q table, the robot will choose the action "move forward" when entering the state $s(0,0,0,0,0,1,0,0,0,0,0,0,0)$ again.

VI. CONCLUSION

This paper developed a hierarchical controller to avoid randomly moving obstacles in autonomous navigation of a robot. The developed method consisted of two parts: a high-level Q-learning controller for choosing an optimal plan for navigation and a low-level, appearance-based visual servo (ABVS) controller for motion execution. The ABVS controller was used to train the robot to avoid obstacles in a simulated state-action scenario. Once the training was done, the experience (Q table) was exploited to choose an optimal motion plan according a ε-greedy maximum Q-value policy. The use of robot learning ability in collision avoidance was a novel feature, under a combined system framework of planning and visual servo control. The approach took advantage of the on-board camera of the robot whose finite field of view was naturally suitable for the Q-learning algorithm. In view of the Q-learning controller, knowledge of obstacle movement and a control law for the ABVS controller were not needed. This was a significant computational advantage. The method was implemented in a simulation system of robot navigation. The results showed that Q-learning successfully converged to an optimal strategy for the robot to establish a proper motion plan.

ACKNOWLEDGMENTS

Funding for the work reported in this paper has come from the Natural Sciences and Engineering Research Council (NSERC) of Canada, Canada Foundation for Innovation (CFI), British Columbia Knowledge Development Fund, and the Canada Research Chair held by C.W. de Silva.

REFERENCES

[1] Ohya, A., Kosaka A. and Kak, A., "Vision-based navigation by a mobile robot with obstacle avoidance using a single-camera vision and ultrasonic sensing," *IEEE Transactions on Robotics and Automation,* , 1998, vol.14, pp. 969–978.

[2] Lapierre, L., Zapata, R. and Lepinay, P., "Combined path following and obstacle avoidance control of a unicycle-type robot," *International Journal of Robotics and Research,* 2007, vol.26, no.4, pp.361-375.

[3] Lee, T.S., Eoh, G.H. Kim J. and Lee, B.H, "Mobile robot navigation with reactive free space estimation," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[4] Chaumette, F. and Hutchinson, S., "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine,* 2006, vol.13, no.4, pp. 82–90.

[5] Allibert, G., Courtial, E. and Tour´e, Y., "Real-time visual predictive controller for image-based trajectory tracking of a mobile robot," *Proceedings of the 17th IFAC World Congres,* 2008.

[6] Becerra, H. M., G. Nicol´as, L. and Sag¨u´es, C., "A Sliding-Mode-Control law for mobile robots based on epipolar Visual servoing From Three Views," *IEEE Transanctions on Robotics*, 2011, vol.27, no.1, pp.175–183.

[7] Cherubini, A. and Chaumette, F., "Visual navigation with obstacle Avoidance," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[8] Cherubini, A. and Chaumette, F., "Visual Navigation of a Mobile Robot with Laser-based Collision Avoidance," *The International Journal of Robotics Research*, 2012, vol.32, no.2, pp.189–205.

[9] Spong, M. W., Hutchinson, S. and Vidyasagar, M., *Robot Modeling and Control,* 2006, Wiley Press, New York.

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