A DECENTRALIZED REINFORCEMENT LEARNING CONTROLLER FOR COLLABORATIVE DRIVING

Luke Ng, Chris Clark, Jan P. Huissoon

Department of Mechanical Engineering, Automation & Controls Group
University of Waterloo
Waterloo, Ontario, Canada
l4ng@engmail.uwaterloo.ca

Abstract: Research in the collaborative driving domain strives to create control systems that coordinate the motion of multiple vehicles in order to navigate traffic both efficiently and safely. In this paper a novel individual vehicle controller based on reinforcement learning is introduced. This controller is capable of both lateral and longitudinal control while driving in a multi-vehicle platoon. The design and development of this controller is discussed in detail and simulation results showing learning progress and performance are presented.

Keywords: autonomous vehicles, co-ordination, decentralized control, machine learning, robot control.

1. INTRODUCTION

In major cities throughout the world, urban expansion is leading to an increase of vehicle traffic flow. The adverse effects of increased vehicle traffic flow include traffic congestion, driving stress, vehicle collisions, pollution, and logistical delays. Once traffic flow surpasses the capacity of the road system, it ceases to become a viable transportation option. One solution is to build more roads; another is to build a better vehicle - a vehicle that can negotiate traffic, coordinate with other similar 'thinking' vehicles to optimize their speeds so as to arrive at their destination safely and efficiently. The vehicle system described is referred to formally as a collaborative driving system.

Previous studies have addressed collaborative driving as a hierarchical decentralized control problem (Huppe et al 2003), (Halle et al 2004). These approaches use a layered approach with a heuristic controller layer choosing among lower-level behaviours. Alternatively, one could approach collaborative driving as a multi-agent planning problem, and address it using reinforcement learning (Laumonier et al 2006).

Reinforcement Learning is a machine learning technique which uses a numerical reward signal as feedback to modify a control policy (Sutton & Barto 1998). Its promise is that an agent can learn using experience alone in a changing environment. As this technique is a recent development in machine learning, it has had limited success in scaling to real-world problems (Stone & Sutton 2001). A current challenging area of research is to scale reinforcement learning to solve real-world robot control problems (Ng et al 2004), (Kohl & Stone 2004).

In this paper, a novel low-level vehicle controller is introduced. Its key feature is that it adapts its control policy using reinforcement learning for both lateral and longitudinal control of the individual vehicle. This controller is meant to be deployed in a vehicle group for the purpose of collaborative driving. The objective of this paper is to present the design and development of this low-level vehicle controller. In particular, the methodology of mapping reinforcement learning algorithms to this problem domain is explained. In addition, simulation results for the initial learning process are presented along with initial performance data while travelling in a fixed platoon.

2. COLLABORATIVE DRIVING DOMAIN

The collaborative driving research domain aims to coordinate the motion of multiple vehicles in order

[1]This research is funded by the Auto21 Network of Centres of Excellence, an automotive research and development program focusing on issues relating to the automobile in the 21st century. AUTO21 is a member of the Networks of Centres of Excellence of Canada program. Web site: www.auto21.ca.
[2]Part of this research was undertaken at the University of Toronto Institute for Aerospace Studies, Space Robotics Group for the application of multi-robot planetary exploration.

to navigate traffic. The group of vehicles being coordinated forms a single-file formation (platoon) in which the lead vehicle's role is to manage and guide the group as it travels along a road (Varaiya 1993).

Issues being addressed in collaborative driving include: longitudinal control (maintaining vehicle spacing), lateral control (lane changing and turning), insertion and exit into and out of the platoon, human-in-the-loop platoon guidance, fully autonomous platoon guidance, vehicle configurations, road configurations/conditions, sensor fusion, communication, and scalability. This paper represents a first step to approaching the problem of collaborative driving from the bottom up. Thus, we begin by focusing on the problem of individual vehicle control while travelling in a platoon. Further work will address the issue of platoon maintenance, inter-vehicle communication, and autonomous control.

3. REINFORCEMENT LEARNING

3.1 The Markov Decision Process

Collaborative driving can be considered a problem of acting or planning in the presence of uncertainty (i.e. in a changing environment). The Markov Decision Process (MDP) provides a formal framework for modelling this type of control problem and for deriving an optimal solution (i.e. an optimal controller) (Bellman 1957).

MDPs are characterized by states s, actions a, and transitions $\sigma(s,a)$. For the problem to be considered Markov, s must be able to completely describe the current situation without a dependence on path. That is, the state does not depend on previous states. Transitions σ, are predictions of the next state based on the current state and the current action being performed, $s' = \sigma(s, a)$. For a robot agent, the transitions represent the agent's model of the environment.

3.2 The Policy

The action for a given state is governed by the agent's current policy π, a mapping from states to actions. An agent starts at an initial state s_0 and follows π until the goal, referred to as the terminal state s_{term}, is reached. The process of going from s_0 to s_{term} is known as an episode. For every MDP, there exists at least one policy which maps the *best* action for every state this is referred to as the optimal policy π^*, the solution to the MDP.

In order for the agent following a policy π to evaluate the effectiveness of its actions, a reward signal r from the environment associated with each s is provided to the agent. If an accurate transition model σ is provided to the agent, the agent can predict how much future reward will be received for the remainder of the episode by a summation of the rewards associated with visiting the remaining states in an episode following a given policy. The form of the return R typically used includes a discount rate γ, which allows future rewards to be discounted,

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (1)$$

where $0 \leq \gamma \leq 1$. The return is the basis for determining which action is best to take for a given state.

The state-value function, $V^\pi(s)$ is the expected accumulation of discounted reward received at each state leading to the terminal state by following policy π, where $E_\pi\{\}$ denotes the expected value.

$$V^\pi(s) = E_\pi\{R_t \mid s_t = s\} = E_\pi\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s\} \quad (2)$$

It expresses the desirability of being in a particular state for a given policy. When following the optimal policy π^*, the state-value function is maximized. If an accurate transition model σ is available, only the state s is important since actions a can be mapped directly to the next state s'. Otherwise, the state and action must be paired (s,a), to represent a unique event. Thus a state-action-value function is introduced, $Q^\pi(s,a)$, which allows the agent to assess the desirability of following a given action a while in a particular state s for a policy π.

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

3.3 The Algorithms

The algorithms used to determine the optimal policy π^* for a given MDP are called Reinforcement Learning (RL) algorithms. These algorithms follow a common approach. The agent begins with an initial policy π_0 which may be far from optimal, and an estimate of its state-value function $V^\pi(s)$ or state-action-value function $Q^\pi(s,a)$ is inaccurate. As the agent follows π_0 it receives rewards r based on its state s. Using this new information, its estimate of $V^\pi(s)$ or $Q^\pi(s,a)$ is iteratively improved. Concurrently, the improved $V^\pi(s)$ or $Q^\pi(s,a)$ can be used to iteratively improve the policy until the agent arrives at the optimal policy π^*.

The convergence of this maximization process requires that all states and actions be visited infinitely in order for estimates of $V^\pi(s)$ or $Q^\pi(s,a)$ to reach their actual values. To ensure this convergence criterion, policies leading to π^* are ε-soft, meaning that there is a ε probability that a random action is selected. Therefore, all actions and states will be reached as $t \to \infty$.

In this paper, the RL algorithms considered are called Temporal-Difference (TD) Learning Algorithms. TD-Learning algorithms are a class of RL algorithms that do not require a transition model. Experience is used to update $Q^\pi(s,a)$ every step of a given episode. TD-Learning algorithms can learn quickly however, they can become trapped in local minima as experiences are used immediately (called boot-strapping) as opposed to other methods which

average experiences over the entire episode (i.e. Monte Carlo methods).

Two versions of TD-Learning algorithms are evaluated for implementation in the collaborative driving domain. The first is the Q-Learning algorithm (Watkins 1989) shown in Figure 1, which is considered an off-policy method as it estimates $Q^{\pi}(s,a)$ using the max_a policy. The max_a policy returns the action with the highest value for the given state. The second is called SARSA (Rummery & Niranjan 1994), shown in Figure 2, and is considered an on-policy method as it estimates $Q^{\pi}(s,a)$ using the current policy with the aid of the next state and next action. In theory, if the optimal policy of an MDP is unique, these two approaches should lead to this same policy what will differ is how quickly they will reach this policy.

Initialize, $Q(s, a)$
Repeat (for each episode):
  Initialize s
  Repeat (for each step in the episode):
    Choose a from s using policy derived from Q (e.g. ε-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$
  until s is terminal

Fig. 1. Q-Learning Algorithm.

Initialize, $Q(s, a)$
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g. ε-greedy)
  Repeat (for each step in the episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g. ε-greedy)
    $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$
    $s \leftarrow s'$; $a \leftarrow a'$
  until s is terminal

Fig. 2. SARSA Algorithm

4. CONTROL

The collaborative driving problem can be considered a hierarchy of two control problems. The lower level problem is the control of the individual vehicle, while the higher level problem is the management of the platoon when multiple vehicles are driving in single file formation. This paper focuses exclusively on the lower controller. The design of this controller is described in detail in the following paragraphs.

The problem of lateral and longitudinal control at the individual level is addressed with a decentralized controller modelled using a Markov Decision Process (MDP). Table 1 summarizes the states/inputs and actions/outputs of the MDP while Figure 3 illustrates the variables used in determining the states and actions with respect to the vehicles.

The states and actions exist in the world as continuous normalized values. The states are converted into discrete binary strings of varying precision so they can exist as discrete variables within the MDP framework. This process of conversion is analogous to analog-to-digital conversion and digital-to-analog conversion used in digital signal processing. The MDP outputs actions in the form of discrete binary strings that are converted into continuous normalized values.

Table 1. States and actions of the lower-level MDP

State:	Expression:	Description:
s_1	θ_i / θ_{max}	Normalized difference in angle to preceding vehicle.
s_2	$(d_t - d_{min}) / (d_{max} - d_{min})$	Normalized distance to preceding vehicle's minimum distance.
s_3	$\|\|v_i\|\|$	Current vehicle speed
Action:		
a_1	k_θ	Gain for steering angle
a_2	k_v	Gain for speed

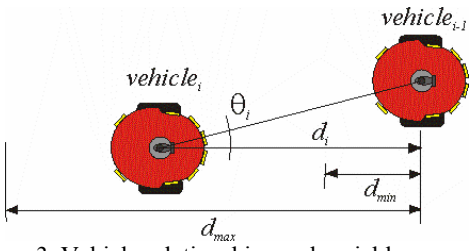

Figure 3. Vehicle relationships and variables

The collaborative driving problem at the individual level can be seen as a problem of maintaining equilibrium for two independent variables: i) the alignment of the proceeding vehicle and ii) the distance to the preceding vehicle Intuitively, the steering command should be proportional to the error in alignment to the proceeding vehicle. Likewise the speed command should be proportional to the error between actual distance and desired distance to the preceding vehicle. Therefore, the actions that are required to be learned are the proportionality gains of the steering and speed commands for different situations or states. The steering and speed functions are expressed below, where i is the rank in the platoon of n vehicles.

$$\theta_{cmd} = a_1 s_1 = k_\theta \theta_i / \theta_{max} \quad (4)$$

$$v_{cmd} = a_2 s_2 = k_v (d_t - d_{min})/(d_{max} - d_{min}) \quad (5)$$

The reward function conveyed to the MDP is a function of the observed states. It is expressed as a superposition of two separate piecewise continuous functions as shown in equations 5, 6, and 7.

$$R_{tot} = R_1(s_1) + R_2(s_2) \quad (5)$$

$$R_1(s_1) = \begin{cases} 1 - |s_1| & if\ |s_1| < 1 \\ -1 & if\ |s_1| = 1 \end{cases} \quad (6)$$

$$R_2(s_2) = \begin{cases} 1 - s_2 & if\,(0 < s_2 < 1) \\ -1 & if\,(s_2 = 1) \\ s_2 & if\,(s_2 \le 0) \end{cases} \quad (7)$$

The reward function is a key factor in determining the optimal policy, that is, it communicates to the MDP the task to be performed. In this MDP, continuous rewards are used to favour the elimination of both the angle error and the distance error. The resulting reward function produces the surface in Figure 4.

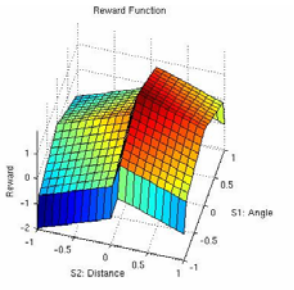

Fig. 4. Reward surface for MDP

In general the process of learning begins with a ε-soft policy, $a(s)$ = ε-greedy. This policy is initially set to random, that is every action *a,* has equal probability of being selected and ε is the exploration factor, where $0 \le \varepsilon \le 1$. Prior to selecting an action using ε-greedy, a random number n is generated where $0 \le n \le 1$ if $n > \varepsilon$, ε-greedy should return the action with the maximum Q for the state s provided if there is no prior visit to state s or if $n \le \varepsilon$, an *exploration start* occurs and a random action a is selected. As all states and actions are visited the ε-greedy policy becomes the optimal policy π^* with the exploration starts disabled.

The manner in which the optimal policy is learned is dependent on the learning algorithm. In designing this controller, two TD-Learning algorithms are studied, since there are no rules to indicate which algorithms work best for which situations. TD-Learning methods are chosen since this type of solution method utilizes experience immediately. SARSA and Q-Learning (see Figure 1 and 2) are the two of the most common TD-Learning algorithms, with many variations in existence. In this paper the basic implementation of each is evaluated.

5. MODELLING

In order to derive an optimal control policy through reinforcement learning, an environment is required. This environment is provided in simulation its purpose is to provide the learning algorithm with simulated state information, a transition model to go from action to subsequent state, and reward data with which to improve its policy with. The benefit of simulation is twofold, as it also allows one to evaluate the performance of the controller in situations which may not be feasible to achieve in reality.

In this study a commercial rapid prototyping tool for mobile robots is used called Webots[tm], by Cyberbotics Ltd. (Michel 2004). Using Webots[tm], a vehicle model was created along with two different training environments. Webots[tm] contains a library of sensors and robot models with which to build robot experiments with. It integrates with the Open Dynamics Engine (ODE), an open-source rigid body simulation library which models dynamics using lagrange multipliers (Baraff 1996).

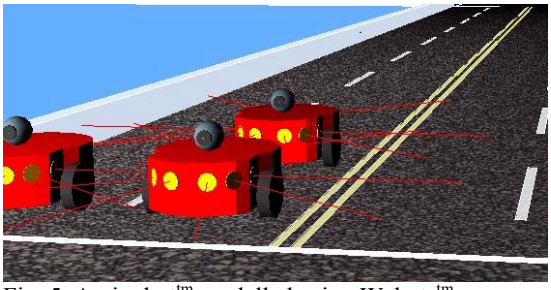

Fig. 5. Amigobot[tm] modelled using Webots[tm].

4.1 Vehicle Model

The vehicle used in this study is a small holonomic differentially driven two-wheeled mobile robot called the Amigobot[tm], manufactured by MobileRobots Inc. Our laboratory has access to up to five of these research robots, on which the controller will ultimately be deployed on. Table 2 list key parameters used in modelling the Amigobot[tm].

Table 2. Webots[tm]/ODE Model Parameters

Object	Parameters:		
Body Mass	m = 3.5 kg		
Bounding Cylinder	h = 0.132m	r = 0.14m	
Wheels:	m = 0.05kg	r = 0.05m	W = 0.025m
Sonar:	6 front	2 rear	R = 0.4m
Camera:	Color	320x240	$\theta_{max} = \pm 20°$

As the application for the controller is collaborative driving, commands such as steering angle θ_{steer} and speed V are used to mimic an automobile. However, the Amigobot[tm] is driven with differential wheel speeds. Therefore steering and speed commands are mapped to the left and right wheel speeds via equations 8 and 9.

$$V_{left} = (V / V_{max} - \theta_{steer} / \theta_{max})\, k_{wheel} \quad (8)$$

$$V_{right} = (V / V_{max} + \theta_{steer} / \theta_{max})\, k_{wheel} \quad 9)$$

4.2 Environment

Two distinct environment models are created in this study: one for initial learning (called *Arena*), the other for performance testing (called *Track*). The *Arena* is a large open 10m x10m flat floor with a barrier to prevent the vehicles from escaping. This environment contains two vehicles, a lead vehicle and a learning vehicle which follows it. Both static and dynamic target training are conducted in this environment. The custom MDP software allows policies to be saved and reloaded, to facilitate learning.

The *Track* is a simulated two lane road with barriers on either side to prevent escape. Five vehicles are modelled in this world and placed at the start position. The end of the road contains a "Warp Gate" which when reached sends the vehicles back to the starting position. Therefore, a repeating road can be simulated using only a straight 50m section of track.

6. RESULTS

6.1. Learning

Two control policies are obtained through static target training through two different reinforcement learning algorithms. Q-Learning with $\varepsilon = 0.1$, $\gamma = 0.1$ and $\alpha = 0.1$ (learning rate) is used to learn a control policy π_1 which can approach a static target to within 1m. The location of the target is randomized for 100 episodes. This is repeated for the SARSA algorithm under the identical conditions to arrive at π_2.

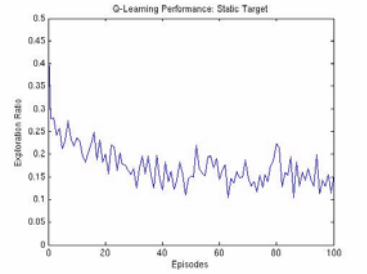

Fig. 6. Q-Learning performance (π_1).

Fig. 7. SARSA: Learning performance (π_2).

Figure 6 and 7 show the progress of learning or each algorithm. The exploration ratio represents the ratio of exploration starts to total steps for a given episode,. This begins high as the policy is unknown and as $t \rightarrow \infty$, it should approach ε, showing the policy has been learned. From the simulation data, both algorithms appear to learn at more or less the same speed, and we cannot say one is faster than the other for this application.

6.2. Longitudinal Performance

The longitudinal control is evaluated by deploying four copies of a policy on four vehicles. The vehicles are initially arranged in single file separated by 0.4m and all at rest. The lead vehicle is set to reach a constant speed of 0.3m/s while each vehicle attempts to follow the preceding vehicle. The vehicle separations are recorded throughout the evaluation. At t = 100 sec, the lead vehicle is stopped to observe the behaviour of the platoon as the vehicles come to rest. This evaluation is executed for both policy π_1 and π_2 and shown in Figure 8 and 9 respectively.

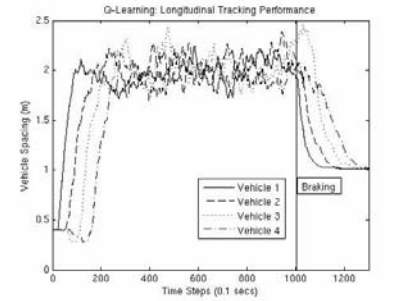

Fig. 8. Q-Learning: Longitudinal control (π_1).

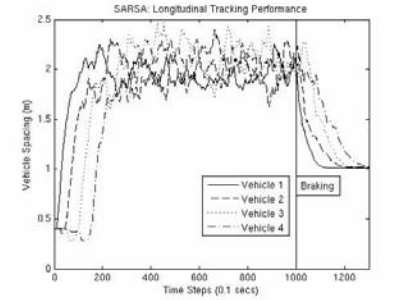

Fig. 9. SARSA: Longitudinal control (π_2).

For this application, the minimal vehicle spacing is considered 1m while the maximum distance is 5m. It is interesting that all vehicles try to adjust their distance to 2m while travelling at about 0.3m/s. After braking, the vehicles attempt to keep a distance of 1m as this results in more reward.

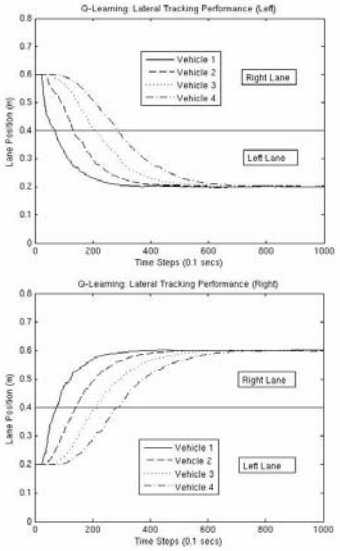

Fig. 10. Q-Learning: Lateral control (π_1).

6.3. Lateral Performance

The lateral control is evaluated by deploying four copies of a policy on four vehicles. The vehicles are initially arranged in single file separated by 2m and all moving at a constant speed 0.3m/s in one lane. A lead vehicle is placed on the adjacent lane, this is equivalent to a unit step input; each vehicle then

tracks the preceding vehicle so as to accomplish a lane change. The vehicles' lateral positions are recorded throughout the evaluation. This evaluation is executed for both policy π_1 and π_2. Figure 10 shows the left and right lane change for π_1 while Figure 11 shows the left and right lane change for π_2.

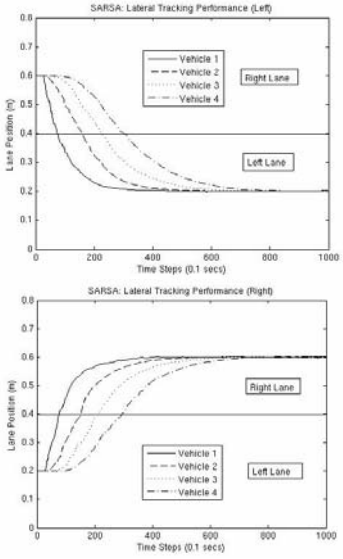

Fig. 11. SARSA: Lateral control (π_2).

The data show a very smooth transition to the other lane with no overshoot. The response is identical for both policies and symmetrical.

7. CONCLUSION

From these initial simulation results, we can draw a few conclusions. The difference in learning performance between the Q-Learning and SARSA algorithms is indiscernible, although, if learning continued for 100s of more episodes we might observe some differences. For now we conclude that they give similar learning performance for this application.

From the very preliminary performance evaluations of the two obtained policies, one from Q-Learning, the other from SARSA, very similar performance is observed. One could conclude that they are in fact the same policy for this MDP. This follows in theory that if the optimal policy for an MDP is unique then different RL algorithms will lead to the same optimal policy. Only more evaluations about different operating points would prove that the same optimal policy has been reached.

8. FUTURE WORK

This paper has presented some preliminary results in our study of collaborative driving control. While it appears that reasonable vehicle behaviour can be achieved via machine learning, our future objectives include looking at longer learning periods for better policies, more detailed evaluation of the obtained policies, stability analysis, the control of higher-level platoon management, and system performance under varying initial platoon configurations and road conditions.

REFERENCES

Baraff, D. (1996). Linear-Time Dynamics using Lagrange Multipliers. In: *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '96)*. pp 137-146. New Orleans, LA, USA, Aug 1996.

Bellman, R. E. (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.

Halle, S., J. Laumonier and B. Chaib-draa. (2004). A Decentralized Approach to Collaborative Drive Coordination. In: *Proceedings of 7th IEEE International Conference on Intelligent Transportation Systems (ITSC'2004)*. Washington, DC, USA, Oct 2004.

Huppe, X., J. de Lafontaine, M. Beauregard and F. Michaud. (2003). Guidance and Control of a Platoon of Vehicles Adapted to Changing Environment Conditions. In: *Proceedings IEEE Conference on Systems, Man, and Cybernetics*. pp 3091-3096.

Kohl, N. and P. Stone. (2004). Policy Gradient Reinforcement Learning for Fast Quadrupedal Locomotion. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. pp 2619-2624. May 2004.

Laumonier, J., C. Desjardins and B. Chaib-draa. (2006). Cooperative Adaptive Cruise Control: a Reinforcement Learning Approach. In: *Proceedings of 4th Workshop in Traffic and Transportation, AAMAS'06.* Hakodate, Hokkaido, Japan, May 2006.

Michel, O. (2004). Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. In: *International Journal of Advanced Robotic Systems*. **Vol 1 Number 1** pp 39-42.

Ng, A.Y., A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. (2004). Sutonomous inverted helicopter flight via reinforcement learning. In: *Proceedings of International Symposium on Experimental Robotics*.

Rummery, G.A. and M. Niranjan (1994). On-Line Q-Learning using connectionist systems. In: *Technical Report CUED/F-INFENG/TR 166.* Engineering Department, Cambridge University.

Stone, P. and R.S. Sutton. (2001). Scaling Reinforcement Learning toward RoboCup Soccer. In: *Proceedings of The Eighteenth International Conference on Machine Learning (ICML 2001)*. pp 537-544. Willianstown, MA, USA, June 2001.

Sutton, R.S. and A.G. Barto. (1998). Reinforcement Learning: An Introduction. A Bradford Book. The MIT Press. Cambridge, MA, USA.1998.

Varaiya, P. (1993). Smart cars on smart roads: problems of control. In: *IEEE Transactions on Automatic Control.* **Vol 32**, Mar 1993.

Watkins, C.J.C.H. (1989). Learning from Delayed Rewards. Ph.D. thesis. Cambridge University.