Multi-Agent Systems for Traffic and Transportation Engineering

Ana L. C. Bazzan
Instituto de Informática, UFRGS, Brazil

Franziska Klügl
Örebro University, Sweden

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Director of Editorial Content:	Kristin Klinger
Senior Managing Editor:	Jamie Snavely
Managing Editor:	Jeff Ash
Assistant Managing Editor:	Carole Coulson
Typesetter:	Jeff Ash
Cover Design:	Lisa Tosheff
Printed at:	Yurchak Printing Inc.

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com/reference

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanbookstore.com

Library of Congress Cataloging-in-Publication Data

Multi-agent systems for traffic and transportation engineering / Ana L.C. Bazzan and Franziska Klugl, editor.
p. cm.
Includes bibliographical references and index.

Summary: "This book aims at giving a complete panorama of the active and promising crossing area between traffic engineering and multi-agent system addressing both current status and challenging new ideas"--Provided by publisher.

ISBN 978-1-60566-226-8 (hbk.) -- ISBN 978-1-60566-227-5 (ebook)

1. Transportation--Data processing. 2. Transportation engineering--Data processing. 3. Traffic engineering--Data processing. 4. Intelligent agents (Computer software) I. Bazzan, Ana L. C. II. Klügl, Franziska.
TA1230.H36 2009
388.3'10285--dc22

2008042399

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter XI
Learning Agents for Collaborative Driving

Charles Desjardins
Laval University, Canada

Julien Laumônier
Laval University, Canada

Brahim Chaib-draa
Laval University, Canada

ABSTRACT

This chapter studies the use of agent technology in the domain of vehicle control. More specifically, it illustrates how agents can address the problem of collaborative driving. First, the authors briefly survey the related work in the field of intelligent vehicle control and inter-vehicle cooperation that is part of Intelligent Transportation Systems (ITS) research. Next, they detail how these technologies are especially adapted to the integration, for decision-making, of autonomous agents. In particular, they describe an agent-based cooperative architecture that aims at controlling and coordinating vehicles. In this context, the authors show how reinforcement learning can be used for the design of collaborative driving agents, and they explain why this learning approach is well-suited for the resolution of this problem.

INTRODUCTION

Modern automotive transportation technologies have faced, in recent years, numerous issues resulting from the increase of vehicular traffic and having important consequences on passenger safety, on the environment and on the efficiency of the traffic flow.

In response, both manufacturers and public institutions have focused on such issues through research and development efforts, and have come up with many solutions. Among them, as mentioned in the introductory chapters, the field of Intelligent Transportation Systems (ITS) has gathered particular interest in the past twenty years. This chapter concerns a specific domain of ITS, which aims at designing fully autonomous vehicle controllers.

Many terms have been used to describe this field and its related technologies, such as Collaborative Driving Systems (CDS), Advanced Vehicle Control and Safety Systems (AVCSS) and Automated Vehicle Control Systems (AVCS). According to Bishop (2005), these systems could be defined as Intelligent Vehicle (IV) technology. Bishop characterized IV systems by their use of sensors to perceive their environment and by the fact that they are designed to give assistance to the driver in the operation of the vehicle. This definition of Intelligent Vehicles describes both Autonomous Vehicle Control and Collaborative Driving systems that we consider in this chapter.

Of course, the agent abstraction can be directly adapted to the definition of IV, as agents have the ability to sense their environment and make autonomous decisions to take the right actions. In the past, work related to the problem of autonomous vehicle control has already considered using intelligent agents. What we propose in this chapter is to show how agent technology can be used to design intelligent driving systems. More precisely, we will detail the design of an agent architecture for autonomous and collaborative driving based on the use of reinforcement learning techniques. We intend to show that reinforcement learning can be an efficient technique for learning both low-level vehicle control and high-level vehicle coordination as it enables the design of a controller that can efficiently manage the complexity of the application, i.e. the number of possible vehicle states and the number of coordination situations.

The next section of this chapter surveys the field of autonomous vehicle control and collaborative driving. It also details what has been done in this field in relation to agent technology. The third section briefly explains agent learning techniques while the fourth and final section describes how reinforcement learning can be used to build agents that can drive and coordinate themselves with others autonomously.

SURVEY OF COLLABORATIVE DRIVING SYSTEMS BASED ON AGENT TECHNOLOGY

This section first surveys what has been done in the field of autonomous vehicle control and collaborative driving systems. Then, it describes how the software agent abstraction and machine learning algorithms have already been used in the design of such systems.

Autonomous Vehicle Control and Collaborative Driving Systems

In response to the problems related to the increase of vehicular traffic, most industrialized countries have decided in recent years to adopt a road-map detailing the future of their investments in Intelligent Transportation Systems (ITS) research. Starting in the early ’90s, this resulted in the fact that many research projects, often in the form of partnerships between academia and industry, began addressing the design of autonomous vehicle control systems. Research has rapidly led to the development of various applications, as detailed in Table 1.

Already, vehicle manufacturers have integrated some of these technologies in vehicles. For example, many luxury cars are now equipped with Adaptive Cruise Control (ACC) systems, automated parking technologies and even lane-keeping assistance systems. Technologies that have been included in vehicles are, for the moment, used in the form of driving-assistance systems where most of the driving task still belongs to the driver.

Of course, a great amount of research is still being done in this field in order to implement these technologies in consumer products. Clearly, it seems inevitable that the industry will, in a few decades, move towards fully automated vehicles. However, a couple of technological hurdles must be addressed before such sophisticated systems can become a reality.

Currently, the next step towards the implementation of fully autonomous collaborative driving systems is the development of efficient communication technology. Clearly, a robust communication protocol, for both vehicle-to-vehicle (V2V) and road-to-vehicle (R2V) communication, is a pre-requisite for collaboration. As a result, many research institutions have already been working on the development and on the implementation of a standardized communication protocol named DSRC (Dedicated Short-Range Communication). Evidently, a lot of research is still being done in that field, and we refer to Tsugawa (2005) for more details on the state of the art of inter-vehicle communications.

Table 1. Autonomous vehicle control technologies and their description (Bishop, 2005)

Technology	**Description**
Collision Detection and Avoidance	This technology uses sensors to monitor the surroundings of the vehicle and to detect possible collisions. The driver is alerted of possible accidents. In the future, these systems could even take action directly on the vehicle to avoid collision.
Lane-Keeping Assistance	This technology uses computer vision systems to detect the curvature of the highway. It can react accordingly, with small adjustments to steering, in order to keep the center of the current lane.
Adaptive Cruise Control (ACC)	This technology uses a laser sensor to detect the presence of a front vehicle. The system adapts the vehicle's cruising velocity in order to avoid collision. Once the obstacle is gone, the vehicle goes back to its initial, desired velocity
Cooperative Adaptive Cruise Control (CACC)	This technology adds a communication layer to ACC systems. Information about the acceleration of a front vehicle is shared and is used to reduce the distance between vehicles.
Platooning	This technology takes CACC to the next level by using communication to exchange acceleration data of an important number of vehicles travelling in a platoon formation.
Automated Longitudinal and Lateral Vehicle Control	This technology uses fully automated controllers to act on a vehicle's longitudinal and lateral components.
Collaborative Driving	This technology is the ultimate goal of autonomous vehicle control. It uses inter-vehicle communication in order to share sensor information and driving intentions with surrounding vehicles (not necessarily a platoon) and select an optimal driving action.

Research Projects

Many research projects have been active in the development and design of autonomous vehicle control, collaborative driving and related technologies.

Perhaps the most famous and influential program in this field is the program of the University of California at Berkeley called PATH (Partners for Advanced Transit and Highway). This program regroups numerous research projects that share the ultimate goal of solving the issues of transportation systems through the use of modern technologies. PATH projects have designed and tested an important range of solutions related to vehicle control. They have studied solutions to complex problems such as automated longitudinal (Raza & Ioannou, 1997; Lu et al., 2000) and lateral vehicle control (Peng et al., 1992), cooperative collision warning systems (Sengupta et al., 2007) and platooning (Godbole & Lygeros, 1994; Sheikholeslam & Desoer, 1990). Bana (2001), has also worked on the use of vehicle communications for advanced vehicle coordination. For more details about the history of PATH and its future research directions, we refer to Shladover (2007). Finally, PATH is also famous in part because it has implemented and demonstrated an autonomous platooning control system as early as in 1997, as part of the Demo '97 event (NAHSC, 1998).

Another important research program has been Japan's Advanced Cruise-Assist Highway Systems Research Association (AHSRA). Similarly to PATH, this program has focused on the development of intelligent systems for the infrastructure, but has also worked on Advanced Security Vehicle (ASV) systems which promote the development and integration of intelligent systems in vehicles. The next step of their project consists in linking both types of systems using a communication architecture. Their program is also well-known for its implementation and demonstration of ASV technologies in the Demo2000 (Tsugawa et al., 2000) event. Moreover, since AHSRA regroups many manufacturers, a large number of the technologies developed through this program have rapidly been integrated in Japanese vehicles.

Many European countries have also been active in this field. For instance, recent work at the TNO Automotive (a research institute of The Netherlands) through the CarTalk2000 project, with partners DaimlerChrysler and Siemens, has focused on the development of communication systems and their application to autonomous vehicle control (de Bruin et al., 2004; Hallouzi et al., 2004).

Of course, the projects described here only offer a glimpse of all the research that has been done on this topic. A lot of other research organizations have also financed projects in this field, such as Italy's ARGO (Broggi et al., 1999), Canada's Auto21 (Auto21, 2007) and European's CHAUFFEUR projects (Schulze, 2007), just to name a few.

Design of Intelligent Vehicles Using Agents and Machine Learning

As we have described earlier, the agent abstraction is especially adapted to the problem of automated vehicle control and collaborative driving. It is not surprising to see that a number of research projects have considered using agents to design such control systems. Moreover, since agents need a decision-making mechanism, the use of agents has often been in conjunction with machine learning techniques. This section overviews previous work on the use of agents and machine learning techniques for autonomous vehicle control and coordination.

Machine Learning for Vehicle Control

One of the first interesting applications of machine learning to the problem of vehicle control was Pomerleau's ALVINN (Pomerleau, 1995). Pomerleau has designed a supervised learning system based on computer vision that featured a neural network which received, as inputs from the vision system, patterns representing the road ahead. The task of the network was to learn to match vision patterns to an accurate driving action. Examples were given by watching a real person driving.

The PATH program, through its Bayesian Automated Taxi (BAT) (Forbes et al., 1995) project has also studied the use of agents and machine learning for autonomous driving in traffic. They have shown that the use of a decision theoretic architecture and of dynamic Bayesian networks has produced a good solution to the problems of sensor noise and uncertainty about the other vehicles' behavior.

Later, Forbes also introduced a longitudinal agent controller (Forbes, 2002) based on reinforcement learning. This controller has been compared to a hand-coded controller, and results showed that the hand-coded controller was generally more precise than the learned controller, but was less adaptable in some situations.

Another interesting approach to longitudinal vehicle control was developed by Naranjo et al. (2003) as part of Spain's AUTOPIA project. Naranjo and his colleagues designed a longitudinal controller based on fuzzy logic. Their controller used inter-vehicle communication to share positioning information of a lead vehicle. It was even embedded in a vehicle and tested in demo sessions of the IEEE (Institute of Electrical and Electronics Engineers) Intelligent Vehicles Conference of 2002.

Machine Learning for Vehicle Coordination

The problem of coordination between vehicles has also received much interest from many researchers as this problem is especially adapted to multi-agent learning algorithms.

Among the numerous examples is work by Ünsal et al. (1999). These researchers have tackled the problem by using multiple stochastic learning automata as a mean to control the longitudinal and lateral motion of a single vehicle. Using reinforcement learning, these automata were able to learn to act in order to avoid collisions. The interactions between the automata have been modeled using game theory, with the objective of optimizing the traffic flow.

In his work, Pendrith (2000) presented a distributed variant of the Q-learning algorithm and applied it to a lane change advisory system. The author considered using a local perspective to gather state information, by considering the relative velocities of the surrounding vehicles. Whereas the solution provided by the algorithm increases the traffic efficiency, the problem of this algorithm is the lack of learning stability.

Moriarty & Langley (1998) proposed a traffic management approach where vehicles select by themselves the lane which optimizes the performance of the traffic flow. The authors have used a combination of reinforcement learning and neuro-evolution methods to keep a set of possible strategies for the vehicles. They have shown that their approach optimizes the velocities of the cars while reducing the number of lane changes.

Finally, Blumer et al. (1995) have used a neural network and an expert system to control vehicles from a coordination point of view (changing lanes, joining a platoon, etc.). The neural network was used to classify traffic situations and a reinforcement learning algorithm was used to evaluate the risk of the situation observed in order to choose the adequate action.

Agent Abstraction

Agents are autonomous software entities that try to achieve their goals by interacting with their environment and with other agents (Russell & Norvig, 2002). With their ability for autonomy and social interactions, agents are a logical choice of mechanism to rely on in order to embed in vehicles a deliberative engine adapted for control and collaboration. Indeed, this abstraction is especially adapted to the problem of collaborative driving that we address here, as vehicle controllers must autonomously make decisions in a decentralized manner while interacting with other vehicles in order to reach their goals of optimizing safety and traffic flow efficiency.

The agent abstraction can also be used to model the driving task using a deliberative architecture, and many different approaches have already been considered. For example, Rosenblatt (1995) has proposed a framework based on a centralized arbitration of votes from distributed, independent, asynchronous decision making processes. This framework has been used for obstacle avoidance by vehicles. In related work, Sukthankar et al. (1998) have focused on tactical driving using several agents that are specialized on one particular task (e.g. change lane agent or velocity agent). A voting arbiter aggregates the recommendation of all agents to choose the best vehicle action. Similarly, work by Ehlert (2001) describes tactical driving agents based on the subsumption approach (Brooks, 1991) and uses behavioral robotics to consider the real-time aspects of the driving task.

A different agent architecture has been proposed by Hallé & Chaib-draa (2005). Their work features a deliberative architecture based on team work (Tambe & Zhang, 2000) and is used for platoon management. This approach relies on a three level architecture (Guidance, Management and Traffic) as in PATH's architecture. In Hallé and Chaib-draa's approach, each vehicle is assigned a specific role in the platoon (Leader, Follower, Splitter, etc.) according to its current task. They have also compared their approach to a centralized and a decentralized platoon and they have given advantages and disadvantages of each type of platoon organization.

Of course, the papers we have presented here on the use of machine learning and of the agent abstraction applied to vehicle control and coordination only represent an overview of what has been done in this field. Nonetheless, it clearly illustrates what can be done when applying agent abstraction and machine learning to vehicle control.

LEARNING AND AGENTS

The resolution of the problems of autonomous vehicle control and of collaborative driving using intelligent agents requires the use of methods that are adapted to making decisions in a complex environment. One important problem that agents must face is the presence, in most environments, of uncertainty. In recent years, reinforcement learning has gathered much interest for the resolution of such problems as it can be used in this context to obtain efficient control policies.

Thus, this section will briefly present the Markov Decision Processes (MDP) model and the corresponding reinforcement learning algorithms classically used to find an optimal solution for a single agent. Afterwards, we introduce multi-agent models and describe algorithms that can learn in situations where interaction and coordination between agents is possible.

Markov Decision Processes

To take action, autonomous agents rely on a deliberation mechanism to select the appropriate action to take according to the current perception of the environment. Since driving can be considered as a sequential task where decisions need to be taken at fixed intervals of time, the framework of Markov Decision Processes (MDPs) is an efficient candidate to model this problem. More precisely, MDPs are sequential decision problems in which the goal is to find the best actions to take to maximize the agent's utility (Sutton & Barto, 1998).

The Markov property is needed to find the optimal solution of an MDP via classic dynamic programming or reinforcement learning approaches. This property is satisfied if the current state of the agent encapsulates all knowledge required to make a decision. More precisely, an environment is said to be markovian if its evolution can be described only by the current state and by the current action of the agent.

The resolution of an MDP yields a policy, which is a function that maps states to actions and which actually represents the behaviour of the agents. When the dynamics of the system (represented by the probabilities of going from current state *s* to next state *s'* when taking action *a*) are known, it is possible to use the Value Iteration algorithm (Russell & Norvig, 2002) to obtain a policy that maximizes the expected reward that the agent can obtain when executing it from starting state *s* (this policy is called the optimal policy).

The Q-Learning algorithm is also particularly interesting for the resolution of an MDP. It is a model-free approach that enables an agent to learn to maximize its expected reward without the availability of the transition and the reward functions that both characterize knowledge of the environment. With this algorithm, the agent learns an optimal action policy simply by trying actions in the environment and by observing their results. This algorithm is based on the notion of Q-value *Q(s,a)* which represents the reward an agent can expect to obtain when it is in state *s* and selects action *a*.

The downside of these algorithms is that they face the "curse of dimensionality". This curse refers to the fact that the size of the state space (the number of *Q(s, a)* pairs) can grow exponentially with the number of variables contained in the states and with the number of possible actions. This renders convergence nearly impossible for complex problems. Moreover, the use of a Q-values table means that continuous environments cannot be treated and need to be discretized.

Policy-gradient algorithms can address some of these issues. Instead of updating a value function in order to obtain the optimal function, these algorithms work by updating directly a parameterized stochastic policy according to the gradient of a policy's performance with respect to the parameters (the performance of a policy is generally defined as the expected reward one can get by following this policy). The advantages of these methods are that they can easily treat continuous state variables and that there is no problem related to the growth of the state space. For more details, we refer the reader to both Baxter & Bartlett (2001) and Williams (1992), as these authors make a good overview of this family of learning algorithms.

Multiple Agents

When multiple agents are involved, their interactions need to be handled since each agent needs to take into account the actions of others for efficient action selection. Usually, we can distinguish two cases: cooperative interactions, where all agents share the same goals, and non-cooperative interactions, where agents may have different or even opposite goals. In this section, we will only focus on the cooperative case.

When several cooperative agents act in the same environment, a decentralized MDP (DEC-MDP) can be used to describe the interaction of these agents (Bernstein et al., 2002). DEC-MDPs adapt some concepts of MDPs to deal with multiple agents and partially observable domains. In DEC-MDPs, observations have a special property: each agent can observe only a part of the current system state and each joint observation corresponds to a unique system state. Note that in this model, any optimal solution maximizes the social welfare, i.e. the sum of all agent rewards.

As far as we know, there exists no reinforcement learning algorithm that can find an optimal solution of a DEC-MDP without knowing the model of the environment. All working algorithms are based on dynamic programming (Bertsekas, 2000) and can only solve problems of small size because the DEC-MDP model is known as being an intractable problem (Bernstein et al., 2002). However, when agents are able to exactly observe the global state of the environment, the Friend Q-Learning algorithm introduced by Littman (2001) allows building an optimal policy for all agents.

Notice that even if this algorithm converges to the optimal joint policy, agents need some information about the others in order to achieve a good coordination. In general, individual states, individual actions and sometimes individual rewards need to be transmitted by communication between agents so that they can learn good policies. This multi-agent learning algorithm will be used later as part of the layer that manages vehicle coordination.

DESIGN OF COLLABORATIVE DRIVING AGENTS

In this section, we present how agents making decisions based on reinforcement learning algorithms can be used to design an autonomous vehicle controller and a collaborative driving system. First, we present our architecture and the different layers it relies on to manage vehicle control. Then, we detail the design of both a low-level vehicle controller and a high-level coordination module. Finally, we describe the results we obtained by executing the policies learned for both modules.

Architecture Design

For the past thirty years, manufacturers have integrated classic Cruise Control (CC) systems into vehicles to automatically maintain a driver's desired cruising velocity. More recently, constructors have introduced Adaptive Cruise Control (ACC) systems that make use of sensors to detect the presence of obstacles in front of a vehicle (Bishop, 2005). These systems are designed to react automatically to obstacles by taking direct action on the vehicle to adjust its current velocity in order to keep a safe distance behind the preceding vehicle.

Cooperative Adaptive Cruise Control systems (CACC), which integrate the use of inter-vehicle communication in the control loop, are often seen as the next step towards autonomous control systems (Bishop, 2005). These systems use wireless communication for the broadcast of positioning, velocity, acceleration and heading information to other vehicles nearby, to improve the receiver's awareness of the environment. By providing this extra information that would normally be out of the range of standard sensors, communication helps vehicles make better driving decisions and increase both traffic efficiency and safety.

In particular, CACC systems benefit from the use of communication to assure the string stability of a group of vehicles. This expression signifies that vehicles do not propagate and amplify perturbations of a front vehicle's velocity. Thus, for example, vehicles do not have to brake more than the preceding vehicle when observing changes in velocity. Non-stability eventually leads to vehicles needing to brake to a stand-still in order to avoid collision, which is often what causes traffic jams. Sheikholeslam and Desoer (1990) have showed that communicating acceleration actions of preceding vehicles through inter-vehicle communication is necessary to observe the stability of a stream of vehicles separated by constant spacing.

The Cooperative Adaptive Cruise Control (CACC) architecture presented here is thus based on this previous work of the automotive industry on vehicle control. The system, which is described in more detail in work by Desjardins et al. (2007), is actually an autonomous, intelligent agent that takes decisions in order to control the vehicle. This agent relies on two layers for decision-making and on a communication module to interact with other vehicles.

The two control layers work at different abstraction levels yet are complementary at coordinating interactions and at achieving cooperation between vehicles. First, the *Coordination Layer* is responsible for the selection of high-level driving actions. It uses information from other communicating vehicles to select an action that is the best response it can take according to the other vehicles' actions in order to maximize local and global security and traffic efficiency criteria. When such an action has been chosen, the low-level vehicle controller, also named the *Action Layer*, is responsible for selecting an action that has a direct effect on the vehicle's actuators. Figure 1 shows how our CACC architecture acts as part of the basic control loop of the navigational system of a vehicle.

When the current low-level action has terminated (either by success of by failure), the *Action Layer* notifies the *Coordination Layer*. This termination is then broadcast to the neighborhood to inform other vehicles. When all neighbors of a vehicle have finished their respective action, the *Coordination Layer* is

Figure 1. CACC system architecture

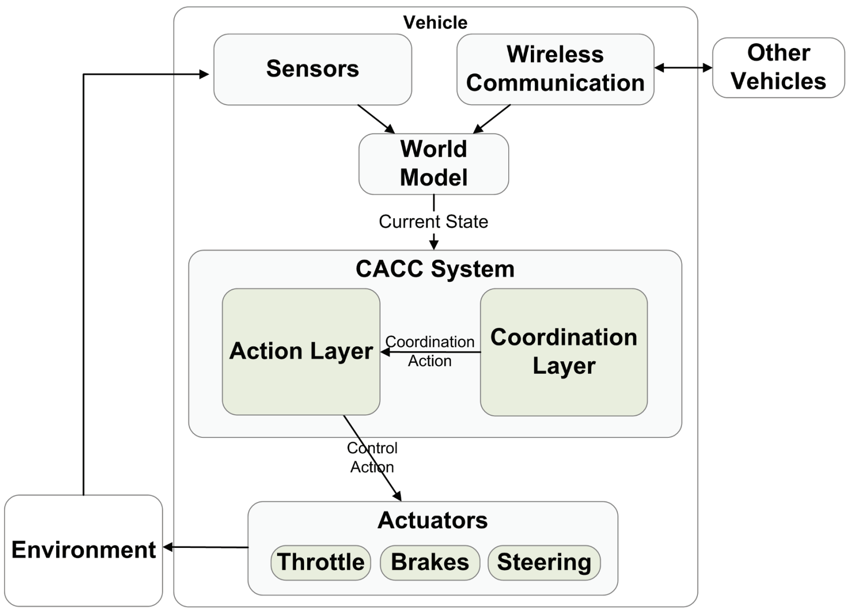
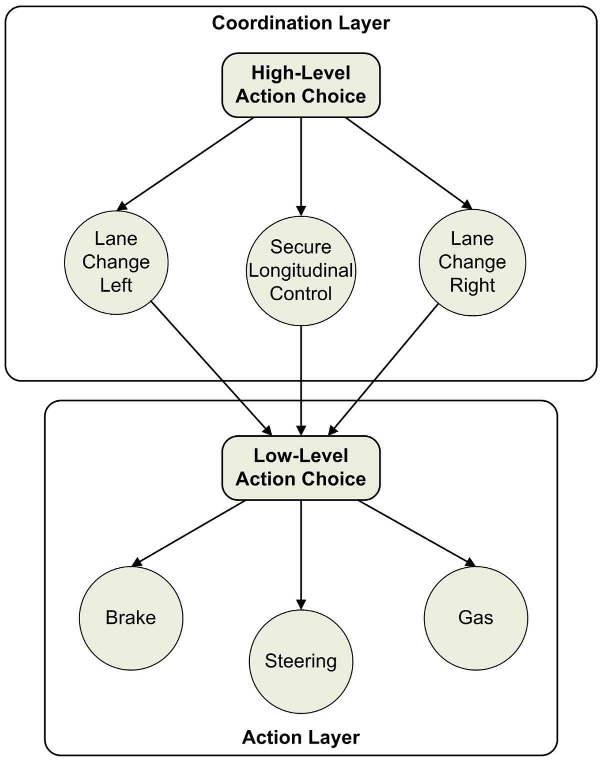


Figure 2. CACC system architecture interactions



able to take another coordination action according to the current state. The state diagram of Figure 2 illustrates the possible transitions that can be triggered by the *Coordination Layer* for a single vehicle.

The exact behavior of each layer has been designed using reinforcement learning algorithms. This learning approach is particularly useful since it allows the agent vehicle to adapt to its environment even if it does not know its dynamics. More specifically, the *Action Layer* uses algorithms to learn the selection of the best low-level actions according to the environment's state in order to achieve the high-level action selected, while the *Coordination Layer* uses learning to optimize the agent interactions.

In the following subsections, we present the design of both layers in detail.

Design of the Action Layer

For the design of our system's *Action Layer*, we have focused on offering a control policy that enables secure longitudinal velocity control. In particular, instead of solving directly the complex problem of Cooperative Adaptive Cruise Control, the work we present here tries to solve a simpler problem by designing an Adaptive Cruise Control (ACC) system, as we intend to show that our approach based on reinforcement learning can lead to good results.

First, we have considered for the inputs (state variables in the MDP framework) of the system the time headway, which gives the distance in time from a front vehicle (as illustrated in Eq. 1), and its difference between two timesteps, which indicates whether the follower has been closing in or going farther from its front vehicle (as given by Eq. 2).

Headway = Hw , Position = Pt , Velocity = V

$$Hw = \frac{(Pt_{Leader} - Pt_{Follower})}{V_{Follower}} \tag{1}$$

$$\Delta Hw = Hw_t - Hw_{t-1} \tag{2}$$

The headway information is perceived by a laser sensor, and detects vehicles in front in a range of up to *120* meters. Through our experiments, we make the hypothesis that there are no delays in the sensory system (as sensor delay will be addressed in future work).

Of course, this ACC state definition can easily be extended by using the communication system to propagate information about the state of surrounding vehicles (position, velocity, acceleration and heading). More specifically, we would like to integrate information about a lead vehicle's acceleration as inputs of this process, so that our system becomes a fully-functional Cooperative Adaptive Cruise Control (CACC) system.

For both the ACC and CACC cases, we will compute the control policies using reinforcement learning. This kind of learning is advantageous and efficient since it enables us to make an abstraction of the vehicle physics but still learn a valuable control policy. This is particularly useful when learning a control policy in an environment containing complex vehicle physics similar to the one used for our experiments (which we briefly detail at the beginning of the Results section).

The reward function we use gives negative rewards when the vehicle is too far from or too close to a secure distance (*2* seconds, a common value in ACC systems (Bishop, 2005)). Positive rewards are given when the vehicle is in the desired range. To direct the exploration of the vehicle to interesting places of the state space, we also give a positive reward to the vehicle if it is too far from the goal but is closing up.

An interesting characteristic of this learning task is that the choice of these state variables was carefully considered. As a result, the behaviour learned does not depend on the current velocity of the vehicles and should generalize to any driving scenario. The only fixed aspect of the controller that would not change with different scenarios is the distance from which the vehicle is following, which depends on the goal region defined by the reward function.

Finally, we also design manually a basic lane change policy, which can be triggered whenever needed by the vehicle *Coordination Layer*. The design of this layer is described in detail in the next section.

Design of the Coordination Layer

The goal of vehicle coordination is to handle dynamically the interactions between cars on the road in order to obtain an intelligent collaborative driving system. To achieve this, the *Coordination Layer* uses policies defined by the *Action Layer* and chooses at each step which policy should be applied in order to improve the coordination. In this subsection, we describe the method we considered for the design of coordination policies. To solve this problem, we use multi-agent learning algorithms and DEC-MDP models, and we introduce the notion of distance of observation between vehicles. Basically, with communication and sensors, each vehicle only has a limited view of its surrounding environment, and can choose an action which will give good coordination results.

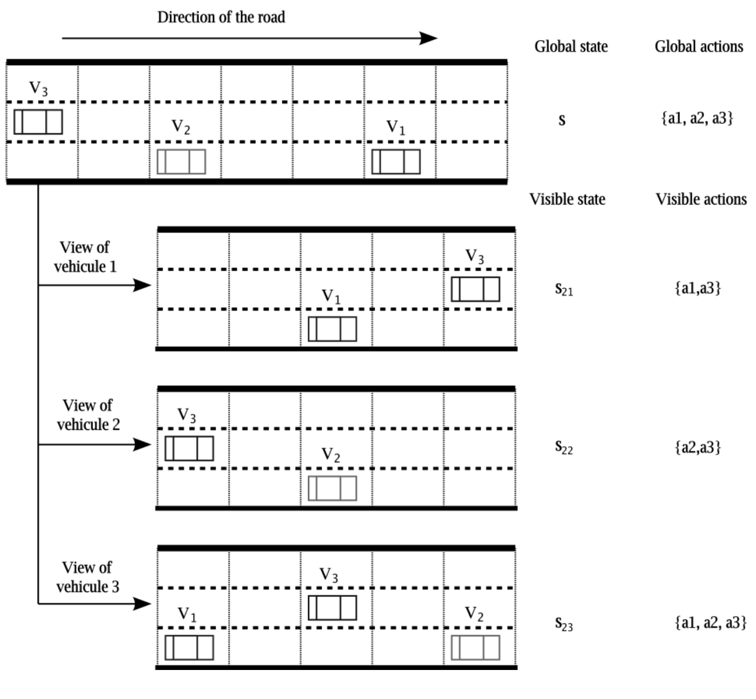
More formally, based on the DEC-MDP model described previously, we make assumptions about the observations of the vehicles, splitting these into two categories: observations over world states and observations over actions. Each observation is assumed to be perfect but only for a sub-part of the environment. Moreover, each agent only has a partial view of the other agents and cannot perceive the complete environment to learn the optimal actions. To define these partial views, we define a neighborhood function *neigh*, which returns the set of visible agents at a certain distance of observation from a central agent. Thus, observations are defined by the union of the exact states of visible agents. By this formulation, we assume that the need for coordination is higher when two agents are close than when they are far from each other. We also assume that every agent is in its own neighborhood and if an agent is in the neighborhood of another, the opposite is also true. Note that a maximal distance d_{max} is reached when each agent can observe all other existing agents.

Applied to the vehicle coordination problem, the functions calculating the partial state and the joint action are defined by the sensors and the communication of the vehicles. Figure 3 shows the partial view (state and action) for each vehicle where the global environment state is composed of 3 vehicles V_1, V_2 and V_3. In this figure, s_i^2 represents the partial vision of the vehicle *i*, which explains why the view s_3^2 is centered on Vehicle 3. Since the road is modeled as a ring, Vehicle 3 can observe Vehicle 2 in front of it and can observe Vehicle 1 behind it. At each step, the agent receives the information from other vehicles (velocities, positions) and the actions that have been chosen for the next step of the interaction.

Once all information needed to construct a partial state and joint action is received, the *Coordination Layer* decides to act by sending its command to the low-level vehicle controller. A vehicle can be ordered to follow the preceding vehicle, to keep a constant velocity or to change lanes to the right or to the left. All these actions correspond to the policies offered by the *Action Layer*.

Since the resolution of a DEC-MDP is known as an intractable problem, we will rather present an algorithm which finds an approximated joint policy using the distance of observation. Our algorithm,

Figure 3. Joint and partial states of a vehicle coordination scenario for a distance of observation of 2



called Partial Friend Q-Learning (PFQ), is based on Friend Q-Learning (Littman, 2001), a multi-agent version of Q-Learning. The basic idea is to apply Friend Q-Learning on partial views and partial joint actions instead of on fully observable states and joint actions to limit the number of possible *Q(s,a)* pairs. At each step, the agent chooses its action contained in the joint action that maximizes the Q-Value in the current state. Then, it observes partial states, partial joint actions and rewards, and updates the Q-Value as usual. In the end, the algorithm computes a policy π^d for each agent and for a fixed distance *d*. Further details of the coordination approach can be found in Laumônier & Chaib-draa (2006). From a vehicle coordination point of view, this algorithm allows us to take into account only a limited part of the environment by neglecting the influence of cars farther away. Thus, changes in the environment far away have no influence on the resulting policy.

Results

To test our architecture, we designed a microscopic traffic simulator in which vehicles are accurately modeled. It features vehicle physics and dynamics based on a single-track model (Kiencke & Nielsen, 2000). This model integrates both longitudinal and lateral vehicle movements and uses a wheel model that is complex enough to simulate with precision the behavior of a vehicle.

The simulator also includes an inter-vehicle communication system and a sensory system in order for vehicles to perceive their environment. The inter-vehicle communication module is a pre-requisite to an efficient CACC system as it makes possible extensive cooperation between vehicles. Both the *Action Layer* and the *Coordination Layer* rely on this module to share information and achieve good performance. The communication layer is loosely based on the DSRC protocol, which addresses many issues related to wireless inter-vehicle communications.

Actions of the vehicles in the simulator are controlled by acting directly on their actuators. This means that the longitudinal actions available to vehicles are to accelerate or to brake by pressing on the corresponding pedal. It is also possible not to take an action at the current time. As for the use of the steering wheel, it leads to the possible lateral actions of the vehicle. Before selecting a driving action, the *Action* and *Coordination Layers* can use sensors and communication to perceive the environment. We make the hypothesis that there are no delays or noise in the system whether it is from sensors, actuators or communication. As explained in the conclusion below, taking care of the issues of sensor delay and noise will be addressed in the following steps of the development of our architecture.

This simulation environment was used for learning control and coordination policies for both the *Action* and *Coordination Layers* of our system. How these experiments were done exactly for each layer is described in the following sections.

Vehicle Control

The *Action Layer* used for low-level vehicle control is designed using reinforcement learning. To obtain a control policy, we put the controller in "learning mode" in our simulated environment.

We tested a "Stop & Go" scenario where a leading vehicle accelerates to a velocity of *20 m/s*, slows down to *7 m/s* and then accelerates again, this time to a *20 m/s* cruising velocity. Our learning agent had to try actions in order to find the best longitudinal following policy. The goal was to reach a secure distance of *2* seconds behind a preceding vehicle, using only a front sensor, which effectively models the behavior of an ACC system.

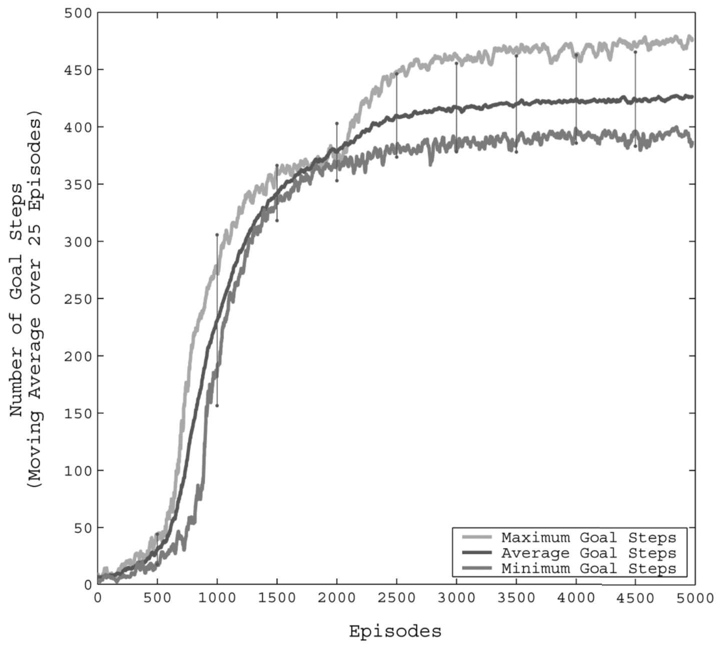
The learning task definition corresponded exactly to what was presented in the "Design of the Action Layer" section. Experiments to learn an efficient control policy have been done using the OLPOMDP policy-gradient algorithm (Baxter & Bartlett, 2001). This reinforcement learning algorithm generates a stochastic parameterized policy (a policy that returns probabilities of selecting the actions in a particular state). To represent this policy, we have used a neural network, and the parameters of the policy are actually the weights of the network. As a result, the algorithm modifies the network's weights in order to increase the probability of selecting the actions that give us positive rewards.

Figure 4 illustrates data related to the execution of *10* learning simulations of *5,000* episodes. Since the algorithm is actually a stochastic gradient descent method, multiple learning simulations were needed in order to compare the resulting policies. Thus, the figure shows the worst, the average and the best policy obtained through the learning phase. Figure 4 also illustrates the fact that the learning algorithm did optimize the number of steps in which the vehicle is located in the desired "safe" region, as, by the end of the learning episodes, the vehicle is in the goal region for approximately *475* steps over *500*, which can be considered as a near-optimal behavior.

After the learning phase, we executed a "Stop & Go" scenario with two vehicles, the follower being controlled by using the learned ACC policy. Figure 5 illustrates the velocities of both vehicles during this execution scenario. This figure illustrates the fact that the learned policy was able to precisely match the velocity of the front vehicle, even when it did accelerate or brake.

Furthermore, Figure 6 shows the associated headway metric of the second vehicle during the execution scenario. It clearly shows that the learned policy resulted in an efficient behavior, with the headway oscillating closely around the desired value for the duration of the simulation.

Figure 4. ACC learning results



Work still needs to be done to achieve our goal of designing a complete longitudinal CACC controller but, for now, the results we have obtained with our Adaptive Cruise Control (ACC) system show that reinforcement learning can be used to provide efficient vehicle following controllers.

Vehicle Coordination

The PFQ algorithm has been tested on a simplified three vehicles scenario as described in Figure 3, where each vehicle had to choose the best lane in order to optimize the velocity of every vehicle. This coordination scenario uses simpler dynamics than the single track model. Moreover, we discretize the positions and velocities of the vehicles and, for each car, we note *Y* the longitudinal position (in meters, assuming that a car is a $1\ m^2$ square) and *X* the current lane. We discretize also the velocities to the set $V = 0, 4, 8, 12, 16, 20\ m/s$. Learning coordination allows us to design an efficient controller which can take into account the actions of the other vehicles situated at a close range. Here, we summarize these results to show that each vehicle only needs to observe a subset of the other vehicles, those that are close to itself, to learn a near-optimal coordination policy.

With the results of those simulations, we can compare empirically the performance of a coordination policy learned in a fully-observable environment (using Friend Q-Learning) with the performance of an approximated coordination policy learned using observations of a subset of the environment (using our approach, PFQ). Here, we compare the algorithms on two situations: the scenario S_1 is defined by size $X = 3$, $Y = 7$, by the set of velocities $V = 0, 4, 8, 12, 16, 20\ m/s$ and by the number of agents $N = 3$. In the second scenario S_2, we enlarge the number of lanes and the length of the road ($X = 5$, $Y = 20$, $V = 0, 4, 8, 12, 16, 20\ m/s$ and $N = 3$). Consequently, in these problems, the maximal distance that we can use to approximate the total problem is $d_{max} = 3$ for S_1 and $d_{max} = 10$ for S_2. In the initial state (Figure 3), ve-

Figure 5. ACC vehicle velocities

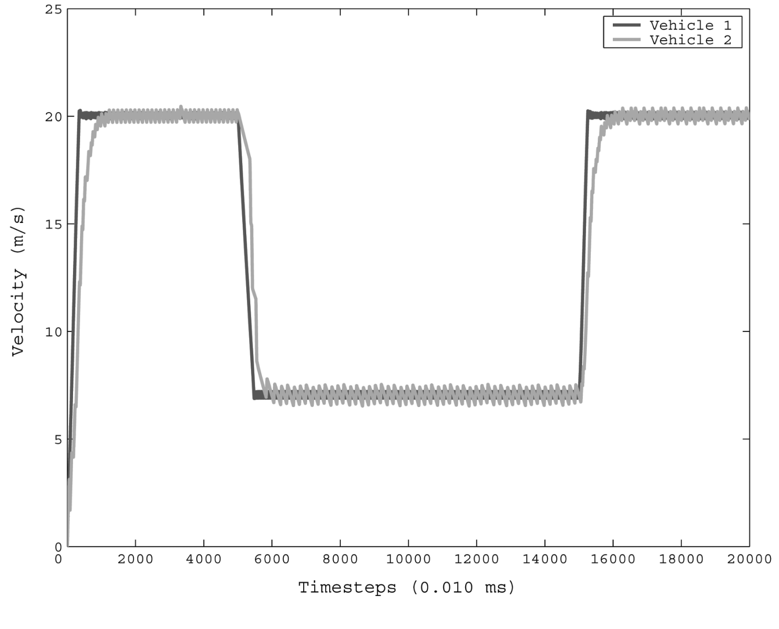
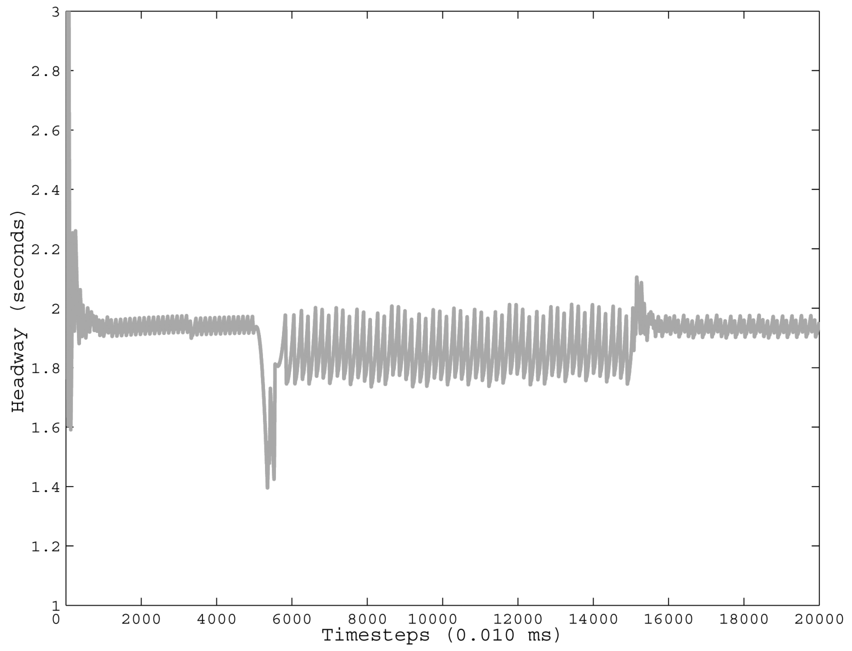


Figure 6. ACC headway results



locities of the agents are $V_1 = 4\ m/s$, $V_2 = 8\ m/s$ and $V_3 = 12\ m/s$. We present, for all results, the average velocity of all vehicles, averaged over *25* learning simulations, with each episode lasting *10* steps.

Figure 7 shows the results of PFQ with distance from $d = 0$ to $d = 3$. This algorithm is compared to the total observation problem resolved by Friend Q-learning. For $d = 0$, $d = 1$ and $d = 2$, PFQ converges to a local maximum, which increases with d. In these cases, the approximated values are respectively of *76%*, *86%* and *97%* of the optimal velocity. When $d = 3$, which is when the local view is equivalent to the totally observable view, the average velocity converges to the optimal average velocity. Thus, without observing everything around them (distance $d = 2$) vehicles are able to coordinate themselves and learn a near optimal policy while reducing the number of vehicles taken into account in the coordination.

Practically, the observation distance is determined by the distance of communication between vehicles. In general, using communication protocol like DSRC, the distance of communication depends on the density of vehicles. Indeed, in order to keep the number of messages relatively low, vehicles can only send to their close neighbors if there are many vehicles around. By doing this, we limit the number of vehicles taken into account in our reinforcement learning algorithms. This limitation is also coherent with the fact that current communication and sensor systems are not designed to handle the perception of remote vehicles. Consequently, we are able to design a coordination layer with good efficiency limiting the number of states in which the optimal policy should be found. The collaborative driving policy learned using a total distance of observation ($d = 3$) is represented by Figure 8. We can observe that, with this near optimal policy, Vehicle 3 learned to pass Vehicle 2 and Vehicle 1 learned to let Vehicle 2 to pass.

Figure 7. Velocity for partial friend q-learning

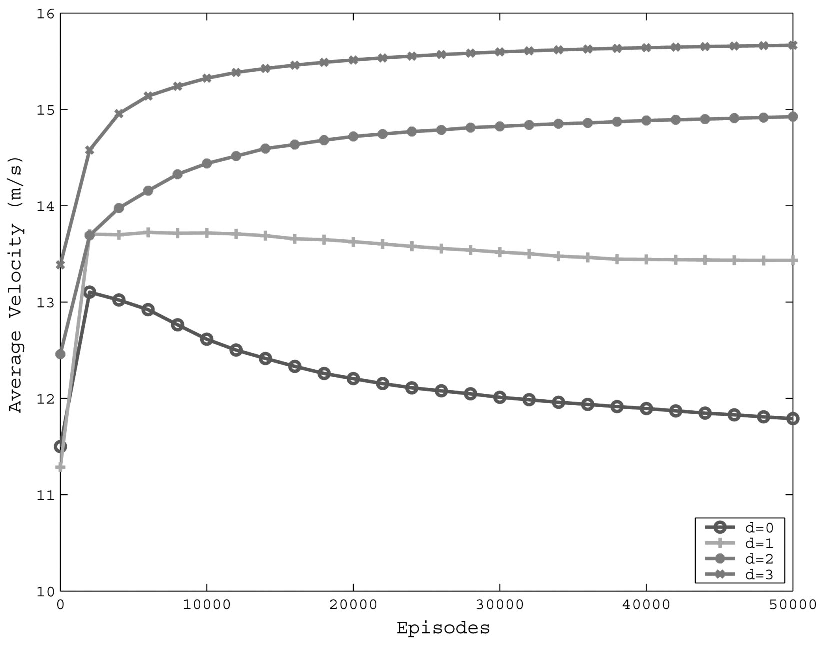
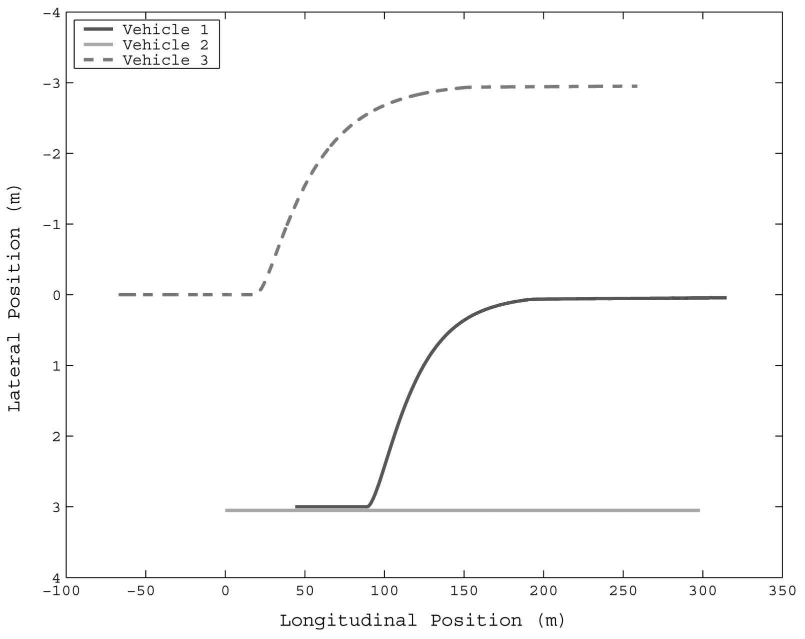


Figure 8. Coordination between 3 vehicles. Vehicle 3 learned to pass Vehicle 2 and Vehicle 1 learned to let Vehicle 2 pass.



CONCLUSION

In this chapter, we proposed a system for autonomous vehicle control and collaborative driving based on the use of agent technology and of machine learning. More specifically, we presented a multi-layered architecture that relies on both an *Action Layer* and a *Coordination Layer*: the *Action Layer* is used to manage low-level vehicle control actions such as braking, accelerating or steering, while the *Coordination Layer* is responsible for high-level action choice by integrating cooperative decision-making between vehicles. These two layers were designed using agent and multi-agent reinforcement learning techniques. Finally, we showed that the integration of reinforcement learning techniques at all levels of our autonomous driving controller gives efficient results for vehicle control and coordination. This approach clearly facilitates the efforts of the system's designer, as the complex details related to vehicle control and related to the numerous possibilities of inter-vehicle interactions are automatically handled by the learning algorithm.

Unfortunately, even though our approach was tested on a realistic vehicle dynamics simulator, we obviously did not take into account all of the requirements needed for the implementation of our system in a real vehicle. For example, we assumed the sensors of the vehicle to be perfect and without noise. In practice, however, sensors like GPS and lasers have limited precision. Obviously, this can lead to a degradation of the efficiency of the *Action* and the *Coordination Layers's* policies. Therefore, future work could consider solving this particular problem, which could be done by using Partially Observable Markov Decision Processes (POMDPs). This framework generalizes MDPs and can be used to find control policies under uncertainty and partial observability of the environment. Moreover, the control of the *Action Layer* should consider continuous actions instead of discrete ones in order to improve the efficiency of the vehicle following behavior. As for the *Coordination Layer*, experiments should be done on more complex scenarios in order to improve performance in high-density vehicular traffic. In this case, some approximation techniques could be considered in order to find an efficient coordination policy for a large number of vehicles.

REFERENCES

Auto21 (2007). Retrieved July 17th, 2008, from: http://www.auto21.ca/

Bana, S. V. (2001). *Coordinating Automated Vehicles via Communication.* PhD thesis, University of California at Berkeley, Berkeley, CA.

Baxter, J. & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 319–350.

Bernstein, D., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research, 27*(4), 819–840.

Bertsekas, D. P. (2000). *Dynamic Programming and Optimal Control, Vols. 1 & 2*, 2nd ed., Nashua, NH: Athena Scientific.

Bishop, W. R. (2005). *Intelligent Vehicle Technology and Trends*. Norwood, MA: Artech House.

Blumer, A., Noonan, J., & Schmolze, J. G. (1995). *Knowledge based systems and learning methods for automated highway systems.* (Technical Report), Waltham, MA: Raytheon Corp.

Broggi, A., Bertozzi, M., Fascioli, A., Lo, C., & Piazzi, B. (1999). The argo autonomous vehicle's vision and control systems. International *Journal of Intelligent Control and Systems, 3*(4), 409–441.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence Journal, 47*, 139–159.

de Bruin, D., Kroon, J., van Klaveren, R., & Nelisse, M. (2004). Design and test of a cooperative adaptive cruise control system. *In Proceedings of IEEE Intelligent Vehicles Symposium* (pp. 392–396).

Desjardins, C., Grégoire, P.-L., Laumônier, J., & Chaib-draa, B. (2007). Architecture and design of a multi-layered cooperative cruise control system. *In Proceedings of the Society of Automobile Engineering World Congress (SAE'07).*

Ehlert, P. A. (2001). *The agent approach to tactical driving in autonomous vehicle and traffic simulation.* Master's thesis, Knowledge Based Systems Group, Delft University of Technology, Delft, The Netherlands.

Forbes, J., Huang, T., Kanazawa, K., & Russell, S. J. (1995). The batmobile: towards a bayesian automated taxi. *In Proceedings of International Joint Conference on Artificial Intelligence* (pp. 1878–1885), Morgan Kaufmann.

Forbes, J. R. (2002). *Reinforcement Learning for Autonomous Vehicles*. PhD thesis, University of California at Berkeley, Berkeley, CA.

Godbole, D. N., & Lygeros, J. (1994). Longitudinal control of a lead car of a platoon. *IEEE Transaction on Vehicular Technology, 43*(4), 1125–1135.

Hallé, S. & Chaib-draa, B. (2005). A Collaborative Driving System based on Multiagent Modelling and Simulations. *Journal of Transportation Research Part C (TRC-C): Emergent Technologies, 13*(4), 320–345.

Hallouzi, R., Verdult, V., Hellendorn, H., Morsink, P. L., & Ploeg, J. (2004). Communication based longitudinal vehicle control using an extended kalman filter. *In Proceedings of International Federation of Automatic Control Symposium on Advances in Automotive Control* (pp. 745-750).

Kiencke, U., & Nielsen, L. (2000). *Automotive control systems: for engine, driveline and vehicle*. Berlin, Germany: Springer-Verlag.

Laumônier, J., & Chaib-draa, B. (2006). Partial local FriendQ multiagent learning: application to team automobile coordination problem. In L. Lamontagne and M. Marchand (Ed.), *Canadian AI, Lecture Notes in Artificial Intelligence* (pp. 361–372). Berlin, Germany: Springer-Verlag.

Littman, M. (2001). Friend-or-Foe Q-learning in General-Sum Games. In C.E. Brodley and A. P. Danyluk (Ed.), *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 322–328), San Francisco, CA: Morgan Kaufmann.

Lu, X.-Y., Tan, H.-S., Empey, D., Shladover, S. E., & Hedrick, J. K. (2000). *Nonlinear longitudinal controller development and real-time implementation* (Technical Report UCB-ITS-PRR-2000-15).

Los Angeles, CA: University of Southern California, California Partners for Advanced Transit and Highways (PATH).

Moriarty, D., & Langley, P. (1998). Learning cooperative lane selection strategies for highways. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 684–691), Menlo Park, CA: AAAI Press.

NAHSC (1998). *Technical Feasibility Demonstration Summary Report (Technical Report).* Troy, MI, USA: National Automated Highway System Consortium.

Naranjo, J., Gonzalez, C., Reviejo, J., Garcia, R., & de Pedro, T. (2003). Adaptive fuzzy control for inter-vehicle gap keeping. *IEEE Transactions on Intelligent Transportation Systems, 4*(3), 132–142.

Pendrith, M. D. (2000). Distributed reinforcement learning for a traffic engineering application. C. Sierra, M. Geni and J.S. Rosenschein (Ed.), *Proceedings of the Fourth International Conference on Autonomous Agents* (pp. 404-411), New York, NY: ACM Press.

Peng, H., bin Zhang, W., Arai, A., Lin, Y., Hessburg, T., Devlin, P., Tomizuka, M., & Shladover, S. (1992). *Experimental automatic lateral control system for an automobile (Technical Report UCB-ITS-PRR-92-11)*, Los Angeles, CA: University of Southern California, California Partners for Advanced Transit and Highways (PATH).

Pomerleau, D. (1995). Neural network vision for robot driving. In S. Nayar and T. Poggio (Eds.), *Early Visual Learning* (pp. 161-181), New York, NY: Oxford University Press.

Raza, H., & Ioannou, P. (1997). *Vehicle following control design for automated highway systems (Technical Report UCB-ITS-PRR-97-2)*, Los Angeles, CA: University of Southern California, California Partners for Advanced Transit and Highways (PATH).

Rosenblatt, J. K. (1995). DAMN: A distributed architecture for mobile navigation. In H. Hexmoor and D. Kortenkamp (Eds.), *Proceedings of the American Association of Artificial Intelligence Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Menlo Park, CA: AAAI Press.

Russell, S. J., & Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Schulze, M. (2007). *Summary of chauffeur project.* Retrieved July 17th, 2008, from http://cordis.europa.eu/telematics/tap_transport/research/projects/chauffeur.html

Sengupta, R., Rezaei, S., Shladover, S. E., Cody, D., Dickey, S., & Krishnan, H. (2007). Cooperative collision warning systems: Concept definition and experimental implementation. *Journal of Intelligent Transportation Systems, 11*(3), 143–155.

Sheikholeslam, S. & Desoer, C. A. (1990). Longitudinal control of a platoon of vehicles. *In Proceedings of the American Control Conference, 1,* 291–297).

Shladover, S. E. (2007). Path at 20 – history and major milestones. *IEEE Transactions on Intelligent Transportation Systems, 8*(4), 584–592.

Sukthankar, R., Baluja, S., & Hancock, J. (1998). Multiple adaptive agents for tactical driving. *Applied Intelligence, 9*(1), 7–23.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press.

Tambe, M., & Zhang, W. (2000). Toward flexible teamwork in persistent teams: extended report. *Journal of Autonomous Agents and Multi-agents Systems, 3*, 159–183.

Tsugawa, S. (2005). Issues and recent trends in vehicle safety communication systems. *International Association of Traffic and Safety Sciences Research, 29*, 7–15.

Tsugawa, S., Kato, S., Matsui, T., Naganawa, H., & Fujii, H. (2000). An architecture for cooperative driving of automated vehicles. *In Proceedings of IEEE Intelligent Transportation Systems* (pp. 422–427).

Ünsal, C., Kachroo, P., & Bay, J. S. (1999). Simulation study of multiple intelligent vehicle control using stochastic learning automata. *IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans, 29*(1), 120–128.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*, 229–256.