21

DAQL-Enabled Autonomous Vehicle Navigation in Dynamically Changing Environment

Chi Kit Ngai and Nelson H. C. Yung
The University of Hong Kong, Hong Kong, China

1. Introduction

Many autonomous tasks can be considered as having to satisfy multiple goals simultaneously. In particular, Autonomous Vehicle (AV) navigation can be considered as a task having to satisfy at least two goals in an environment. The first goal is to plan a path for an agent to move from an origin to a destination that takes the shortest number of navigation steps. If the environment is static and the destination is stationary, then this shortest path is constant and can be planned in advance if the environment is known a priori, or estimated as the agent explores the environment if it is initially unknown. If the environment or the destination is dynamically changing, then the shortest path is no longer constant. This problem may still be considered as a path planning issue if the environment at each sampled time is known. However, the problem is more appropriately dealt with by incorporating a second goal that aims to avoid collisions between the agent and its neighboring obstacles while executing an overall shortest path strategy towards the destination. The collision avoidance (CA) problem has been well studied in the context of static known or unknown environments (Latombe, 1991; Ge & Cui, 2000; Oriolo et al., 1998; Ye et al., 2003). In the case of dynamic environments (DE) (Stentz, 1994; Stentz, 1995; Yang & Meng, 2003; Minguez & Minguez, 2004; Minguez, 2005), the focus at present is on dynamic environment (DE) that is slowly changing with fairly low obstacle density.

In theory, if the agent samples the environment fast enough, any environment would appear as a static environment and the navigation problem can be solved using existing solutions for static environments. In practice, this condition is difficult to achieve particularly when obstacles are moving at speeds higher than the agent or sampling rate is low. To deal with this situation, an obvious approach is to explicitly consider obstacle motions. Fiorini & Shiller (Fiorini & Shiller, 1998) proposed the concept of Velocity Obstacles that enables obstacle motions between two time steps to be considered in their formulation. Like other similar algorithms (Mucientes e al., 2001; Yamamoto et al., 2001; Feng et al., 2004; Qu et al., 2004), they assumed that objects move in a constant velocity. Shiller et al. (Shiller et al., 2001; Large et al., 2002) further proposed the non-linear velocity obstacle concept which assumes that obstacles can have variable speed. Moreover, they described the obstacles' trajectories using circular approximation. Although it may not always capture the correct movement of obstacles, it is an attempt to predict obstacle motions between two time steps. Similarly, Zhu's hidden Markov model (Zhu, 1991) and Miura's probabilistic model (Miura et al., 1999) also attempted the same. The idea of considering obstacles motion within two time steps explicitly proves to be vital in enhancing the agent's CA ability in reality. Motivated by

this idea, we propose in this chapter a new approach, which incorporates two major features that are not found in solutions for static environments: (1) actions performed by obstacles are taken into account when the agent determines its own action; and (2) reinforcement learning is adopted by the agent to handle destination seeking (DS) and obstacle actions.

Reinforcement Learning (RL) (Sutton & Barto, 1998) aims to find an appropriate mapping from situations to actions in which a certain reward is maximized. It can be defined as a class of problem solving approaches in which the learner (agent) learns through a series of trial-and-error searches and delayed rewards (Sutton & Barto, 1998; Kaelbling, 1993; Kaelbling et al., 1996; Sutton, 1992). The purpose is to maximize not just the immediate reward, but also the cumulative reward in the long run, such that the agent can learn to approximate an optimal behavioral strategy by continuously interacting with the environment. This allows the agent to work in a previously unknown environment by learning about it gradually. In fact, RL has been applied in various CA related problems (Er & Deng, 2005; Huang et al., 2005; Yoon & Sim, 2005) in static environments. For RL to work in a DE containing multiple agents, the consideration of actions of other agents/obstacles in the environment becomes necessary (Littman, 2001). For example, Team Q-learning (QL) (Littman, 2001; Boutilier, 1996) considered the actions of all the agents in a team and focused on the fully cooperative game in which all agents try to maximize a single reward function together. For agents that do not share the same reward function, Claus and Boutilier (Claus & Boutilier, 1998) proposed the used of JAL. Their results showed that by taking into account the actions of another agent, JAL performs somewhat better than the traditional QL. However, JAL depends crucially on the strategy adopted by the other agents and it assumes that other agents maintain the same strategy throughout the game. While this assumption may not be valid, Hu and Wellman proposed Nash Q-learning (Hu & Wellman, 2004) which focuses on a general sum game that the agents are not necessarily working cooperatively. Nash equilibrium is used for the agent to adopt a strategy which is the best response to the other's strategy. This approach requires the agent to learn others Q-value by assuming that the agent can observe other's rewards.

In this chapter, we propose an improved QL method called Double Action Q-Learning (DAQL) (Ngai & Yung, 2005a; Ngai & Yung, 2005b) that similarly considers the agent's own action and other agents' actions simultaneously. Instead of assuming that the rewards of other agents can be observed, we use a probabilistic approach to predict their actions, so that they may work cooperatively, competitively or independently. Based on this, we further develop it into a solution for the two goal navigation problem in a dynamically changing environment, and generalize it for solving multiple goal problems. The solution uses DAQL when it is required to consider the responses of other moving agents/obstacles. If agent action would not cause the destination to move, then QL (Watkins & Dayan, 1992) would suffice for DS. Given two actions from two goals, a proportional goal fusion function is employed to maintain a balance in the final action decision. Extensive simulations of the proposed method in environments with single constant speed obstacle to multiple obstacles at variable speed and directions indicate that the proposed method is able to (1) deal with single obstacle at any speed and directions; (2) deal with two obstacles approaching from different directions; (3) cope with large sensor noise; (4) navigate in high obstacle density and high relative velocity environments. Detailed comparison with the Artificial Potential Field method (Ratering & Gini, 1995) reveals that the proposed method improves path time and the number of collision-free episodes by 20.6% and 23.6% on average, and 27.8% and 115.6% at best, respectively.

The rest of this chapter is organized as follows: Section 2 introduces the concept of the proposed DAQL-enabled reinforcement learning framework. Section 3 describes the implementation method of the proposed framework in solving the autonomous vehicle

navigation problem. Section 4 presents the simulation procedures and results with comparisons with related method. Finally, conclusions are given in Section 5.

2. DAQL-enabled multiple goal reinforcement learning

2.1 General overview

Autonomous navigation is inherently a multiple goal problem involving destination seeking, collision avoidance, lane/wall following and others. Fig. 1 depicts the concept of multiple goal Reinforcement Learning with totally G goals. A multiple-goal scenario can be generalized such that both conventional QL and DAQL can be used for learning depending on the nature of the environment. The individual Q-values are eventually fused to produce a final action. For instance, limit the vehicle navigation problem to two goals: DS and CA. If obstacles and destination are non-stationary, then both goals can be dealt with by DAQL, whereas if they are all stationary, then QL suffice. Here, this general concept is illustrated by assuming that the destination is stationary and the obstacles are mobile. As such, QL is used for DS and DAQL is used for CA.

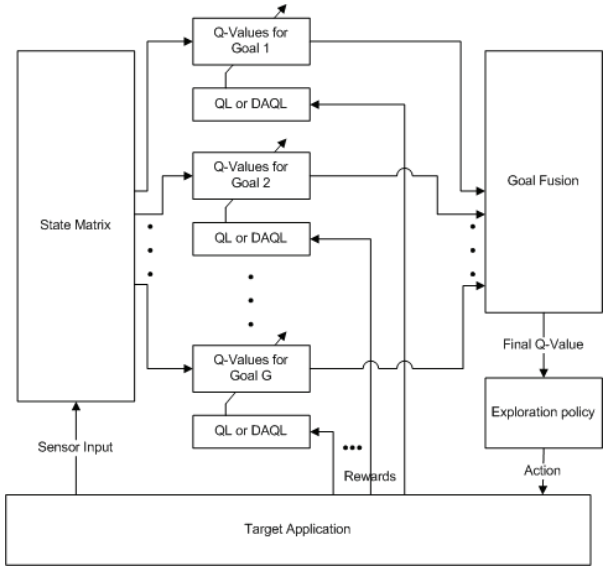


Fig. 1. Concept of multiple goal reinforcement learning.

2.2 Reinforcement learning framework

An effective tool for mapping states (that describe the environment) to actions (that are taken by an agent) and carrying out appropriate optimization (based on a value function) is the Markov Decision Process (MDP) model. It is a model for sequential decision making under uncertainty. In an MDP, the transition probability and the reward function are determined by the current state and the action selected by the agent only (Puterman, 1994). It can be explained by considering a specific time instant of an agent and its environment as depicted in Fig. 2. At each time step t, the agent observes the state $s_t \in S$, where S is the set of possible states, then chooses an action $a_t \in A(s_t)$, where $A(s_t)$ is the set of actions available in s_t,

based on s_t and an exploration policy (e.g. greedy policy). The action causes the environment to change to a new state (s_{t+1}) according to a transition probability, $P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$. At the end of a sampling time T, the environment returns a reward or penalty to the agent according to a reward function, $R^a_{ss'} = E\{r_{t+1} \mid a_t = a, s_t = s, s_{t+1} = s'\}$. The agent then faces a similar situation in the next time instant.

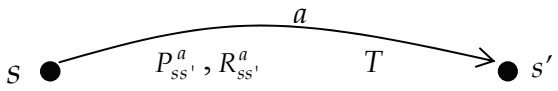


Fig. 2. State diagram of the MDP model given that s_t=s, s_{t+1}=s', and a_t=a.

In RL, the value function is introduced to estimate the value for the agent to be in a given state. It is the expected infinite discounted sum of reward that the agent will gain as follows (Sutton & Barto, 1998):

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

where $E_{\pi}\{\}$ is the expected value when policy π is adopted and R_t is the discounted sum of future rewards; γ is the discounting factor and r_{t+k+1} is the reward (or penalty) received at time (t+k+1). Policy π is a mapping from each state-action pair to the probability $\pi(s,a)$ of taking action a when in state s. To solve the RL task, an optimal policy should be determined that would result in an a_t with the highest expected discounted reward from s to the end of the episode. The optimal value function corresponding to the optimal policy is then achieved by maximizing the value function that represents the expected infinite discounted sum of reward:

$$V^*(s) = \max_{a \in A(s)} \sum_{s'} P^a_{ss'}\left(R^a_{ss'} + \gamma V^{\pi}(s')\right) \tag{2}$$

The corresponding action-value function is given as:

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

and the optimal action-value function is given as:

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

2.3 Q-learning

Q-Learning (Watkins & Dayan, 1992) is one of the efficient methods for solving the RL problem through the action-value function in Eqt. (4). In QL, the agent chooses a_t according to policy π and the Q-values corresponding to state s_t. After performing action a_t in state s_t and making the transition to state s_{t+1}, it receives an immediate reward (or penalty) r_{t+1}. It then updates the Q-values for a_t in s_t using the Q-values of the new state, s_{t+1}, and the reward r_{t+1} as given by the update rule:

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

QL has been proven to converge to optimal action-value with probability one if each action is executed in each state an infinite number of times (Kaelbling et al., 1996; Watkins & Dayan, 1992), and works reasonably well in single agent environment, where the agent is the only object that is able to evoke a state transition.

2.4 Double action Q-learning

In general, it is fair to assume that a DE consists of static obstacles (e.g. walls) and dynamic agents/obstacles. In this case, the assumption that state transition is solely caused by an agent is not exactly appropriate (Littman, 2001; Boutilier, 1995; Claus & Boutilier, 1998). In other words, state transition in a DE may be caused by the action taken by the agent, $a^1{}_t \in A_1(s_t)$, and a collective action taken by the other agents/obstacles, $a^2{}_t \in A_2(s_t)$, where $A_1(s_t)$ and $A_2(s_t)$ are the set of actions available in s_t for the agent and the obstacle in the environment respectively. Fig. 3 depicts a new MDP that reflects this relationship, whereas $a^2{}_t$ describes the action performed by an obstacle. The net state transition at each time step is the result of all the action pairs taken together.

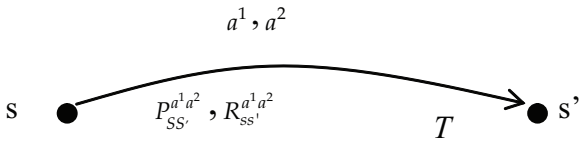


Fig. 3. State diagram of the new MDP model given that $s_t=t$, $s_{t+1}=s'$, $a^1{}_t=a^1$, and $a^2{}_t=a^2$.

The seven parameters of the new MDP are: T, s_t, s_{t+1}, $a^1{}_t$, $a^2{}_t$, $P_{ss'}^{a^1a^2}$ and $R_{ss'}^{a^1a^2}$, where $P_{ss'}^{a^1a^2} = \Pr\{s_{t+1}=s' \mid s_t=s, a_t^1=a^1, a_t^2=a^2\}$ is the transition probability from s to s′, when the agent takes action a^1 and the environment takes action a^2; and $R_{ss'}^{a^1a^2} = E\{r_{t+1} \mid a_t^1=a^1, a_t^2=a^2, s_t=s, s_{t+1}=s'\}$ is the reward received as a result.

In this new model, state changes when either (or both) the agent or the environment has taken its action. To reflect the fact that state transition is now determined by a^1 and a^2, the new value function is formulated below:

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

where $E_{\pi^1,\pi^2}\{\}$ represents the expected value when policy π^1 is adopted by the agent and policy π^2 is adopted by the environment. Similarly, there exists an optimal value function when an optimal policy pair π^1 and π^2 is applied. Although there may be more than one pair, we called all the optimal pairs π^{1*} and π^{2*}. They have the optimal value function $V^*(s)$ defined as:

$$
\begin{aligned}
V^*(s_t) &= \max_{a^1 \in A_1(s), a^2 \in A_2(s)} E_{\pi^{1*},\pi^{2*}} \left\{ R_t \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2 \right\} \\
&= \max_{a^1 a^2} E_{\pi^{1*},\pi^{2*}} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2 \right\} \\
&= \max_{a^1 a^2} E_{\pi^{1*},\pi^{2*}} \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2 \right\} \\
&= \max_{a^1 a^2} E \left\{ r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t^1 = a^1, a_t^2 = a^2 \right\} \\
&= \max_{a^1 a^2} \sum_{s'} P_{ss'}^{a^1 a^2} \left[R_{ss'}^{a^1 a^2} + \gamma V^*(s') \right]
\end{aligned}
\tag{7}
$$

The corresponding optimal action-value function is given as:

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

Using the same technique as QL, the function $Q^*(s_t, a^1_t, a^2_t)$ can be updated continuously that fulfils the purpose of RL. The QL type update rule for the new MDP model is given below:

$$
Q(s_t, a_t^1, a_t^2) \leftarrow Q(s_t, a_t^1, a_t^2) + \alpha \left[r + \gamma \max_{a_{t+1}^1, a_{t+1}^2} Q^*(s_{t+1}, a_{t+1}^1, a_{t+1}^2) - Q(s_t, a_t^1, a_t^2) \right] \tag{9}
$$

Although a^2_t is involved in calculating Eqt. (7), (8) & (9), it is inherently uncontrollable by the agent and therefore maximizing a^2_t in (7) and a^2_{t+1} in (8) & (9) is meaningless. Instead, an approximation to the optimal action-value function by using the observed a^2_{t+1} is found and maximizing Eqt. (8) by a^1_{t+1} subsequently. As such, the new update rule for DAQL is:

$$
Q(s_t, a_t^1, a_t^2) \leftarrow Q(s_t, a_t^1, a_t^2) + \alpha \left[r_{t+1} + \gamma \max_{a_{t+1}^1} Q(s_{t+1}, a_{t+1}^1, a_{t+1}^2) - Q(s_t, a_t^1, a_t^2) \right] \tag{10}
$$

where s_t, a^1_t are known in t, a^2_t , s_{t+1}, and r_{t+1} are known in t+1, and a^2_{t+1} can only be known in t+2. Therefore, the learning is delayed by two time steps when compared with conventional QL, but with a^2_t and a^2_{t+1} appropriately included.

When comparing Eqt. (5) with (10), the difference between DAQL and QL is that action a^2_t has been explicitly specified in the update rule. The optimal value function as a result of maximizing a^1_t only, while a^2_t is considered explicitly but unknown is given below:

$$
\begin{aligned}
&V^{*\prime}(s) = \max_{a^1} E_{\pi^{1*}\pi^2} \left\{ R_t \mid s_t = s, a_t^1 = a^1 \right\} = \max_{a^1} E_{\pi^{1*}\pi^2} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t^1 = a^1 \right\} \\
&= \max_{a^1} E_{\pi^{1*}\pi^2} \left\{ r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} \mid s_t = s, a_t^1 = a^1 \right\} = \max_{a^1} E_{\pi^2} \left\{ r_{t+1} + \gamma V^{*\prime}(s_{t+1}) \mid s_t = s, a_t^1 = a^1 \right\} \\
&= \max_{a^1} \sum_{a^2} \pi^2(s, a^2) \sum_{s'} P_{ss'}^{a^1 a^2} \left[R_{ss'}^{a^1 a^2} + \gamma V^{*\prime}(s') \right]
\end{aligned}
\tag{11}
$$

The corresponding optimal action-value function is:

$$\begin{aligned} Q^*{}'(s,a^1,a^2) &= E_{\pi^2}\left\{ r_{t+1} + \gamma \max_{a^{1'}} Q^*(s_{t+1},a^{1'},a^{2'}) \,|\, s_t = s, a_t^1 = a^1, a_t^2 = a^2 \right\} \\ &= \sum_{s'}\sum_{a^{2'}} \pi^2(s',a^{2'}) P_{ss'}^{a^1 a^2} \left[R_{ss'}^{a^1 a^2} + \gamma \max_{a^{1'}} Q^*{}'(s',a^{1'},a^{2'}) \right] \end{aligned} \tag{12}$$

It can be seen that Eqt. (4) is a special case of Eqt. (12). The DAQL formulation learns the expected Q-values by maximizing the future Q-values with $a^1{}_t$ over actual $a^2{}_{t+1}$ through time iterations. Therefore, if the current state is known and $a^2{}_t$ can be predicted, $a^1{}_t$ can be selected by using proper exploration policy (e.g. greedy policy):

$$a_t^1 = \arg \max_{a_t^1} (Q(s_t, a_t^1, a_t^2)) \tag{13}$$

To predict obstacles' action, an AR model is applied, which allows the calculation of the expected Q-value. In case that other obstacles' actions are not predictable, such as when they move randomly, we assumed that $a^2{}_t$ has equal probability in taking any of the $|A_2(s)|$ actions.

2.5 Goal fusion

The purpose of goal fusion (GF) is to derive a single final action from the actions of different goals. Available methods for the coordination of goals include simple summation or switch of action value function (Uchibe et al., 1996), mixtures of local experts by supervised learning (Jacobs et al., 1991), and multiple model based reinforcement learning (Doya et al., 2002). Here, we adopt a modified summation method to coordinate multiple goals. A GF function based on this is formulated as follow:

$$Q_{final}(a_t^1) = \begin{bmatrix} \dfrac{Q_1(a_t^1)}{\sum_{a^1}\left|Q_1(a^1)\right|} & \dfrac{Q_2(a_t^1)}{\sum_{a^1}\left|Q_2(a^1)\right|} & \cdots & \dfrac{Q_G(a_t^1)}{\sum_{a^1}\left|Q_G(a^1)\right|} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_G \end{bmatrix} \tag{14}$$

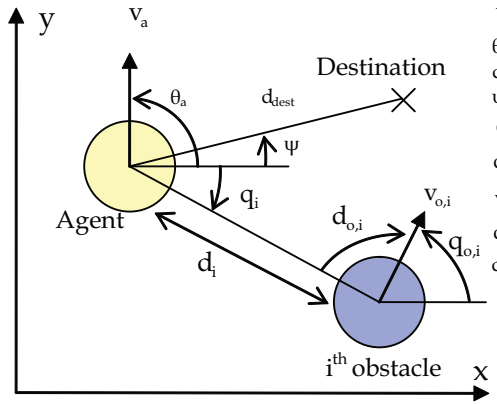
Where $\beta_1+\beta_2+\ldots+\beta_G=1$, G is the number of goals to be achieved and $Q_1(a^1{}_t),\ldots,Q_G(a^1{}_t)$ are the Q-values of the G goals respectively. The importance of the goals with respect to the whole task is represented by the value of β. A more important goal is represented by a larger β while a less important goal is represented by a smaller β.

3. Autonomous navigation through moving obstacles

3.1 Geometrical relations between agent and environment

The control variables of the agent and the ith obstacle at time t are depicted in Fig.4. It is assumed that there are N moving obstacles in the environment and that obstacle distances can be sensed by distance sensors on the agent, which have a minimum and maximum detectable distance of $d_{s,min}$ (10cm) and $d_{s,max}$ (500cm) respectively. Further assume that only M obstacles in the environment can be detected, where $M \leq N$. The location of the ith obstacle is denoted by distance $d_i \in D_o$ where $D_o=[d_{s,min}, d_{s,max}] \subset \Re$ and angle $\theta_i \in \Theta$ where $\Theta=[0,2\pi] \subset \Re$.

We assume that the agent is $d_{dest} \in \Re^+$ away from the destination and is at an angle $\phi \in \Theta$. The four parameters: d_i, θ_i, d_{dest}, and ϕ are quantized into states. The state set for the relative location of the destination is $l_{dest} \in L_{dest}$ where $L_{dest} = \left\{ \left(\tilde{d}_{dest}, \tilde{\varphi} \right) | \, \tilde{d}_{dest} \in D_{dest} \text{ and } \tilde{\varphi} \in \Theta_q \right\}$, D_{dest}={$i \mid i$=0,1,⋯,11} and Θ_q={$j \mid j$=0,1,⋯,15}. The state set for obstacle location is $s_i \in S_i$ where $S_i = \left\{ \left(\tilde{d}_i, \tilde{\theta}_i \right) | \, \tilde{d}_i \in D_q \text{ and } \tilde{\theta}_i \in \Theta_q \right\}$, D_q={$k \mid k$=0,1,…,9} and Θ_q ={$j \mid j$=0,1,…,15}. Quantization is achieved as follows and depicted in Fig. 5 for ϕ and θ_i:



v_a : Velocity of agent
θ_a: Heading angle of agent
d_{dest}: Distance between agent and destination
ψ : Angle between agent and destination
d_i : Distance between agent and the i^{th} obstacle
q_i : Angleof d_i w. r. t. the horizontal axis.
$v_{o,i}$: Velocity of the i^{th} obstacle
$q_{o,i}$: Heading angle the i^{th} obstacle
$d_{o,i}$: Relative heading angle of the i^{th}obstacle w. r. t.d_i.

Fig. 4. Control variables of agent and the i^{th} obstacle.

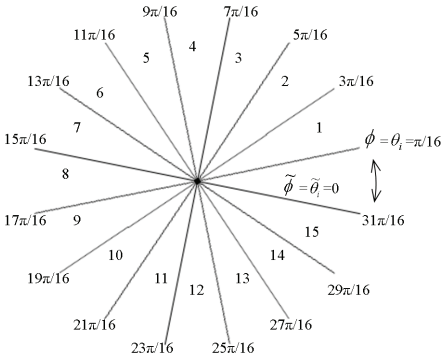


Fig. 5. Quantization of ϕ and θ_i into $\tilde{\phi}$ and $\tilde{\theta}_i$ respectively.

$$\tilde{d}_{dest} = \begin{cases} \lfloor d_{dest} / 5 \rfloor & \text{for } d_{dest} < 10 \\ \lfloor d_{dest} / 10 \rfloor + 1 & \text{for } 10 \le d_{dest} < 100 \\ 11 & \text{for } d_{dest} \ge 100 \end{cases} \tag{15}$$

$$\tilde{\phi} = \begin{cases} \left\lfloor \dfrac{\phi + \pi/16}{\pi/8} \right\rfloor & \text{for } 0 \le \phi < 31\pi/16 \\ 0 & \text{for } 31\pi/16 \le \phi < 2\pi \end{cases} \tag{16}$$

$$\tilde{d}_i = \lfloor d_i / 5 \rfloor \tag{17}$$

$$\tilde{\theta}_i = \begin{cases} \left\lfloor \dfrac{\theta_i + \pi/16}{\pi/8} \right\rfloor & \text{for } 0 \le \theta_i < 31\pi/16 \\ 0 & \text{for } 31\pi/16 \le \theta_i < 2\pi \end{cases} \tag{18}$$

There are altogether 192 states for L_{dest} and 160 states for S_i. The output actions are given by $a \in A$ where $A=\{(|v_a|,\theta_a) \mid |v_a| \in V_a$ and $\theta_a \in \Theta\}$, $V_a=\{m \times v_{max}/5 \mid m=0,1,...,15\}$, $\Theta_a=\{n\pi/8 \mid n=0,1,\cdots,15\}$, and v_{max} is the maximum agent speed. For $|\bar{v}_a|=0$, the agent is at rest despite of θ_a, resulting in only 81 actions. For DAQL, we assume that obstacles have speed $v_o \in \Re^+$ and heading angle $\theta_o \in \Theta$. They are quantized to $a^2{}_i \in A_o$ where $A_o = \left\{ (\tilde{v}_o, \tilde{\theta}_o) \mid \tilde{v}_o \in V_q \text{ and } \tilde{\theta}_o \in \Theta_q \right\}$, $V_q=\{l \mid l=0,1,...,10\}$, and $\Theta_q=\{j \mid j=0,1,\cdots,15\}$. Quantization is achieved as follows:

$$\tilde{v}_o = \begin{cases} \lfloor v_o + 5 \rfloor & \text{for } 0 \le v_o < 105 \\ 100 & \text{for } v_o \ge 105 \end{cases} \tag{19}$$

$$\tilde{\theta}_o = \left\lfloor \frac{\theta_o}{\pi/8} \right\rfloor \tag{20}$$

where there are altogether 161 actions for each obstacle as observed by the agent. The concept of factored MDP (Boutilier et al., 2000; Guestrin et al. 2001) can be applied if necessary to reduce the number of states required.

3.2 Destination seeking

For convenience, destination is assumed stationary here, otherwise actions performed by the moving destination may be considered as in the case of obstacles, which the same DAQL formulation applies.

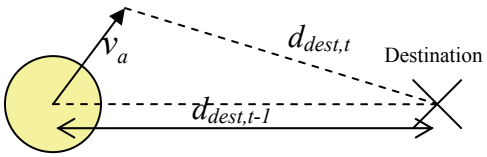


Fig. 6. Change in d_{dest} from t-1 to t.

The purpose of using reinforcement learning in destination seeking is for the agent to learn the limitation of the underlying vehicle mechanics such as limited acceleration and deceleration. The crux of the QL formulation for DS is that the agent is punished if its trajectory towards the destination contains more steps than necessary. With reference to Fig. 6, let us define $\Delta d_{dest}=d_{dest,t-1}-d_{dest,t}$, where $d_{dest,t-1}$ is the distance between the agent and destination at t-1, $d_{dest,t}$ is the distance at t; and the agent travels at v_a from t-1 to t. If the agent performs a shortest path maneuver, then $|v_a|T=\Delta d_{dest}$, otherwise $|v_a|T>\Delta d_{dest}$ and the worst case is when the agent has moved away from the destination, i.e., $\Delta d_{dest}=-|v_a|T$. Let us define d_{extra} as:

$$d_{extra} = |v_a|T - \Delta d_{dest} \tag{21}$$

where $0 \le d_{extra} \le 2|v_a|T$. The normalized reward function of the agent is thus defined as:

$$r_{DS,t} = (\Delta d_{dest} - d_{extra}) / (v_{a,\max} T) \tag{22}$$

where $-3 \le r_{DS,t} \le 1$. In Eqt. (22), d_{extra} is a penalty to the agent in order to ensure that it follows the shortest path to travel to the destination. The reward function is further shifted to $1 \le r_{DS,t} \le 0$ by $r_{DS,t} \leftarrow (r_{DS,t} - 1)/4$, so that the Q-values calculated are in line with those from CA. By using the QL update rule, the agent can learn to use the most appropriate action in the current state to reach the destination using the most direct path, as depicted in Fig. 7.

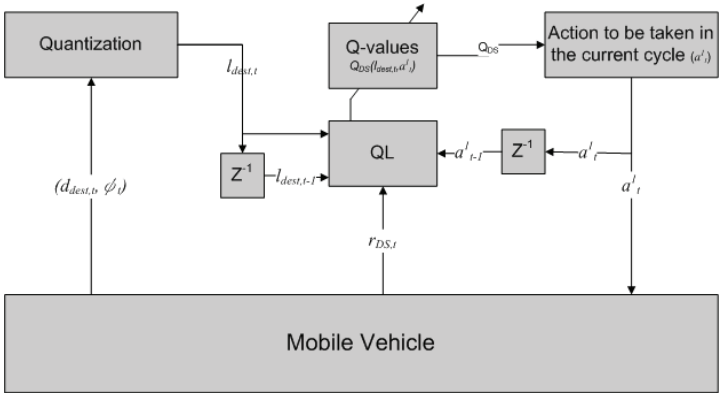


Fig. 7. QL for destination seeking.

3.3 Collision avoidance

Given multiple mobile obstacles in the environment, DAQL is most applicable here. The reward function adopted by DAQL represents punishment (-1) to the agent when collision occurs:

$$r_{CA,i,t} = \begin{cases} 0 & \text{if no collision occured} \\ -1 & \text{if collision occured} \end{cases} \tag{23}$$

When $r_{CA,i,t}$ is available, the agent uses the DAQL update rule to learn CA, as depicted in Fig. 8. Given obstacles' actions in two time steps (t-2 & t-1), the agent updates its Q-values ($q_i(s_{i,t}, a^1_t, a^2_{i,t})$) at t. If there are M obstacles that are detectable by the agent, the DAQL update rule is applied M times and the results are combined based the parallel learning concept introduced by Laurent & Piat (Laurent & Piat, 2001; Laurent & Piat, 2002). Their proposal of taking the sum of all the Q-values from all the obstacles is used, as oppose to taking the maximum Q-value over all the obstacles, as given in the following:

$$Q_{CA}(a^1_t) = \sum_i q_i(s_{i,t}, a^1_t, a^2_{i,t}) \tag{24}$$

where $Q_{CA}(a^1_t)$ is the overall Q-value set for the entire obstacle population when the agent takes a^1_t; $q_i(s_{i,t}, a^1_t, a^2_{i,t})$ is the Q-value set due to the ith obstacle; $s_{i,t}$ is the state of the ith obstacle observed by the agent at time t; and $a^2_{i,t}$ is the action performed by the ith obstacle at t. Since all M obstacles share a single set of Q-values, the Q-values are updated M times in one time step. As a^2_t is not known at t, it has to be predicted, which can be treated independently

from RL, i.e. the agent predicts from the environment's historical information, or it can be based on concepts (rules learn from examples) and instances (pools of examples). To incorporate the predicted a^2_t, Eqt. (24) is modified as follows:

$$q_i(a^1_t) = \sum_{a^2_{i,t}} p_{a^2_{i,t}} q_i(s_{i,t}, a^1_t, a^2_{i,t}) \tag{25}$$

$$Q_{CA}(a^1_t) = \sum_i q_i(a^1_t) \tag{26}$$

where $p_{a^2_{i,t}}$ is the probability that the environment takes action $a^2_{i,t}$. The expected value of the overall Q-value is obtained by summing the product of the Q-value of each obstacle when they take action $a^2_{i,t}$ with their probability of occurrence. The combined Q-value for the entire DE, $Q_{CA}(a^1_t)$, is the summation of Q-values of each obstacle.

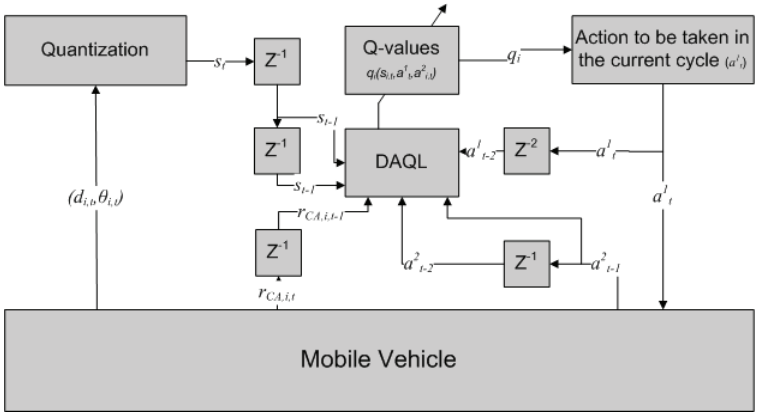


Fig. 8. DAQL for single obstacle.

3.4 Prediction

To predict $a^2_{i,t}$, a linear prediction technique based on the autoregressive (AR) model is adopted. We assume that the accelerations of obstacles are slowly changing in the time interval T between two time steps. A 1st order AR model (Kehtarnavaz & Li, 1988; Ye, 1999) is used to model the acceleration $a_i(t)$:

$$a_i(t) = B_{i,t} a_i(t-1) + e(t) \tag{27}$$

where $e(t)$ is the prediction error and $B_{i,t}$ is a time-dependent coefficient and is estimated adaptively according to the new distance measurements. The acceleration is thus approximated by a combination of velocity and position representations:

$$\begin{aligned} a_i(t) &= \frac{1}{T}[v_i(t) - v_i(t-1)] \\ &= \frac{1}{T^2}\{[r_i(t) - r_i(t-1)] - [r_i(t-1) - r_i(t-2)]\} \\ &= \frac{1}{T^2}[r_i(t) - 2r_i(t-1) + r_i(t-2)] \end{aligned} \tag{28}$$

where $v_i(t)$ and $r_i(t)$ are the velocity and position of the ith obstacle at time step t, respectively. Substituting Eqt. (28) into (27) gives a 3rd order AR model:

$$r_i(t) - (2 + B_{i,t})r_i(t-1) + (2B_{i,t} + 1)r_i(t-2) - B_{i,t}r_i(t-3) = e(t) \tag{29}$$

Therefore, the next position of the ith obstacle at time t+1 can be predicted by the following equation if the coefficient $B_{i,t}$ is known:

$$\hat{r}_i(t+1) = r_i(t) + v_k(t)T + \hat{B}_{i,t}a_k(t)T^2 \tag{30}$$

where $\hat{B}_{i,t}$ is time-dependent and is updated by the adaptive algorithm in (Shensa, 1981). The coefficient $\hat{B}_{i,t}$ can thus be determined by the following equations:

$$\hat{B}_{i,t} = \Delta_{i,t}R_{i,t}^{-1} \tag{31}$$

$$\Delta_{i,t} = \lambda\Delta_{i,t-1} + a_k(t)a_k^T(t-1) \tag{32}$$

$$R_{i,t} = \lambda R_{i,t-1} + a_k(t-1)a_k^T(t-1) \tag{33}$$

where $0<\lambda\leq 1$ is a weighting factor close to 1. Since $a_k(t)$, $\Delta_{k,t}$, $R_{k,t}$ and λ are all known, $\hat{B}_{i,t}$ can be predicted and thus $\hat{r}_i(t+1)$ can be predicted, from which the action performed by the ith obstacle at t can be predicted and the probability $p_{a_{i,t}^2}$ can be determined. A probability of 1 is given to the predicted action and 0 is given to all other actions.

3.5 Fusion of DS and CA

Given two sets of Q values from DS and CA, they are combined by using β - a parameter that varies between 0 and 1, to balance the influence of the two goals, as given in Eqt. (34), where $Q_{CA}(a^1_t)$ and $Q_{DS}(a^1_t)$ are normalized.

$$Q_{final}(a_t^1) = (1-\beta)\frac{Q_{CA}(a_t^1)}{\sum_{a^1}\left|Q_{CA}(a^1)\right|} + \beta\frac{Q_{DS}(a_t^1)}{\sum_{a^1}\left|Q_{DS}(a^1)\right|} \tag{34}$$

For β closer to 1, $Q_{final}(a^1_t)$ is biased towards DS, giving the agent better DS performance but poorer CA performance. Conversely, for β closer to 0, $Q_{final}(a^1_t)$ is biased towards CA, giving the agent poorer DS performance but better CA performance. The final decision of the agent is made by using the ε-greedy policy as shown in Eqt. (35). Fig. 9 and Fig. 10 depict the functional diagram and pseudo code of the proposed method respectively for multiple obstacles.

$$a_t^1 = \begin{cases} \arg\max_{a_t^1} Q_{final}(a_t^1) & \text{with probability } 1\text{-}\varepsilon \\ \text{random} & \text{with probability } \varepsilon \end{cases} \tag{35}$$

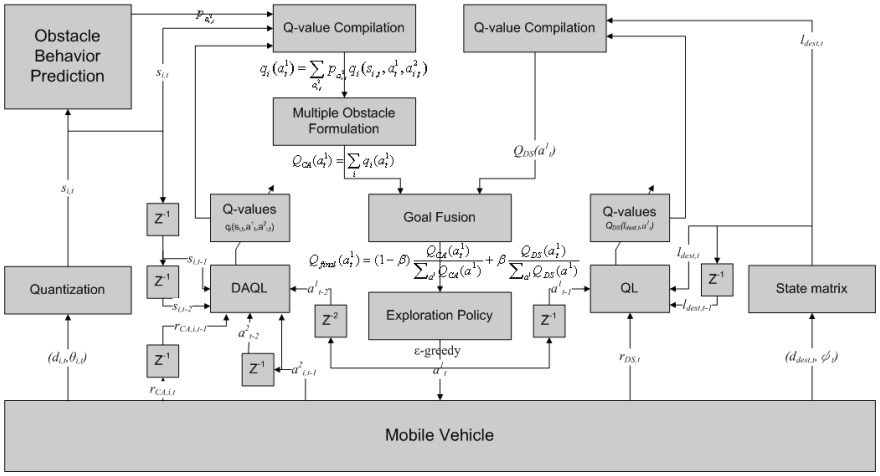


Fig. 9. Functional diagram of the proposed method.

Initialize $q_i(s,a^1,a^2)$ arbitrarily
Repeat (for each episode)
Initialize a^1, a^2, r, l_{dest}, s_i, and s_i'
Repeat (for each step of episode):

Get $p_{a_i^{2'}}$ by predicting the action $a^2{}_i$' that will be performed by the ith obstacle

Calculate $q_i(a^{1\prime}) = \sum_{a_i^{2\prime}} p_{a_i^{2\prime}} q_i(s_i, a^{1\prime}, a_i^{2\prime})$

Calculate $Q_{CA}(a^{1\prime}) = \sum_i q_i(a^{1\prime})$

Determine $Q_{DS}(a^{1\prime})$

Calculate $Q_{final}(a^{1\prime}) = (1-\beta)\frac{Q_{CA}(a^{1\prime})}{\sum_{a^1}\left|Q_{CA}(a^1)\right|} + \beta\frac{Q_{DS}(a^{1\prime})}{\sum_{a^1}\left|Q_{DS}(a^1)\right|}$

Choose $a^{1\prime}$ from s using policy derived from $Q_{final}(a^{1\prime})$

Take action $a^{1\prime}$ using ε-greedy exploration policy
Observe the new state l_{dest}', s_i'', r' and $a^2{}_i$'
Determine the action $a^2{}_i$ that have been performed by the ith obstacle

$$q_i(s_i,a^1,a_i^2) \leftarrow q_i(s_i,a^1,a_i^2) + \alpha\left[r_{CA,i} + \gamma \max_{a^{1\prime}} q_i(s',a^{1\prime},a_i^{2\prime}) - q_i(s,a^1,a_i^2)\right]$$

$$Q_{DS}(l_{dest},a) \leftarrow Q_{DS}(l_{dest},a) + \alpha\left[r_{DS} + \gamma \max_{a'} Q_{DS}(l_{dest}{}',a') - Q_{DS}(l_{dest},a)\right]$$

$l_{dest} \leftarrow l_{dest}$', $s_i \leftarrow s_i$', s_i' $\leftarrow s_i$'', $a^1 \leftarrow a^{1\prime}$, $a^2 \leftarrow a^{2\prime}$, $r_{CA,i} \leftarrow r_{CA,i}$'
until l_{dest}' is terminal

Fig. 10. Pseudo code of the proposed method.

4. Simulations and results

4.1 Simulation conditions

In this simulation, length has unit of cm and time has unit of second. The agent and obstacles are assumed to be circular with diameter of 100 cm, and the environment is 2500×2500 (cm²), as depicted in Fig. 11. The numbers in the figure represent the location of the agents and targets in every 10 s. The maximum speed of the agent ($v_{a,max}$) is assumed to be 50 cm/s, with a maximum acceleration and deceleration of 20 cm/s^{-2}. The agent is required to start from rest, and decelerate to 0 cm/s when it reaches the destination.

To acquire environmental information, a sensor simulator has been implemented to measure distances between agent and obstacles. The sensor simulator can produce either accurate or erratic distance measurements of up to 500 cm, at T interval (typically 1s) to simulate practical sensor limitations. The other parameters are set as follows: α for both DS and CA learning is set to 0.6 for faster update of Q-values; γ is set to 0.9 for CA and 0.1 for DS; β of 0.1 is set to have strong bias towards CA, in the expense of longer path; ε is set 0.5 for DS and 0.1 for CA.

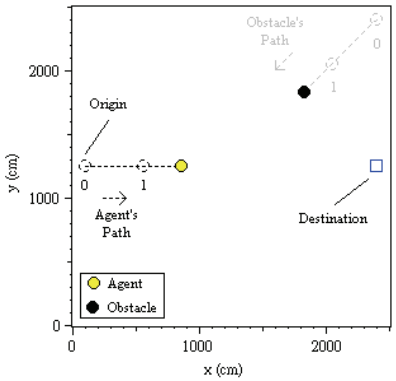


Fig. 11. Simulation environment.

4.2 Factors affecting navigation performance

The factors that affect an agent's performance in a DE are: relative velocity, relative heading angle; separation; and obstacle density. They define the bounds within which the agent can navigate without collision from an origin to a destination. The relative velocity of obstacle as observed by the agent can be defined as: $\vec{v}_{r,i} = \vec{v}_{o,i,\max} - \vec{v}_{a,\max}$, where $\vec{v}_{o,i,\max}$ and $\vec{v}_{a,\max}$ are velocity vectors of the ith obstacle (O_i) and agent (A) respectively. In essence, $\vec{v}_{r,i}$ represents the rate of change in separation between A and O_i. Given , heading angle of O_i w.r.t. the line joining the centres of O_i and A, and δ_a, heading angle of A as depicted in Fig.12, relative heading angle is defined as $\psi=\pi-(\delta_a+\delta_{o,i})$. It should be noted that ψ equals π when A and O_i are moving towards each other, $-\pi$ when A and O_i are moving away from each other, and 0 when both are moving in the same direction. Let d_i be the separation between A and O_i. It determines the priority A would adopt when considering O_i among other obstacles. If $d_{s,max}$ is the maximum sensor measurement range of A, and if $d_{s,max}<d_i$ then O_i simple does not exist from A's point of view. Obstacle density can be defined as $D=N\pi r_o^2/(A_{env}-\pi r_a^2)$, where N

is the number of obstacle in the environment and A_{env} is the area of the closed environment. We also assume that the obstacles are identical and have a radius of r_o, and A has a radius of r_a. Given A_{env}=25002, r_o=r_a=50, D=0.00125N.

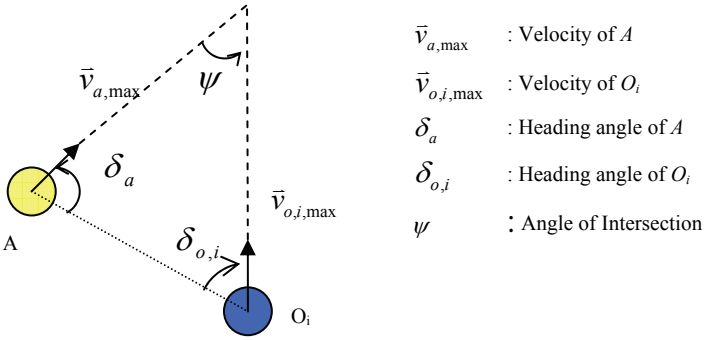


Fig. 12. Heading angles of O_i and A.

4.3 Training for destination seeking

First, the agent was trained by randomly generated origin and destination (O-D) pairs in the environment without obstacles, where each training episode consisted of the agent successfully travelled from O to D. Second, 100 episodes were used to train the agent to obtain a set of Q-values and another 100 episodes of different O-D pairs were used to evaluate the path length versus the shortest path based on the trained Q-values without learning and exploration. Step 2 was repeated 100 times to get an average performance over 10,000 episodes. Fig.13 depicts the mean % difference between the actual and the shortest paths.

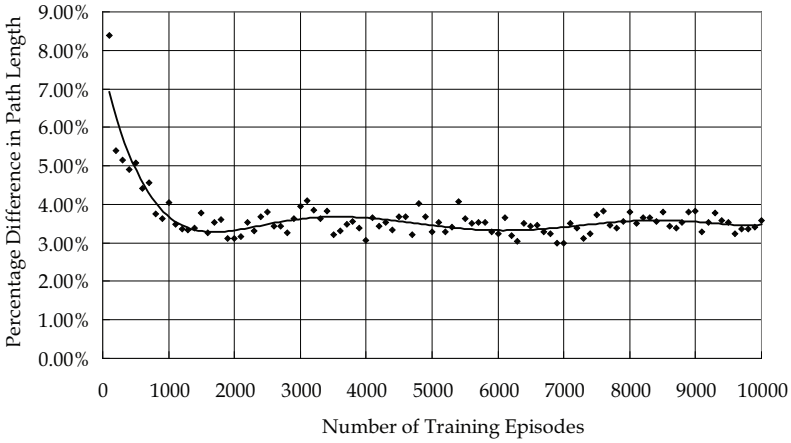


Fig. 13. Percentage difference in path length.

It can be seen that given sufficient training, the agent achieves a path difference of 3-4%. This is so because of the discrete actions the agent adopted in the simulation. In Fig.13, the data are curve fitted with a 6th order polynomial (solid line), from which a cost function is applied to

determine the optimal number of training required. The cost function is defined as $C=f(x)\times\ln(E)$ and plotted in Fig. 14, where $f(x)$ is the polynomial function for the mean % difference and E is the number of episodes. From Fig. 14, minimum cost is achieved when the number of training episodes is around 1500. The Q-values for DS that correspond to this are used in all subsequent simulations.

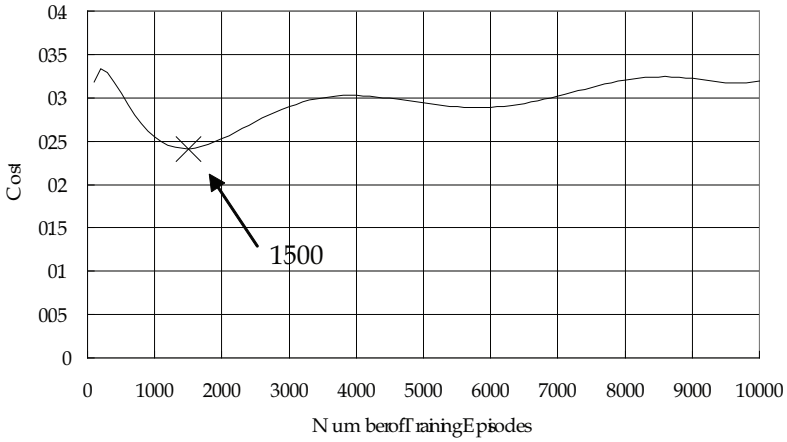


Fig. 14. Cost function.

4.4 Training for collision avoidance

For different environmental factors, we trained the agent with Q-values for CA set to zeros initially for 10000 episodes in each case. After training, simulation results are obtained by allowing the agent to maneuver in an environment with the same set of environmental factors without learning and exploration. When obstacles are present, the agent travels between a fixed OD pair. The agent learnt from the rewards or punishments when it interacted with the obstacle. When the agent reached the destination, an episode was terminated and the agent and obstacle were returned to their origins for the next episode.

Furthermore, to illustrate the behavior of the agent in a more complex environment which involves multiple sets of different environmental factors at the same time, environments with randomly moving obstacles are constructed. Q-values for CA are set to zeros initially and the agent is trained for 10000 episodes in each test case. After training, simulation results are obtained by allowing the agent to maneuver in the same environment without learning ability and exploration. In each training episode, the agent was required to travel to a fixed destination from a fixed origin through the crowd of randomly moving obstacles which were randomly placed in the environment, and the termination condition was the same as before.

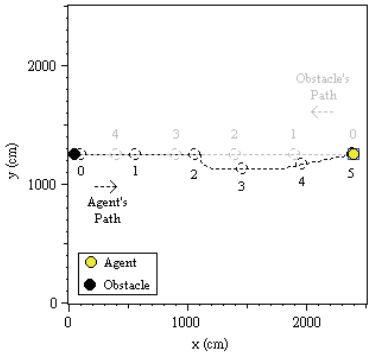
4.5 Obstacles at constant velocity

This simulation investigates how the agent reacts to one or more obstacles at constant velocity with an initial separation of larger than 500 cm. The AR model in Section 3.4 was used for obstacle action prediction. For one obstacle, two v_o values and two ψ were considered: 50 cm/s and 100 cm/s; and π and $¾\pi$. The simulation was repeated for v_o=50

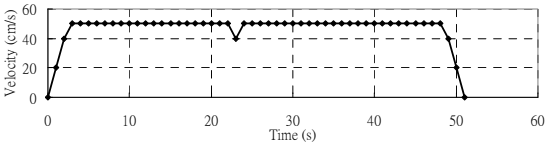
cm/s when two obstacles were present at different heading angles. It was also repeated for a group of obstacles having the same heading angle. These cases are tabulated in Tables 1 to 3.

Case	ψ (rad)	v_o (cm/s)	$\left\|\bar{v}_{r,i}\right\|$ (cm/s)
A	π	50	100
B	π	100	150
C	$3\pi/4$	50	92.39
D	$3\pi/4$	100	139.90

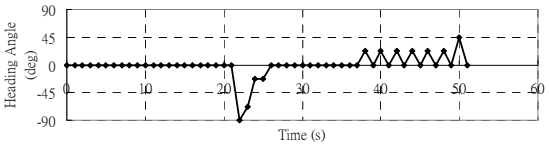
Table 1. Summary of simulation parameters with ONE obstacle.



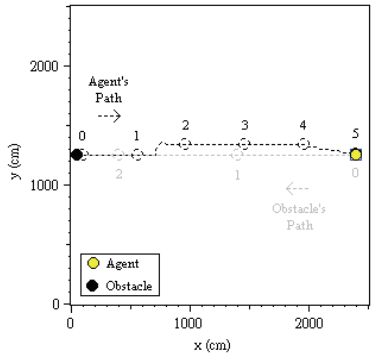
(a) Path (Case A)



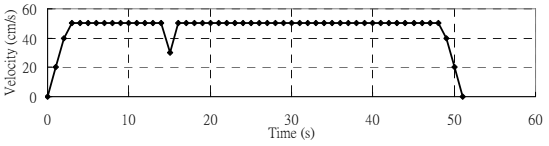
(b) Velocity profile (Case A)



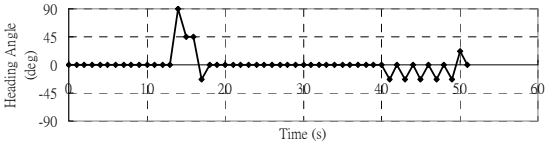
(c) Heading angle profile (Case A)



(d) Path (Case B)



(e) Velocity profile (Case B)

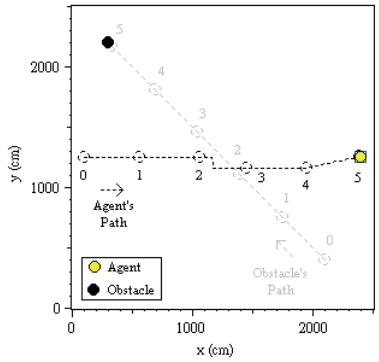


(f) Heading angle profile (Case B)

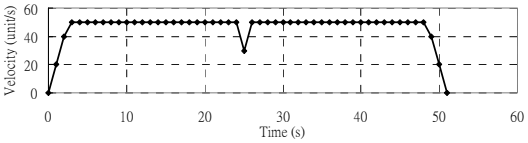
Fig. 15. Simulation results of Cases A and B.

1. Cases A and B: The obstacle moved directly towards the agent at different velocities respectively, as depicted in Fig. 15. For Case A, the obstacle moved at the same speed as the agent. The agent maintained at a maximum speed until the obstacle was within range. It responded appropriately as seen from its Velocity and Heading angle profiles. The agent responded with a gradual change in heading angle to avoid collision. It remained at the changed course for a while before converging to the destination. For Case B, as the obstacle

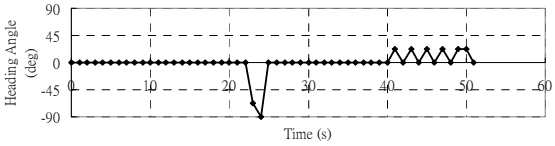
moved faster than the agent, the agent responded earlier with a larger change in velocity. As the CA event ended faster, the agent in Case B reached the destination earlier.



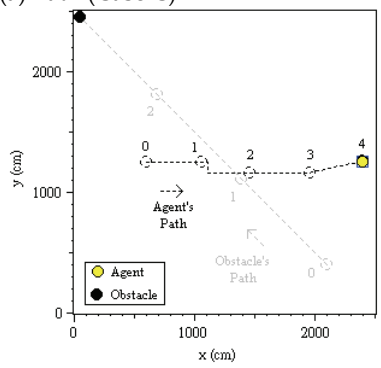
(a) Path (Case C)



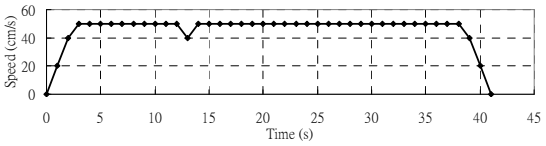
(b) Velocity profile (Case C)



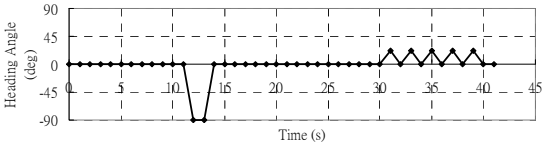
(c) Heading angle profile (Case C)



(d) Path (Case D)



(e) Velocity profile (Case D)



(f) Heading angle profile (Case D)

Fig. 16. Simulation results of cases C and D.

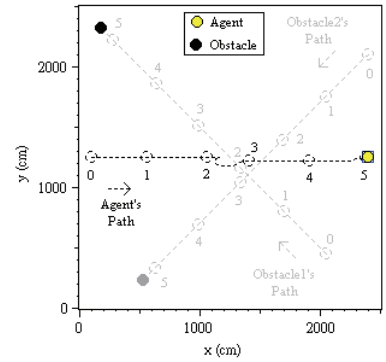
2. Cases C and D: The obstacle crossed path with the agent at an angle of ¾π, as depicted in Fig. 16. For Case C, when obstacle speed is the same as the agent, the agent moved to the right slightly to let the obstacle pass. For Case D, the agent responded earlier and also decided to let the obstacle passed first. As the obstacle moved faster in this case, the velocity and heading angle changes of the agent were larger.

Case		ψ (rad)	v_o (cm/s)	$\lvert\vec{v}_{r,i}\rvert$ (cm/s)
E	Obstacle 1	$3\pi/4$	50	92.39
	Obstacle 2	$3\pi/4$	50	92.39
F	Obstacle 1	$\pi/2$	50	70.71
	Obstacle 2	$\pi/2$	50	70.71

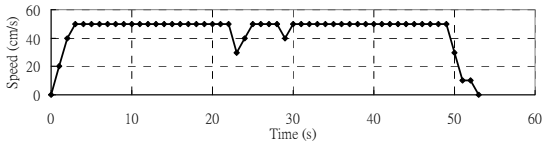
Table 2. Summary of simulation parameters with TWO obstacles.

3. Cases E and F: To deal with two obstacles simultaneously. The obstacles moved at speed v_o=50 cm/s in both cases, but at different heading angles. Case E, as depicted in Fig. 17(a-c),

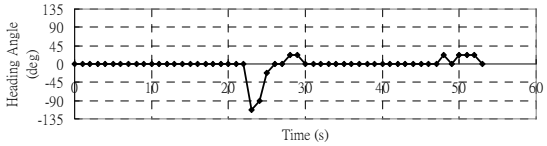
consists of two obstacles moved at an intersecting angle of ¾π with respect to the agent. As can be seen from its V and H profiles, there are two responses: one at t=23s when Obstacle 1 approached the agent first, and one at t=29s when Obstacle 2 followed. The speed changes in both responses were minor, while the agent stepped backward in the first instance to avoid collision. For Case F, two obstacles moved perpendicularly to the agent as depicted in Fig. 17(d-f). There were two distinct responses (at t=9s and 22s), both of which required slowing down and change in heading angle to let the obstacle pass first.



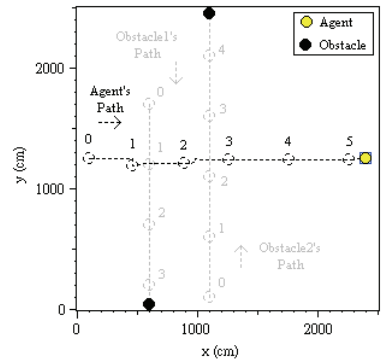
(a) Path (Case E)



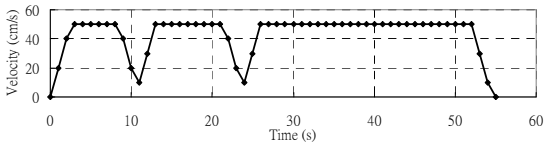
(b) Velocity profile (Case E)



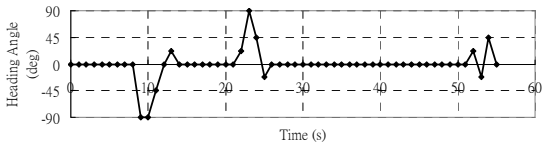
(c) Heading angle profile (Case E)



(d) Path (Case F)



(e) Velocity profile (Case F)



(f) Heading angle profile (Case F)

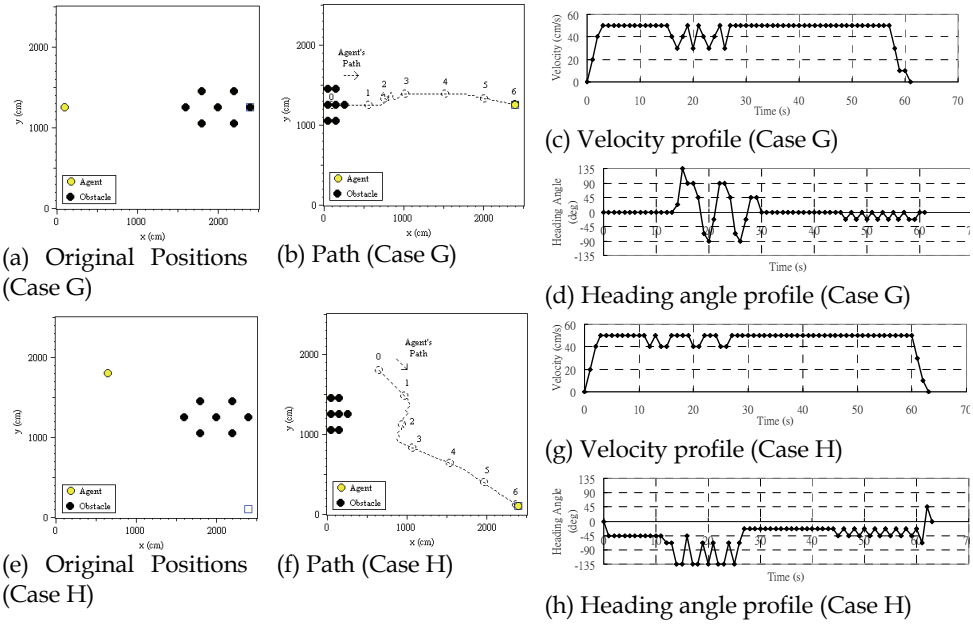
Fig. 17. Simulation results of cases E and F.

Case	ψ (rad)	v_o (cm/s)	$\lvert\vec{v}_{i,r}\rvert$ (cm/s)
G	π	50	100
H	$3\pi/4$	50	92.39

Table 3. Summary of simulation parameters with a GROUP of obstacles.

4. Cases G and H: To deal with a larger number of obstacles in the DE. In Case G, seven obstacles moved in a cluster towards the agent at v_o=50 cm/s. From the path diagram as depicted in Fig. 18(b), as the obstacles were well apart, the agent found no difficulty in navigating through them, as shown in its V and H profiles. For Case H, the cluster of seven

obstacles moved at an angle of ¾π with respect to the agent. Again, the agent navigated through the cluster appropriately, without collision.



(a) Original Positions (Case G)
(b) Path (Case G)
(c) Velocity profile (Case G)
(d) Heading angle profile (Case G)
(e) Original Positions (Case H)
(f) Path (Case H)
(g) Velocity profile (Case H)
(h) Heading angle profile (Case H)

Fig. 18. Simulation results of cases G and H.

4.6 Obstacles at variable velocity

The objective of this simulation is to study agent behavior in handling a simple randomly changing environment. In Cases I and J, a single obstacle moved at varying velocity directly towards the agent ($\psi=\pi$). The obstacle's velocity ranges are 0-50 cm/s and 0-100 cm/s respectively, in step of 10 cm/s. The agent was evaluated over 1,000 episodes in the same environment in each case. A summary of the two cases is given in Table 4.

Case	ψ .rad)	v_o (cm/s)	$\left\|\vec{v}_{r,i}\right\|$.(cm/s)	Number of collision-free Episodes	Mean Path Time (s)
I		0-50	50-100	976	62.97
J		0-100	50-150	957	59.32

Table 4. Simulation parameters with ONE obstacle at random speed.

The results show that for Case I, the proportion of collision-free episodes is 97.6% and a mean path time of 62.97s. A collision-free episode is one that the agent travels to the destination without causing any collision. When compared with the shortest path time (50s), the agent used an extra of 12.97s more. For Case J, the obstacles moved faster in a wider range. As a result, the number of collision-free episodes was reduced to 95.7%, but the mean path time was also reduced to 59.32s. This can be explained as because of the faster moving obstacles, the agent experienced more collisions, but managed less convoluted paths.

4.7 Inaccurate sensor measurement

In this simulation, we investigate how the proposed method tolerates inaccuracy in sensor measurements. As in Cases A & B at three different speeds (v_o=10, 50 or 100 cm/s), the output of the sensor simulator was deliberately corrupted by a Gaussian noise function that has a mean (μ) of $\mu=d_i$ and standard deviation (σ) of $n\times\mu$ where n=0, 0.1, 0.2, 0.3, 0.4, 0.5, and 0.6 (Ye et al., 2003). For each set of n and v_o, Q-values for CA are set to zeros initially and the agent was trained for 10,000 episodes. After training, and the agent was evaluated in the same environment for 1,000 times with different n and v_o. Table 5 depicts the simulation summary.

n	v_o = 10 cm/s ($\lvert\vec{v}_{r,i}\rvert$ = 60 cm/s)		v_o = 50 cm/s ($\lvert\vec{v}_{r,i}\rvert$ = 100 cm/s)		v_o = 100 cm/s ($\lvert\vec{v}_{r,i}\rvert$ = 150 cm/s)	
	Collision-free Episodes	Mean Path Time (s)	Collision-free Episodes	Mean Path Time (s)	Collision-free Episodes	Mean Path Time (s)
0	1000	57	1000	57.00	1000	53.00
0.1	1000	56.79	1000	54.43	1000	54.71
0.2	999	55.40	987	56.05	957	59.89
0.3	979	62.37	996	58.45	987	61.49
0.4	997	65.55	979	58.33	725	80.45
0.5	991	57.52	989	59.04	816	71.31
0.6	995	61.93	954	65.58	576	92.01

Table 5. Robustness to sensor noise.

From Table 5, for n<0.2, none of the obstacle speed would cause collision. For n≥0.2, collision began to appear. At low speed, the number of collisions can be kept small with a worst case of 2.1%. For $|\vec{v}_o|$ =50 cm/s, the number of collision-free episodes was reduced to 95.4% at n=0.6. For $|\vec{v}_o|$ =100 cm/s, it went down to 57.6%, or almost half of the episodes have collisions. This is logical as slow obstacles are easier to avoid compared with fast obstacles, and inaccurate sensor measurements make it harder to avoid collision.

For mean path time, it generally increases when n increases, although minima appear at n=0.2 for low speed, n=0.1 for medium speed and n=0 for high speed. As in Case A, the mean path time is longer when obstacle speed is low because of more convoluted paths. As n increases, the agent learnt to respond earlier to such inaccuracy and resulted in shorter paths. However, for larger n, the agent travels extra steps in order to cope with the large sensor error, which resulted in even longer path. The same applies when obstacle speed is relatively higher, except that the minima appear when n is smaller because the agent responded earlier in this case.

4.8 Randomly moving obstacles and performance comparison

The purpose of this simulation is to evaluate the proposed method in an environment with up to 50 moving obstacles, and compare it against another navigation method that is designed to work in such a complex environment. Obviously, those that work on static environment (Pimenta et al., 2006; Belkhouche et al. 2006; Jing et al. 2006), those that consider relatively simple cases with very low obstacle density (Soo et al. 2005; Kunwar & Benhabib, 2006), or those that assume perfect communication among agents, e.g. robot soccering (Bruce & Veloso, 2006), are unsuitable. A suitable candidate is the artificial potential field method proposed by Ratering & Gini (R&G) (Ratering & Gini, 1995), which was simulated in a relatively complex environment with high density of multiple moving

obstacles. To enable the comparison, obstacles size was reduced to 20 cm in diameter, and the obstacles were placed and moved randomly in speed and direction, as in (Ratering & Gini, 1995). The origin and destination of the agent were located at the lower left hand corner and upper right corner of the environment, respectively. Since the obstacles moved randomly, the prediction was not used in the proposed method. Different obstacle density D and obstacle velocity v_o were studied, and results are tabulated in Table 6. Each result shown in the table was derived from 100 episodes after training. The R&G method used static potential filed and dynamic potential fields to handle static and moving obstacles respectively, and their results are also depicted in Table VI.

Obstacle speed v_o (cm/s)	No. of obstacles	Obstacle Density D	$\left\|\bar{v}_{r,i}\right\|$ (cm/s)	**Average path time (s)**	St. dev. path time	**No. of collision-free episodes**	Average no. of collisions	St. dev. No. of collisions
10	10	0.0005	40-60	**68.25 (79.07)**	16.16 (13.40)	**99 (99)**	0.01 (0.02)	0.1 (0.20)
10	20	0.001	40-60	**80.1 (93.92)**	50.27 (20.93)	**100 (95)**	0 (0.06)	0 (0.28)
10	30	0.0015	40-60	**92.71 (110.19)**	59.19 (27.35)	**97 (98)**	0.11 (0.02)	0.83 (0.14)
10	40	0.002	40-60	**99.24 (126.23)**	53.56 (34.25)	**95 (92)**	0.07 (0.09)	0.33 (0.32)
10	50	0.0025	40-60	**111.55 (135.06)**	58.39 (41.06)	**94 (82)**	0.15 (0.25)	0.63 (0.61)
30	10	0.0005	20-80	**68.58 (80.75)**	7.84 (11.98)	**99 (99)**	0.01 (0.01)	0.1 (0.10)
30	20	0.001	20-80	**75.03 (96.17)**	14.27 (23.43)	**99 (95)**	0.01 (0.05)	0.1 (0.22)
30	30	0.0015	20-80	**80.12 (110.95)**	16.29 (28.18)	**96 (89)**	0.07 (0.18)	0.43 (0.59)
30	40	0.002	20-80	**89.58 (116.94)**	28.13 (28.91)	**94 (80)**	0.08 (0.46)	0.34 (1.11)
30	50	0.0025	20-80	**91.93 (125.46)**	21.07 (32.30)	**92 (72)**	0.14 (0.59)	0.62 (1.18)
50	10	0.0005	0-100	**69.62 (85.10)**	8.60 (18.56)	**91 (92)**	0.25 (0.46)	1.53 (2.41)
50	20	0.001	0-100	**74.39 (97.56)**	10.68 (19.37)	**88 (75)**	0.2 (0.74)	0.64 (1.56)
50	30	0.0015	0-100	**84.19 (111.48)**	15.47 (23.09)	**85 (63)**	0.17 (1.44)	0.43 (3.16)
50	40	0.002	0-100	**93.67 (123.48)**	24.95 (29.11)	**77 (37)**	0.43 (2.66)	0.98 (3.60)
50	50	0.0025	0-100	**101.31 (127.72)**	29.13 (29.49)	**69 (32)**	0.68 (3.22)	1.64 (3.61)

Table 6. Cases of randomly moving obstacles in a fixed area. (Numbers in brackets show the results of R&G Method (Ratering & Gini, 1995))

In general, for the same $|\vec{v}_{r,i}|$, the no. of collision-free episodes decreases as D increases for the proposed method. Obviously, more obstacles in a fixed area increase the chance of collision. This is also true when $|\vec{v}_{r,i}|$ increases. In the extreme, only 69% of episodes are collision-free when $|\vec{v}_{r,i}|$ =0-100 cm/s and D=0.0025 (max). When compared with the R&G method, when D is very low, differences in no. of collision-free episodes between the two methods are insignificant. However, when D is larger (>10 obstacles), the proposed method performed consistently better. This is also the case when $|\vec{v}_{r,i}|$ increases. On average, the improvement on the no. of collision-free episodes is 23.63%, whereas the best is slightly over 115% for the largest D.

For average path time, it increases as D increases. This is to be expected as there are more obstacles and more CA actions that resulted in longer path time. On the other hand, for small $|\vec{v}_{r,i}|$ and large D, clustering of obstacles becomes a real possibility that can block the agent's path. This is confirmed by the large standard deviation of path time when compared with other larger $|\vec{v}_{r,i}|$. Although the R&G method employed the adjustable hill extent method to deal with this issue, their average path times are in fact longer. When $|\vec{v}_{r,i}|$ is large, obstacle clustering is reduced, but their speed makes it necessary to make more convoluted path to avoid them, therefore the resultant path time is longer, with smaller standard deviation. Again, there is a minimum in average path time at medium $|\vec{v}_{r,i}|$ depending on D. When compared with R&G method, an average improvement of 20.6% is achieved.

5. Conclusion

In this chapter we have presented a multiple goal reinforcement learning framework and illustrated on a two-goal problem in autonomous vehicle navigation. In general, DAQL can be applied in any goals that environmental response is available, whereas QL would suffice if environmental response is not available or can be ignored. A proportional goal fusion function was used to maintain balance between the two goals in this case. Extensive simulations have been carried out to evaluate its performance under different obstacle behaivors and sensing accuracy. The results showed that the proposed method is characterized by its ability to (1) deal with single obstacles at any speed and from any directions; (2) deal with two obstacles approaching from different directions; (3) cope with large sensor noise; (4) navigate in high obstacle density and high relative velocity environment. Detailed comparison of the proposed method with the R&G method reveals that improvements by the proposed method in path time and the number of collision-free episodes are substantial.

6. Acknowledgement

The authors would like to thank the anonymous referees for their thorough review of the paper and many constructive comments. The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administration Region, China (Project No.HKU7194/06E), and was partially supported by a postgraduate studentship from the University of Hong Kong.

7. References

Belkhouche F., Belkhouche B., Rastgoufard P. (2006). Line of sight robot navigation toward a moving goal, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 36, 255-267

Boutilier C. (1996). Planning, learning and coordination in multi-agent decision processes, *Sixth conference on Theoretical Aspects of Rationality and Knowledge (TARK `96)*, pp. 195-201, The Netherlands

Boutilier C., Dearden R., Goldszmidt M. (2000).Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121, 49-107

Bruce J.R., Veloso M.M. (2006). Safe multirobot navigation within dynamics constraints, *Proceedings of the IEEE Special Issue on Multi-Robot Systems*, 94, 1398-1411

Claus C., Boutilier C. (1998). The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems, *Fifteenth National Conference on Artificial Intelligence*, pp. 746-752

Doya K., Samejima K., Katagiri K., Kawato M. (2002). Multiple model-based reinforcement learning. *Neural Computation*, 14, 1347-1369

Er M.J., Deng C. (2005). Obstacle avoidance of a mobile robot using hybrid learning approach. IEEE Transactions on Industrial Electronics, 52, 898-905

Feng Z., Dalong T., Zhenwei W. (2004). Multiple obstacles avoidance for mobile robot in unstructured environments, *Proceedings of the 2004 IEEE International Conference on Robotics, Automation and Mechatronics*, vol.141, pp. 141-146, Singapore

Fiorini P., Shiller Z. (1998). Motion planning in dynamic environments using velocity obstacles. International Journal of Robotics Research, 17, 760-772

Ge S.S., Cui Y.J. (2000). New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16, 615-620

Guestrin C., Koller D., Parr R. (2001). Multiagent Planning with Factored MDPs, *Proceedings of the 14th Neural Information Processing Systems (NIPS-14)*, pp. 1523-1530

Kaelbling L.P. (1993). *Learning in embedded systems*, MIT Press, Cambridge, Mass.

Kaelbling L.P., Littman M.L., Moore A.W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4, 237-285

Hu J.L., Wellman M.P. (2004). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4, 1039-1069

Huang B.Q., Cao G.Y., Guo M (2005). Reinforcement Learning Neural Network to the Problem of Autonomous Mobile Robot Obstacle Avoidance, *Proceedings of the Fourth IEEE International Conference on Machine Learning & Cybernetics*, pp. 85-89, Guangzhou

Jacobs R.A., Jordan M.I., Nowlan S.J., Hinton G.E. (1991). Adaptive Mixtures of Local Experts. *Neural Computation*, 3, 79-87

Jing R., McIsaac K.A., Patel R.V. (2006). Modified Newton's method applied to potential field-based navigation for mobile robots. *IEEE Transactions on Robotics*, 22, 384-391

Kehtarnavaz N., Li S. (1988). A collision-free navigation scheme in the presence of moving obstacles, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 808-813

Kunwar F., Benhabib B. (200).Rendezvous-Guidance Trajectory Planning for Robotic Dynamic Obstacle Avoidance and Interception. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 36, 1432-1441

Large F., Sekhavat S., Shiller Z., Laugier C. (2002). Towards real-time global motion planning in a dynamic environment using the NLVO concept, *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.601, pp. 607-612, Lausanne, Switzerland

Latombe J.-C. (1991). *Robot motion planning*, Kluwer Academic Publishers, Boston

Laurent G., Piat E. (2001).Parallel Q-learning for a block-pushing problem, *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol.281, pp. 286-291, USA

Laurent G.J., Piat E. (2002). Learning mixed behaviours with parallel Q-learning, *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol.1001, pp. 1002-1007, Lausanne, Switzerland

Littman M.L. (2001). Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, 2, 55-66

Minguez J., Montano L. (2004). Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20, 45-59

Minguez J. (2005). The obstacle-restriction method for robot obstacle avoidance in difficult environments, *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2284-2290

Miura J., Uozumi H., Shirai Y. (1999). Mobile robot motion planning considering the motion uncertainty of moving obstacles, *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics*, vol.694, pp. 692-697, Tokyo, Japan

Mucientes M., Iglesias R., Regueiro C.V., Bugarin A., Carinena P., Barro S. (2001). Fuzzy temporal rules for mobile robot guidance in dynamic environments. *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, 31, 391-398

Ngai D.C.K., Yung N.H.C. (2005a). Double action Q-learning for obstacle avoidance in a dynamically changing environment, *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, pp. 211-216, Las Vegas, USA

Ngai D.C.K., Yung N.H.C. (2005b). Performance evaluation of double action Q-learning in moving obstacle avoidance problem, *Proceedings of the 2005 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 861, pp. 865-870, Hawaii, USA

Oriolo G., Ulivi G., Vendittelli M. (1998). Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 28, 316-333

Pimenta L.C.A., Fonseca A.R., Pereira G.A.S., Mesquita R.C., Silva E.J., Caminhas W.M., Campos M.F.M. (2006). Robot navigation based on electrostatic field computation, *IEEE Transactions on Magnetics*, 42, 1459-1462

Puterman M.L. (1994). *Markov decision processes : discrete stochastic dynamic programming*. John Wiley & Sons, New York

Ratering S., Gini M. (1995). Robot Navigation in a Known Environment with Unknown Moving Obstacles. *Autonomous Robots*, 1, 149-165

Qu Z.H., Wang J., Plaisted C.E. (2004). A new analytical solution to mobile robot trajectory generation in the presence of moving obstacles. *IEEE Transactions on Robotics and Automation*, 20, 978-993

Shensa M. (1981). Recursive least squares lattice algorithms--A geometrical approach. *IEEE Transactions on Automatic Control*, 26, 695-702

Shiller Z., Large F., Sekhavat S. (2001). Motion planning in dynamic environments: obstacles moving along arbitrary trajectories, *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, vol.3714, pp. 3716-3721, Seoul

Soo J. E., Seul J., Hsia T.C. (2005). Collision avoidance of a mobile robot for moving obstacles based on impedance force control algorithm, *Proceedings of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems 2005 (IROS 2005), pp. 382-387

Stentz A. (1994). Optimal and efficient path planning for partially-known environments, *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, vol. 3314, pp. 3310-3317, 1994

Stentz A. (1995). The focused D* algorithm for real-time replanning, *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, IJCAI '95, Montreal, Canada, August 1995

Sutton R.S. (1992). *Reinforcement learning*, Kluwer Academic Publishers, Boston

Sutton R.S., Barto A.G. (1998). *Reinforcement learning : an introduction*, MIT Press, Cambridge, Mass.

Uchibe E., Asada M., Hosoda K. (1996). Behavior coordination for a mobile robot using modular reinforcement learning, *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems '96*, vol.1323, pp. 1329-1336

Watkins C.J.C.H., Dayan P. (1992). Q-Learning. *Machine Learning*, 8, 279-292

Yamamoto M., Shimada M., Mohri A. (2001). Online navigation of mobile robot under the existence of dynamically moving multiple obstacles, *Proceedings of the 4th IEEE Int. Symposium on Assembly & Task Planning*, pp. 13-18, Soft Research Park, Japan

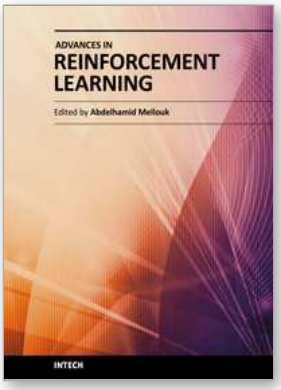
Yang S.X., Meng M.Q.H. (2003). Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach. *IEEE Transactions on Neural Networks*, 14, 1541-1552

Ye C. (1999). *Behavior-Based Fuzzy Navigation of Mobile Vehicle in Unknown and Dynamically Changing Environment*. Pd.D. Thesis, Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

Ye C., Yung N.H.C., Wang D.W. (2003). A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics*, 33, 17-27

Yoon H.U., Sim K.B. (2005). Hexagon-Based Q-Learning for Object Search with Multiple Robots, *Proceedings of Advances in Natural Computation: First International Conference*, ICNC 2005, pp. 55-66, Changsha, China

Zhu Q.M. (1991). Hidden Markov Model for Dynamic Obstacle Avoidance of Mobile Robot Navigation. *IEEE Transactions on Robotics and Automation*, 7, 390-397



Advances in Reinforcement Learning

Edited by Prof. Abdelhamid Mellouk

ISBN 978-953-307-369-9
Hard cover, 470 pages
Publisher InTech
Published online 14, January, 2011
Published in print edition January, 2011

Reinforcement Learning (RL) is a very dynamic area in terms of theory and application. This book brings together many different aspects of the current research on several fields associated to RL which has been growing rapidly, producing a wide variety of learning algorithms for different applications. Based on 24 Chapters, it covers a very broad variety of topics in RL and their application in autonomous systems. A set of chapters in this book provide a general overview of RL while other chapters focus mostly on the applications of RL paradigms: Game Theory, Multi-Agent Theory, Robotic, Networking Technologies, Vehicular Navigation, Medicine and Industrial Logistic.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chi Kit Ngai and Nelson H. C. Yung (2011). DAQL-Enabled Autonomous Vehicle Navigation in Dynamically Changing Environment, Advances in Reinforcement Learning, Prof. Abdelhamid Mellouk (Ed.), ISBN: 978-953-307-369-9, InTech, Available from: http://www.intechopen.com/books/advances-in-reinforcement-learning/daql-enabled-autonomous-vehicle-navigation-in-dynamically-changing-environment

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821