CHARLES DESJARDINS

Cooperative Adaptive Cruise Control: A Learning Approach

Mémoire présenté
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès sciences (M.Sc.)

Faculté des sciences et de génie
UNIVERSITÉ LAVAL
QUÉBEC

Janvier 2009

©Charles Desjardins, 2009

Résumé

L'augmentation dans les dernières décennies du nombre de véhicules présents sur les routes ne s'est pas passée sans son lot d'impacts négatifs sur la société. Même s'ils ont joué un rôle important dans le développement économique des régions urbaines à travers le monde, les véhicules sont aussi responsables d'impacts négatifs sur les entreprises, car l'inefficacité du flot de traffic cause chaque jour d'importantes pertes en productivité. De plus, la sécurité des passagers est toujours problématique car les accidents de voiture sont encore aujourd'hui parmi les premières causes de blessures et de morts accidentelles dans les pays industrialisés. Ces dernières années, les aspects environnementaux ont aussi pris de plus en plus de place dans l'esprit des consommateurs, qui demandent désormais des véhicules efficaces au niveau énergétique et minimisant leurs impacts sur l'environnement.

Évidemment, les gouvernements de pays industrialisés ainsi que les manufacturiers de véhicules sont conscients de ces problèmes et tentent de développer des technologies capables de les résoudre. Parmi les travaux de recherche en ce sens, le domaine des Systèmes de Transport Intelligents (STI) a récemment reçu beaucoup d'attention. Ces systèmes proposent d'intégrer des systèmes électroniques avancés dans le développement de solutions intelligentes conçues pour résoudre les problèmes liés au transport automobile cités plus haut.

Ce mémoire se penche donc sur un sous-domaine des STI qui étudie la résolution de ces problèmes grâce au développement de véhicules intelligents. Plus particulièrement, ce mémoire propose d'utiliser une approche relativement nouvelle de conception de tels systèmes, basée sur l'apprentissage machine. Ce mémoire va donc montrer comment les techniques d'apprentissage par renforcement peuvent être utilisées afin d'obtenir des contrôleurs capables d'effectuer le suivi automatisés de véhicules. Même si ces efforts de développement en sont encore à une étape préliminaire, ce mémoire illustre bien le potentiel de telles approches pour le développement futur de véhicules plus "intelligents".

Abstract

The impressive growth, in the past decades, of the number of vehicles on the road has not come without its share of negative impacts on society. Even though vehicles play an active role in the economical development of urban regions around the world, they unfortunately also have negative effects on businesses as the poor efficiency of the traffic flow results in important losses in productivity each day. Moreover, numerous concerns have been raised in relation to the safety of passengers, as automotive transportation is still among the first causes of accidental casualties in developed countries. In recent years, environmental issues have also been taking more and more place in the mind of customers, that now demand energy-efficient vehicles that limit the impacts on the environment.

Of course, both the governments of industrialized countries and the vehicle manufacturers have been aware of these problems, and have been trying to develop technologies in order to solve these issues. Among these research efforts, the field of Intelligent Transportation Systems (ITS) has been gathering much interest as of late, as it is considered an efficient approach to tackle these problems. ITS propose to integrate advanced electronic systems in the development of intelligent solutions designed to address the current issues of automotive transportation.

This thesis focuses on a sub-field ITS since it studies the resolution of these problems through the development of Intelligent Vehicle (IV) systems. In particular, this thesis proposes a relatively novel approach for the design of such systems, based on modern machine learning. More specifically, it shows how reinforcement learning techniques can be used in order to obtain an autonomous vehicle controller for longitudinal vehicle-following behavior. Even if these efforts are still at a preliminary stage, this thesis illustrates the potential of using these approaches for future development of "intelligent" vehicles.

Avant-propos

Par la présente, j'aimerais remercier tous ceux qui m'ont supporté dans mon travail de recherche au cours des deux dernières années.

Tout d'abord, je me dois de souligner le support qui m'a été offert par mon directeur de recherche, M. Brahim Chaib-draa. Ce dernier a toujours été présent pour répondre à mes questions et pour me faire progresser dans mes travaux.

Évidemment, je dois aussi souligner l'apport des étudiants du DAMAS pour les idées et les discussions. En particulier, je remercie Julien Laumônier et Pierre-Luc Grégoire, avec qui j'ai été en étroite collaboration relativement au projet AUTO21.

Finalement, je remercie ma famille: Yves et Louise, Philippe et Hélène, ainsi qu'Éloi et Clément, pour leur encouragement et leur support inconditionnel. De plus, je remercie mes amis, plus particulièrement Alex, Alex et Ghis, ainsi que les gars de Plum Shark, qui m'ont permis de me changer les idées lorsque c'était le temps.

Charles Desjardins

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Rarely a new technology has revolutionized our way of life as much as the invention of the automobile. Numbers speak for themselves when describing the importance of the emergence of motor vehicles as a mean of transportation: the estimated number of registered vehicles reached an astonishing 247,421,120 cars in the U.S in 2005 [Bureau of Transportation Statistics, 2007]. In Canada the number of registered vehicles in 2006 is evaluated at 19,499,843 [Statistics Canada, 2006]. Not only are those North American numbers impressive, but the global number of vehicles on the planet is also bound to increase dramatically in the next decades, with emerging countries such as China and India rapidly embracing automotive transportation systems.

Over the past 50 years in industrialized countries, the growth in the use of automotive transportation and of personal mobility technologies has had a deep and fundamental impact on our way of life. It dictated the geographical organization of urban regions and was responsible for the apparition of many phenomenons such as: (i) urban sprawl; (ii) traffic jams; (iii) atmospheric pollution.

Yet, the biggest impact of automotive transportation technologies on society has been economical. Today, the production and manufacture of many goods at an affordable cost is in part based around the fact that automotive transportation can help carry materials and products to global locations. As a result, the efficiency of the traffic flow has become a primary concern for many businesses as it is closely related to the efficiency of their supply chain and to the delivery of their products to the marketplace in a timely manner.

Over the years, society has thus become more and more dependent on the availability of vehicular technology. We will see, in this chapter, that the ever growing number of vehicles resulting from this dependence has brought its lot of negative aspects that affect our everyday lives.

This introductory section provides an overview of automotive research and development that focuses on addressing the issues that plague today's vehicular transportation. First, Section 1.1 will describe the problems faced by the industry and will detail their impact on society. Then, Section 1.2 details how research in many different fields of study has provided possible solutions to these issues. Particularly, we will focus on recent research in Intelligent Transport Systems (ITS), and more specifically on the development of Intelligent Vehicle (IV) technologies, as these systems offer interesting solutions to many of the obtacles faced by automotive transportation. Section 1.3 will briefly describe how modern artificial intelligence and machine learning techniques can be applied to the design of Intelligent Vehicles. Afterwards, Section 1.4 will describe the AUTO21 research project as part of which the work described in this thesis was conducted. Finally, we conclude this introductory chapter by detailing the objectives of this thesis (Section 1.5) and its organization (Section 1.6).

1.1 Automotive Transportation Issues

As briefly mentionned in the previous section, current automotive transportation systems face many obstacles that, unfortunately, have important repercussions. This section will describe these problems and will illustrate their effects on today's society.

1.1.1 Traffic Flow Efficiency

A first important concern faced by automotive transportation is related to the increase in the number of vehicles on the road. Of course, the fact that more and more vehicles have been sharing the limited road network has significant consequences on the efficiency of the traffic flow. However, the limited ressources of the road network is not the only cause of the overall degradation of the traffic flow in urban areas. For instance, consider the traffic jams that we can often observe, around crowded off-ramps, even when there are no accidents. This stems from the fact that human drivers, obviously, have slow reaction times. When a driver perceives that surrounding vehicles are braking, the time he takes to react often means that he will have to break harder than the vehicle it follows in order to keep a secure distance. This delay will propagate back and amplify when going through a string of vehicles with the effect that, further down the stream, vehicles have to slow down considerably and might even have to stop completely in order to avoid collision. Thus, through dense traffic, only a small slowdown can ultimately amplify up to a point where it generates a traffic jam.

Moreover, since humans are limited in their ability to anticipate changes in their environment, drivers often select sub-optimal actions due to an inappropriate assessment of a situation. This, again, has negative effects on the traffic conditions.

Obviously, the poor efficiency of the traffic flow results in a number of negative economical and societal impacts. According to a report from Transport Canada [2007], the costs related to congestion delays in major urban areas of Canada is estimated at somewhere between 2.0 to 3.3 billion dollars in 2002. In the United States, a report from the Texas Transportation Institute [Schrank and Lomax, 2007] estimates the number of hours lost due to congestion in urban areas at 4.2 billion hours for 2005. It is obvious that the availability of technologies designed to increase the efficiency of the traffic flow would be highly beneficial and are thus urgently sought.

1.1.2 Environment

Another important issue linked to the increase in the number of vehicles on the road is the negative environmental impacts of automotive transportation. Indeed, vehicles are a major determinant of atmospheric pollution. Gas emissions from vehicular traffic is generally considered as the main factor in the presence of smog in cities. Since episodes of smog have been linked to respiratory diseases, it is clear that a reduction in gas emissions would be beneficial for the health of many citizens.

Of course, the negative influence of automotive transportation on the environment can directly be linked to the poor efficiency of the traffic flow. Gas emissions from vehicles' idle engines in traffic jams contribute to the degradation of the air quality, and statistics on the matter clearly illustrate these effects. In 2005, the quantity of wasted fuel due to congestion in urban areas in the United States has been evaluated at over 2.9 billions of gallons [Schrank and Lomax, 2007]. In Canada, Transport Canada [2007] estimates the quantity of wasted fuel due to traffic jams at 500 millions liters for the year 2002.

Clearly, the reduction of emissions, consequent of an increase in the efficiency of the traffic flow, would go a long way towards cleaner air, a greener environment, and an increase in the quality of life of citizens of urban regions around the world.

1.1.3 Safety

It is well known that the most important issue that automotive transportation technologies face is related to safety. Every year, a large number of vehicles are involved in accidents, resulting in an important number of casualties. These accidents often come from the fact that humans poorly anticipate changes in the driving environment, making it difficult for drivers to make appropriate choices when in dense traffic or in any kind of critical situation.

According to Transport Canada [2006], there has been $2,923$ deaths and $210,629$ injuries related to motor vehicle crashes in Canada, for the year of 2005 alone. Numbers for the United States are even more devastating, as the Bureau of Transportation Statistics reports a total number of fatalities of $43,443$ in 2005, and evaluates the number of persons injured in crashes at $2,699,000$ during the same period [Bureau of Transportation Statistics, 2007].

The current trend in these statistics show that the ratio of the number of injuries and fatalities over the number of vehicles on the road has constantly been decreasing over the past 20 years, which reflects efforts by manufacturers and governments towards increasing safety. However, a lot of research and development still needs to be carried regarding safety in automotive transportation, as fatalities resulting from vehicular accidents still represents one of the main cause of injuries and deaths in modern societies.

1.2 Solutions

As we have just seen, vehicular transportation has significant effects on today's society, and finding solutions to these problems remains on the top of the agenda of both the governments of developped countries and of the automotive industry. By passing legislation enforcing higher safety and environmental standards for vehicles and by investing money in joint research and development activities with manufacturers, governments have made their share of efforts towards the resolution of current transportation problems. In response, the industry has focused on these issues by both conforming to these safety and environmental standards, and by proceeding to active research in many different fields of study that could potentially offer innovative solutions.

For instance, among the numerous solutions considered, research has been conducted on the development of alternative fuels. Moreover, research for alternative means of powering vehicles has also been popular. Future technologies such as the electric or hydrogen powered car will surely play an important role in the reduction of greenhouse gas emissions resulting from our current dependence on oil.

Another good example of an area where active research has recently been carried is in materials engineering. This field focuses on the development of lighter materials, which could result in a significant economy in gas consumption by reducing the weight of vehicles. Moreover, this field also tries to develop materials that have the ability to absorb, through deformation, the force resulting from an impact, ultimately improving the safety of passengers.

Clearly, this brief overview of two automotive transportation research areas illustrates how the industry has proposed solutions in order to address the issues of modern automotive transportation. In this thesis, we focus on another key research area, named Intelligent Transportation Systems (ITS), which has stimulated significant interest in recent years. These systems propose to use modern sensing, communication and computational technologies in order to develop intelligent applications whose design goal is to solve some of the issues described previously.

In particular, the ITS sub-field of Intelligent Vehicles (IV) is currently the focus of active research as the integration of intelligent systems directly in vehicles could have many positive impacts. By sensing the environment and providing situation assessment, these systems could give valuable information to drivers, and help them select actions that could increase the efficiency of the traffic flow and diminish the possibility of accidents. This, in turn, could also have a positive effect on the environment by reducing the quantity of fuel wasted in traffic jams.

Yet, the development of IV technologies remains a complex problem, and numerous design approaches can be considered. As the next section will show, this thesis will focus on using modern machine learning techniques to obtain a IV system for autonomous vehicle control.

1.3 Machine Learning for Autonomous Vehicle Control

With its recent interest in solving complex control problems, the field of machine learning has been an interesting new-comer in IV research. Indeed, through the use of the agent abstraction and of reinforcement learning, modern machine learning is seen as an effective and elegant solution to the problem of autonomous vehicle control.

The first interesting aspect of modern machine learning resides in the fact that the agent abstraction applies itself naturally to the domain of autonomous vehicles. By definition, agents are autonomous and social entities that can interact with their environment in order to reach their design goals. Clearly, an autonomous vehicle can be seen as an agent that can sense its environment through the use of sensors, that can communicate with other vehicles through the use of wireless inter-vehicle communication and that can act to optimize safety and traffic efficiency metrics. Thus, the use of agents is obviously an efficient representation mechanism for the problem of autonomous vehicle control.

Another interesting aspect of modern machine learning is that recent research on the use of Markov Decision Processes (MDPs) has provided to agents a robust decision-making framework grounded on solid mathematical notions. In particular, MDPs are especially adapted for the modeling of sequential decision problems, making them a suitable representation for the resolution of numerous problems. Moreover, the popularity of MDPs in reinforcement learning research has resulted in the development of many algorithms that can solve learning tasks in different types of environments. More specifically, algorithms now exist to find efficient control policies for model-free learning tasks, which are useful when the agent does not know the exact dynamics of the system. With these algorithms, agents can learn by direct interaction with their environment. Since the driving environment is complex and difficult to describe precisely using analytical mathematics, the use of model-free reinforcement learning algorithms to obtain a control policy becomes an interesting alternative.

Finally, the interest of the machine learning community for the automatization of driving is undeniable, as the complexity of this task offers great challenges to researchers. Among the issues that must be resolved in order to obtain efficient control policy is the problem of handling both continuous state and action spaces. Moreover, the fact that real sensors are noisy and have delayed perception are important factors that should also be taken into account during the learning process.

1.4 The AUTO21 NCE

AUTO21 [Auto21, 2008] is a Network of Centres of Excellence funded by the Canadian federal government. It promotes collaboration between the Canadian automotive industry and academic institutions, with the goal of enhancing Canada's competitivity on the global automotive market. It aims at building knowledge by forming highly qualified personel in many different fields related to automotive research and development. This network regroups 43 universities across Canada and over 240 industry and government partners working on 54 research projects gathered under 6 specific themes.

The work presented in this thesis was done as part of Theme F, entitled *Intelligent Systems and Sensors*. As its name implies, this theme focuses on research and development related to ITS, and regroups many different projects ranging from development of sensors, GPS and communication systems to their integration into complete autonomous and intelligent applications. More specifically, our work was done under the *Vehicle Communications and Applications* project. This project focuses on the development of wireless vehicle-to-vehicle (V2V) and road-to-vehicle (R2V) communication protocols and on their integration in intelligent transportation and vehicle control systems.

This thesis was conducted at the DAMAS Laboratory [DAMAS, 2008], a research team from the Computer Science and Software Engineering Department at Laval University involved, for this AUTO21 project, in the development of applications of communication protocols into intelligent systems using machine learning techniques.

1.5 Thesis Objectives

It is under the premises exposed in this introductory section that the current thesis studies the use of machine learning techniques applied to the domain of Intelligent Transportation Systems. More precisely, it investigates the ways that reinforcement learning algorithms can be used to learn, by direct interaction with the environment, a control policy for a Cooperative Adaptive Cruise Control (CACC) system designed for secure following of a front vehicle. To achieve this, we will have to:

- survey previous work in the field of autonomous vehicle control;
- study classic reinforcement learning approaches;
- study methods to handle continuous state variables in reinforcement learning;
- develop a vehicle simulator to conduct learning experiments;
- experiment with the learning framework (state space definition, algorithm parameters, etc.) in order to obtain an efficient control policy.

1.6 Thesis Organization

This thesis is organized as follows. Chapter 2 focuses on Intelligent Transport Systems and Autonomous Vehicle Control Systems research. It reviews current research done in this field and also details previous work in machine learning for the resolution of various control problems. Next, Chapter 3 focuses on the theory behind the machine learning techniques we plan to use to obtain our automated vehicle control policies. It also reviews reinforcement learning theory and details value function approximation and policy-gradient algorithms that can be employed in order to handle continuous state variables. Chapter 4 details the implementation of a vehicle simulator designed to learn vehicle control policies. Afterwards, Chapter 5 presents the learning tasks considered and the learning algorithms used to obtain the control policies. It also describes the results we obtained. Finally, Chapter 6 concludes and specifies some of the future work that could be done to improve our approach.

Chapter 2

Intelligent Vehicles

This second chapter surveys what has been done in the field of Intelligent Vehicles (IV). First, we will define the scope of this research topic and how it relates to the larger field of Intelligent Transportation Systems (ITS). Next, we will describe IV research projects that have focused on the development of Autonomous Vehicle Control Systems (AVCS). Finally, we will also survey some of the work done to design vehicle control systems using the software agent abstraction and machine learning algorithms.

2.1 Intelligent Transportation Systems

As briefly discussed in the introductory chapter, most industrialized countries have recently adopted road-maps detailing the future of their investments in Intelligent Transportation Systems (ITS) research in response to the problems related to the increase of vehicular traffic. This focus was motivated by the fact that, by definition [ITS Canada, 2008], ITS are systems that make use of modern technologies to improve the security and the efficiency of vehicular transportation.

To achieve this, ITS rely on the use of information acquisition technologies to observe the surrounding environment and to gather information about its current state. Thus, many lasers, ground detectors and cameras are currently used by ITS to sense their surroundings. Other environment sensing possibilities are on the horizon, with the growing development of computer vision systems. As a result, a whole range of modern sensing technologies will soon be able to provide mapping of a dynamical environment with enough precision that numerous advanced applications will become possible, even, in some cases, for handling safety-critical situations.

Future ITS will also depend on information sharing between actors involved in the system. Indeed, thanks to recent advances in wireless communication technologies, it will soon be possible for vehicles to communicate efficiently with the infrastructure (using both road-to-vehicle (R2V) or vehicle-to-road (V2R) communications) but also with other vehicles (using vehicle-to-vehicle (V2V) communication). Clearly, the availabitity of inter-vehicle communications will be a pre-requisite for future cooperative systems as they will offer to vehicles extended awareness of their environment that is needed to make autonomous driving decisions.

Computers are also an important part of current and future ITS, as they offer the processing power needed to analyze the data gathered by both sensors and communication systems. In particular, computers enable the development of autonomous decision-making engines that can compute an adequate course of action according to the current state of the environment.

Many countries already use ITS for various purposes. Among the existing applications of ITS in Canada [Fu et al., 2003] is, for example, the adaptive traffic signal controller used by the city of Edmonton to monitor the current state of the traffic flow on one of its most important artery. After having observed the traffic conditions, the system selects an adequate response by choosing a traffic-light pattern to apply from a bank of pre-designed patterns. Another example is the D.R.I.V.E.S. system which is currently active in the cities of Toronto [Globis Data Inc., 2008b] and Montreal [Globis Data Inc., 2008a]. This system is a real-time traffic information tool, available on the web, that displays on a map the state of the traffic flow on the surrounding highways. Users can rapidly observe the areas where there is significant delay, and can adjust their route accordingly.

Other ITS applications include a number of similar intelligent technologies, going from electronic highway toll management to real-time parking availability information. In the next section, we will see that ITS research has also focused on the integration of intelligent systems inside vehicles.

2.2 Intelligent Vehicles

There is a consensus among actors of the automotive transportation on the fact that the current issues of transportation systems are directly related to the limitations of human drivers in sensing and decision-making in complex dynamical environments. As a result, the automatization of the driving task in the future is considered as inevitable, as it would solve these issues by reducing the cost in human lives of traffic accidents and by optimizing the efficiency of the traffic flow.

Thus, part of ITS research has been dedicated to embedding intelligent systems directly inside vehicles. Like ITS systems, Intelligent Vehicles use existing sensors, such as lasers, to provide detection of obstacles around the vehicle, while current GPS systems also offer geographical information about the position of a vehicle. On top of this advanced sensor technology, Intelligent Vehicles will soon be able to use wireless systems for vehicle-to-vehicle communication, which will serve as a mean to share information about the environment and extend the range of current sensors. Of course, inter-vehicle communications will be a pre-requisite for most safety applications, as they offer a more accurate representation of a vehicle's surroundings. Indeed, the extended information brought by communication will be used by intelligent vehicles in order to deliberate on the selection of the best action to take according to safety and traffic efficiency concerns.

Formally, Bishop [2005] defined Intelligent Vehicles (IV) as systems that sense the environment and provide either information, direct vehicle control or both, to assist the driver in optimal vehicle operation. IV systems operate at what is called the tactical level of driving, meaning that such control systems can act directly on the vehicle's throttle, brakes or steering. This contrasts with strategic decisions such as route choice, that are normally supplied by on-board navigation systems.

A lot of research is currently being done by vehicle manufacturers in order to implement Intelligent Vehicle technologies in consumer products. For the moment, existing IV systems are primarily designed as Advanced Driver Assistance Systems (ADAS). Such systems offer driving aids in the form of advice to drivers or through limited automatization of driving. ADAS are mostly characterized as "convenience systems", as they alleviate the burden of driving, often with a focus on increased safety.

Already, vehicle manufacturers have integrated some of these intelligent technologies in their products. For example, many luxury vehicles (BMW's 3, 5 and 7 Series [BMW, 2008], the Lexus LS430/460 [Lexus, 2008], Mercedes-Benz S-Class [Mercedes-Benz, 2008] and Jaguar's XK-R [Richardson et al., 2000], just to name a few) are now equipped with numerous assistance systems ranging from Adaptive Cruise Control (ACC) to automated parallel parking technologies. Descriptions of current driver-assistance applications (and future, in the case of the CACC system) are given in Table 2.1, as we detail Automated Parking, Collision Avoidance, Lane-Keeping Assistance, Adaptive Cruise Control and Cooperative Adaptive Cruise Control technologies.

Technology	**Description**
Automated Parking	This technology uses sensors and computer vision to sense the surroundings of the vehicle. The vehicle detects the available space, and can proceed to parallel parking by itself, using direct control of the vehicle's steering, while throttle and brakes are still operated by the driver.
Forward Collision Detection and Avoidance	This technology uses sensors to monitor the surroundings of the vehicle and detect possible accidents. Upon the detection of a collision, the driver is asked to take corrective actions, although some systems have limited braking authority.
Lane-Keeping Assistance (LKA)	This technology uses computer vision systems to detect the curvature of the highway and reacts accordingly, with small adjustments to the steering wheel, as 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 infrastructure to ACC systems. Information about accelerations of a preceding vehicle is shared and serves to reduce the inter-vehicle distance, which increases the efficiency of the traffic flow.

Table 2.1: List of Advanced Driver Assistance Systems (ADAS) and their definition (after Bishop [2005]).

IV research is not limited to driver assistance systems, as these do not offer complete solutions to the issues of vehicular transportation. Even if it might still be a few decades away, vehicle manufacturers, along with public research institutions, are already focusing on Automated Vehicle Control Systems (AVCS) technologies. These systems are designed to take critical driving decisions autonomously and to execute the selected actions without external help from drivers. Table 2.2 illustrates some of the AVCS systems that have been proposed through this early research, by detailing Automated Vehicle Control, Platooning and Collaborative Driving technologies.

Technology	**Description**
Automated Vehicle Control	With this technology, vehicles drive using fully automated longitudinal and lateral controllers without exchange of information with other vehicles.
Platooning	This technology takes CACC to the next level by using communication to exchange acceleration data with an important number of vehicles travelling under a platoon formation. Inter-vehicle distances can be decreased to allow small gaps between vehicles. As a result, platooning often considers fully automated vehicles.
Collaborative Driving	This technology constitutes 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) as to take an optimal driving action.

Table 2.2: List of Autonomous Vehicle Control Systems and their definition (after Bishop [2005]).

2.2.1 Adaptive and Cooperative Adaptive Cruise Control Systems

Among the existing and future Intelligent Vehicle technologies, this thesis is particularly interested in Adaptive and Cooperative Adaptive Cruise Control (ACC and CACC systems). These systems have been at the heart of many research and development efforts over the recent years because ACC and CACC systems are considered as the first step towards the full automatization of the driving task. Indeed, their design involves complicated issues that need to be addressed before going on to more complex vehicle control problems that include extended cooperation between vehicles, such as platooning or collaborative driving.

Adaptive Cruise Control systems, as illustrated in Figure 2.1, use sensors to detect the presence of obstacles. If, upon sensing, a front vehicle is detected, the system adjusts the vehicle's cruising velocity in order to keep a desired inter-vehicle distance set by the driver. When the obstacle has left the front of the vehicle, the system automatically gets back to its previous cruising velocity.

The desired headway can usually be set to values between 1.0 to 2.2 seconds from the front vehicle, as this represents values proposed by governing agencies for highway driving [Bishop, 2005]. For instance, the Jaguar XK-R's ACC system can be set to headway values between 1.0 and 2.0 seconds [Richardson et al., 2000].

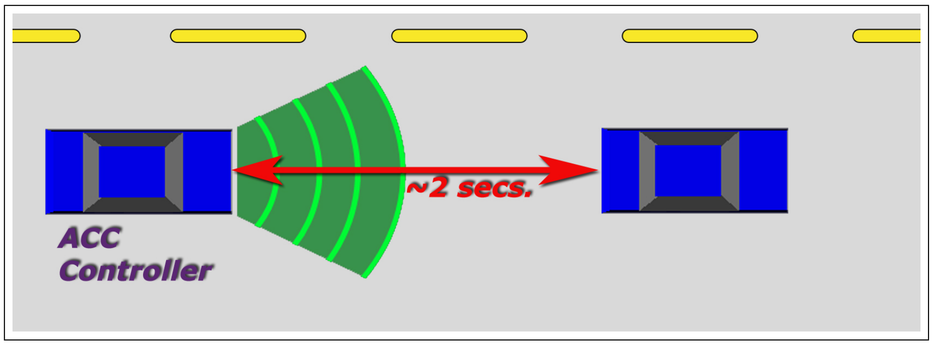


Figure 2.1: ACC system.

Whereas older ACC systems only work at velocities over 40 km/h [Bishop, 2005], modern ACC systems (called low-speed or Stop & Go ACC) enable driving in situations such as traffic jams, where vehicles must often modify their current velocity to relatively slow speed. However, these systems have only a limited influence on the vehicle, and the driver is again the one to take most of the driving decisions (steering, hard braking and emergency situations). Moreover, for safety considerations, the system gives to the driver the entire responsibility of the car when it gets under a certain minimum speed.

Despite their obvious advantage related to driver comfort, simulation studies about the effects of ACC systems on traffic conditions have shown that with enough market penetration, large headway values (considered to be around 2.0 seconds) cause a degradation of the efficiency of the traffic flow [VanderWerf et al., 2002]. As a result, this has motivated researchers to design Cooperative Adaptive Cruise Control (CACC) systems (as illustrated in Figure 2.2) that integrate inter-vehicle wireless communications in their control loop. These systems extend the range of current sensors by offering additional information about surrounding vehicles, such as acceleration, heading and yaw rate, that is not available from today's sensors.

Under the assumption of the availability of a reliable communication signal giving additional environment information, it becomes possible for the decision-making process

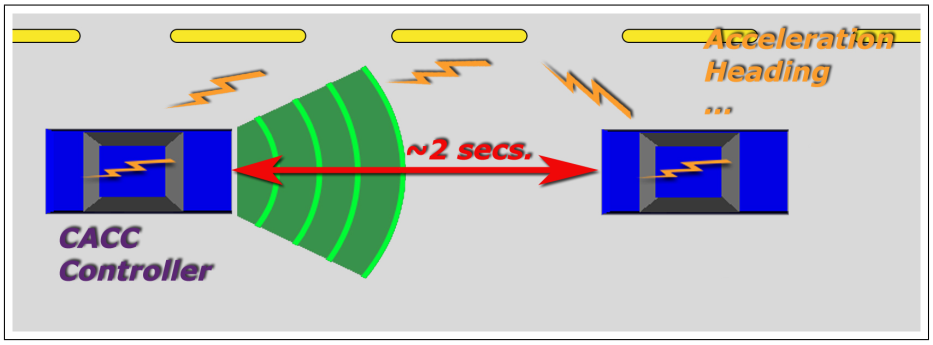


Figure 2.2: CACC system.

of CACC systems to select actions that reduce the inter-vehicle distance while still keeping a secure behavior. Moreover, it has been shown that the presence of an inter-vehicle communication channel for a group of autonomous vehicles using the same control system is important with regards to string stability.

String stability is an important characteristic of vehicle control systems as it describes the fact that, when multiple vehicles equipped with the system are following one another, the error to the desired spacing values does not amplify as it propagates back through the string of vehicles [Rajamani, 2005]. When a controller is not string stable, a general degradation of the efficiency of the traffic flow is often observed, as braking actions of a front vehicle need to be amplified down the stream to avoid collision, up to a point where vehicles have to stop completely, causing traffic jams.

On the theoretical front, it has been shown that inter-vehicle communication is a prerequisite to the preservation of the string stability of a group of vehicle equipped with an ACC system using a constant spacing policy [Rajamani, 2005]. Rajamani also notes that string stability as been proven for systems using a variable spacing policy (such as using a constant time-gap policy) without communication. In that case, communication can still be useful in order to reduce the inter-vehicle distance to increase the efficiency of the traffic flow while preserving a secure behavior.

Unfortunately, CACC systems are still at the research and development stage, as many important advances are needed before these systems can hit the road, especially in relation to inter-vehicle communication and to sensor fusion. Even if ACC and CACC systems are only considered as advanced driving assistance systems (ADAS), research has used these systems as a basis for the design of vehicle control systems that consider a complete automatization of the driving task. We will detail, in the next section, some of the research projects that have focused on the field of ACC, CACC and more complex AVCS.

2.3 Survey of AVCS Research Projects

For the time being, implementations of CACC systems and Autonomous Vehicle Control Systems (AVCS) built on current ACC systems have been limited to prototypes tested in closed circuits, as a number of technological hurdles need to be addressed before such sophisticated systems become a reality on the road. Many of the research projects that have been working toward AVCS design have had to dedicate their efforts to specific engineering issues that hinder the development of autonomous vehicle controllers.

First of all, since most AVCS applications must share information with their environment for efficient behavior, the availability of a robust communication protocol handling both vehicle-to-vehicle (V2V) and road-to-vehicle (R2V) communication is a pre-requisite. Currently, many research projects are working on the development and on the implementation of an inter-vehicle communication protocol. As part of this research, the DSRC protocol (Dedicated Short-Range Communication) has recently been proposed [ASTM International, 2003] as a standard for the future development in inter-vehicle communications.

Of course, work still needs to be done in order to improve this protocol, especially in order to find solutions to important issues that concern wireless communications in highly dynamical environments. For example, current research focuses on finding better addressing and routing schemes for packets [Festag et al., 2004; Borgonovo et al., 2003] as moving vehicles provide an additionnal difficulty in handling communications that are efficient and certain. We will touch some projects that have focused on inter-vehicle communications in the next subsection, but a detailed state-of-the art of this field is given by Tsugawa [2005] and by Luo and Hubaux [2004].

Second, the availability of multiple sensors and wireless inter-vehicle communication uncovers issues related to the representation of all this information. In hope to address this problem, research in the field of sensor fusion has tried to come up with ways of binding the data coming from multiple sources to obtain the most accurate representation of a system's current state. Moreover, sensor fusion also tries to deal with problems related to keeping this information up-to-date even with the sensor delays and with the different resolution offered by the sensors. Thus, many of the research projects addressing the design of AVCS have had to find solutions to this problem, and it remains an issue that generates a lot of research.

Finally, another major technical issue concerns autonomous vehicle control in itself. Many fields of study, from mechanical engineering to computer science, have been interested in integrating the use of sensors and communication for decision-making. Research regarding this issue has tried to address both low-level vehicle control (which answers the problem of selecting throttle, brake and steering actions to respond to the current driving task) and high-level action decisions (which selects driving tasks to accomplish using vehicle coordination), with the goal to select the best possible actions to take in order to optimize safety and traffic flow efficiency.

The next subsections will survey some of the research projects that have been particularly active and that have made significant contribution to AVCS research.

2.3.1 California's PATH

Perhaps the most famous and influential program related to autonomous vehicle control systems is PATH [PATH, 2008] (Partnership for Transit and Highways), a joint venture of the California Department of Transportation and of the University of California at Berkeley. Started in 1986, this multi-disciplinary research effort regroups numerous projects that all share the ultimate goal of solving the issues of transportation systems through the use of modern technologies.

PATH is well-known for its work on Advanced Vehicle Control and Safety Systems (AVCSS) and on Automated Highway Systems (AHS). Thus, projects have focused on solutions to complex problems such as automated longitudinal [Raza and Ioannou, 1997; Lu et al., 2000; Rajamani and Zhu, 1999] and lateral [Peng et al., 1992] vehicle control, cooperative collision warning systems [Sengupta et al., 2007] and platooning [Godbole and Lygeros, 1994; Sheikholeslam and Desoer, 1990]. While developping autonomous control systems, researchers have also studied the effect of AVCS deployment on the traffic flow [Shladover et al., 2001; VanderWerf et al., 2004].

PATH projects have also studied the issue of inter-vehicle communication. Thus, research related to this field has addressed multiple topics going from the design of communication protocols for safety messaging [Xu et al., 2005; Sengupta et al., 2004], to the communication requirements for high-level vehicle coordination [Bana, 2001], to the study of the effects on the traffic flow of systems that rely on communication for automation or driving assistance [Xu et al., 2002]. Of course, these are just some examples of inter-vehicle communication work done in the context of the PATH research network.

Closer to the topic of the current thesis, some PATH projects have also touched the issue of autonomous controller design. Through the use of control laws developped analytically, Drew [2002] has designed a CACC system and has given, through simulation, results detailing their controller's performance. The BAT Project [Forbes et al., 1995, 1997; Forbes, 2002] has also focused on controller design, but its solution considered the use of the agent abstraction and of real-time machine learning algorithms to take autonomous decisions (again through simulation).

Finally, PATH researchers were also the first to propose the platooning concept [Varaiya, 1993]. On that matter, PATH has really impacted the research in this domain as they were the first to implement and demonstrate in a closed-circuit an autonomous platoon as early as 1997, as part of the Demo '97 event [Consortium, 1998]. More specifically, one of the demonstration for this event featured eight automated vehicles driving one behind another in platoon formation without any sort of human intervention.

For more detail about PATH's previous and future research directions, we refer the reader to Shladover [2007] who does a good retrospective of this program.

2.3.2 Japan's AHSRA

Another important program participating in intelligent vehicle research is Japan's Advanced Cruise-Assist Highway Systems Research Association [AHSRA, 2008]. Its goal is to develop the technology needed in order to support cooperative vehicle-highway safety systems. These systems wish to link Automated Highway System (AHS) research with Advanced Safety Vehicle (ASV) research through the use of road-to-vehicle and vehicle-to-vehicle communications.

This program is also well-known for its implementation and demonstration of ASV technologies in the Demo 2000 [Tsugawa et al., 2000] event. Over 35 different ASV technologies were tested during this event. Since the AHSRA regroups Japanese manufacturers, many of the technologies developed in relation to the ASV initiative such as lane-keeping assistance and adaptive cruise control systems have rapidly been integrated in vehicles.

The efforts of this organization have also focused on using inter-vehicle communication for enhanced ASV applications. For example, Kato et al. [2002] have worked on the design of a cooperative driving system based on dGPS positioning (GPS technology that adds differential positioning with base stations for increased precision) and DSRC wireless inter-vehicle communications. This system included both a longitudinal and a lateral control system and it was tested on several platooning scenarios during the Demo2000 event.

2.3.3 The Netherland's TNO and Germany's CarTalk2000

Of course, Europe has also been involved in AVCS research. Here, we will showcase the efforts of The Netherland's Organization for Scientific Research [TNO, 2008]. The TNO is a research organization that touches many domains, the automotive industry being one of them. Researchers of this center have been collaborating with both universities and the automotive industry on various research projects, and they have been active on the design of autonomous vehicle systems. For example, the TNO's partnership with Germany's CarTalk2000, which also involves industrial partners from Daimler-Chrysler, Fiat and the information technology expert Siemens, has focused on the use of inter-vehicle communication to integrate extended information on surrounding vehicles.

To evaluate their protocols, TNO's researchers have tested different applications that rely on the availability of communications. For example, they have proposed a Communication-Based Longitudinal Control system, built on top of current ACC control systems by integrating communication [Morsink et al., 2003]. In their work, velocities and accelerations of front vehicles are transmitted by communication. Vehicles receiving this information can adjust their velocity in order to keep a safe headway. The controller they have developped uses acceleration information from up to four front vehicles in order to select a desired acceleration. They have also addressed the problem of sensor fusion, for binding front laser perceptions with communication messages, using an Extended Kalman Filter [Hallouzi et al., 2004]. Their system was integrated in vehicles and tested in various Stop & Go situations.

In other work, they have also evaluated the impact of advanced driver assistance systems on the traffic flow with a microscopic traffic simulator [Visser, 2005]. They were able to evaluate how the traffic flow efficiency is affected by different market penetration rates of their proposed communication-based driving assistance system.

2.3.4 Other Work

All the work listed above represent just the tip of the iceberg of what has been and is currently being done in the field of intelligent vehicles and autonomous vehicle control systems. Of course, a lot of other organizations have also financed similar research projects in this field, such as Italy's ARGO [Broggi et al., 1999], Canada's Auto21 [Auto21, 2008], Germany's FleetNet [Festag et al., 2004] and European's CHAUFFEUR projects [Schulze, 2007].

For more information, we refer the interested reader to other surveys on the state-of-the-art of intelligent vehicle and related technologies: Tsugawa [2005] has done a good job of illustrating the previous and future research on inter-vehicle communication, Vahidi and Eskandarian [2003] have surveyed the advances in ACC and intelligent collision avoidance systems, while Bishop [2005] has touched nearly every aspect of intelligent vehicles design.

2.4 Intelligent Vehicles and Machine Learning

As we have described earlier, the field of computer science has recently shown great interest in the problem of autonomous vehicle control. This section will detail some research projects that have already applied artificial intelligence and machine learning techniques to design intelligent vehicle controllers. First, in Section 2.4.1, we will show how the agent abstraction has been applied naturally to this problem. Since agents need a deliberation mechanism for decision making, they are often used in conjunction with modern machine learning techniques, as we will see in Section 2.4.2. Finally, Section 2.4.3 will overview what has been done towards the use of machine learning for autonomous vehicle coordination.

2.4.1 Intelligent Vehicles and the Agent Abstraction

Agents are autonomous software entities that try to achieve their goals by interacting with their environment and with other agents [Russell and Norvig, 2002]. With this ability for autonomy, the agent abstraction is a logical choice of mechanism to embed in vehicles a deliberative engine adapted either for driver assistance or for full automation of the driving task. Moreover, their social ability makes this abstraction especially adapted to the problem of collaborative driving, where decisions should ideally be made in a decentralized manner, but with constant interaction with other vehicles in order to fulfill the goals of increasing safety and traffic flow efficiency that they share.

With those characteristics, it is not surprising that the agent abstraction has already been used by researchers to model autonomous driving tasks and many papers have proposed different approaches and architectures for agent driving. For example, Ehlert [2001] has designed agents working at a tactical-level of driving (which corresponds to high-level decisions such as lane-changing, that take into account other vehicles on the road). These agents can make decisions about the actions they should take in real-time. To achieve this, a short-term planner computes the next positions of the moving objects detected by sensors. Then, reactive rules similar to the subsumption architecture [Brooks, 1991] are used to propose some actions. Finally, an arbiter analyzes the different actions proposed by each behavior rule and selects the best action.

Researchers from Carnegie-Mellon University have considered another approach, named SAPIENT (Situation Awareness Planner Implementing Effective Navigation in Traffic), that makes driving decisions using many agents collaborating together to drive a single vehicle [Sukthankar et al., 1998]. Each agent first focuses on the perception of a simulation object, and must evaluate the effect of its possible actions with regard to this object. Finally, a voting mechanism observes the agents' responses and must select the best action possible.

A different agent architecture has been proposed by Hallé [2005], whose work features multiple agents making decisions using a deliberative architecture based on team work [Tambe and Zhang, 2000]. This approach is then used for platoon formation management. Tasks are managed by attributing in real-time a dynamic role to each vehicle in the platoon according to the task that must be completed. The author has compared the performance of his approach by using it on both a centralized and a decentralized platoon, and he has given advantages and disadvantages corresponding to each type of platoon organization.

These are just a few examples of the use of the agent abstraction applied to autonomous driving. Of course, agents are still used today to describe autonomous vehicle control problems, but focus of researchers has shifted to the design of efficient algorithms for control or coordination using modern machine learning. The next section will detail previous work in which such techniques have been used to address the problem of autonomous vehicle control.

2.4.2 Machine Learning for Intelligent Vehicle Control

Already, a number of research efforts have proposed to use machine learning in order to address the problem of action selection for autonomous vehicle control. This problem is especially interesting for the machine learning community as it offers many challenges such as the complexity of the dynamics and the uncertainty resulting from the actions of surrounding vehicles. As we will see in this section, machine learning is rapidly developing as an efficient tool to solve such complex problems, justifying the emergence of these techniques as solution methods for the vehicle control problem.

One of the first application of machine learning for vehicle control has been the ALVINN (Autonomous Land Vehicle in a Neural Network) system [Pomerleau, 1995]. Pomerleau used a supervised learning system featuring a neural network that received as inputs patterns from a computer vision system. The network had to learn to match vision patterns of the road ahead, captured by the vision system, to an accurate driving action that keeps the vehicle on the road. Training examples were obtained by watching the driving behavior of a real person. In some of the experiments, vehicles equipped with ALVINN networks were able to drive without human intervention for distances of 35 kilometers. This work has been brought even further with the design of MANIAC (which stands for Multiple ALVINN Networks In Autonomous Control) [Jochem et al., 1993]. This system used several ALVINN-like networks, pre-trained for different types of roads, and combined the output coming from the different ALVINN in order to use knowledge from each network.

Panwai and Dia [2005a,b, 2007] have also used neural networks, but this time, to design an autonomous vehicle controller that tries to copy human drivers' vehicle following behavior. The work of these authors, however, was a little bit different, as it focused on the study of the traffic flow, using large-scale (macroscopic) simulators. These simulation tools try to simulate accurately traffic conditions as if a large number of vehicles were driven by humans. The model they proposed selects actions (brake and throttle) according to the agents' perceptions, by using a neural network to learn to match human behavior data gathered previously. They have compared different neural network architectures for action selection against classical car-following models, such as the Gipps model [Wilson, 2001] and the controller they obtain seems to outperform current human-driver simulation models in the accurate representation of their behavior.

Another approach based on neural networks has been proposed by Huang and Ren [1999]. It features a vehicle controller that uses multiple neural fuzzy networks (NFN), combining standard artificial neural networks and fuzzy logic into a unified framework. The learning process tunes the parameters of both the neural network and the fuzzy logic rules using a mix of genetic and gradient algorithms. The performances of such a controller was finally tested through simulation. Similar work by Dai et al. [2005] has used reinforcement learning to learn an approximate value function, which is then used to tune the parameters of the fuzzy logic controller. The authors have shown that their initial fuzzy logic controller performs better at vehicle control after its parameters have been modified using the learned value function.

The use of fuzzy logic to design autonomous vehicle control systems has also been considered by Naranjo et al. [2003, 2006a,b], as part of Spain's AUTOPIA project. These researchers have designed a longitudinal vehicle controller based on fuzzy logic and which integrates inter-vehicle communication to share positioning information of a lead vehicle. Their resulting ACC controller was even embedded in a vehicle and tested in demo sessions of the IEEE IV2002 Conference. Their approach is quite similar to what we propose in this thesis, as an intelligent system is used to make abstraction of the heavy mathematical analysis needed to design such a system using control theory.

The PATH program, through its Bayesian Automated Taxi (BAT) project [Forbes et al., 1995, 1997] has also studied the use of agents and machine learning for autonomous driving in traffic, this time through the use of a Bayesian decision-making architecture relying on dynamic probabilistic networks (DPN) for inference. The authors of this project indicate that their solution was able to handle pretty well the problems of sensor fusion and of sensor noise. Further work on the BAT project has focused more particularly on solving the driving task using reinforcement learning. Thus, Forbes [2002] has proposed an instance-based mechanism that approximates a value function by storing relevant examples. This example set is updated when new experience results in a more accurate value function. Moreover, the authors have proposed to use DPNs in order to learn the model of an environment with continuous state and actions. When tested in a complex driving simulator, their learned controllers performed significantly better than hand-coded controllers.

Of course, these PATH researchers were not the only ones to consider using such learning algorithms to design autonomous vehicle controllers. In recent years, the use of reinforcement learning to solve autonomous vehicle control problems has gathered more and more interest. For example, Alton [2004] has focused on using a nearest-neighbor policy representation and a policy search algorithm for learning to steer a simulated truck with a trailer. Similar work includes efforts of Pyeatt and Howe [1998], who describe a two-layer architecture based on low-level behaviors and on a high-level controller. Each low-level behavior (steering, accelerating, passing) was obtained using the Q-Learning algorithm, with policies represented by neural networks. The high-level control mechanism had the task to select the appropriate low-level behavior to launch and to switch from a behavior to another using a hand-coded method. Another related approach has been proposed by Coulom [2002], as part of his work on reinforcement learning and motor control. Using the same race car simulator as Pyeatt and Howe [1998], Coulom has focused on the design of a vehicle controller using the TD(λ) algorithm, this time along with a single neural-network for value function approximation. His approach was different as it was enhanced by the use of numerous additional features given to the network as inputs, in order to make learning easier.

Obviously, the reinforcement learning community has also focused on a large number of other optimal control problems that offer the same issues and challenges as autonomous vehicle control. The characteristics of these other problems make them interesting environments for the development and test of new learning techniques, which in turn could obviously be applied to AVCS design.

Autonomous helicopter and unmanned aerial vehicle (UAV) control has been an environment that has attracted many reinforcement learning researchers as control of such vehicles is a complex problem that features continuous, non-linear and stochastic dynamics. Thus, Ng and his students designed controllers for helicopters and similar autonomous machines. They have first proceeded in approximate identification of the dynamics model of the helicopter by observing physical data obtained during a test flight with a professional pilot. Then, they have used a policy search approach to reinforcement learning named PEGASUS [Ng and Jordan, 2000]. With this approach, they were able to learn maneuvers challenging even to advanced human pilots and used in remote helicopter control competitions [Ng et al., 2004b]. They were also able to learn inverted flight maneuvers [Ng et al., 2004a] using the same technique. Ng and his team achieved even more challenging maneuvers using a slightly different approach based on the resolution of the MDP by Differential Dynamic Programming (DDP) [Abeel et al., 2007].

Autonomous control of various other vehicles has also been an active domain of application of reinforcement learning. For example, using the Sarsa(λ) algorithm and a technique called shaping (which aims at reproducing a complex behavior by introducing to the agent problems of increasing difficulty, ultimately solving the desired problem), Randlov and Alstrom [1998] have been able to learn efficient control policies in a bicycle simulator.

Other researchers have also tackled learning control policies for a remote control car and a complex flight simulator [Abeel et al., 2006], while El-Fakdi et al. [2005] have learned to control a simulated, unmanned underwater vehicle. The former has proposed a policy-gradient algorithm where real-life experience is used to evaluate the current policy and where an approximate model is used to compute its gradient, while the latter has used a simpler approach of classical policy gradient algorithms coupled with a neural network function approximator. Both have been successful in learning an efficient control policy for their respective domain of interest.

Finally, the field of robot locomotion is also a typical control problems that has its share of complexity and that has been of interest to researchers. For example, work has been done towards learning low-level locomotion of a Sony Aibo dog-like robot. In this learning task, the robot was able to learn to move its four independent legs in order to move on a soccer field [Kohl and Stone, 2004]. Field [2005] has focused on learning a similar locomotion behavior, but this time on a simulated scorpion-like, 6-legged hexapod robot. Work from Peters and Schaal [2006] has focused on learning to move a seven-degrees of freedom robotic arm to hit a baseball. All of these approaches focused on learning using policy gradient methods as they are well adapted to handling such robotic tasks.

This brief overview of the use of machine learning to solve not only autonomous vehicle control but control problems in general show that most of these applications offer interesting, real-life uses of machine learning theory. Of course, these do not come without their lot of issues, such as the necessity to handle continuous variables and complex dynamics. Despite these problems, current results have shown that autonomous control using machine learning is a promising field that is sure to gather many more research efforts in the near future.

2.4.3 Machine Learning for Intelligent Vehicle Coordination

Even if it is not directly related to experiments of this thesis, it is important to note that the machine learning community has also been active in research toward vehicle coordination. This is an interesting domain of application for agent and multi-agent learning algorithms as actors of the system must learn to take decisions based on local information about the state of the environment and about surrounding agents in order to select the best action for all vehicles. Of course, the ultimate goal of such systems is to reduce the number of accidents and to increase the throughput of the traffic flow.

Among the work in this field, Moriarty and Langley [1998] have proposed an approach in which cars learn distributed control strategies for lane selection using reinforcement learning. The policies they obtain optimizes the velocities of the vehicles while reducing the number of lane changes. Similarly, Ünsal et al. [1999] have tackled the problem of lane selection by using multiple stochastic learning automata to control the longitudinal and lateral movement of a single vehicle. The interactions of these automata were modeled using game theory, and results show that they were able to learn to act in order to avoid collisions. Similar work from Pendrith [2000] has focused on the design of another lane change advisory system, but this time based on a distributed variant of the Q-Learning algorithm (DQL) where multiple agents update a shared policy. However, all of these efforts were done using a simplifed driving environment.

Work from colleagues at Laval University's DAMAS Laboratory in the field of multi-agent coordination applied to vehicle coordination is also quite interesting. Laumônier and Chaib-draa [2006] have evaluated the influence of the range of agent observations on the problem of vehicle coordination. They have proposed an algorithm that extends the FriendQ multi-agent learning algorithm by only taking into account the state and actions of observable agents into the deliberation process. They have shown, both in a simplifed driving simulator and in a more complex single-track vehicle environment, that such an approximated approach can still yield good results for the lane change problem.

The preceeding paragraphs only gave a few examples of work done on vehicle coordination based on machine learning techniques. Of course, theory on multi-agent learning is still in its early stages and much research still needs to be done before it can be applied successfully to real vehicles. However, it is clear that since it is particularly difficult to characterize the dynamics of multi-agent interactions, learning approaches become interesting solutions for the collaborative driving problem, as agents can figure out by themselves on how to coordinate.

2.5 Summary

In this chapter, we have presented an overview of the field of Intelligent Vehicles (IV). First, we have described how IV relates to the larger domain of Intelligent Transportation Systems (ITS). Next, we have detailed some Advanced Driver Assistance Systems (ADAS) and Autonomous Vehicle Control Systems (AVCS), with particular focus on Adaptive and Cooperative Adaptive Cruise Control systems. Then, we have seen in Section 2.3 that many research projects, most of them joint efforts from both industrial partners and public research institutions, have been working on various issues related to these systems. Finally, we have also shown that machine learning has recently gathered much interest as a solution to the problem of autonomous vehicle control. This is the approach we propose to take to design the fully autonomous longitudinal CACC system that we detail in Chapter 5. In the mean time, the next chapter overviews the machine learning theory we will rely on for the design of our vehicle controller.

Chapter 3

Reinforcement Learning and Policy Gradient Methods

In recent years, the use of the agent abstraction has grown in popularity among machine learning researchers, as its characteristics make it an efficient and practical representation that is naturally adapted to the resolution of many problems requiring autonomous decision making. By definition, intelligent software agents are social entities that can act autonomously in their environment to fulfill their design goals [Russell and Norvig, 2002; Wooldridge, 2002].

Of course, to select the appropriate actions to take in order to reach their goals, agents need a decision-making architecture. As detailed in the numerous chapters of Russell and Norvig [2002], many artificial intelligence techniques can be considered for action selection, going from probabilistic reasoning using Bayesian networks to Constraint Satisfaction Problems (CSPs) to various planning and search approaches.

However, to act autonomously in stochastic, uncertain and complex environments with possibly unknown dynamics, the use of reinforcement learning has become the solution of choice. With this technique, an agent can learn, by direct interaction with its environment, which actions are preferable depending on the current state of the world. This can result in highly efficient behavior in situations were it might be difficult for a human designer to tell which action should be selected in order to maximize the performance criterion of the agent in the long term.

The current chapter focuses on describing the reinforcement learning theory that serves as the basis for the decision-making architecture we will use to solve our autonomous vehicle control problem. First, Section 3.1 will focus on the general reinforce-

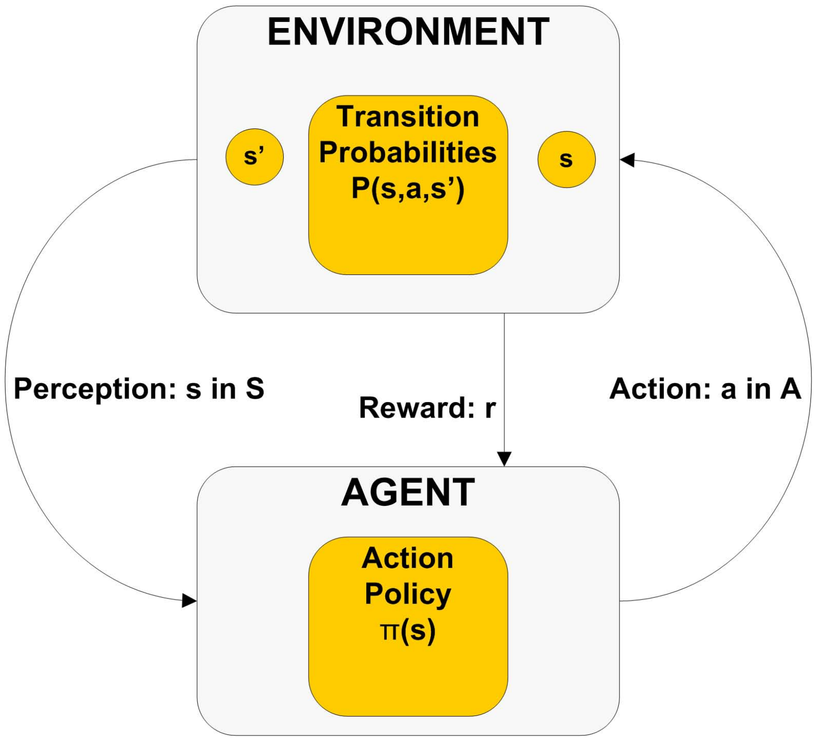


Figure 3.1: Agent interactions in reinforcement learning.

ment learning framework, detailing Markov Decision Processes and classical learning algorithms. Then, Section 3.2 will describe the issues related to tabular representations of value functions and will detail how these problems can be addressed using function approximation. Finally, Section 3.3 will detail policy-gradient algorithms, which will be used to design our autonomous vehicle controllers, as they are especially adapted to the resolution of complex control problems.

3.1 Reinforcement Learning

Reinforcement learning is a modern machine learning technique that concentrates on the resolution of sequential decision problems. With this technique, an agent can learn through direct interaction with its environment. When the agent takes an action a using its current policy $\pi(s)$, it receives a reinforcement, under the form of a reward signal r that indicates whether this action was good or bad in the specific state it was taken. These interactions of a learning agent with its environment are summarized in Figure 3.1.

The goal of reinforcement learning is to learn a policy mapping the current state of the system to the action that maximizes the expected reward an agent can receive in its environment. As we will see in this section, numerous algorithms have been proposed in order to compute agent control policies.

It is interesting to note that reinforcement learning is closely related to control theory. This domain, which has many uses in engineering, tries to manipulate the inputs of a dynamical system in order to match the desired values at the output of the system. As a subfield of control theory, optimal control tries to design controllers that optimize an objective function that depends on the different variables of the system. Clearly, this definition of optimal control is quite similar to the reinforcement learning process, where the goal is to take actions, according to the current state of the system, in order to optimize the expected reward. When the dynamics of the system are known, optimal control methods can use mathematical tools to solve the problem. However, when the dynamics of the system to control are unknown, the problem becomes much more complicated. Resolution relies on the use of adaptive optimal control methods, but these have serious limitations when facing non-linear systems. As indicated by Sutton et al. [1992], reinforcement learning has thus become the solution of choice to tackle such problems as this technique can yield efficient control policies by direct interaction with the system.

The current section will first study the framework of Markov Decision Processes, which is at the heart of most reinforcement learning approaches. Afterwards, we will describe the different learning algorithms and some value function approximation techniques often required to solve problems with large state spaces. Finally, we will also detail policy-gradient algorithms, which have recently gathered much interest as they represent an efficient method to solve control problems.

3.1.1 Markov Decision Processes

Markov Decision Processes (MDPs) are statistical, sequential decision processes used to solve problems in which an agent tries to find the best actions to take in order to maximize its expected reward [Sutton and Barto, 1998].

Formally, a MDP can be described as:

- $\mathcal{S}$, a finite set of states;
- $\mathcal{A}$, a finite set of actions;

- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, an immediate reward function;
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, the transition probabilities from a state to another when taking an action.

The current state of the system in which the agent acts corresponds to a particular set of values taken by its features (state variables). Thus, the set of states describes all the possible combinations of values that can be taken by these features. This set grows exponentially with the number of state variables. The set of actions lists all the possibilities that the agent can interact with the environment to modify its current state. Taking action a in state s leads the agent to a subsequent state s' with probability $\mathcal{P}(s, a, s')$. Finally, through its reward function, the environment returns a scalar signal r to the agent, indicating whether the action taken was good or bad.

Aside from these elements, efficient resolution of MDPs depends on the presence of the Markov property. This property is satisfied if the current state of the system encapsulates all of the knowledge required to make a decision, and it is needed to find the optimal solution of an MDP via classic dynamic programming or reinforcement learning approaches. Formally, this property is illustrated by Equation 3.1.

$$Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, r_1, s_0, a_0\right\} = Pr\left\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\right\} \quad (3.1)$$

Other important terminology in the realm of MDPs includes the notion of value of a state $V(s)$. This value corresponds to the expected reward that an agent can obtain after visiting the current state s. Of course, it should be noted $V^{\pi}(s)$, since the expected reward always depends on the actions taken after visiting state s, which are dictated by the current policy $\pi(s)$ mapping environment states to agent actions.

To compute the value function $V^{\pi}(s)$ associated to the current policy $\pi(s)$, we generally use the Bellman equation, as seen in Equation 3.2. This equation indicates that the value of a particular state s corresponds to a sum, over all possible actions proposed by the policy $\pi(s)$, of the value of all possible outcomes of that action, weighted by the probability of taking that action under policy $\pi(s)$. The outcome of taking action a corresponds to the probability of transitionning to state s' following this action, multiplied by a sum of the reward obtained for this transition and of the discounted value of the next state s'. Thus, to summarize, this equation calculates the value of a state when using policy $\pi(s)$ by summing over the value of all possible transitions according to their respective probabilities.

$$V^{\pi}(s) \quad = \quad \sum_{a} \pi(s, a) \sum_{s'} \mathcal{P}^{a}_{ss'} \left[\mathcal{R}^{a}_{ss'} + \gamma V^{\pi}(s')\right]. \qquad (3.2)$$

To solve an MDP, we are particularly interested in the computation of an optimal policy π^* that maximizes the expected reward that an agent can get. Evaluating such policies yields the optimal value function, $V^*(s)$. However, the computation of the optimal value function can be done without knowledge of the policy, by using the Bellman Optimality Equation (as seen in Equation 3.3). Clearly, this equation shows that the optimal value function corresponds to the value of the action that maximizes the expected reward in that state. The optimal policy $\pi^*(s)$ then corresponds to taking that particular action, as seen in Equation 3.4.

$$V^*(s) \quad = \quad \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s') \right]. \tag{3.3}$$

$$\pi^*(s) = \arg\max_a V^*(s) \tag{3.4}$$

It is interesting to note that MDPs are a specific case of the more general Partially Observable Markov Decision Process (POMDP) framework. POMDPs include a variable that describes probabilities over the agent observations. MDPs just make the assumption of full observability of the environment, always setting this probability to 1.

3.1.2 Algorithms

The numerous algorithms that have been proposed to solve MDPs can be separated in two distinct classes: model-based, that can be used when the transition probabilities are known, and model-free, that can be used when the agent does not have explicit knowledge of the environment's dynamics.

When the transition probabilities $\mathcal{P}$ for transitioning from one state to another are known, MDPs can be solved using classic dynamic programming approaches. Among these algorithms is the well-known Value Iteration (VI) algorithm [Bellman, 1957], as illustrated in Algorithm 1. With the values of the states initialized to zero, the algorithm iterates on the set of states and updates the values using the Bellman Optimality Equation 3.3. This algorithm will converge to the optimal value of the states, though we can stop updating the values when they only change by a small amount ϵ.

Unfortunately, transition probabilities are not always known and, in that case, we have to rely on other MDP resolution techniques. These approaches, called "model-free" algorithms, have the ability to learn an optimal policy by direct interaction with the environment. A classic example of model-free learning is the Q-Learning algorithm [Watkins, 1989], which converges to an optimal action policy simply by trying

Algorithm 1 Value Iteration (after [Sutton and Barto, 1998]).

```
$\epsilon \leftarrow$ small positive number.
repeat
    $\Delta \leftarrow 0$
    for all $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^*(s')\right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
until $\Delta < \epsilon$
```

actions in the environment and observing their results. This algorithm is based on the notion of Q-value $Q(s,a)$ that represents the expected reward when the agent is in state s and chooses action a. This new representation is necessary since the algorithm must evaluate independently the effect of each action in a specific state.

Q-Learning is a form of Temporal-Differences learning (TD-Learning), which is based on the idea that we take a previous prediction of state value and correct this prediction using current estimates. Thus, as the name implies, there is an evolution of our estimates through time according to newly sampled transition values. The Q-Learning update rule shows that, from the current sampled transition (s, a, s'), we make a small modification to the current value of the (s, a) pair so that it gets closer to currently expected value of $r + \gamma \max_{a'} Q(s', a')$. The convergence of the Q-Learning algorithm has been proven (see Watkins and Dayan [1992]) for as long as all (s, a) pairs keep on being infinitely updated.

Algorithm 2 Q-Learning (after [Sutton and Barto, 1998]).

```
for each episode do
    repeat
        Take action $a$ using the current policy, with some exploration.
        Receive reward $r$.
        Observe next state $s'$.
        $Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
        $s \leftarrow s'$
    until $s$ is a terminal state.
end for
```

Of course, model-free algorithms are characterized by the fact that they require some sort of exploration of the state space. In order to find out the best actions, the agent must not always be greedy towards the current value function, but must explore to see if other actions could achieve better results in the long-term. Thus, during learning, the action choice often relies on a ϵ-greedy approach, which exploits the current policy most of the time but explores with a small probability ϵ.

However, we will see in the next subsection that both model-based and model-free approaches that rely on tabular representation of the value function face an important issue that needs to be addressed in order to be efficient at learning in complex, high-dimensional environments.

3.2 Value Function Approximation

Standard tabular representation of a value function, as normally used by the Value Iteration and Q-Learning algorithms, must keep track of all the possible states that can be encountered in the environment. This technique might be good for learning in low-dimension environments but, as the size of the state space grows exponentially with the number of possible state variables (an issue name the "curse of dimensionality"), it rapidly becomes inefficient when facing state spaces of high-dimensionality. Moreover, solving a problem requiring high-resolution of the state variables might rapidly lead to an intractable problem.

It becomes even more problematic for these representations to treat continuous environments, as they must be discretized. Discretization of the state space can lead to the hidden state issue, which signifies that the problem becomes non-Markovian since important features of a state are missing for accurate action selection. In this case, hidden state is the result of perceptual aliasing, which occurs when numerous environment states, that should normally discriminate between different agent actions, are discretized to the same value. In that case, it is clear that the agent will have a hard time learning an accurate policy (see McCallum [1995] for further detail on hidden state and perceptual aliasing).

Clearly, to tackle problems in complex environments, we must consider the use of value function approximation techniques. Instead of using a table to keep track of each possible state (or state-action pair) of the environment, these methods consider the use of a function to compute an approximated state value. Most of the time, function approximators are parametrized by weights, which are modified, using optimization algorithms, in order to minimize the approximation error between the estimated function and the desired output value.

A definite advantage of these representation methods is that they can generalize from previous experience. As a result, an approximation of the value function made with a limited number of visited states can be used to estimate the value of states over all the state space. Other than generalization, another advantage of value function approximation comes from the fact that using weights results in an important reduction in the ressources needed to represent the policy, as the number of weights is generally much lower than the possible number of states that must be kept in memory when using a tabular representation.

The current section will detail how function approximators can be used with reinforcement learning algorithms. It sustains Section 3.3, which details how parametrized policy representations that rely indirectly on value function approximation can be used with the policy-gradient family of algorithms.

3.2.1 Preliminary Theoretical Considerations

Theory on the use of function approximation to estimate the value of states comes in large part from the supervised learning domain, where the goal is generally to minimize the estimation error over a data set.

Formally, what we want to achieve is to use training examples $(\vec{x}, y)$ to predict the output y associated to an input $\vec{x} = \{x_1, x_2, \ldots, x_m\}$ corresponding to a vector of features that defines the current state of the system. The function we want to learn must return the approximated output value y', as seen in Equation 3.5.

$$F(\vec{x}) = y' \tag{3.5}$$

To evaluate the performances of a function approximator, the mean-squared error (as seen in Equation 3.6) between the approximated value and the desired value on a set of n examples is often used.

$$MSE = \frac{1}{n} \sum_{i \in n} \left[F(\vec{x_i}) - y_i\right]^2 \tag{3.6}$$

When applied to reinforcement learning, we define by $V_\theta(s)$ the value function approximator. This function is parametrized by a vector of weights $\vec{\theta} = \{\theta_1, \theta_2, \ldots, \theta_m\}$. For instance, Equation 3.7 shows a possible linear approximator, which affects a particular weight to each of the state variables x_i. Of course, parametrized function approximators are not restricted to this linear form, and various tools exist to represent the value function, such as Tile Coding, Radial Basis Function (both described in [Sutton and Barto, 1998]) and neural networks (which are covered in the next subsection).

$$V_\theta(s) = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \tag{3.7}$$

No matter what kind of representation is used, the goal of value function approximation is to minimize the error between the true value function $V^\pi(s)$ and its approximation $V_\theta(s)$. Since reinforcement learning obtains training examples in an on-line fashion, we must be able to calculate the approximation error of a single example (instead of calculating the error over a complete training set n such as in Equation 3.6). The resulting error function that must be minimized for the reinforcement learning problem is given in Equation 3.8.

$$E(s) = \frac{\left[V^\pi(s) - V_{\vec{\theta}}(s)\right]^2}{2} \tag{3.8}$$

To minimize this error function, the weights of the approximate value function are modified by calculating the gradient of the error function with regards to the parameters $\vec{\theta}$. Equation 3.9 gives the gradient of the error function.

$$\frac{\partial E(s)}{\partial \vec{\theta}} = -(V^\pi(s) - V_{\vec{\theta}}(s))\frac{\partial V_{\vec{\theta}}(s)}{\partial \vec{\theta}} \tag{3.9}$$

Then, to reduce the error, the approximator's parameters are moved in the inverse direction of the gradient (because the gradient indicates the direction for which the error grows the most rapidly). In this case, the technique is called "stochastic gradient descent", since the weights are modified using an estimation of the gradient from a single training example. Equation 3.10 shows the resulting update rule.

$$\vec{\theta} = \vec{\theta} + \alpha(V^\pi(s) - V_{\vec{\theta}}(s))\frac{\partial V_{\vec{\theta}}(s)}{\partial \vec{\theta}} \tag{3.10}$$

One important problem remains with this weight update method: we do not know what the true value function $V^\pi(s)$ is. This issue can be addressed using the same estimation we made for the Q-Learning algorithms (and TD methods in general). This estimation suggests that the current value function should move toward the value of the transition just sampled, which we assume to be the true value of the policy. Thus, we actually modify the weights using the TD error $r + \gamma max_{a'} Q(s', a') - Q(s, a)$. As a result, update rule 3.11 can be used with the Q-Learning algorithm.

$$\vec{\theta} = \vec{\theta} + \alpha(r + \gamma \max_a Q_{\vec{\theta}}(s, a) - Q_{\vec{\theta}}(s, a))\frac{\partial Q_{\vec{\theta}}(s, a)}{\partial \vec{\theta}} \tag{3.11}$$

However, convergence of reinforcement learning methods to the exact value function when using approximation is definitely a complicated issue. Tsitsiklis and Van Roy [1997] suggest that convergence not only depends on the type of the approximator, whether it is a linear or non-linear function of its parameters, but also depends on the distribution of the states visited for value function updates.

This distribution of visited states is actually dependent on the algorithm used. On-policy algorithms update the current value function using a state distribution that is generated using the current policy. For example, we see that the update rule 3.12 of the Sarsa algorithm modifies the state value using $Q(s_{t+1}, a_{t+1})$, where a_{t+1} is sampled from the current policy. On the other hand, off-policy algorithms update the function approximator using a distribution of the states that is not sampled using the current policy. For example, the Q-Learning algorithm is an off-policy algorithm, since the value function is updated (as seen in Equation 3.13) by taking the action that maximizes the value in the next state, $\max_a Q(s_t, a)$, instead of using an action from the current policy.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{3.12}$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{3.13}$$

Thus, Tsitsiklis and Van Roy [1997] have proven the convergence of temporal differences algorithms with linear approximators, when using an on-policy algorithm, to a set of parameters that corresponds to a near-optimal, minimal error function. The authors have also given bounds on the error of the value function obtained. Moreover, they have also shown that divergence is possible for linear approximators when using off-policy updates. This case of divergence has been corroborated by some simple examples [Boyan and Moore, 1995; Baird, 1995; Tsitsiklis and Van Roy, 1996] that show divergence of the approximate value function for different off-policy algorithms. Finally, they have also illustrated that non-linear approximators may diverge with on-policy and off-policy updates. Even if theoretical results show that, in some cases, divergence can possibly be observed, the use of function approximation can still be succesful at efficiently approximating a value function.

The next subsection will describe Artificial Neural Networks (ANNs). ANNs are non-linear function approximation methods and thus, as state previously, are not guaranteed to converge to a good approximation of the value function. However, we will see in Section 3.3 that when coupled with policy-gradient algorithms, neural networks can be used to learn efficient behavior based on a rough approximation of the value function.

3.2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a statistical learning mechanism that uses interconnected adaptive neurons in order to model non-linear relationships between training data. Many network architectures have been proposed, each having particular characteristics. Some networks are used for supervised learning (such as the multi-layer perceptron [Rumelhart et al., 1986] and Fuzzy ARTmap network [Carpenter et al., 1992]), while other architectures are better adapted to unsupervised learning (such Kohonen's Self-Organizing Map [Kohonen, 1990] and Growing Neural Gas (GNG) networks [Fritzke, 1995]).

These neural network architectures can be used to solve many problems such as function approximation, classification (which tries to learn the correlation between example features and their corresponding class) and clustering (which tries to idenfity groupings of examples without knowing their respective class).

This section focuses on the multi-layer percetron, a neural network which is especially adapted for function approximation. It overviews its architecture and describes its backpropagation learning algorithm, which is used later on in conjunction with policy-gradient algorithms. Of course, neural networks architectures and learning algorithms have already been covered in depth in litterature, and we refer the reader to Haykin [1998] and Russell and Norvig [2002] for further detail.

Network Architecture

The artificial neural network, based on the concept of multiple interconnected neurons, is a direct analogy to the human brain's inner working. Much like electrical signals in the human brain, artificial neurons receive input signals of different strength and are activated when the input stimulus is strong enough.

A neuron's activation value n is a weighted sum of its input signal, a vector of features $\vec{p}$, according to its connection weights $\vec{w}$. This sum also includes the neuron's bias of fixed input value -1 and its connection weight b. The result of this sum, n, is then passed to an activation function $f(n)$ that calculates the final output value a of the neuron. Figure 3.2, illustrates the behavior of a single neuron, while Equation 3.14 shows the calculation of its activation level n and Equation 3.15 illustrates the calculation of its output value a.

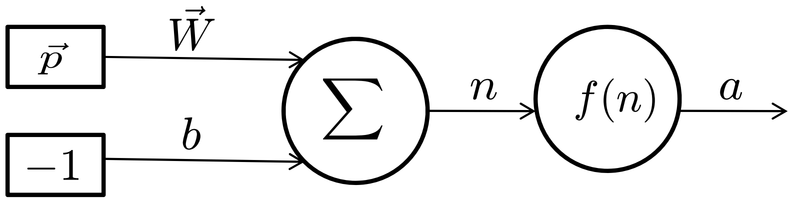


Figure 3.2: Single neuron diagram.

$$n = \vec{w}^T\vec{p} - b \tag{3.14}$$
$$a = f(n) \tag{3.15}$$

Obviously, the single neurons previously introduced might become the atomic units of more complex and powerful neural networks. For example, the multi-layer perceptron, which is a directed, acyclic graph (called a feedforward network), connects multiple neurons together, increasing the complexity of the functions that can be learned. The topology presented in Figure 3.3 corresponds to multiple layers of neurons: input units are connected to one or more hidden layers, and, finally, a output layer returns the network's value related to the current input. This value, of course, depends on the network connection weights and on the activation functions that are used by the different layers.

An interesting property of standard feedforward networks is that it has been proven that networks using a single hidden layer of neurons with sigmoidal activation functions and an output layer of linear neurons have the power to approximate any function of arbitrary complexity [Cybenko, 1989]. Of course, the quality of the approximation depends on the number of neurons on the hidden layer and on the power of the learning algorithm.

Backpropagation Algorithm

Practical applications of neural networks have been limited until the discovery of the backpropagation learning algorithm in 1986, by Rumelhart et al. [1986]. With this technique, much more powerful uses of neural networks could be designed since it became possible to learn efficiently more complex functions. This algorithm is based on the

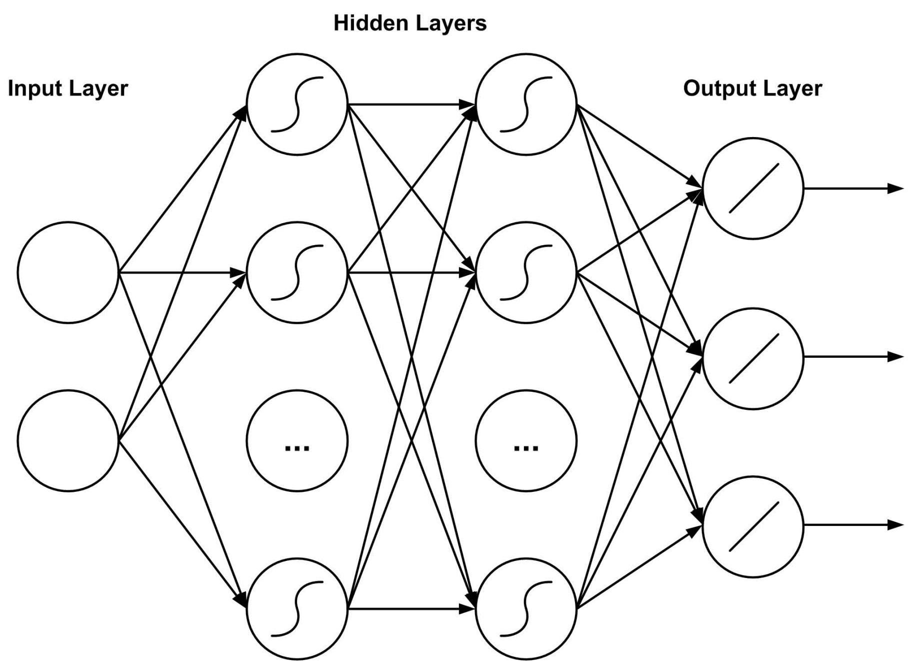


Figure 3.3: Multi-layer perceptron diagram.

gradient descent approach described earlier. First, the current inputs must be propagated forward in the network to calculate its outputs. Then, the error of the network's approximation is evaluated and compared to the desired output associated to the training examples. In general, the backpropagation algorithm uses the mean-squared error as a performance measure.

The algorithm then proceeds in calculating the gradient of this error according to the output layer's weights. Once this gradient at the output of the network has been calculated, the backpropagation algorithm propagates it to all of the weights of the neural network. Notice that the name backpropagation comes from the fact that we propagate this error gradient in the inverse direction of the feedforward network's usual signal flow, from the output of the network back to the input.

We must also note that the gradient calculation requires that all of the neurons' activation functions be differentiable. As a result, the linear and sigmoidal functions are widely used with neural networks as they possess this property.

The next section presents the policy-gradient methods, since they will later be used in our context of autonomous vehicle controllers, and illustrates how neural networks are actually used with this family of reinforcement learning algorithms.

3.3 Policy-Gradient Algorithms

The policy-gradient family of algorithms has gathered significant interest in recent years in the reinforcement learning community. Starting with the early work of Williams [1992] and with renewed interest later on from many researchers [Kimura et al., 1995; Marbach, 1998; Baxter and Bartlett, 2001; Sutton et al., 2000; Weaver and Tao, 2001], the policy-gradient approach has focused on reinforcement learning through the direct modification of a parametrized policy by using the gradient of a performance function.

Policy-gradient algorithms benefit from the same advantages of methods that rely on value function approximation. They are an efficient technique to learn when the model of the environment is unknown and they can generalize from previous experience using a parametrized representation to map all possible states to actions.

Interest in these methods also comes from the fact that they have many advantages over the value function approximation approach. First, learning in high-dimensional environments where the number of state variables and actions is important often results in complex value functions. In some cases, the control policy can be easier to represent than its corresponding value function [Anderson, 2000], and methods that modify directly the policy can thus be much better at learning an efficient behavior.

Another interesting advantage of policy-gradient methods is related to their convergence guarantees. While algorithms using value function approximation have relatively limited convergence properties (as briefly described in Section 3.2.1), policy-gradient methods behave efficiently with parametrized representations. Stochastic optimization theory even indicates that gradient descent methods are guaranteed to converge to a locally optimal solution as long as the gradient estimation is close to the true gradient [Baxter et al., 2001].

Moreover, policy-gradient algorithms are adapted to the fact that some learning tasks can sometimes be non-Markovian due to the hidden state problem (as seen earlier in Section 3.2, hidden state describes the fact that important information is missing for accurate action selection). In fact, work from Kimura et al. [1997] suggests that policy-gradient methods behave well in this context, since they are based on the POMDP framework, which is particularly efficient at treating noise in dynamics (since they handle uncertainty over actions) and noise in sensors (as they handle uncertainty of observations).

However, these methods do have a few limitations. Marbach [1998] indicates that their convergence guarantees are limited to locally optimal solutions. Moreover, he notes that the use of a particular parametrized representation actually limits the set of possible policies we can obtain. Unfortunately, there is no guarantee on the quality of the optimal policy from this limited set. Thus, the choice of policy representation remains an issue that is problem-dependent and that must often be addressed through empirical efforts.

Nonetheless, all of the advantages of policy-gradient algorithms made them a perfect fit for the resolution of complex control problems. Already, researchers have used this approach for problems such as autonomous helicopter flight [Bagnell and Schneider, 2001], robot locomotion [Field, 2005; Kohl and Stone, 2004] and unmanned underwater vehicle control [El-Fakdi et al., 2005]. Clearly, it seems that these algorithms could efficiently solve the problem of Cooperative Adaptive Cruise Control which is the topic of this thesis.

The current section will present the policy-gradient reinforcement learning algorithms. First, Section 3.3.1 will detail the general approach for the estimation of the policy-gradient. Then, Section 3.3.2 will describe the specific implementation we consider, which uses a neural network for policy representation.

3.3.1 Estimation of the Policy-Gradient

Policy-gradient methods are model-free algorithms that learn by modifying directly the weights of a parametrized stochastic policy in the direction that maximizes a performance criterion. The computation of this direction corresponds to the gradient of the policy's performance according to its parameters, and is the most important aspect of these learning algorithms.

As noted by Baxter and Bartlett [2000], the exact calculation of the gradient is possible only in environments for which the transition probabilities are known. Moreover, this operation is intractable for all but small problems, as it requires the inversion of the stochastic transition probability matrix which is computationally expensive for state spaces of more than a few states. Thus, work of reinforcement learning researchers on policy-gradient algorithms has primarily focused on finding ways to estimate the gradient of a policy's performance through simulation.

In this brief presentation of gradient estimation methods, we will first detail the REINFORCE algorithm [Williams, 1992], which was the first technique proposed to estimate, from direct experience with the environment, the gradient of a policy's performance for episodic learning tasks. Afterwards, we will detail the GPOMDP algorithm [Baxter and Bartlett, 2001], which proposes modifications to REINFORCE in order to estimate the gradient of infinite-horizon learning tasks. Finally, we will also present the OLPOMDP algorithm [Baxter et al., 2001], an online version of GPOMDP.

Note that throughout this presentation, we will omit most of the advanced details related to Markov Chain theory that serves as the theoretical foundation of these algorithms. For a formal analysis of these approaches, we refer the reader to Williams [1992], Marbach [1998] and Baxter and Bartlett [2001].

REINFORCE

To detail the steps leading to the REINFORCE algorithm, we first define the performance of a policy which we denote $\rho(\vec{\theta})$, as the performance ρ depends on the policy parameters $\vec{\theta}$. As seen in Equation 3.16, this performance corresponds to the expected reward $r(X)$ that can be obtained by following this policy.

$$\rho\left(\vec{\theta}\right) = E\left[r\left(X\right)\right] \tag{3.16}$$

Equation 3.17 shows that this expectation can be expressed as a sum, over all possible trajectories X, of the reward obtained through this trajectory weighted by the probability of observing this trajectory, which we will note $q_{\vec{\theta}}(X)$. This probability depends not only on the environment's transition probabilities, but also on the choices made using the stochastic parametrized control policy. Thus, we parametrize these probabilities by the policy's weights $\vec{\theta}$.

$$\rho\left(\vec{\theta}\right) = \sum_{X} r\left(X\right) q_{\vec{\theta}}\left(X\right) \tag{3.17}$$

In order to find the direction that optimizes this value, we need to find the gradient of this performance function by differentiating it according to the weights $\vec{\theta}$, as seen in Equation 3.18.

$$\nabla_{\vec{\theta}}\rho\left(\vec{\theta}\right) = \sum_{X} r\left(X\right) \nabla_{\vec{\theta}} q_{\vec{\theta}}\left(X\right) \tag{3.18}$$

By multiplying the left term of this equation by $\frac{q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)}$, we see that the gradient calculation corresponds to the expectation of Equation 3.20.

$$\nabla_{\vec{\theta}}\rho\left(\vec{\theta}\right) = \sum_{X} r(X)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} q_{\vec{\theta}}(X) \tag{3.19}$$

$$= E\left[r(X)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)}\right] \tag{3.20}$$

Then, averaging over N trajectories generated according to probabilities $q_{\vec{\theta}}(x)$ (as seen in equation 3.21) gives an unbiased estimation of the gradient (which means that with $N \to \infty$, the estimation converges toward the true gradient).

$$\hat{\nabla}_{\vec{\theta}}\rho\left(\vec{\theta}\right) = \frac{1}{N}\sum_{i=1}^{N} r(X_i)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X_i)}{q_{\vec{\theta}}(X_i)} \tag{3.21}$$

Under the Markov Decision Processes framework, we can define the probability $q_{\vec{\theta}}(X)$ of observing a specific trajectory X by the product of all the individual transition probabilities $p(x_{t+1}|x_t)(\vec{\theta})$ from state x_t to state x_{t+1} included in this trajectory, as seen in equation 3.22.

$$q_{\vec{\theta}}(X) = p(x_0)p_{\vec{\theta}}(x_1|x_0)p_{\vec{\theta}}(x_2|x_1)\dots p_{\vec{\theta}}(x_T|x_{T-1}) \tag{3.22}$$

$$= p(x_0)\Pi_{t=0}^{T-1} p_{\vec{\theta}}(x_{t+1}|x_t) \tag{3.23}$$

However, remember that, as noted earlier, these transitions depend both on the environment's dynamics and on the policy used for control. As a result, we can rewrite Equation 3.23 as Equation 3.24, where $p(x_{t+1}|x_t)$ are the transition probabilities of the environment and $\pi_{\vec{\theta}}(x_t, a_t)$ are the action selection probabilities of the stochastic policy.

$$q_{\vec{\theta}}(X) = p(x_0)\Pi_{t=0}^{T-1} p(x_{t+1}|x_t)\pi_{\vec{\theta}}(x_t, a_t) \tag{3.24}$$

We can use this result to replace the trajectory's probability $q_{\vec{\theta}}(X)$ in Equation 3.21. In that case, we can show that the ratio $\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)}$ actually corresponds to a sum, over the trajectory, of ratios $\frac{\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t,a_t)}{\pi_{\vec{\theta}}(x_t,a_t)}$ that are only dependent on the action selection probabilities of the policy. This result is illustrated in Equation 3.25 and its demonstration is given in Appendix A.

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} = \sum_{t=0}^{T-1}\frac{\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)} \tag{3.25}$$

Thus, like the name policy-gradient implies, we end up differentiating the policy's representation according to its parameters. Further details about this transformation are given by Baxter and Bartlett [2001], Field [2005] and Peters and Schaal [2006]. Of course, they indicate that this particular modification is an important aspect of policy-gradient algorithms, as it renders possible the optimization of the policy's performance without any knowledge or full observability of the system's dynamics.

With this modification, Equation 3.25 can be accumulated recursively over the course of a trajectory using an intermediate variable $\vec{z}_t$ (as seen in Equation 3.26). This variable, called the eligibility trace, contains a sum of gradients in the space of the policy parameters resulting from each transition.

$$\vec{z}_{t+1} = \vec{z}_t + \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)} \tag{3.26}$$

Using the same technique, rewards can also be accumulated recursively through a trajectory, using an intermediate variable r_t. The cumulative reward of a trajectory at time t can be obtained using a function $\phi(r_t, x_{t+1})$, as seen in Equation 3.27. This function can compute either a sum of rewards (Eq. 3.28), an average reward (Eq. 3.29) or a discounted reward (Eq. 3.30).

$$r_{t+1} = \phi(r_t, x_{t+1}) \tag{3.27}$$

$$\begin{aligned}
\phi(r_t, x_{t+1}) &= r_t + r(x_t, a_t) && (3.28)\\
\phi(r_t, x_{t+1}) &= r_t + \frac{r(x_t, a_t)}{T+1} && (3.29)\\
\phi(r_t, x_{t+1}) &= \gamma r_t + r(x_t, a_t) && (3.30)
\end{aligned}$$

With these two new variables, the gradient estimator of Equation 3.21 can now be expressed by Equation 3.31. Note that in this equation, r_i and $\vec{z}_i$ respectively denote the reward and the eligibility trace cumulated over trajectory i.

$$\hat{\nabla}_{\vec{\theta}} \rho \left(\vec{\theta}\right) = \frac{1}{N} \sum_{i=1}^{N} r_i \vec{z}_i \tag{3.31}$$

All of these steps lead to the classic REINFORCE algorithm, as shown in Algorithm 3. At each step of a simulated trajectory, the algorithm evaluates the eligibility of the policy's weights, stores it in $\vec{z}_t$ and also stores the reward obtained. When the algorithm finishes a trajectory, it calculates the current gradient estimation vector $\vec{\Delta}_{j+1}$ by attributing the reward obtained through the trajectory to all the weights according to their eligibility. Then, the reward and weight eligibility are reset and the next episode

Algorithm 3 REINFORCE policy-gradient algorithm (after [Baxter and Bartlett, 2001]).

```
1:  Initialize the trajectory count $j = 0$.
2:  Initialize the trajectory reward $r_0 = 0$.
3:  Initialize the eligibility trace $\vec{z}_0 = 0$.
4:  Initialize the gradient estimation sum $\vec{\Delta}_0 = 0$.
5:  for all transitions $x_t \rightarrow x_{t+1}$ of trajectory $j$ do
6:      if $x_{t+1}$ is a recurrent state $i^*$ then
7:          Update the gradient estimation: $\vec{\Delta}_{j+1} = \vec{\Delta}_j + r_t \vec{z}_t$.
8:          Reset the eligibility traces: $\vec{z}_{t+1} = 0$.
9:          Reset the trajectory reward: $r_{t+1} = 0$.
10:         $j \leftarrow j + 1$.
11:     else
12:         Update the eligibility trace: $\vec{z}_{t+1} = \vec{z}_t + \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}$.
13:         Update the trajectory reward: $r_{t+1} = \phi(r_t, x_{t+1})$.
14:     end if
15: end for
16: Return $\frac{\vec{\Delta}_N}{N}$
```

begins. When the algorithm has been through its N trajectories, it returns an estimate of the gradient that can be used to update the weights of the parametrized policy.

This algorithm returns an unbiased estimator of the policy-gradient. Such an approach is interesting because it only necessitates K parameters for the numbers of weights in the policy, K parameters to store the eligibility trace $\vec{z}_t$ of these weights and one parameter to store the cumulative reward of a trajectory r_t.

However, this algorithm does have several drawbacks which have prevented it to be used in practice on real world problems. As noted by Baxter and Bartlett [2001], these drawbacks come from the fact that this algorithm is dependent on the presence of a recurrent state i^*. This recurrent state is used to identify the end of a trajectory and indicates when the gradient estimation should be updated and when the eligibility should be reset.

First, when facing partial observability the algorithm cannot perceive that it has just entered a recurrent state. In this case, the algorithm does not know that it should update its gradient estimation and reset both its current reward and eligibility trace. Thus, chances are that the gradient estimation will be updated infrequently, only when the recurrent state will have been observed.

Another similar problem comes from the fact that acting in large state spaces often results in rare visits to the recurrent state. Again, this causes infrequent updates of the gradient estimation.

Both of these issues have the effect of degrading the performances of the algorithm because, as detailed by Baxter et al. [2001], the variance of the gradient estimation is dependent on the time between accesses to a recurrent state. Thus, when recurrence time is high (due to large state spaces or to partial observability), accurate estimation of the gradient will need an important number of samples, resulting in an extremely slow learning process.

Of course, modifications to this basic policy-gradient algorithm were developed. In the next section, we will detail the GPOMDP algorithm of Baxter and Bartlett [2001], which proposes to modify this REINFORCE algorithm in order to address the issue of dependence on a recurrent state.

GPOMDP

The GPOMDP algorithm [Baxter and Bartlett, 2001] builds on Williams' REINFORCE by adding support for infinite-horizon learning tasks. Again, this approach can learn through interaction with the environment, but this time, it is independent of the identification of a recurrent state. Instead of updating the gradient estimation after each trajectory (such as done by REINFORCE), GPOMDP updates the gradient after a number of transitions T. In order to do so, its authors propose to discount eligibility traces rather than to keep an history of this value over a complete trajectory. This results in a biased estimator, but with much more efficient performances for real applications. Finally, the authors have also proven convergence of such an algorithm.

The modification they propose is to introduce a new variable, β, that is used in the recursive calculation of the eligibility traces, as seen in Equation 3.32. As described in [Baxter and Bartlett, 2001; Baxter et al., 2001], this variable corresponds to a trade-off between bias and variance of the gradient estimator. When $\beta \to 1$, the estimator is similar to REINFORCE's estimation: it has a small bias but large variance. On the other hand, when $\beta \to 0$, the gradient estimation discounts the eligibility trace, resulting in a larger bias, but also in a reduced variance of the estimator. Careful setting of this value depends on the underlying Markov Chain, and theoretical details about the implications of this variable are also given in [Baxter and Bartlett, 2000, 2001; Baxter et al., 2001].

$$\vec{z}_{t+1} = \beta \vec{z}_t + \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)} \tag{3.32}$$

The resulting algorithm can be seen in Algorithm 4.

Algorithm 4 GPOMDP policy-gradient algorithm (after [Baxter and Bartlett, 2001]).

```
1:  Initialize $\beta \in [0, 1)$.
2:  Initialize $T > 0$.
3:  Initialize $\vec{\theta}_0$.
4:  Initialize $\vec{z}_0 = 0$.
5:  Initialize $\vec{\Delta}_0 = 0$.
6:  for all steps $t = 0..T-1$ of a trajectory do
7:      Get the current observation $x_t$ of the environment.
8:      Select action $a_t$ with policy $\pi_{\vec{\theta}}(x_t, a_t)$.
9:      Observe the resulting state $x_{t+1}$.
10:     Get the resulting reward $r_{t+1}$.
11:     Update the eligibility trace: $\vec{z}_{t+1} = \beta \vec{z}_t + \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}$.
12:     Update the gradient estimation: $\vec{\Delta}_{t+1} = \vec{\Delta}_t + r(x_{t+1}) \vec{z}_{t+1}$.
13: end for
14: return $\frac{\vec{\Delta}_T}{T}$.
```

However, the authors indicate (in [Baxter and Bartlett, 2000]) that the number of transitions T needed before updating the gradient estimation is still required to be large in order for it to have low variance. To address this problem, algorithms similar to GPOMDP have proposed to use a reward baseline as a mean to reduce variance of the gradient estimation [Kimura et al., 1997; Weaver and Tao, 2001]. This baseline represents a value against which the current observed reward is compared. For example, Weaver and Tao [2001] have proposed to substract the estimated average reward b to the observed reward $r(x_{t+1})$ (as seen in Equation 3.33). They have shown empirically that this approach performs better than GPOMDP.

$$\vec{\Delta}_{t+1} = \vec{\Delta}_t + (r(x_{t+1}) - b)\, \vec{z}_{t+1} \tag{3.33}$$

OLPOMDP

The OLPOMDP algorithm, again proposed by Baxter and Bartlett [2000], is an online version of GPOMDP. Instead of accumulating the gradient estimation over a certain number of state transitions T before updating the policy, it modifies at each step the weights of the policy representation. This stochastic gradient algorithm, presented in Algorithm 5, has been shown to converge close to a local optima [Baxter and Bartlett, 2000].

Algorithm 5 OLPOMDP policy-gradient algorithm (after Baxter and Bartlett [2000]).

```
1:  Initialize $\beta \in [0, 1)$.
2:  Initialize $T > 0$.
3:  Initialize $\vec{\theta}_0$.
4:  Initialize $\vec{z}_0 = 0$.
5:  for all steps $t = 0..T-1$ of a trajectory do
6:      Get the current observation $x_t$ of the environment.
7:      Select action $a_t$ with policy $\pi_{\vec{\theta}}(x_t, a_t)$.
8:      Observe the resulting state $x_{t+1}$.
9:      Get the resulting reward $r_{t+1}$.
10:     Update the eligibility trace: $\vec{z}_{t+1} = \beta \vec{z}_t + \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}$.
11:     Update the policy parametrization: $\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha r(x_{t+1}) \vec{z}_{t+1}$.
12: end for
13: return $\vec{\theta}_T$.
```

3.3.2 OLPOMDP Using a Neural Network Policy Representation

The current section will briefly illustrate how the OLPOMDP algorithm can be implemented using a neural network to represent the policy. In this particular case, the weights of the neural network correspond to the policy parameters. This approach will be of interest as we will refer to this architecture later to learn autonomous vehicle control policies.

The first thing to remember about policy-gradient algorithms is that they are based on the use of stochastic parametrized policies. Thus, the neural network's task consists in receiving the current state features as inputs, and returning the probabilities of the discrete actions available to the agent. In order to do so, the network computes the values of the different actions, and relies on an additional layer that uses the soft-max function to compute their probability. The resulting neural network architecture is illustrated in Figure 3.4.

The soft-max function, shown in Equation 3.34, is particularly useful in the policy-gradient context as it meets the differentiability requirements of the OLPOMDP algo-

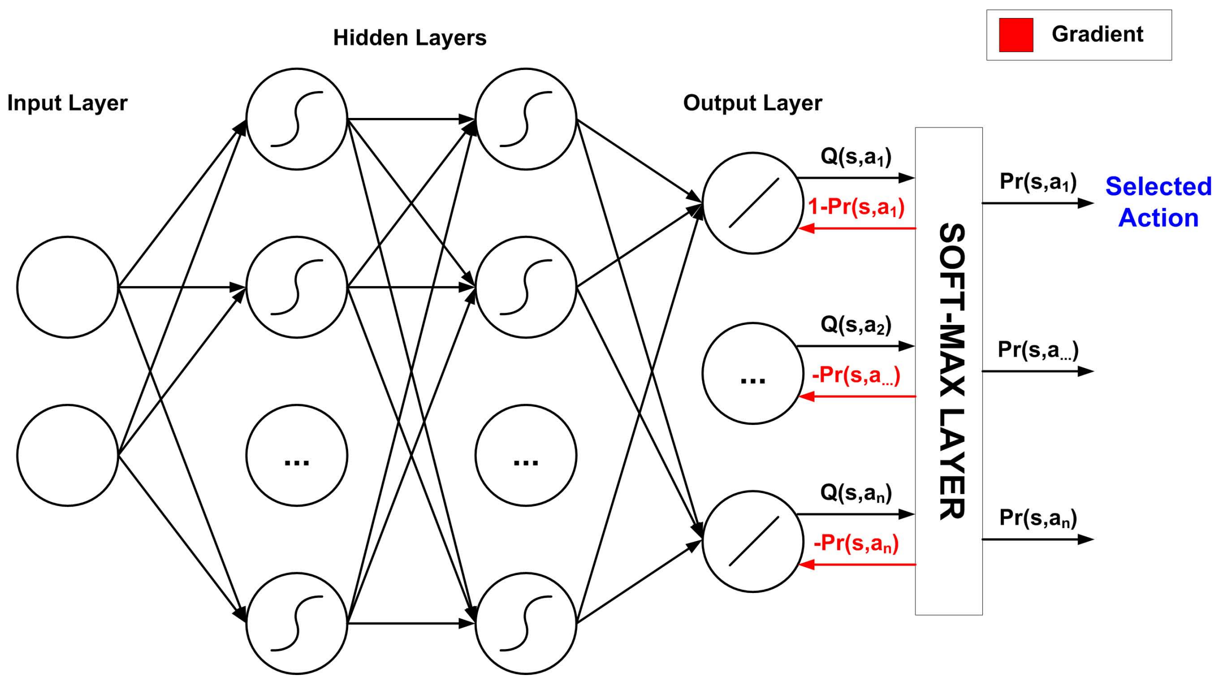


Figure 3.4: Neural network with a soft-max output layer.

rithm (and of all policy-gradient algorithms for that matter).

$$\pi\left(s, a_i\right) = \frac{e^{Q(s,a_i)}}{e^{Q(s,a_1)} + \ldots + e^{Q(s,a_m)}} \tag{3.34}$$

In fact, differentiating this function according to the action values $Q(s, a_i)$ gives a simple and elegant result, as illustrated in Equation 3.35 (the proof is given in Appendix B).

$$\nabla \pi_\theta(s, a_i) = \begin{cases} \pi\left(s, a_i\right)\left(1 - \pi\left(s, a_i\right)\right) & \text{if } a_i = a_j, \\ -\pi\left(s, a_i\right)\pi\left(s, a_j\right) & \text{if } a_i \neq a_j. \end{cases} \tag{3.35}$$

Of course, to compute policy-gradient $\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t, a_t)$, one needs to differentiate the policy according to all of its weights. In the case of a neural network representation, it is preferable to use the soft-max gradient of Equation 3.35, which is actually the gradient of the network's output, and find the corresponding local gradients of the weights of the network. This can be easily achieved by using the standard backpropagation algorithm, which was briefly described in section 3.2.2. Thus, the soft-max gradient is propagated back through the network to find the gradient of each weight. However the policy-gradient algorithm defines the eligibility (local gradient of the weights for a single step) as $\frac{\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t,a_t)}{\pi_{\vec{\theta}}(x_t,a_t)}$. As a result, the soft-max gradient divided by $\pi_{\vec{\theta}}(x_t, a_t)$ is the actual value to be backpropagated, as illustrated in Equation 3.36. These values are also illustrated in red on Figure 3.4.

$$\frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(s, a_i)}{\pi_{\vec{\theta}}(s, a_i)} = \begin{cases} 1 - \pi_{\vec{\theta}}(s, a_i) & \text{if } a_i = a_j, \\ -\pi_{\vec{\theta}}(s, a_j) & \text{if } a_i \neq a_j. \end{cases} \tag{3.36}$$

Once the backpropagation algorithm has evaluated the local gradient of each neuron, the eligibility trace $\vec{z}_{t+1}$ can be updated. Finally, the weights can be modified using the OLPOMDP update rule $\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha r(x_{t+1})\vec{z}_{t+1}$.

Using neural networks with the GPOMDP and OLPOMDP algorithms has been an approach considered by the authors [Baxter and Bartlett, 2001; Baxter et al., 2001]. Moreover, implementation details are also given in [El-Fakdi et al., 2005], as these researchers have used a similar architecture for the autonomous control of an underwater vehicle.

3.4 Summary

In this chapter, we introduced the theory that will be behind the autonomous vehicle controller that will be described in Chapter 5.

First, we have presented the Markov Decision Process framework which formalizes the reinforcement learning problem. We have also detailed classic resolution approaches for both model-based and model-free problems. Then, we detailed an important issue related to the use of these algorithms to solve complex applications in the real-world. In fact, we described that solving high-dimensional problems with a tabular representation of the value function is often intractable. Thus, we have presented some value function approximation techniques and we have described how artifical neural networks are appropriate for such approximation purposes.

Finally, we have detailed the policy-gradient methods, which propose an alternate way of learning a policy without using an explicit value function. These methods are based on the direct modification of a parametrized stochastic policy in order to increase its performance. This approach solves some of the problems of value function approximation and has been used for the resolution of complex control problems. Policy-gradient algorithms will be revisited in our experiments section.

Chapter 4

Vehicle Simulation Environment

To fulfill the goal of this thesis which aims to design an autonomous longitudinal vehicle controller using reinforcement learning, we have to run numerous experiments in a vehicle simulator that is adapted for the task. To this end, we have adopted the following requirements for the simulator we plan to use:

(a) the behavior of the simulated vehicles should be as close as possible to that of real vehicles;

(b) the simulator should be flexible enough to let us embed in simulated vehicles a reinforcement learning engine that can be used to select driving actions;

(c) the simulator should be able to run at "faster-than-real-time" in order to execute a large number of learning simulations in decent time.

A study of existing vehicle simulation tools justified the development from the ground up of a new vehicle simulator, with the main reason being the lack of flexibility of current simulators for the integration and implementation of reinforcement learning algorithms.

In this chapter, we describe the simulator that we have developped as a testbed for learning autonomous vehicle behavior. First, Section 4.1 will make an overview of the simulator and will detail how it coordinates all of its operations. Then, Section 4.2 will detail the physics engine that is responsible for the accurate motion of the vehicles. Afterwards, Section 4.3 will present both the sensor and the communication managers that are used by vehicles for the perception of their environment. Finally, Section 4.4

will briefly describe the reinforcement learning architecture that we have embedded in simulation vehicles for action selection, in order to obtain our autonomous controller.

4.1 Simulator Overview

To address our needs for a precise simulation tool in which we can learn autonomous vehicle control, we have built, using the C++ programming language, a microscopic vehicle simulator. We selected a microscopic simulation approach because it focuses on modeling the behavior, more or less accurately depending on the precision needed, of each simulated vehicle. This comes in contrast with macroscopic vehicle simulation methods, that consider the use of equations that model globally the traffic conditions. Macroscopic simulators are mainly used to simulate a large amount of vehicles for traffic flow studies [Ehlert, 2001].

Another important characteristic of our simulator is that we have chosen a discrete-time approach. This means that simulations are divided in time steps Δt, which represent a certain amount of simulated time that has elapsed between two executions of the simulator's main control loop. Whenever the loop ends an iteration, the current time is increased, and then the loop restarts in order to update the position of the objects for the following time step. Since the computation of these updates do not rely on actual time but rather on the simulation time step value Δt, this method has the advantage that it can efficiently run at "faster-than-real-time".

In our case, we have used a time step value of 10 milliseconds (corresponding to a 100 Hz update frequency), which was required for accurate physics modeling. However, we designed our control loop so that we could decouple the refresh rate of the different modules from the simulator's main loop. Thus, sensor, communication and learning modules were able to update at different frequencies. For example, we could update sensors and communication at every 100 milliseconds, while taking learning decisions at every 250 milliseconds. We will see, in Section 4.4 and Chapter 5, that separating the update rate of the different modules was particularly important for the efficiency of the learning module.

Now, let's observe in detail the simulator's inner workings as illustrated in Figure 4.1. This figure clearly shows that the simulator actually relies on a simple control loop that takes care of triggering the various update methods for every simulation object. First, the simulator calls a method that updates the internal perception of each vehicle. In the second step, the simulator executes the action choice method of every object.

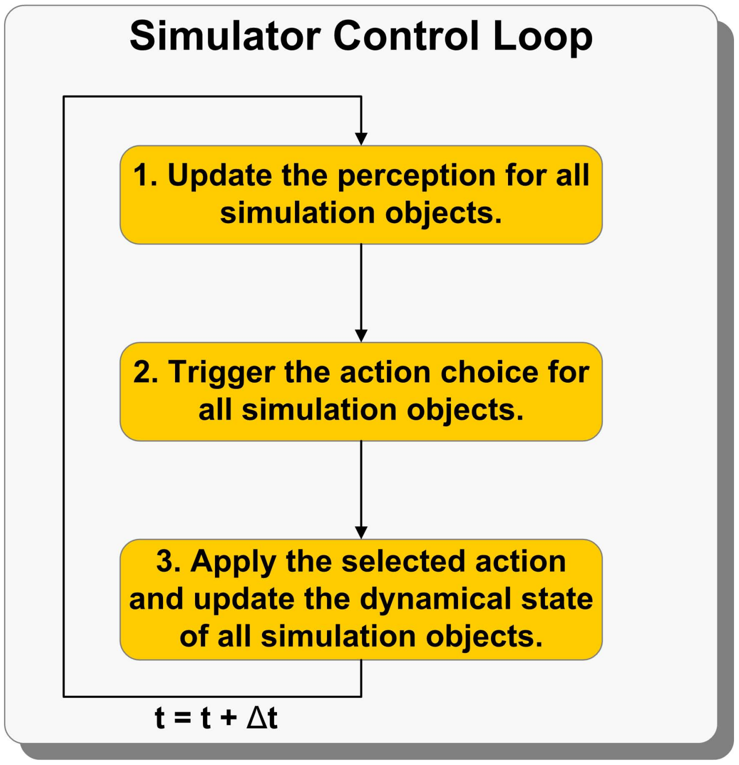

Simulator Control Loop
1. Update the perception for all simulation objects.
2. Trigger the action choice for all simulation objects.
3. Apply the selected action and update the dynamical state of all simulation objects.
t = t + Δt

Figure 4.1: Simulator control loop.

It is in this method that the vehicles use their updated perception in order to select their next action to take. Finally, after having applied the actions of every object, the simulator updates their position according to the dynamics engine, increments the current time step, and proceeds to the simulation of the next frame.

Figure 4.2 shows the resulting control loop from the point of view of a single vehicle. This figure gives more detail about the action selection process, as it illustrates the fact that the decision-making module was designed around two separate layers. First, the *Coordination Layer* is responsible for the selection of "high-level actions" such as lane-changing and secure vehicle-following. This layer was created for experiments conducted at DAMAS Laboratory by Laumonier [2008] on learning multi-agent interactions in the context of a collaborative driving system. Once this layer has chosen the appropriate action to take, it transmits it to the *Action Layer*, which must achieve this action by selecting the appropriate "low-level actions" that correspond to the vehicle's steering, brakes and throttle. However, the current thesis only focuses on learning a policy for the "secure vehicle-following" high-level action. Thus, for the remaining of this thesis, we will ignore the presence of the *Coordination Layer* in our simulator's control loop. Instead, we refer to Desjardins et al. [2007, 2008] for more details about the interactions between both action selection layers.

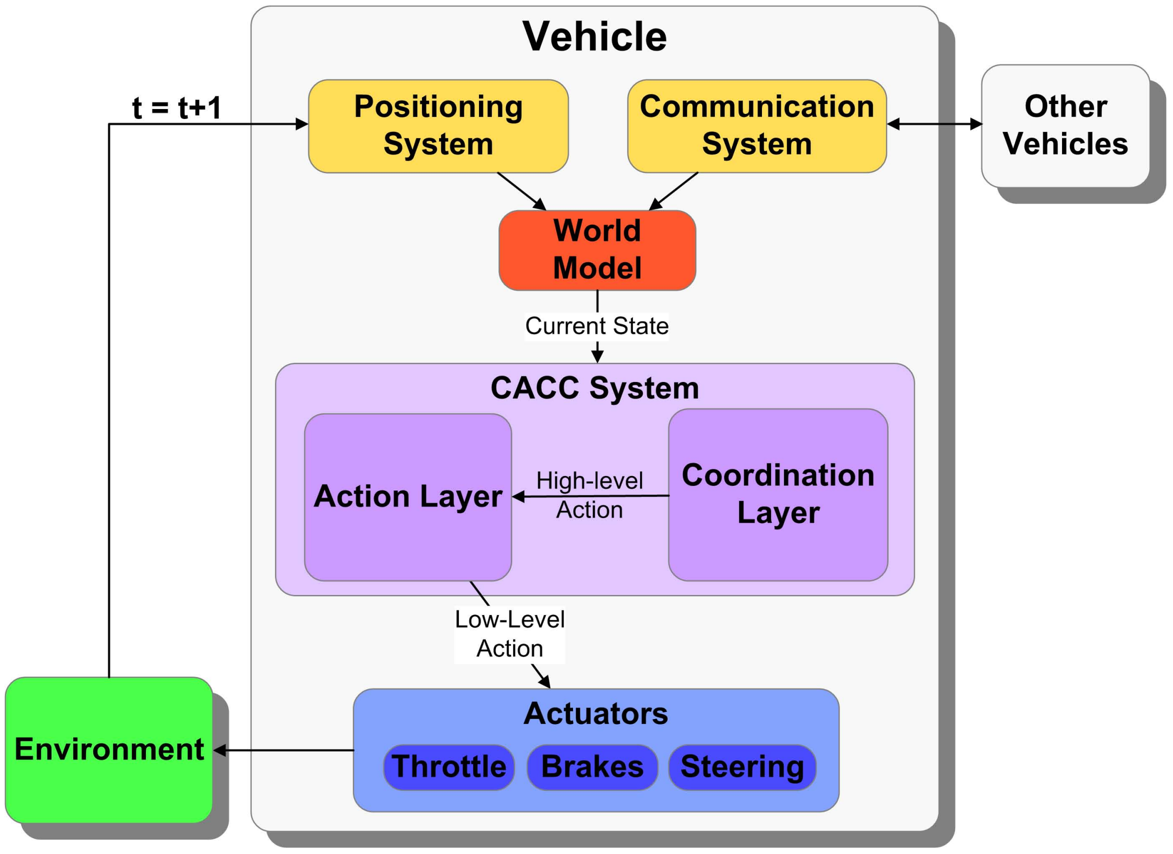
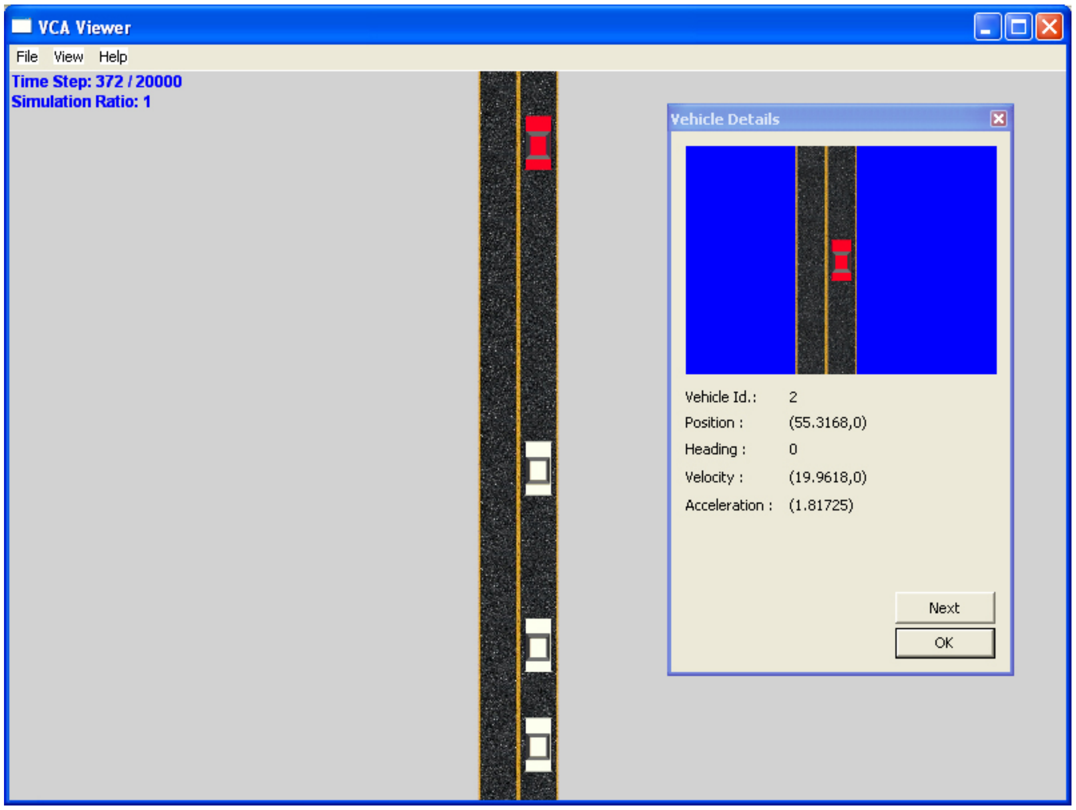


Figure 4.2: Vehicle control loop.

We defined the learning simulations to run using scenarios, which are simple configuration files that contain all the information needed for the simulator to execute. These scenario files specify the map of the road network to use, the number of simulation episodes to execute, the number of vehicles in the simulation, and also specifies the location of each of these vehicles' own configuration file.

Vehicle configuration files list everything needed to initialize a simulated vehicle by describing its initial position, the sensors and the communication protocols it uses, but also, most importantly, these files contain all the parameters required for the execution of the deliberation mechanism. For automated vehicles (such as the front vehicles used in Chapter 5), these files list the velocity profile to be followed through the simulation. When using a learning algorithm, the configuration file details all about the state definition, the reward function and the learning algorithm to be used.

Another important aspect of our simulator is that it is equipped with an efficient log system that enables the gathering of simulation data. Using this system, we were able to obtain, for each simulation, information about the dynamical state of vehicles



Time Step: 372 / 20000
Simulation Ratio: 1

Vehicle Id.:	2
Position :	(55.3168,0)
Heading :	0
Velocity :	(19.9618,0)
Acceleration :	(1.81725)

Figure 4.3: Simulation viewer.

(position, velocity, acceleration, etc.), but also regarding the execution of the learning algorithms.

Finally, along with the simulator, we have designed, using the DirectX programming API [Microsoft, 2008], a simulation viewer that presents visually the behavior of vehicles. The viewer works by first loading the log files corresponding to a particular simulation, and then by updating on screen the position of vehicles according to what is contained in these files. A screenshot of this simulator is given in Figure 4.3.

4.2 Vehicle Dynamics Engine

In order to show that it is possible to learn an autonomous vehicle controller in a complex environment, our simulator should model with accuracy the motion of real vehicles. More specifically, the dynamics engine should update the state of simulation vehicles, by taking as input commands from their actuators (steering, brakes and throttle) and by computing their resulting position, velocity, and acceleration.

For a computationally efficient simulation, we decided to port to C++ a vehicle dynamics module previously developped in Java by Hallé [2005] as part of previous work for the Auto21 project. This module is based on the "single-track" model, as described by Kiencke and Nielsen [2000]. This model features a non-linear approximation technique that takes particular care in the description of the wheel model and of the vehicle's longitudinal and lateral motion. Even if this "single-track" approach simplifies the calculation by considering the wheels on each axis as a single unit (thus, for example, both front wheels are seen as one), the model is still quite precise as it integrates the notions of friction and of wheel slip in the computation of the resulting wheel traction force. Of course, these wheel dynamics play an important role in the representation of realistic vehicle behavior.

The dynamics module also implements the simulation of a vehicle's driveline, as detailed by Huppé [2004]. By taking as inputs the pressure on the acceleration and brake pedals, the driveline model first computes the engine torque, transmits it to the automatic transmission module, which then calculates the resulting drive torque. Finally, the drive torque is transformed in order to obtain the wheels' angular acceleration that can be used as inputs by the wheel model. Since our driving agents take direct action on the vehicle's acceleration and brake pedals, the presence of a complex, non-linear driveline whose behavior is close to that of a real vehicle is a valuable characteristic for us, as it gives much more realism to the definition of our problem.

However, as noted by Huppé [2004], one limitation of the driveline is that there are some issues with the dynamics when the vehicle goes at very low velocities. As a result, we designed the experiments of Chapter 5 so that this would not be a problem (for example, the front vehicle of our simulations will not make a complete stop, but will rather slow down to a velocity where the dynamics are still correct).

For a complete description of the modules that are part of our vehicle dynamics engine, we refer to Hallé [2005], Huppé [2004] and Kiencke and Nielsen [2000].

4.3 Sensor and Communication Managers

This section describes both the *Sensor Manager* and the *Communication Manager* that are included in every vehicle in order to take care of updating the sensors and the communication system.

The *Sensor Manager* is used to keep track of all sensors embedded on a vehicle. When the simulation environment sends a sensor update command (which corresponds to step 1 of Figure 4.1), the *Sensor Manager* updates every sensor that it lists for the vehicle. However, for extended flexibility, we designed our system so that each sensor could have its own refresh rate. As a result, the *Sensor Manager* only triggers updates on sensors for which the refresh rate has elapsed since their previous update. Moreover, we added another useful feature to sensors, by integrating, in their data structure, information about their previous state. This history has enabled us to access data on the current value of the sensor or to obtain a filtered value corresponding to an average over the previous sensor observations.

For now, the only sensor that we implemented is a front vehicle sensor. Many of such sensors are currently built by third-party manufacturers and are being embedded in vehicles for Adaptive Cruise Control applications. These sensors all work in a similar fashion, by employing either radar or laser technology in order to sense the area in front of a vehicle. Their specifications are usually described by their reach, by their angle of observation and by their update (refresh) rate. The European IV project PReVENT explains how such sensors work in their state-of-the-art on that technology [Strobel et al., 2004], while its companion report [Strobel and Servel, 2004] gives a list of existing sensors and their specifications. Using similar approaches as presented in these papers, we have implemented the specifications of the Bosch long range radar sensor. Thus, we considered a range value of 120 meters, a beam angle of 8 degrees and a sensor refresh rate of 10 Hz (however, Chapter 5 shows that some of our experiments were done using a 4 Hz refresh rate (250 ms) in order to synchronize sensor updates with decision steps of the learning engine). Finally, our front laser implementation did not include the notion of sensor noise or delay.

The other important module for the management of vehicle perceptions is the *Communication Manager*, which is used to handle each car's inter-vehicle communication system. When vehicles decide to transmit information to others, they pass the messages through their communication system. Then, these messages are relayed by the *Communication Managers* to the simulator's global communication system, whose task is to simulate the protocols used for the transfer of messages between the vehicles.

Protocols specify how the reception of messages is affected by factors such as communication range, delay, and packet loss. These parameters are applied by the global communication system to every message to be transmitted. These messages are then transferred accordingly to each receiving vehicle's *Communication Manager*. In our case, we have made the assumption of the availability of a communication channel having a messaging delay of 100 ms, which is a value that has been considered as acceptable for use in future safety critical systems [Bishop, 2005]. The transmission range has been evaluated to 100 meters, since this value is consistent with the 802.11b wireless LAN communication protocol used for inter-vehicle messaging [Günter and Großmann, 2005]. Again, we did not consider noise in the communication channel nor packet loss: the messages are received exactly as they were sent.

To conclude this section on the sensory and communication systems, we must note that the focus of our experiments was not on the integration and implementation of exact specifications and behavior of these systems. However, the approach we developed is still a precise model that is also adequate for our needs related to environment perception. Of course, it could be interesting, in future work, to bring this framework even closer to reality by integrating more accurate sensor and communication models that could include precise simulation of sensor noise, delay, and also of communication noise, delay and packet loss.

4.4 Learning Module

As mentionned earlier, the main reason that justified the development of our own vehicle simulation tool was the need to include the ability to run reinforcement learning algorithms. To meet this design goal, we had to develop our vehicle action choice mechanism accordingly, which we did by embedding a learning module in the *Action Layer* of Figure 4.2. This figure shows that the *Action Layer* receives as inputs, at every time step, a representation of the current state of the environment, which, of course, is used by reinforcement learning algorithms in order to select actions directly on the throttle, brakes and steering. After this action has been selected, the environment computes the updated position, velocity, acceleration and heading of the vehicle.

In order to decouple as much as possible the environment dynamics from the learning module, we added a "post-action" method to the *Action Layer*. This method is called by the simulator's control loop, after the environment and sensor update. Its task is to assign rewards according to the updated perception of the vehicle, and to trigger the computation of the remaining steps of the learning algorithms in order to conclude the simulation of the previous step. The updated control loop architecture from the vehicle point of view can be seen in Figure 4.4.

However, after a some empirical experiments with this architecture, we noticed a lack of efficiency of the learning module. Indeed, the execution of the *Action Layer* at the same frequency than the vehicle dynamics actually made it impossible to learn

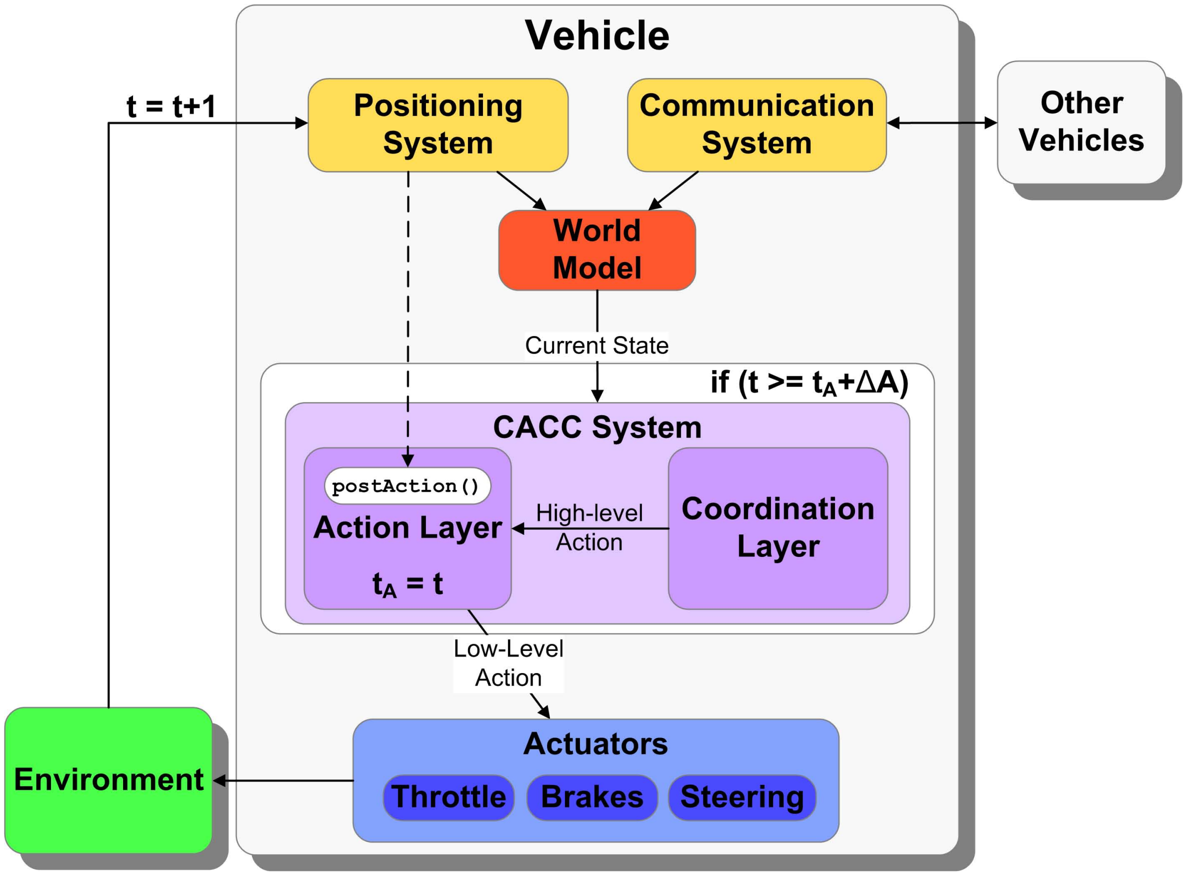


Figure 4.4: Vehicle control loop updated for reinforcement learning.

any kind of behavior. This problem came from the fact that, with our complex vehicle dynamics engine, the effects of a vehicle's actions take more than a single 10 *ms* time step (corresponding to the dynamics' refresh rate) before they can be observed on the velocity of a vehicle. Of course, since it is nearly impossible to learn anything valuable when you cannot observe the result of the actions that are taken, we had to address this particular issue. Thus, as mentionned previously in this chapter, we decoupled the action selection mechanism from the dynamics engine. Figure 4.4 illustrates this modification as it shows that the execution of the *Action Layer* is conditionnal to the elapse of this particular decision-making time interval ΔA.

Finally, another important aspect that we considered for the design of the *Action Choice* module was its flexibility, as we wanted to design an efficient learning engine that could easily be extended to run different algorithms, to use numerous state definitions and to consider different state space representation techniques. Thus, we again relied on the use of configuration files to specify which parameters to load at runtime, as our simulator relied on the Factory Method design pattern [Gamma et al., 1995] along with a hierarchy of abstract classes to instantiate the appropriate elements of our learning framework.

4.5 Summary

In this chapter, we have made an overview of the simulator that we designed in order to conduct our experiments with learning agents.

First, we observed the general architecture of the simulator, showing the main control loop and how its modules are inter-related from the vehicle's point of view. Then, we detailed the aspects of vehicle perception, by presenting both the Sensor and Communication Managers. Finally, we made a brief overview of the characteristics of the learning module.

With these details in mind, the next chapter will describe the experiments that we have done in this simulator in order to obtain an autonomous vehicle control policy.

Chapter 5

Experiments and Results

This chapter details our efforts to design an autonomous Cooperative Adaptive Cruise Control (CACC) system that integrates both sensors and inter-vehicle communication in its control loop in order to keep a secure vehicle-following behavior. To achieve this, we propose to use reinforcement learning and, in particular, the policy-gradient methods that we described in Chapter 3, in order to learn to control a vehicle by direct interaction with the simulated driving environment that was described in Chapter 4.

Thus, each of the sections below presents the simulation setup of a specific learning scenario and evaluates the performance of the resulting policies when used to control vehicles. First, we detail our preliminary work done towards learning an Adaptive Cruise Control (ACC) policy using a simplified vehicle dynamics model. Then, we describe the experimentations done to learn an ACC controller in a complex vehicle dynamics model. Finally, we detail how the ACC learning framework was modified to obtain a Cooperative Adaptive Cruise Control (CACC) by integrating an acceleration signal of a preceeding vehicle, obtained using inter-vehicle communication.

5.1 ACC Controller: A Simplified Model

For our first effort towards learning an autonomous Cooperative Adaptive Cruise Control, we considered the use of an environment model featuring simplified vehicle dynamics. This environment is based on a simple transition function, enabling us to concentrate on the definition of the learning framework without having to face the complexity inherent to acting in a high fidelity vehicle dynamics simulator. Moreover, instead of directly learning a CACC controller, we first focused on learning a simpler autonomous Adaptive Cruise Control (ACC) system, that will serve us as a basis for our future work. The results presented in this section have been detailed previously in [Desjardins et al., 2007] along with the architecture of the CACC system, as seen in Chapter 4.

5.1.1 Simplified Model Environment

The simplified model we have used for this preliminary work on ACC features a road represented by a grid on which the vehicles can move. It is inspired from other simplified vehicle environments used for research such as, for example, work of Ünsal et al. [1999]. The road is 3 meters wide, infinite in length, and is discretized by cells of size 0.25 by 0.25 meters. In this model, the state of a vehicle is defined by its position and velocity. Position corresponds to the location of its center cell while possible vehicle velocities go from 0 to 30 m/s.

The actions available to a vehicle are essentially acceleration actions that modify its velocity. Pressing on the gas pedal has the effect of adding 1 m/s to the vehicle's current velocity, while pressing on the brake pedal subtracts 2 m/s to the current velocity. There is also a "no-op" action, that reduces the vehicle's velocity by substracting 1 m/s to the current value. All actions are considered as deterministic, meaning that their result is always guaranteed when applied. However, when an action cannot be executed, such as trying to accelerate when one is already at the maximum velocity or braking when the vehicle is stopped, the action does not have an effect.

Finally, the discretization in time for decision making and for updating the system's dynamics is of 0.250 seconds. We also make the hypothesis that there are no delays in the system whatsoever, whether it be from sensors or actuators.

5.1.2 Learning Task Description

For learning an Adaptive Cruise Control policy in our simplified environment previously described, we considered a simple acceleration scenario. In this situation, the vehicle trying to learn an ACC control policy is placed 5 meters behind the leading car, and both vehicles are initially stopped. Then, the leading vehicle accelerates to reach a 20 m/s velocity. By definition of ACC systems, the task of the learning vehicle is to accelerate and learn to stabilize itself at a secure distance from the leading vehicle.

Using reinforcement learning, we can obtain an efficient control policy for this behavior by:

(a) using an adequate state representation that gives us all the necessary information for optimal action selection (which refers to the Markov property);

(b) having an action space that can lead the agent to properly explore the state space;

(c) using an appropriate reward function that guides the agent toward its design goal.

All these elements properly defining the learning task will be detailed in the next subsections.

State and Action Spaces

For the definition of the states, we have considered the use of a small and efficient representation, featuring only two state variables, namely the headway (Hw) and the headway derivative (ΔHw). Headway (also called "range" in some research papers) refers to the distance in time from a front vehicle, and its calculation is given in Equation 5.1. This is a pretty standard measurement [Bareket et al., 2003; Naranjo et al., 2006b; Richardson et al., 2000] for inter-vehicle spacing that has the advantage of being dependent of the current velocity of the vehicle. For example, the faster a vehicle goes, the farther it should be from its predecessor in order to respect the prescribed secure distance.

$$Hw = \frac{(Position_{Leader} - Position_{Follower})}{Velocity_{Follower}} \tag{5.1}$$

$$\Delta Hw = Hw_t - Hw_{t-1} \tag{5.2}$$

The headway derivative (also called "range rate"), given in Equation 5.2, contains valuable information about the relative velocity between the two vehicles. It indicates whether the vehicles (front vehicle and its follower) have been moving closer to or farther from each other since the previous update of the value. Both headway and headway derivative are provided by a laser sensor, as described in Section 4.3.

For this simplified version of the ACC control problem, the two state variables are discretized for representation of the policy using a Q-Values table. Headway has a range of 0 to 10 seconds and is discretized in 1000 intervals, while the headway derivative can take values between -0.1 and 0.1, with 10 discrete intervals. This resulted in a relatively good discretization that keeps the number of states of the system to a reasonable size for efficient resolution in decent time. As described in the previous section concerning the dynamics of our simplified environment, the action space contains three actions: a gas action, $G100$ (full pressure on the gas pedal), a braking action, $B100$ (full pressure on the brake pedal) and a "no-op" action, $NO-OP$. The state and action space of our MDP framework could be described formally by Equations 5.3 and 5.4.

$$S = \{Hw, \Delta Hw\} \tag{5.3}$$
$$A = \{G100, B100, NO-OP\} \tag{5.4}$$

Reward Function

The success of the learning task, as noted earlier, also depends on the reward function used by the agent, since this function is mostly used by a learning algorithm to direct the agent in areas of the state space where it will gather the maximum expected reward. In other words, for a learned controller to display efficient vehicle following behavior, the reward function must be designed to give positive reward values to actions that get the agent towards the safe inter-vehicle distance to the preceeding vehicle.

In our case, the task at hand corresponds to a "maintenance" task, as the agent's goal is to stay in a certain area of the state space for as long as possible. However, some failure states that terminate the episode are encountered when the agent gets in parts of the state space where it is considered to be too close from its preceeding vehicle. Thus, the reward function we propose, shown in Figure 5.1, tries to encapsulate this information. Since it is considered that the secure inter-vehicle distance should be around 2 seconds (as detailed in Section 2.2.1), we decided that the large positive reward would be given when the vehicle would enter a safe zone. This zone extends at ± 0.5 seconds from the headway goal of 2 seconds. Moreover, we also considered a smaller zone, at ± 0.1 seconds from the safe distance, where the agent would receive an even more important reward. The desired effect of such a reward function is to advise the agent to stay as close as possible to the safe distance. On the other hand, we give negative rewards to the vehicle when it is located too far from the safe distance or when it finds itself too close from the preceeding vehicle.

However, the use of such a reward function left us with the problem of poor exploration of the state space. Indeed, when simulation starts, the leading vehicle accelerates according to its pre-defined acceleration profile, rapidly leaving the learning vehicle far behind. The learner will ultimately be unable to reach the safe distance in decent time, as it would have to select through exploration a large number of consecutive acceleration

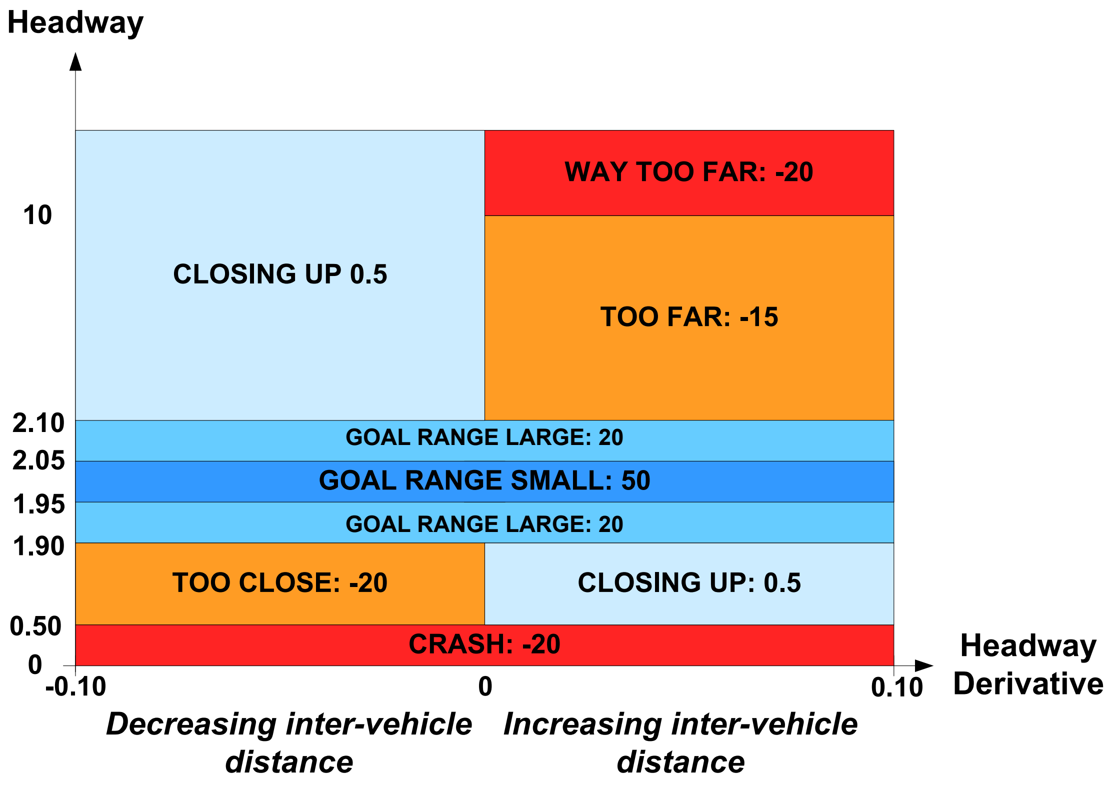


Figure 5.1: Simplified ACC reward function.

actions in order to catch up with the preceeding vehicle. It would become extremely long and, in practice, nearly impossible for the agent to find out how to get to regions of the state space where it can maximize its expected reward.

To address this issue, when the agent is too far from the leading vehicle but selects an action that has the effect of reducing the inter-vehicle distance, it is rewarded with a small positive reward. This has the effect of directing the exploration of the agent, as it rapidly learns to selects acceleration actions that lead to the visit of meaningful states, closer to the goal. This technique is similar to Andrew Ng's reward shaping method [Ng et al., 1999], which speeds up the learning process by giving positive reinforcements to actions that make the agent progress along a desired trajectory.

Algorithm

Finally, the choice of algorithm to use and the appropriate setting of its parameters is also important in order to solve the learning task efficiently. For our simplified problem,

Parameter	**Value**
Exploration Threshold ϵ	0.9
Learning Rate α	0.7
Discount Factor γ	0.9

Table 5.1: Q-Learning parameters for the simplified ACC learning task.

we considered the use of a model-free algorithm. This choice was justified by the fact that we plan to use a model-free approach in future work, as our complex simulator uses dynamics for which the transition function is unknown. Since the state variables we consider are discretized, we can represent the policy using a Q-values table and thus use the Q-Learning algorithm to solve this simplified problem efficiently.

With $30,000$ existing Q-Values pairs ($1,000$ headway values $\times$ 10 headway derivative values $\times$ 3 actions per state) consituting our state space, we considered and observed empirically that setting the number of episodes to $1,000$ and the number of steps per episode also to $1,000$ (thus making a total of $1,000,000$ visits of (s, a) state-action pairs that are comprised in the Q-Values table) is adequate in order to learn a following behavior in this framework.

Finally, we empirically set the Q-Learning parameters to the values given in Table 5.1.

5.1.3 Results

Learning

To learn a control policy in this simplified ACC setting, we first ran $1,000$ episodes of $1,000$ steps of the scenario described in Section 5.1.2 while taking driving decisions using the Q-Learning algorithm. Results of this simulation illustrating the evolution of the number of goals states visited throughout learning phase are given in Figure 5.2.

Execution

Execution of the resulting policy was performed on a Stop & Go scenario. These scenarios are interesting as they represent driving situations that are often seen on urban

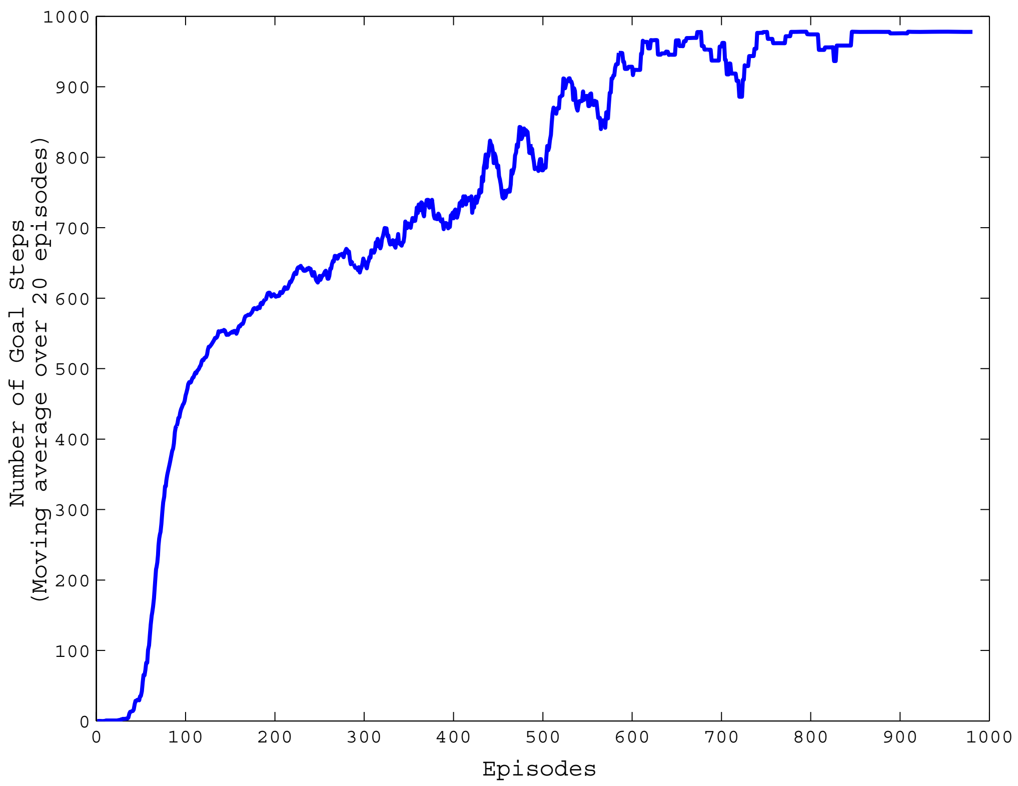


Figure 5.2: Results of 1,000 learning episodes for the simplified ACC task.

roads. They have been used by many researchers for the development of autonomous controllers and/or for the assessment of their efficiency and effects on the traffic flow [Naranjo et al., 2006b; Hallouzi et al., 2004; de Bruin et al., 2004]. In our case, a leading vehicle starting at standstill accelerates to a velocity of 20 m/s, slows down to 5 m/s for 40 seconds (200 timesteps) and then accelerates back to its original cruising velocity.

Then, we placed a vehicle behind the leader and advised it to use the learned ACC control policy to select actions so it should follow its respective predecessor from the defined secure distance. Figure 5.3 shows the velocities of the vehicles during the execution of the scenario, while Figure 5.4 illustrates the corresponding headway of the follower.

5.1.4 Discussion

As we said earlier, this simplified simulation setup was in fact preliminary work towards obtaining a fully-functional CACC system. It let us define our system's architecture,

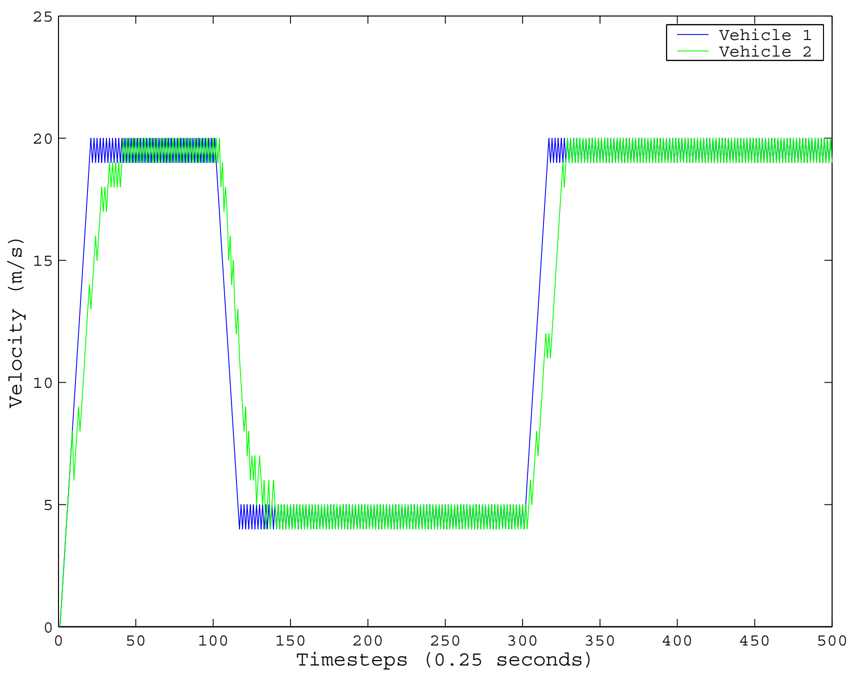


Figure 5.3: Vehicle velocities for a simplified ACC simulation.

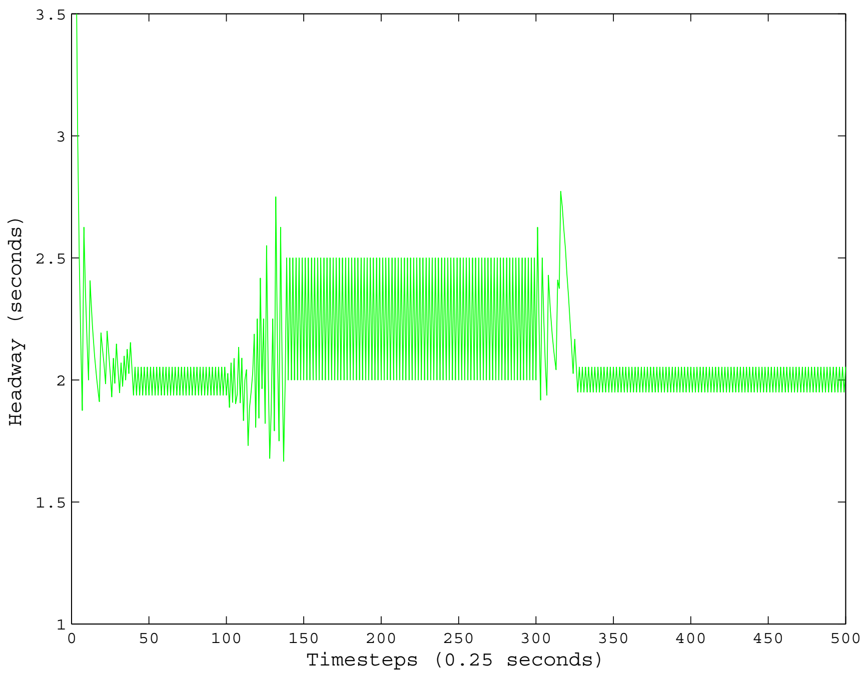


Figure 5.4: Headway of the following vehicle for a simplied ACC simulation.

and it gave us the chance to concentrate on the definition of our reinforcement learning framework.

A quick look at Figure 5.2 shows that the learning simulations were successful in finding a policy that directs a vehicle toward a secure distance from its predecessor. In fact, this figure illustrates that by the end of the learning episodes, the number of goal steps the agent visits is of 978 steps per episode. Thus, the vehicle spends most of its time within the desired inter-vehicle spacing. Moreover, Figure 5.2 also shows that the learning algorithm was able to converge to this solution in approximately 800 episodes, which clearly demonstrates that the learning task can be solved efficiently using our framework.

It is interesting to note that we can consider this behavior as “near-optimal”. With our definition of headway, it is inevitable that the first few steps of an episode, when the vehicle starts at 0 m/s, will generate a headway ratio that is high. Even if it rapidly goes down, it is clearly impossible for the agent to reach its goal in these first few steps. This is obvious when looking at the headway of the follower during policy execution scenarios, as Figure 5.4 shows that the first few simulation steps display a headway ratio of over 4 seconds.

Apart from this extremely high headway at the beginning, we observe, in Figures 5.3 and 5.4, that our policy results in the following vehicle spending most of its time close to the desired value of 2 seconds headway.

However, a few things can be said regarding the headway results. First, we can observe that the learned behavior results in oscillations of the velocity and of the headway (as it depends on the velocity) around the desired values. Unfortunately, this behavior is inevitable in this simplified model since there is no action in our model that can help us achieve a constant velocity. Thus, since the actions have an important effect on velocity, oscillations appear, but actually correspond to the optimal behavior for the given environment.

Another point that we can note from Figure 5.3 is the increase in oscillations around the desired headway when vehicles are at lower velocities (which happens on Figure 5.4 from around timestep 100 to 300). This phenomenon is caused by the sensibility of the headway ratio to low speed: when at low speed, taking an action that increases or diminishes the velocity will always result in larger modifications of the headway ratio than at higher speed.

Finally, we must also address the “spikes” observed in the headway of the follower when the preceeding vehicle has an acceleration that is not null (seen on Figure 5.4 at time 100 and 300). This issue happens because the policy used was learned for a situation where the front vehicle always keeps the same velocity. In policy execution mode, when the front vehicle either accelerates or brakes, the policy still can react fairly well on the basis of the headway derivative, but there clearly is some important information missing in the state definition to keep a good optimization of the safe distance. Of course, learning a control policy in a Stop & Go scenario would have resulted in a slightly different policy, but it surely could not have been better at reacting to the preceeding vehicle's accelerations and braking actions, as critical information regarding the front vehicle's state would still be missing.

Nonetheless, even when taking these shortcomings into account, it is clear that we have achieved to learn an ACC policy that results in a behavior corresponding to what we wanted. The use of a simplified system let us focus on the learning problem and also let us define a framework to solve the task of vehicle following. Most of the work done here serves as a basis to the work presented in the next section, where we try to learn an ACC control policy, but this time in a high-fidelity, complex driving environment featuring continuous state variables.

5.2 Single-Track ACC

In this section, we will go further in trying to solve the problem of autonomous vehicle following, as we address the learning of an ACC policy but this time in a realistic driving environment. As detailed in Chapter 4, this environment features complex vehicle dynamics based on the single-track model.

Again, we consider the learning task using a model-free approach, as we wish to learn an efficient vehicle following policy by direct interaction with this system. However, learning in such a precise environment requires the use of methods better adapted to large state spaces than classic tabular representations as used in the previous experiments. For such a realistic environment, we propose to use policy-gradient algorithms. As mentionned previously (in Section 3.3), these algorithms have gathered significant interest recently for the resolution of control problems in complex environments [Abeel et al., 2006; Field, 2005; El-Fakdi et al., 2005; Peters and Schaal, 2006]. After describing the learning task, the state space, the action space and the reward function, we will detail the algorithm used and its parameters. Finally, we will analyze the results of policy learning and execution.

5.2.1 Learning Task Description

The learning task considered here is similar to the task we executed in our simplified model. However, to address some of the issues of our previous experiments, we modified slightly the scenario in order to learn a control policy directly in a Stop & Go scenario (instead of in an "acceleration-only" scenario as in the preceeding section).

In this specific scenario, a leading vehicle starting at standstill accelerates to a velocity of 20 m/s (72 km/h), slows down to 7 m/s (25.2 km/h) for 40 seconds and then accelerates back to its original 20 m/s cruising velocity. Thus, the learning vehicle now faces modifications of the front vehicle's acceleration. Again, it must learn a policy that lets it keep a secure distance even if it does not receive explicit information regarding these modifications of acceleration. This is analogous to a classic ACC system, where the system relies solely on headway and headway derivative for inter-vehicle gap control.

State and Action Spaces

The state space of the current problem is mostly similar to the definition of the simplified problem of the previous section, except that, since the algorithm we use can handle continuous inputs, we did not need to discretize the state variables. However, we decided to limit the possible range of the values of the state variables. Thus, any value of headway that would exceed 10 seconds was brought back to this 10 seconds value. Similarly, any value of headway derivative exceeding 0.1 or smaller than -0.1 was brought back to these respective values. We have observed that this technique has a positive effect on the behavior of the function approximator used by the algorithm, as all its ressources are used to handle interesting regions of the state space.

The action space for this problem, is identical to the discrete set of actions introduced previously (see Equation 5.4).

Reward Function

The reward function considered here to solve the learning task is identical to the function presented in Section 5.1.2. However, we observed empirically that better results were achieved by slightly modifying the values to those shown in Figure 5.5.

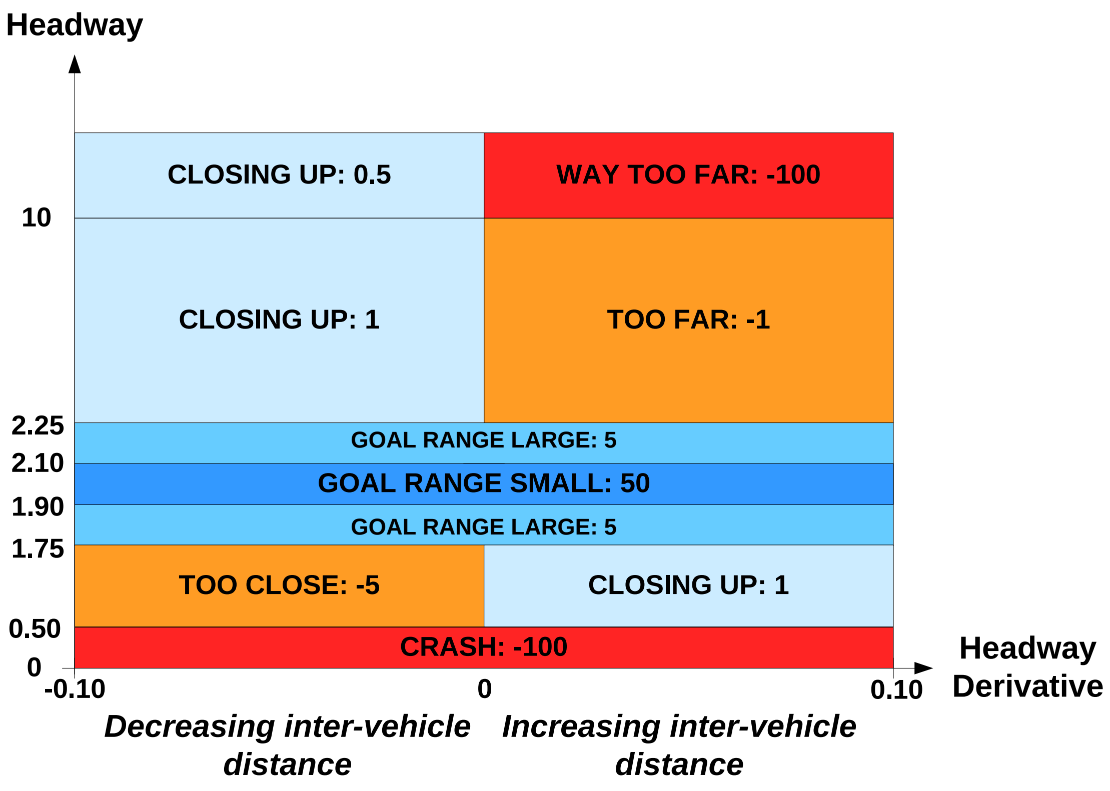


Figure 5.5: Single-track ACC reward function.

Algorithm

One major issue of classic reinforcement learning algorithms as applied in the previous section is that they are often limited when trying to efficiently solve real world problems. The main issue is related to the size of the tabular value function representation, as it grows rapidly with the number of possible values of state variables and actions. In some cases, the state space can grow up to a size where efficient learning, which normally requires infinite visits to (s, a) pairs, becomes impossible. Thus, solving complex problems where the resolution of state variables must be high requires some sort of value function approximation. Unfortunately, function approximation methods applied to classic reinforcement learning algorithms do not have convergence guarantees (simple examples have been shown to diverge), but these methods might still yield good results in practice.

Another interesting approach as a mean to solve control problems that has gained particular interest in recent years has been the use of policy-gradient methods. This class of algorithms is looking to modify the parameters of a stochastic policy in the direction that optimizes the expected reward. These algorithms have the ability to work with continuous state variables since the policy they use is based on a value function approximator. However, these approaches are different, as they do not rely on accurately approximating the value function, but rather rely on modifying it to improve the current policy. Their advantage over classic reinforcement learning methods that use function approximation is that policy-gradient algorithms are guaranteed to converge (albeit only to a local maximum). Of course, such abitilies to handle high resolution, continuous environments are pre-requisites for solving efficiently the problem of autonomous vehicle control considered here.

Another important aspect of policy-gradient methods that justifies our choice of algorithm comes from the fact these are model-free approaches. Hence, the algorithm can learn an efficient behavior by direct interaction with its environment. Of course, this characteristic is also important in the context of our problem, since the driving simulator used to model the environment is quite accurate and features a complex dynamics model for which we do not know the transition function. Since it is quite difficult to solve with precision such a control problem using analytical mathematics, approaches that learn to control a system by direct interaction offer an obvious advantage as they reduce the complexity of the designer's task while still yielding good action policies. Finally, another benefit of using policy-gradient algorithms is that these methods react fairly well to learning under noisy dynamics [Kimura et al., 1997].

For these reasons, we selected the OLPOMDP algorithm (that was described in Section 3.3.1), a policy-gradient algorithm proposed by Baxter et al. [2001], as it is well adapted to solve the problem of finding an ACC control policy in our complex vehicle simulator. This on-line algorithm has the ability to update the weights of the policy using an estimation of the gradient at each iteration. This algorithm is designed to handle partially observable MDP's, but to solve our problem, we made the assumption of having a full observability over the environment.

Parameters that must be set for the algorithm are β, which describes the bias-variance trade-off, α, the learning rate and finally T, the number of iterations of the algorithm. The parameters were set empirically to the values shown in Table 5.2. The number of iterations was at most of $2,500,000$ ($5,000$ episodes of 500 steps), but in practice this number was slightly lower, since some episodes were reset when the agent got too close from its preceeding vehicle. Evaluated over 10 learning simulations, we obtained an average number of iterations of $2,202,718$.

However, these parameters do not entirely specify the behavior of the algorithm, as one of the most important aspect lies in the appropriate definition of the value

Parameter	**Value**
Bias-Variance Trade-Off β	0.9
Learning Rate α	0.00001
Number of Iterations T (average over 10 simulations)	$2,202,718$

Table 5.2: OLPOMDP Parameters for the single-track ACC Learning Task

function approximator. The most important consideration is that the approximator must be differentiable according to its parameters in order to enable the calculation of the gradient of the current policy.

In our case, we have opted for an artificial neural network with 2 inputs for state variables (headway and headway derivative), one hidden layer of 20 neurons and an output layer of 3 neurons (one for the value of each action). The threshold function used for the hidden layer neurons was a sigmoid function, while the output layer nodes used a linear function. This network architecture is adapted to function approximation, as it has been shown that networks having one hidden layer and one output layer using respectively sigmoidal and linear transfer functions have the ability to approximate any given function with a precision that depends on the number of neurons on the hidden layer [Cybenko, 1989]. Since the OLPOMDP algorithm considers the use of stochastic policies, we have added an additional layer that calculates, using a soft-max function, a probability distribution over the actions. With this architecture, the gradient calculation through the backpropagation technique is possible for every weight of the network, as detailed in Chapter 3. Figure 5.6 shows the resulting neural network we used to represent our policy.

5.2.2 Results

Learning

As previously stated, learning control policies using policy-gradient algorithms can be challenging since they can only guarantee convergence toward a local minima [Marbach, 1998]. As a result, we cannot be certain that the policy obtained corresponds to the optimal behavior. Moreover, it can be difficult to evaluate the performances of the learning algorithm since the stochasticity inherent to our action choice makes it possible to obtain totally different policies over the course of consecutive simulations. Thus, two simulations can largely differ, as random action selection gets the agent to explore

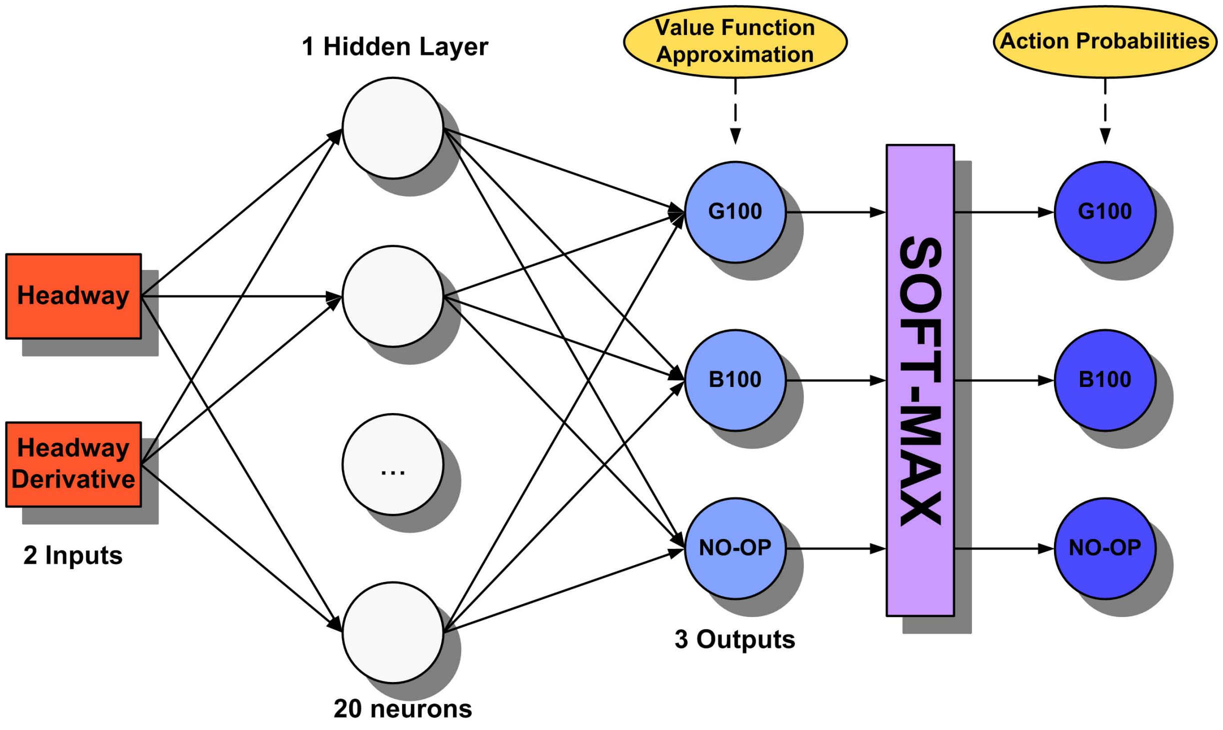


Figure 5.6: Neural network architecture of the single-track ACC controller.

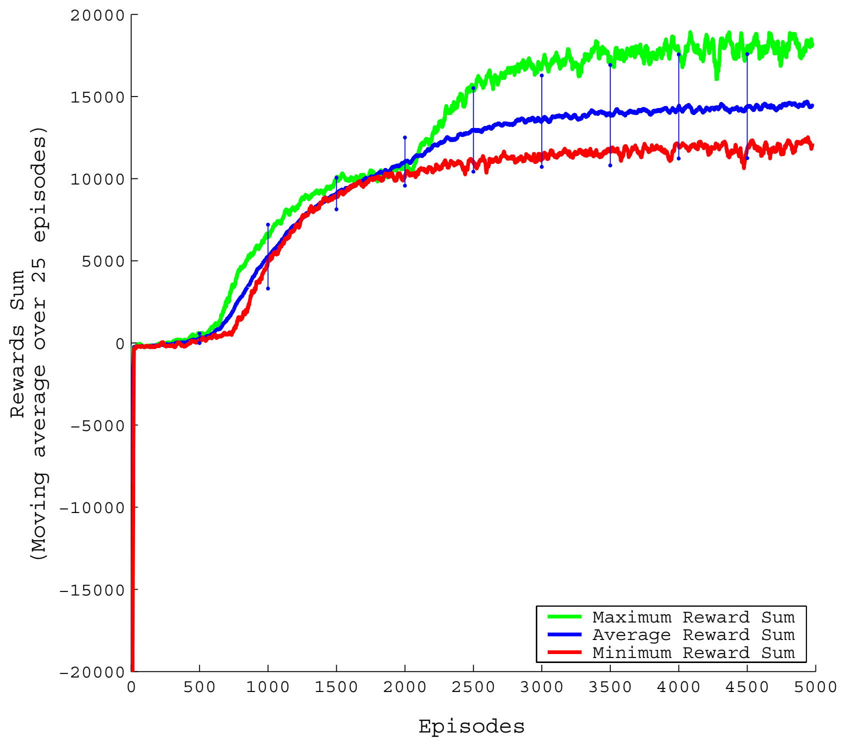


Figure 5.7: Rewards sum for 10 learning simulations of 5, 000 episodes of the single-track ACC task.

different areas of the state space. Some of the resulting policies can be highly successful (possibly reaching a global maximum), while others can get stuck in regions where the expected reward is much lower.

In this context, it is common to present results of multiple learning simulations in order to show how the learning algorithm performs in the environment of interest [Baxter et al., 2001; El-Fakdi et al., 2005; Aberdeen, 2003]. For our problem, we ran 10 learning simulations to compare the resulting policies, each of these simulations corresponding to 5, 000 episodes of a maximum of 500 iterations. As mentionned earlier, since some simulations were reset before reaching the end of the episode (when the learner got too close to its preceeding vehicle), we actually obtained an average number of iterations of 2, 202, 718 over 10 learning simulations. Figure 5.7 presents the maximum, average and minimum reward sum of the 10 simulations, while Figure 5.8 gives the maximum, average and minimum number of goal states reached throughout the learning simulations. Both figures also display the respective standard deviation.

Clearly, these figures also show that different results can be obtained from stochastic optimization algorithms for the same learning task and environment. By getting statis-

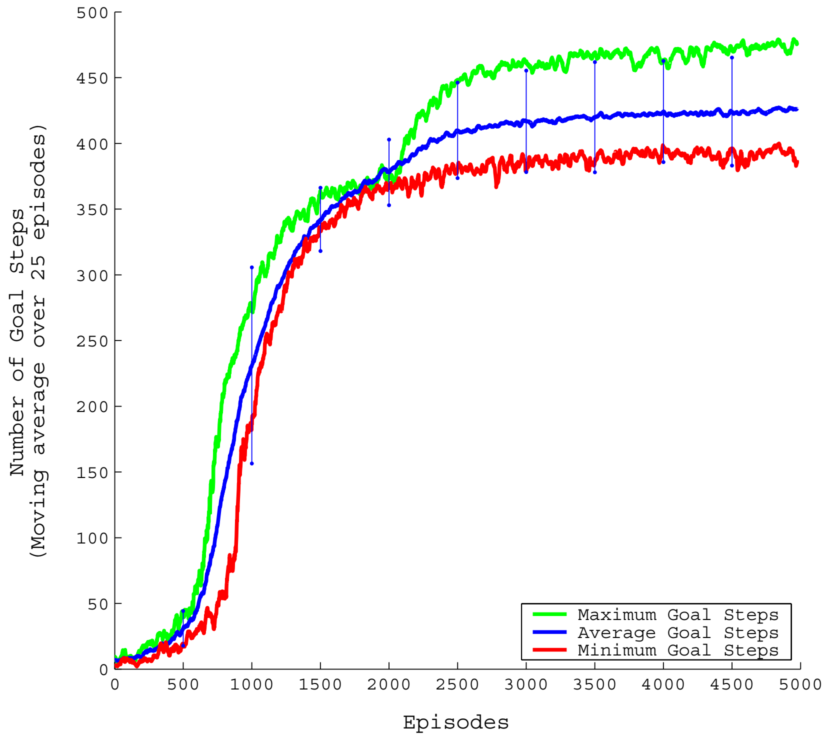


Figure 5.8: Number of goal steps for 10 learning simulations of 5,000 episodes of the single-track ACC task.

tics of the results of learning over a large number of simulations, it could be possible to obtain a certain degree of confidence over the average policies we can obtain and on the possibility of having policies that give expected rewards close to the possible maximum. Unfortunately, as simulations in our complex environment are rather costly (a single learning simulation of $5,000$ episodes takes over 36 hours on a recent laptop with 2.0 Ghz Intel Dual Core processor and 2.0 GB of RAM), we obviously could not make enough simulations to warrant a statistically significant number of simulations. However, since we are looking for the best possible action policy, we are especially interested in the policy that results in the highest expected reward.

Execution

After comparing the results of the 10 learning policies obtained, we selected the best policy (the policy yielding the highest expected reward) in order to study its behavior. A Stop & Go scenario similar to the one used for learning was executed: a following vehicle, using the learned policy, must follow the front vehicle. Again, as the leader accelerates, the follower must keep a secure distance.

To execute the learned policy, the current vehicle's headway and headway derivative were given as inputs to the neural network. The action that was selected corresponded to the one with the highest probability output by the soft-max layer.

Results of this simulation are illustrated here by the velocity profile of the two vehicles (Figure 5.9), and by the follower's headway throughout the simulation (Figure 5.11).

5.2.3 Discussion

The results of the previous section were obtained without taking into account any sort of sensor or actuator delay. However, since the simulator needs a 100 Hz (10 ms) refresh rate for the dynamics to be accurate, we had to decouple the agent's decision making rate from the dynamics engine refresh rate. At this rate, due to the complexity of our dynamics model, taking an action did not have a direct effect on the state at the next timestep, as actions need more time to visibly affect the vehicle. In these conditions, we decided to learn with decisions at 250 ms intervals, rate at which we observed that an action had time to affect the dynamical state of a vehicle. From the assumption of no delay of the sensory system, refreshes of its components were synchronized with

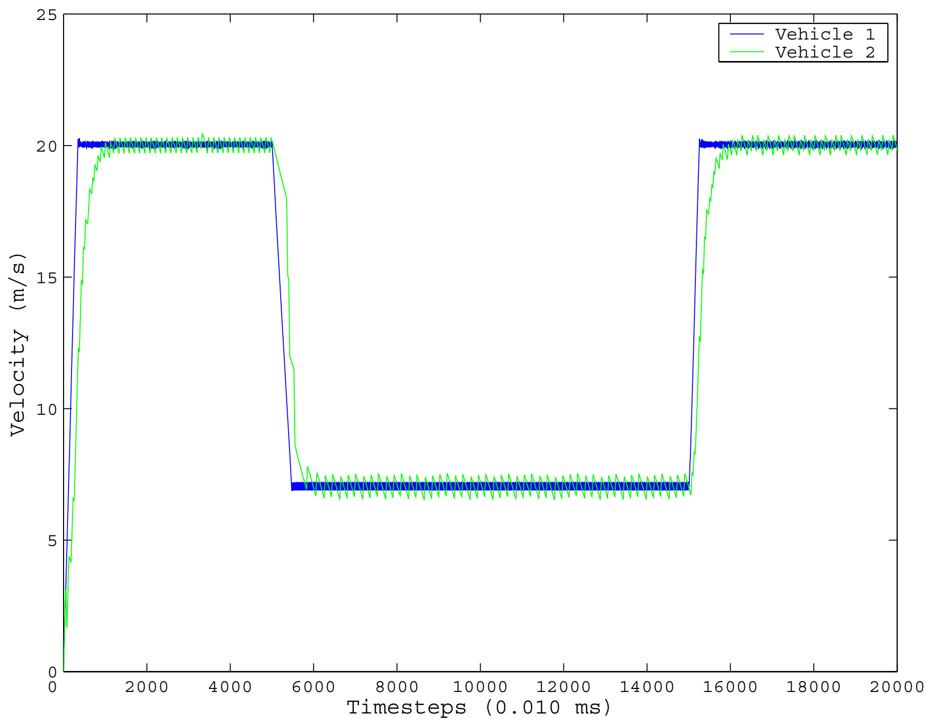


Figure 5.9: Vehicle velocities for a single-track ACC simulation.

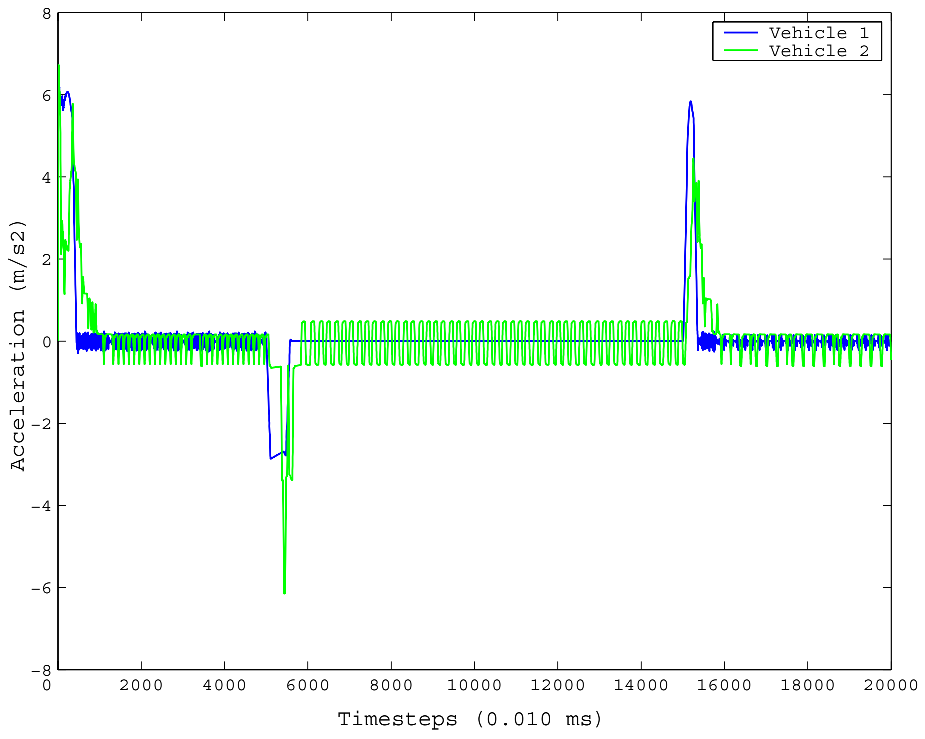


Figure 5.10: Vehicle accelerations for a single-track ACC simulation.

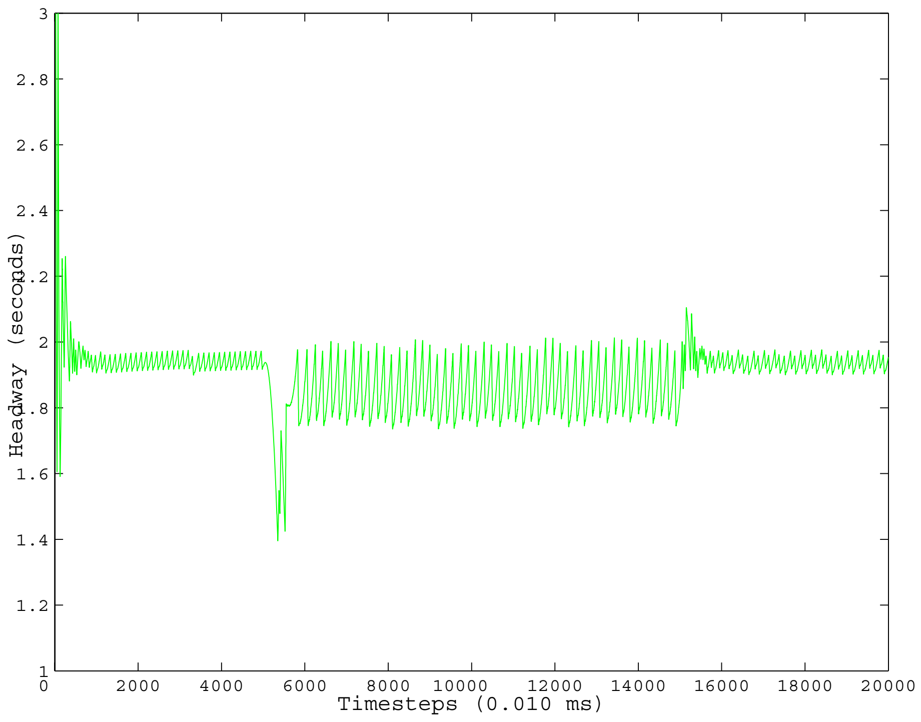


Figure 5.11: Headway of the following vehicle for a single-track ACC simulation.

the decision steps. When an action was selected, it was applied for the whole 250 ms interval. By doing so, it became possible to correlate actions to their effects, enabling efficient learning.

Unfortunately, no matter how we set the delay between decisions, the complexity of our vehicle dynamics model had negative effect on the efficiency of learning. Since the driveline of our vehicle is accurately modeled and takes into account residual acceleration, previous actions applied on the vehicle continue for some time to have an effect on its dynamics. Of course, this added complexity to the learning task, as this dependence on the previous actions means that the Markov property for our currently defined framework is dropped.

Aside from information about the previous actions, other important information is also missing from our state definition in order to efficiently learn a vehicle following behavior. In fact, the learning task as defined here can be considered as non-stationary, since the results of our actions in a particular state can differ, as they not only depend on the short-term action history (as detailed in the previous paragraph), but also on the acceleration of the front vehicle. Clearly, taking a braking action will result in a different next state when the front vehicle is accelerating than when it is going at a constant velocity. This issue is observable on the headway of the follower, as seen in Figure 5.11. From timesteps 5,000 to 5,500, the behavior of the follower shows that the resulting policy has difficulty keeping track with the deceleration of preceding vehicle. In fact, as the front vehicle brakes, the follower gets much closer to the front vehicle than the desired 2 seconds value (Table 5.3 actually gives a minimum headway value of 1.395 seconds). The acceleration profile of the follower (see Figure 5.10) also illustrates this issue, as it shows that since it got too close to the front vehicle, the follower has to brake much harder than the front vehicle in order to rapidly get back to the desired headway value.

This behavior of the learned policy can be ascribed to the fact that, for most of the learning simulation, the front vehicle is at constant velocity or is accelerating. In these situations, when the follower gets closer to the front vehicle, taking "no-op" actions has the effect of slowing down just a little bit, getting the vehicle back in the desired range. Thus, just after the desired 2 seconds region, the learning algorithm correlates "no-op" actions as the most efficient. This can be seen in Figure 5.12, which represents graphically the actions selected by the policy according to both headway and headway derivative state variables (the blue region corresponds to the acceleration actions, the green region corresponds to "no-op" actions while the red region represents the area of the state space where braking actions are selected).

During learning, when the same states are visited as the front vehicle is braking, the result of a "no-op" action is different. In that case, "no-ops" are not drastic enough in slowing down the follower, and a braking action is what should be learned instead. But, as we can see, when the front vehicle brakes, the follower gets too close, meaning that this behavior has not been learned. Our hypothesis is that, since decelerations do not happen often during a simulation, the learning algorithm ignores them and treats as "noise" the "no-op" actions that give poor results when the front vehicle is braking, thus concentrating on using the correct actions for most of the time. Moreover, the fact that there is no state variable to model these different accelerations of the leader also play an important role in this poor behavior.

On the bright side, we see from Figure 5.11 that the learned policy reacts fairly well to accelerations, as the follower directly stabilizes itself at the safe distance both at the beginning of the simulation and at timesteps 15,000 to 15,500. This comes from the fact that the policy's threshold between "no-op" and gas actions is located close to the desired 2 seconds region (as seen in Figure 5.12). Thus, taking acceleration actions up to the 2 seconds gap gets the follower closer to the front vehicle. When past the desired region, the next action the agent will selects will be a "no-op" action, which will have the effect of getting the follower back towards the desired region (since the front

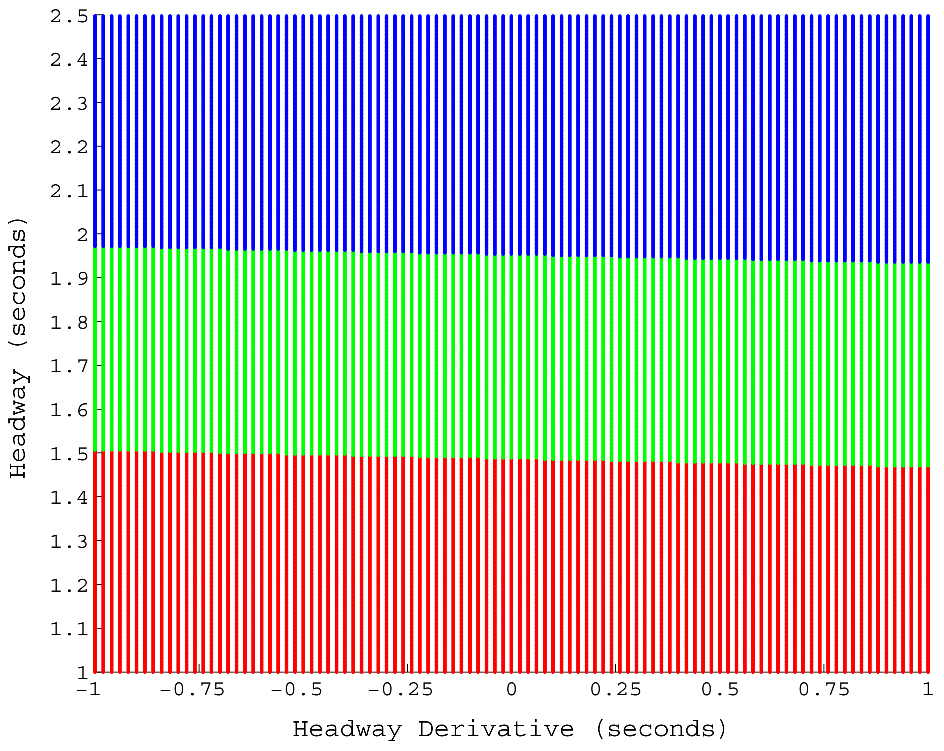


Figure 5.12: Learned single-track ACC policy.

vehicle his accelerating). Thus, oscillations between gas and "no-op" actions actually corresponds to a "near-optimal" behavior when the front vehicle either accelerates or is at a constant velocity. This also explains oscillations of the acceleration profile of the follower (in Figure 5.10) when the front vehicle is at constant velocity.

In practice, learning is still quite efficient as most of the simulation is spent around the desired headway value of 2 seconds. Of course, some oscillations are observed around the desired value, but this is normal as we act using a small number of discrete actions. This is corroborated by Table 5.3 (that gives a summary of the raw data associated to Figure 5.11) which shows that the average headway (measured when the following vehicle was over 5 m/s, which stabilizes the headway ratio) of 1.89 seconds is quite close to the desired value. Also, we again observe large oscillations at slower speeds, but that is again related to the sensibility of our headway ratio to low speeds as detailed in the previous section.

To settle the problem of non-stationarity of the learning task, we introduce in the next section an updated state definition which integrates an acceleration signal of the front vehicle received from inter-vehicle communication, thus transforming our ACC controller into a full-fledged Cooperative Adaptive Cruise Control system.

	Headway (in secs.)
Minimum	1.395
Average	1.892
Maximum	2.260

Table 5.3: Headway values of the following vehicle for a single-track ACC simulation.

5.3 Single-Track CACC

The experiments and results of this section show our efforts toward integrating the acceleration signal of a front vehicle into our autonomous controller. The addition of this signal transforms our ACC system in a CACC system by making the assumption of the availability of a communication channel between vehicles. With this signal, we want to address the issues related to the relatively poor efficiency of our controller when the front vehicle is braking.

5.3.1 Learning Task Description

The learning task considered here is nearly identical to the task of the previous section, as exactly the same scenario is considered. Small modifications to our framework were still necessary and will be detailed in the next paragraphs.

State and Action Spaces

For this problem, we had to modify the state space, in order to integrate the acceleration information into our state definition. We added a state variable, which was bounded, like the other variables, from $-2\ m/s^2$ to $2\ m/s^2$. Moreover, since the accelerations varied widly between consecutive decision steps, we have used a low-pass filter on this variable to make an average of the acceleration over the past second. As an effect, this smoothes the oscillations of this value, resulting in a much more representative reading of acceleration. For its part, the action space was left exactly as in the previous section.

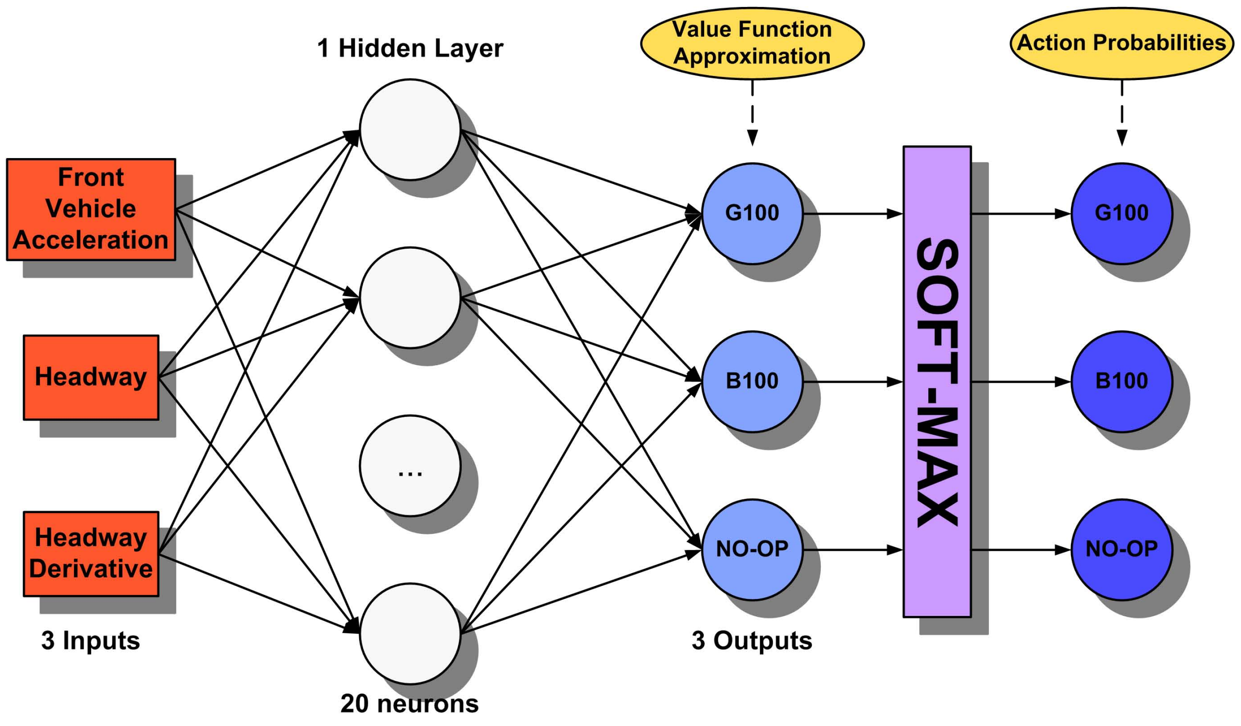


Figure 5.13: Neural network architecture of the single-track CACC controller.

Reward Function

The reward function considered here does not change from the previous definition of Figure 5.5.

Algorithm

Again, we have used the same OLPOMDP algorithm, with exactly the same parameters as in the previous experiments. However, we modified the neural network architecture, to accomodate the presence our new state variable as previously introduced. To this end, we added an input to our network, as shown in Figure 5.13.

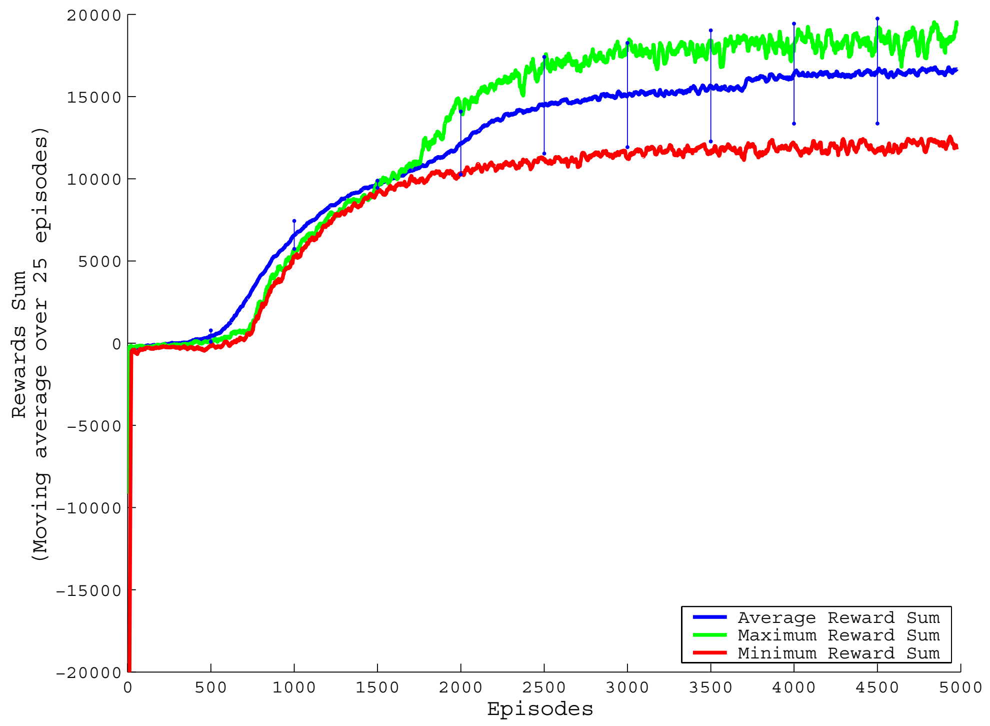


Figure 5.14: Rewards sum for 10 learning simulations of 5, 000 episodes of the single-track CACC task.

5.3.2 Results

Learning

Using the same methodology for the learning process as for the development of the ACC controller, we obtained 10 different control policies, and results of the learning process are given in Figures 5.14 and 5.15.

Execution

Again, similarly to the ACC experiments, we tested our learned policy in the same scenario to what was used for learning. Figure 5.16 shows the velocities of both vehicles during the simulation, Figure 5.17 shows their accelerations, while Figure 5.18 illustrates the headway of the follower.

Afterwards, to observe the performances of our learned policy in real-life conditions,

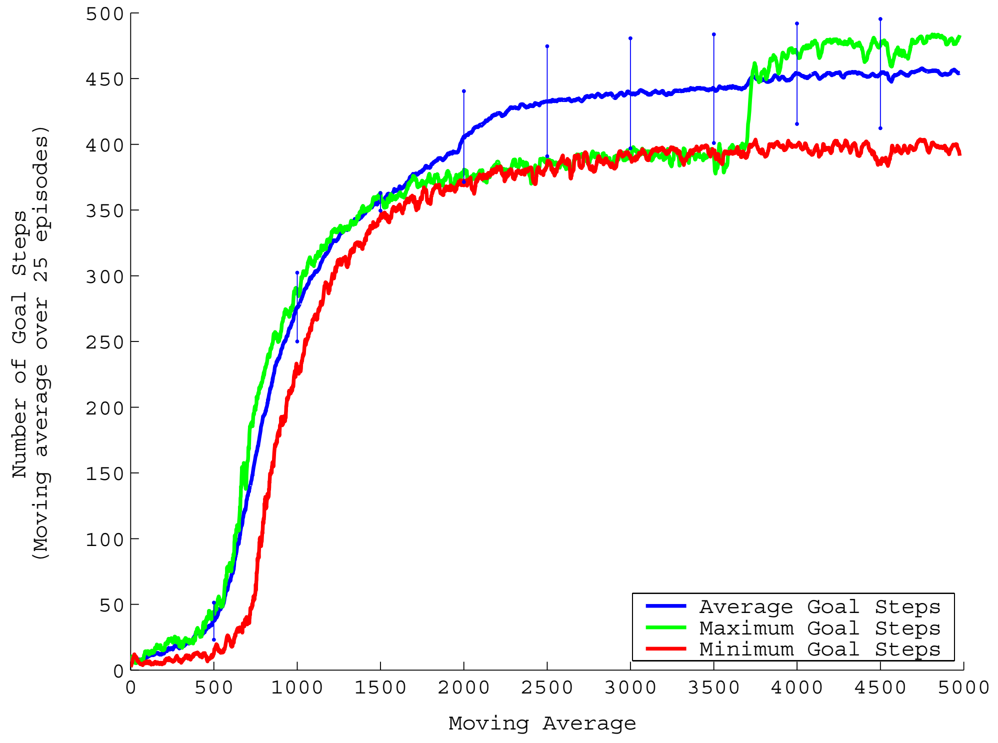


Figure 5.15: Number of goal steps for 10 learning simulations of 5,000 episodes of the single-track CACC task.

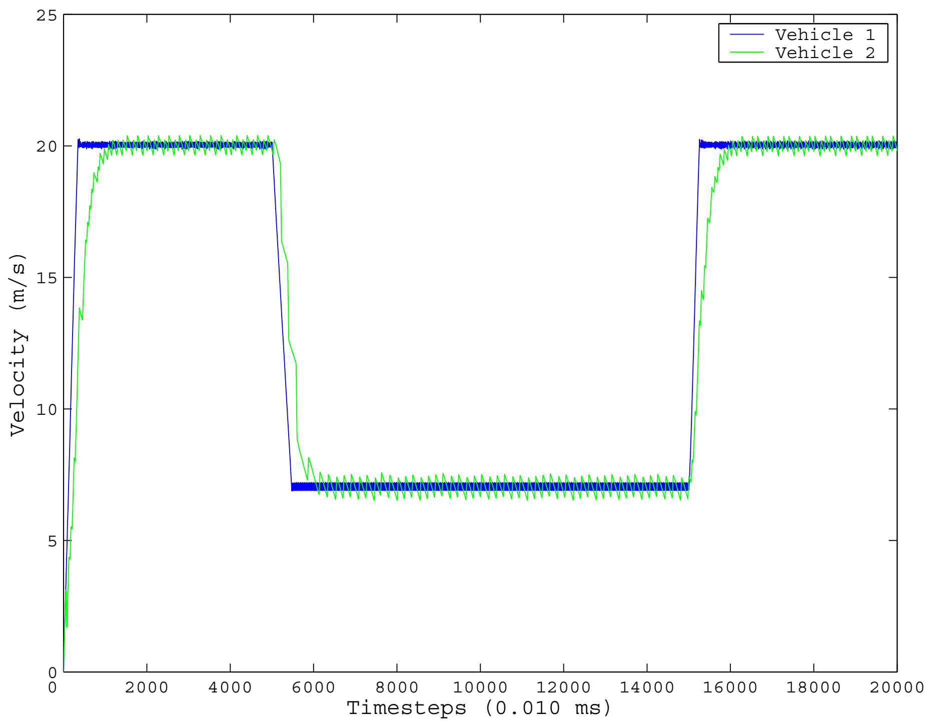


Figure 5.16: Vehicle velocities for a single-track CACC simulation.

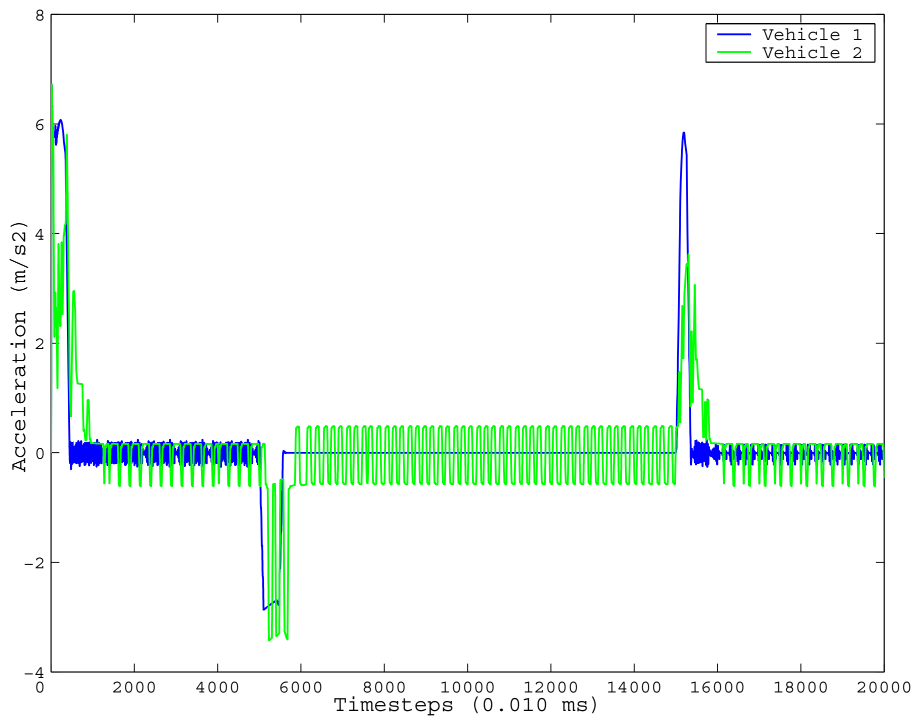


Figure 5.17: Vehicle accelerations for a single-track CACC simulation.

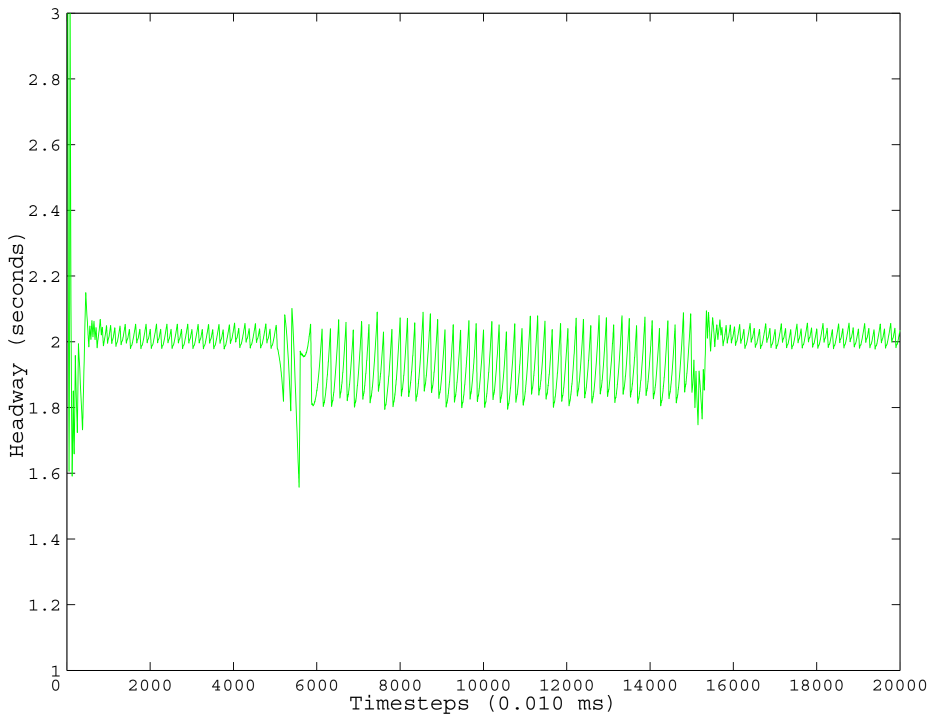


Figure 5.18: Headway of the following vehicle for a single-track CACC simulation.

we put the vehicle in a similar scenario, but this time, we increased the decision frequency to a rate of 20 Hz (a decision is taken every 50 ms). At the same time, to accurately represent modern sensors, we have used a front laser refresh rate of 10 Hz (refresh at every 100 ms), which is a typical frequency value for laser sensors, as seen in many papers from Jurgen [2006]. Communication messages were also given a 10 Hz refresh rate, which corresponds to the estimated requirements of tolerable latency for safety vehicle-to-vehicle communication applications [Bishop, 2005]. Results for this simulation are given in Figures 5.19, 5.20 and 5.21.

5.3.3 Discussion

The results of our experiments relative to the CACC controller were conform to our intuitions. First, Table 5.4 (giving raw data of Figure 5.18) shows that the average headway value is much closer to the desired 2 seconds value, as it is even located in the goal region (2 seconds ± 0.1) that gives the most reward to the agent. Also, maximum and minimum headway values are much closer to the desired value than what we observed for the ACC controller.

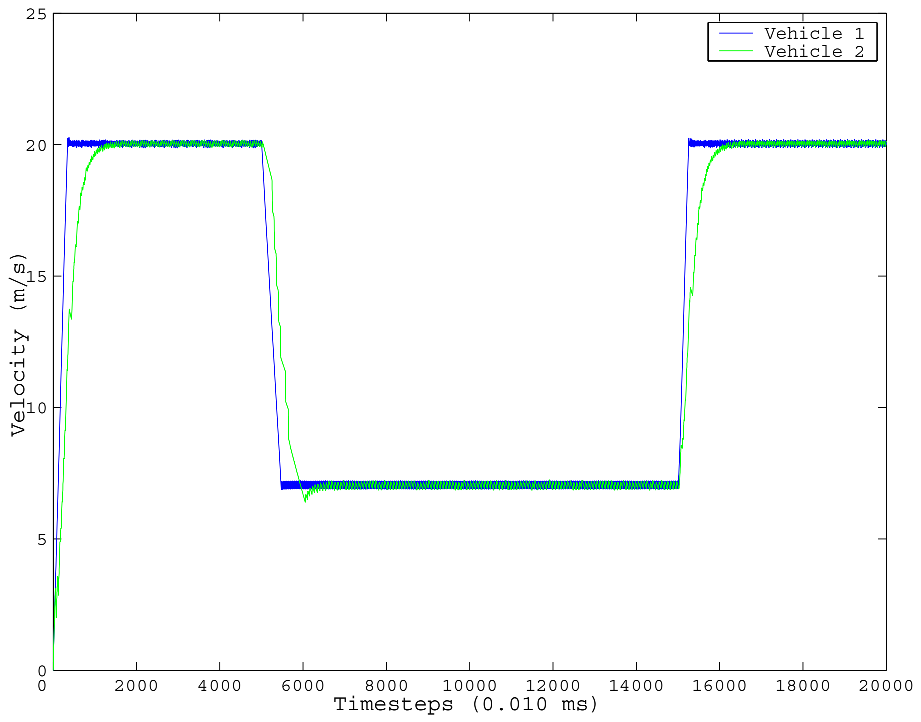


Figure 5.19: Vehicle velocities for a single-track CACC simulation with realistic delays.

	Headway (in secs.)
Minimum	1.558
Average	1.962
Maximum	2.150

Table 5.4: Headway values of the following vehicle for a single-track CACC simulation.

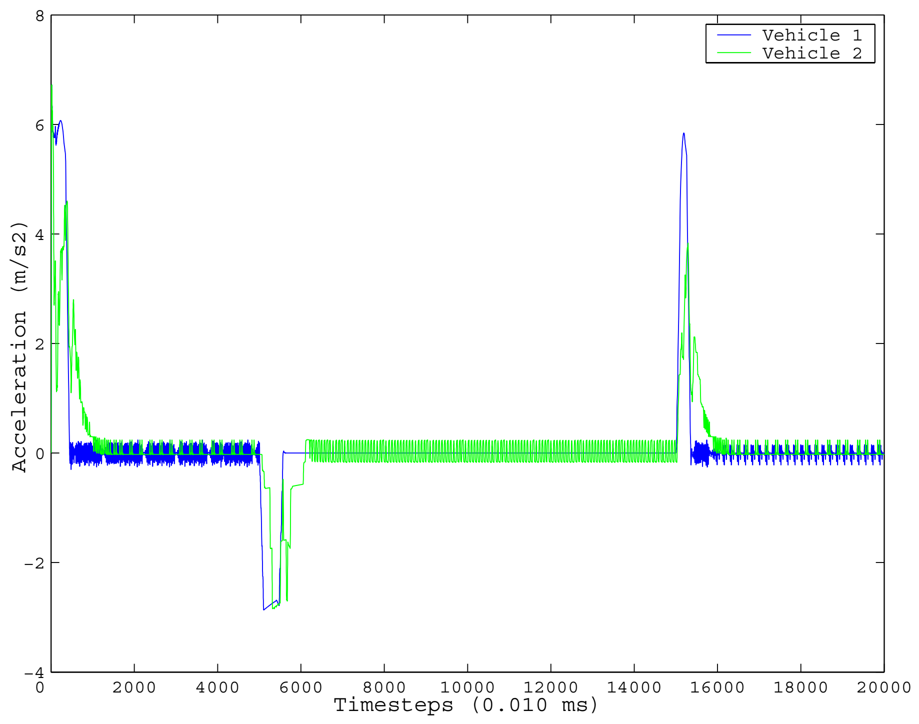


Figure 5.20: Vehicle accelerations for a single-track CACC simulation with realistic delays.

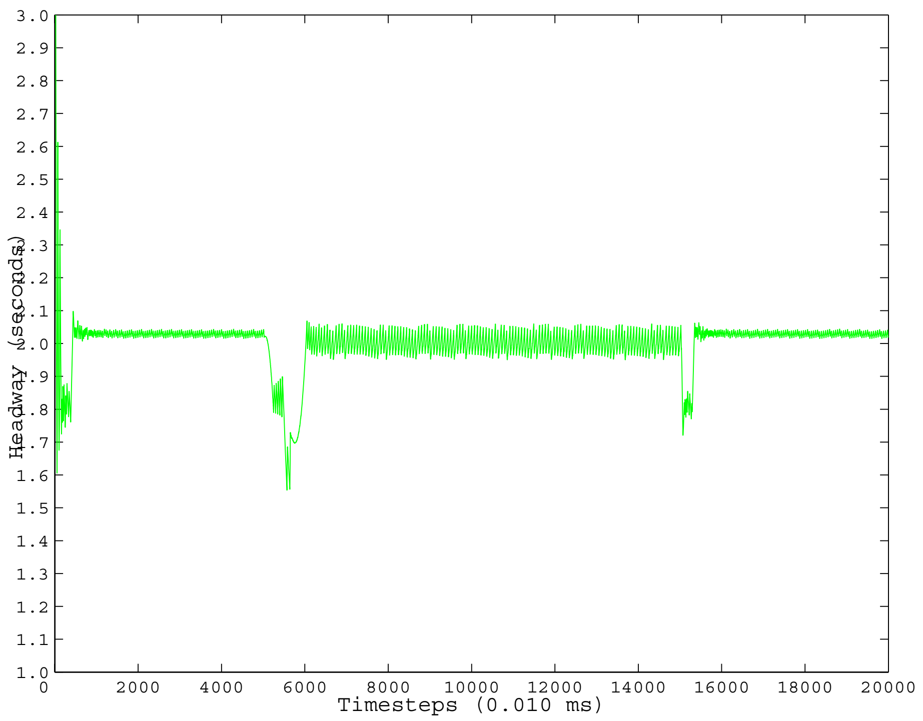


Figure 5.21: Headway of the following vehicle for a single-track CACC simulation with realistic delays.

The headway figure of the follower (in Figure 5.18) shows, from timestep $5,000$ to $5,500$, that when the front vehicle is braking, the follower using the CACC policy performs better at keeping distances than the ACC controller. During this time interval, the headway of the follower oscillates close to the desired value of 2 seconds (approximately 1.85 seconds), which is better than the approximate headway of 1.50 seconds achieved by the ACC controller.

However, we can observe, from timesteps $5,500$ to $5,600$, that the CACC controller steers the vehicle away from the desired headway as it gets closer to its predecessor. This behavior can be attributed to the fact that, at this timestep, the front vehicle has stopped accelerating. Thus, to select actions, the follower's controller observes a constant velocity (acceleration of 0 m/s^2) of the leader and selects actions accordingly. In reality, at this time, the follower is still going a little faster than the front vehicle (as seen in the velocity profile of Figure 5.16). As a result, the follower has the tendency to get closer to the front vehicle, since it uses "no-op" actions until it finally matches the velocity of the front vehicle.

In fact, the CACC behavior is interesting when looking at the acceleration of the follower (in Figure 5.17). This figure shows that, contrary to the ACC controller, the CACC controller does not need to use much harder braking than the leader (around -3 m/s^2 for the CACC compared to -6 m/s^2 for the ACC). This behavior is interesting in the sense that is shows that there is no amplification of accelerations or decelerations, which would result, with many vehicles, in a complete halt of the traffic flow further down the stream. Thus, from this point of view, the presence of the acceleration signal of the leader does enable the learning of a better control policy.

The graphical representation of the resulting policy, shown in Figure 5.22, confirms that the acceleration signal was succesfully integrated in our controller. This figure shows that, depending on the acceleration, there are different thresholds for switching between actions. Of course, this means that a correlation between the different state variables has been succesfully learned. For example, when the front vehicle is braking, changes from acceleration, to "no-op" and then to braking actions happen earlier (when the follower is farther from the goal region) than when the front vehicle is either at constant velocity or accelerating.

Even better results were achieved when executing the learned policy with sensor delays corresponding to actual real life values. For this simulation, we made decisions and used sensor refresh rates at higher frequencies than the 4 Hz (250 ms) of the previous simulations. Our intuition was that, with a refresh rate of 10 Hz (100 ms), we would observe a more precise control of our vehicle. This was indeed confirmed

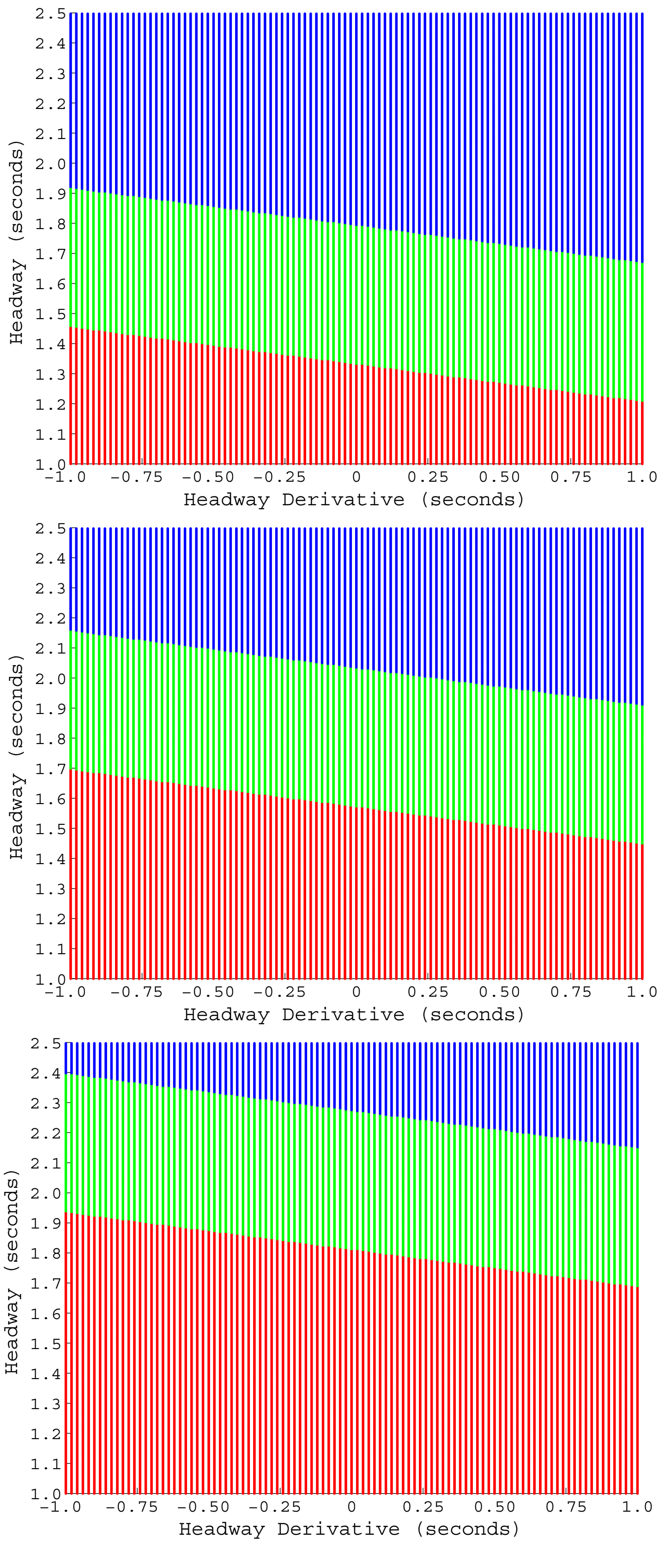


Figure 5.22: Learned single-track CACC policy. Top figure shows acceleration, middle shows constant velocity and the bottom shows braking of the front vehicle.

	Headway (in secs.)
Minimum	1.553
Average	2.000
Maximum	2.100

Table 5.5: Headway values of the following vehicle for a single-track CACC simulation with realistic delays.

	ACC	**CACC**	**CACC with realistic delays**
Minimum Hw	1.395	1.556	1.553
Average Hw	1.892	1.962	2.000
Maximum Hw	2.260	2.150	2.100
Average Hw Error	0.110	0.061	0.039
Hw RMS Error	0.135	0.086	0.066

Table 5.6: Headway and headway error values for the single-track ACC, CACC and CACC with realistic delays simulations.

by results, as shown in Figures 5.19, 5.20 and 5.21. These figures clearly show that the amplitude of the oscillations of headway, velocity and acceleration are smaller than those of the previous experiments. Moreover, Table 5.5 also confirms that the headway values are closer to the desired region than what we achieved with the other controllers (see Tables 5.3 and 5.4).

Unfortunately, the behavior we observed previously when the leader has finished braking, resulting in the follower getting too close because its velocity is still superior to that of the front vehicle, is still an issue. However, the learned policy is a little better at keeping distances than the ACC controller as illustrated by the minimum headway observed.

Table 5.6 compares our controllers, by summarizing their performances using the corresponding headway (Hw) and headway error of their execution on the same Stop & Go scenario. This table illustrates the Root-Mean-Square (RMS) error, which is particularly interesting as it gives more weight to values far from the desired region. Clearly, the CACC controller is generally closer than the ACC controller to the desired goal region, while the CACC executed with realistic decision and sensor delays offers even better performances at staying in close distance from the desired safe distance.

Finally, it could be interesting to measure the performances of our controller against other similar autonomous controllers. Unfortunately, this is difficult to achieve, as our work is entirely done through simulation, and because we have made several assumptions to simplify the problem (no sensor noise, simplified communication protocol not taking many factors into account such as packet loss, etc.) which would have to be lifted in order to compare our controller against other controllers. Moreover, while being precise, our single-track vehicle dynamics model is still an approximation of real vehicular dynamics.

One should notice that most of the existing work on the design of CACC systems have not done much comparisons against other systems, and have only illustrated the performances of their controller by observing its behavior over velocity, acceleration, headway and headway error metrics obtained through execution of their system. For example, prototypes of Spain's Autopia project [Naranjo et al., 2006b] have not been analyzed thoroughly, as they have only showed that their system resulted in an average headway error of 0.13 seconds, a value for which they assume that their system is safe. Hallouzi et al. [2004] also built and tested a CACC system prototype. Evaluation of its performances have focused on analysis of velocity and acceleration profiles of their vehicle following system compared to their front vehicle's behavior.

Notice that some researchers have not focused on the design of controllers, but rather on assessing their effect at a larger scale, on the traffic flow. For instance, Shladover et al. [2001] and Visser [2005] have studied, through simulation, the effect of having many vehicles equipped with autonomous control systems. In order to perform these simulations, they had to model their autonomous controller using mathematical equations. From our side, we did not concentrate on the evaluation of our controller's effects on the traffic flow since the goal of our research was limited to the design a functional CACC controller using reinforcement learning. However, it is clear that studying how our controller fares in the presence of multiple vehicles also equipped with this system would be a great extension to our work.

5.4 Summary

In this chapter, we have shown that it is possible to use reinforcement learning techniques in order to design an efficient autonomous CACC control system. We designed our controller with an incremental approach by:

(a) designing a learning framework for an ACC system in a simplified environment;

(b) porting this learning architecture in a complex vehicle dynamics simulator;

(c) integrating an inter-vehicle communication signal giving additional information about the acceleration of the front vehicle.

Then, by observing the results of the execution of the learned policies, we have been able to show that our CACC controller performed better than our ACC controller at keeping the desired inter-vehicle distance for the duration of a simulation. Of course, some ameliorations could be made to our controller, and the next chapter will conclude this thesis by briefly detailing some of the future work that could be done in order to improve its performances.

Chapter 6

Conclusion

As we have seen, the growing number of vehicles on the road has resulted in a degradation of the quality of life for citizens of urban regions around the planet. Indeed, our dependence on vehicular transportation has led to urban sprawl, accidents, traffic jams and atmospheric pollution, all of which, in turn, have raised numerous considerations such as the economical aspects linked to the poor efficiency of the traffic flow, the safety of passengers and the health problems linked to poor environmental conditions.

The purpose of this thesis was to address these issues using Intelligent Vehicle (IV) technologies. These technologies have recently gathered much interest from the Intelligent Transportation Systems (ITS) research community, as they are seen as an interesting and efficient solution to the problems of automotive transportation. In this thesis, we were particularly interested in designing a Cooperative Adaptive Cruise Control (CACC) system for autonomous vehicle following. However, our work differs from existing solutions as we have proposed a different design approach based on the use of modern machine learning techniques.

Towards reaching this goal, we have first studied work on IV technologies. In particular, we have focused on studying how machine learning can be used to design such technologies. This led us to focus on reinforcement learning as this technique is especially adapted to the design of controllers that can learn to act autonomously by direct interaction with a complex environment (without knowing its model). We have also studied value function approximation and policy-gradient algorithms as these methods are often required in order to efficiently learn in complex, continuous environments.

Through our experiments, we have shown that our approach results in efficient behavior, as vehicles equipped with our system were able to follow a front vehicle at a secure distance for most of a simulation. Moreover, our CACC controller did perform better than our ACC controller, which shows that the integration of an acceleration signal from the front vehicle did result in a more precise gap control policy.

This chapter now concludes by illustrating the contributions brought by this thesis, and by proposing future work that could further improve our vehicle control policies.

6.1 Contributions

This section summarizes the contributions that this thesis has made relatively to the application of machine learning techniques to the field of Intelligent Vehicles.

- A survey detailing previous work in the fields of Intelligent Vehicles and Autonomous Vehicle Control has been made. This survey has been enhanced by a description on how machine learning constitutes an efficient alternative for the design of such systems and how it has already been used in vehicular control and coordination research.
- A survey on the field of reinforcement learning has been made. It has detailed the framework of Markov Decision Processes and has overviewed classic resolution algorithms such as Q-Learning, which can be used to obtain an optimal control policy in unknown environments with discrete state spaces.
- A study of function approximation methods, that have the ability to handle continuous state variables in complex environments, has been made. This study also focused on policy-gradient algorithms, which are model-free approaches based on a parametrized representation of the policy. These algorithms directly modify the policy (instead of using a value-function) in order to increase its expected return. This study was important as function approximation and policy-gradient were later used to improve our approach.
- A vehicle simulator with the ability to integrate reinforcement learning in the simulation loop was developed. It also implemented high-fidelity vehicle dynamics, which were necessary for the design of a complex and realistic learning environment.
- An architecture for a Cooperative Adaptive Cruise Control system relying on two specific control layers has been proposed. The architecture combines both a *Coordination Layer* and an *Action Layer*, that respectively focus on high-level action decisions and on low-level vehicle control.

- An efficient state space representation for the Adaptive and Cooperative Adaptive Cruise Control problems, based on inter-vehicle time distance and for which the learned behavior is independent from the current velocity of vehicles has been proposed. This representation leads to a policy that generalizes the vehicle following behavior learned at a specific velocity profile over all possible velocities of a leading vehicle.
- The efficient learning of a vehicle control behavior using the OLPOMDP policy-gradient algorithm and a neural-network policy representation has been shown.
- Results showing that the learned control policies are efficient at keeping the desired inter-vehicle distance for most of a simulation were obtained. Moreover, we demonstrated that the use of inter-vehicle communication in order to share extended information does help a follower achieve better gap control during accelerations and decelerations of a front vehicle.

6.2 Future Work

Of course, a lot of work could still be done in order to improve the CACC vehicle controller proposed in this thesis. We have separated the potential improvements in three categories, which will be detailed in the next subsections.

6.2.1 Learning Process

First, it is clear that some modifications to the learning process should be made to improve the resulting vehicle following behavior. For instance, improvements could be made to solve the problem of poor behavior of our vehicle following policy that is observed when the front vehicle has stopped braking. As described in Chapter 5, this is due to the fact that, when the front vehicle has stopped accelerating (or decelerating), the learned policy selects "no-op" actions instead of braking actions to stabilize its position around the desired time distance. This issue could potentially be addressed by modifying the topology of the neural network in order to integrate a relative velocity variable as an additional input. Taking this new variable into account when learning the policy could potentially increase the efficiency of the learning process, by resulting in a behavior that reacts adequately to the situation we just described.

One should also consider using recurrent neural networks [Mandic and Chambers, 2001] to address the issue described in the previous paragraph (which is in fact due to the presence of residual acceleration in the vehicle's dynamics model, resulting in actions having an effect on more than the next timestep). Recurrent neural networks integrate a notion of time prediction, by connecting the previous output values to the inputs of the network, thus preserving history of its previous state. With these networks, the selection of future actions would take into account previous actions and learn to estimate their effect over time.

Moreover, issues related to the oscillatory behavior of our vehicle control policy could be addressed by using continuous actions. However, this would require further study to efficiently implement this approach, as it brings additional complexity to the learning process.

It could also be interesting to test our system, but this time using a Radial Basis Function (RBF) neural network [Haykin, 1998] for function approximation. These networks use gaussian neurons that have the advantage of enabling local approximation instead of global approximation (the latter resulting from the use of sigmoid transfer functions). Local approximation offers some advantages, since learning in a specific region of the state space does not affect what has been learned previously in other regions. In this context, Field [2005] has proposed an approach that uses policy-gradient algorithms with gaussian units to represent the policy, and has obtained promising results for low-dimensional control tasks.

However, no matter what kind of transfer function (sigmoidal or RBF) is used, the problem of adequate selection of the policy representation still remains a challenging issue. Indeed, this problem is quite hard as we need to find a parametrized representation which describes a family of possible policies that should ideally include a policy that can be learned and that is relatively close to the desired behavior. On the other hand, we also have to take into account the fact that the parametrized policy we are seeking must be kept as simple as possible so that we obtain a generalized behavior that does not overfit the training data, in such a way as to handle new situations correctly. Concretely, even if neural networks with sigmoidal and RBF neurons have the universal approximation property, we still have to find the best network topology that lets us generalize new experiences adequately and that can yield efficient policies, which is, in itself, a difficult problem as it requires a lot of tuning for the designer.

An interesting alternative to tackle this problem would be to consider using approaches with the ability to learn simultaneously a policy and its representation according to the experience encountered. In this respect, Girgin and Preux [2008] have proposed an interesting concept where a cascade-correlation neural network is used in conjunction with a policy-gradient algorithm. With this architecture, the neural network is able to organize itself by adding hidden neurons to find a representation that results in better performance from the policies that can be obtained. Obviously, an important advantage of this approach is that a good policy representation is obtained without any intervention from the user.

Of course, the generalization of our learned policy to different values of desired headway would be a great extension to our concept, as current ACC systems let users select their desired inter-vehicle distance (this time distance was set at 2 seconds for all of our experiments). This could be done by integrating the desired headway as an input of our neural network. However, learning would have to be done using different desired velocities in order to obtain a policy that knows how to act according the value of this variable.

Finally, our controller could also be completed by an autonomous lateral control system. Again, this could be done using reinforcement learning, and a potential solution would be to use a reward function under the form of a potential function over the width of a lane, similar to current force feedback given by existing lane-keeping assistance system. This reward function would surely direct the driving agent towards learning an adequate lane change policy. Moreover, the integration of an intelligent vehicle coordination system for collaborative decision making, such as proposed by Laumônier and Chaib-draa [2006], could transform our system into a complete driving assistance system.

6.2.2 Simulation

Some elements regarding our simulation framework could also be improved, with the ultimate goal to have an even more realistic environment through which we could make our learning experiments. In fact, the first aspect to consider would be to integrate an even more accurate simulator for sensory and communication systems. In this way, we could lift some of our current assumptions, such as the absence of sensor and communication noise. Of course, this would complexify the learning process, but the resulting environment would be much closer to real-life conditions.

With regards to this issue, the use of Kalman Filtering [Kalman, 1960] would be practical as it is an interesting state estimation method that is adapted to handling sensor noise. This approach, which joins the readings from multiple sensors, could be beneficial for the performances of our system as it aims at creating the most accurate representation of the current state of the environment. However, as detailed by Bishop

[2005], a lot of work still needs to be done to increase the awareness of the environment, which explains the fact that sensor fusion is an extremely active domain of today's intelligent transportation system research.

6.2.3 Controller Evaluation

Finally, our current work can potentially be extended in order to evaluate the performances of the vehicle controllers that we obtained through learning. In particular, future work should focus on comparing our approach to controllers designed using control theory. As explained by Rajamani [2005], many researchers from the field of mechanical engineering have proposed ACC and CACC controllers using this approach, since it offers a solid mathematical background for the design of these systems and for the analysis of the resulting controllers. For instance, the string stability property is a particularly important characteristic that studies the behavior of the controller when placed in the presence of a large number of similar autonomous controllers. Ideally, we would like our controller not to degrade the traffic conditions when used by a large number of vehicles. With control theory, these properties can be proven analytically as the controllers are modeled using a system of equations.

Unfortunately, it is difficult to evaluate such properties for controllers obtained using reinforcement learning. We propose that future work should implement a control theoretic vehicle controller and compare its execution against our reinforcement learning controller. This would show us where our controller stands against a formal approach and possibly identify some advantages or shortcomings of our system.

Even though a control theoretic approach can lead to the assessment of properties such as string stability, we must note that it still makes approximations in the modeling of the system in order to express it using mathematical equations. Thus, it does not handle well the uncertainty of the system.

Ultimately, it would also be interesting to embed and test the performances of our controller in a real vehicle and compare its execution against previous approaches that were embedded in vehicles [Naranjo et al., 2006b; de Bruin et al., 2004; Hallouzi et al., 2004].

Finally, much work still needs to be carried before our system can be used as a fully-autonomous vehicle controller. As a result, it is much more realistic to think that such a system could eventually be embedded in vehicles as an advanced driving assistance system (ADAS), which would still require human interaction but could be used as a basis for the design of future Autonomous Vehicle Control systems.

Bibliography

Abeel, P., Coates, A., Quigley, M., and Ng, A. Y. (2007). An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems 19 (NIPS 19)*, Cambridge, MA. MIT Press.

Abeel, P., Quigley, M., and Ng, A. Y. (2006). Using inaccurate models in reinforcement learning. In *Proceedings of 23rd International Conference on Machine Learning (ICML)*, Pittsburgh, PA.

Aberdeen, D. A. (2003). *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, Australian National University.

AHSRA (2008). Advanced cruise-assist highway systems research association (ahsra). Online available from: http://www.ahsra.or.jp/index_e.html, accessed July 9th, 2008.

Alton, K. (2004). Shaping and policy search for nearest-neighbour control policies with applications to vehicle steering. Master's thesis, University of British Columbia.

Anderson, C. W. (2000). Approximating a policy can be easier than approximating a value function. Technical report, Colorado State University, Fort Collins, CO, USA. CS-00-101.

ASTM International (2003). Astm e2213-03 standard specification for telecommunications and information exchange between roadside and vehicle systems - 5 ghz band dedicated short range communications (dsrc) medium access control (mac) and physical layer (phy) specifications. Online available from: http://www.astm.org, accessed February 27th, 2008.

Auto21 (2008). Online available from: http://www.auto21.ca/, accessed June 11th, 2008.

Bagnell, J. A. and Schneider, J. G. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*, Seoul, Korea.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 30–37, San Francisco, CA. Morgan Kaufmann.

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

Bareket, Z., Fancher, P. S., Peng, H., Lee, K., and Assaf, C. A. (2003). Methodology for assessing adaptive cruise control behavior. *IEEE Transactions on Intelligent Transport Systems*, 3(4):123–131.

Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning in pomdp's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 41–48, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Baxter, J., Bartlett, P. L., and Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artifical Intelligence Research*, pages 351–381.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ. ISBN = 0486428095.

Bishop, W. R. (2005). *Intelligent Vehicle Technology and Trends*. Artech House. ISBN = 1-58053-911-4.

BMW (2008). Online available from: http://www.bmw.com/, accessed May 5th, 2008.

Borgonovo, F., Campelli, L., Cesana, M., and Coletti, L. (2003). Mac for ad-hoc inter-vehicle network: services and performance. In *Proceedings of the IEEE Vehicular Technology Conference*, Orlando, Florida, USA.

Boyan, J. A. and Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems 7 (NIPS 7)*, Cambridge, MA. MIT Press.

Broggi, A., Bertozzi, M., Fascioli, A., Lo, C., and 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.

Bureau of Transportation Statistics (2007). *National Transportation Statistics*. U.S. Department of Transportation - Research and Innovative Technology Administration. http://www.bts.gov/publications/national_transportation_statistics/, accessed on June 9th, 2008.

Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., and Rosen, D. B. (1992). Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5):698–713.

Consortium, N. A. H. S. (1998). National automated highway system consortium technical feasability demonstration summary report. Technical report, National Automated Highway System Consortium, Troy, MI, USA.

Coulom, R. (2002). *Apprentissage par renforcement utilisant des réseaux de neurones, avec des applications au contrôle moteur*. PhD thesis, Institue National Polytechnique de Grenoble.

Cybenko, G. V. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dai, X., Li, C.-K., and Rad, A. B. (2005). An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. *IEEE Transactions on Intelligent Transport Systems*, 6(3):285–293.

DAMAS (2008). Online available from: http://www.damas.ift.ulaval.ca/, accessed June 11th, 2008.

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

Desjardins, C., Grégoire, P.-L., Laumônier, J., and Chaib-draa, B. (2007). Architecture and design of a multi-layered cooperative cruise control system. In *Proceedings of the SAE World Congress*, Detroit, MI, USA.

Desjardins, C., Laumônier, J., and Chaib-draa, B. (2008). Learning agents for collaborative driving. In *Multi-Agent Architectures for Traffic and Transportation Engineering*. In Prep.

Drew, M. C. (2002). Coordinated adaptive cruise control: Design and simulation. Master's thesis, University of California at Berkeley, Berkeley, CA, USA.

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.

El-Fakdi, A., Carreras, M., and Ridao, P. (2005). Direct gradient-based reinforcement learning for robot behavior learning. In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics (ICINCO 2005)*, pages 225–231. INSTICC Press.

Festag, A., Fussler, H., Hartenstein, H., Sarma, A., and Schmitz, R. (2004). Fleetnet: Bringing car-to-car communication into the real world. In *Proceedings of the 11th World Congress on ITS*, Nagoya, Japan.

Field, T. (2005). Policy-gradient learning for motor control. Master's thesis, Victoria University of Wellington.

Forbes, J., Oza, N., Parr, R., and Russell, S. (1997). Feasability study of fully automated vehicles using decision-theoretic control. Technical Report UCB-ITS-PRR-97-18, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

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

Forbes, J. R., Huang, T., Kanazawa, K., and Russell, S. J. (1995). The batmobile: Towards a bayesian automated taxi. In *Fourteenth International Joint Conference on Artificial Intelligence*, pages 418–423.

Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7 (NIPS 7)*, Cambridge, MA. MIT Press.

Fu, L., Henderson, J., and Hellinga, B. (2003). Inventory of intelligent transportation systems (its) in canada. Technical report, ITS Canada.

Gamma, E., Helm, R., Johnson, R., and Vissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison Wesley, Boston, MA. ISBN = 0-201-63361-2.

Girgin, S. and Preux, P. (2008). Basis expansion in natural actor critic methods. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL 2008)*, Villeneuve d'Ascq, France.

Globis Data Inc. (2008a). D.r.i.v.e.s. montreal. Online available from: http://www.globisdata.ca/montreal-traffic.html, accessed July 9th, 2008.

Globis Data Inc. (2008b). D.r.i.v.e.s. toronto. Online available from: http://www.globisdata.ca/toronto-traffic.html, accessed July 9th, 2008.

Günter, Y. and Großmann, H. P. (2005). Usage of wireless lan for inter-vehicle communication. In *IEEE Conference on Intelligent Transportation Systems*, pages 296–301.

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

Hallé, S. (2005). Automated highway systems: Platoons of vehicles viewed as a multi-agent system. Master's thesis, Université Laval, Québec, QC, Canada.

Hallouzi, R., Verdult, V., Hellendorn, H., Morsink, P. L., and Ploeg, J. (2004). Communication based longitudinal vehicle control using an extended kalman filter. In *IFAC Symposium on Advances in Automotive Control*, Salerno, Italy.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation.* Prentice Hall PTR, Upper Saddle River, NJ, USA. ISBN = 0-132-73350-1.

Huang, S. and Ren, W. (1999). Use of neural fuzzy networks with mixed genetic/gradient algorithm for automated vehicle control. *IEEE Transactions on Industrial Electronics*, 46(6):1090–1102.

Huppé, X. (2004). Guidance et commande longitudinale d'un train de voitures adaptées aux conditions routières et climatiques canadiennes. Master's thesis, Université de Sherbrooke, Sherbrooke, QC, Canada.

ITS Canada (2008). Online available from: http://www.itscanada.ca/, accessed February 23rd, 2008.

Jochem, T., Pomerleau, D., and Thorpe, C. (1993). MANIAC: A next generation neurally based autonomous road follower. In *Proceedings of the International Conference on Intelligent Autonomous Systems: IAS–3.*

Jurgen, R. K., editor (2006). *Adaptive Cruise Control.* SAE International, Warrendale, PA. ISBN = 978-0-7680-1792-2.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, 82 (Series D):35–45.

Kato, S., Tsugawa, S., Tokuda, K., Matsui, T., and Fujii, H. (2002). Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications. *IEEE Transactions on Intelligent Transport Systems*, 3(3):213–225.

Kiencke, U. and Nielsen, L. (2000). *Automotive Control Systems: for engine, driveline and vehicle.* Springer-Verlag. ISBN = 3-540-66922-1.

Kimura, H., Yamamura, M., and Kobayashi, S. (1995). Reinforcement learning by stochastic hill climbing on discounted reward. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 295–303, San Francisco, CA. Morgan Kaufmann.

Kimura, H., Yamamura, M., and Kobayashi, S. (1997). Reinforcement learning in pomdps with function approximation. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 152–160, San Francisco, CA. Morgan Kaufmann.

Kohl, N. and Stone, P. (2004). Policy-gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2004)*, pages 2619–2624, New Orleans, LA, USA.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

Laumônier, J. and Chaib-draa, B. (2006). Partial local friendq multiagent learning: Application to team automobile coordination problem. In *Proceedings of Canadian AI, Lecture Notes in Artificial Intelligence*, pages 361–372. Springer-Verlag.

Laumonier, J. (2008). *Méthodes d'apprentissage dans le cadre de la coordination multiagent: application au transport intelligent*. PhD thesis, Université Laval.

Lexus (2008). Lexus ls features. Online available from: http://www.lexus.com/models/LS/features/pricing.html, accessed May 5th, 2008.

Lu, X.-Y., Tan, H.-S., Empey, D., Shladover, S. E., and Hedrick, J. K. (2000). Nonlinear longitudinal controller development and real-time implementation. Technical Report UCB-ITS-PRR-2000-15, California Partners for Advanced Transit and Highways (PATH), University of Southern California.

Luo, J. and Hubaux, J.-P. (2004). A survey of inter-vehicle communication. Technical Report IC/2004/24, School of Computer and Communication Science, Lausanne, Switzerland.

Mandic, D. P. and Chambers, J. A. (2001). *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. John Wiley & Sons, Inc., New York, NY, USA. ISBN = 0-471-49517-4.

Marbach, P. (1998). *Simulation-Based Optimization of Markov Decision Processes*. PhD thesis, Laboratory for Information and Decision Systems, MIT.

McCallum, A. K. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, Rochester, New York.

Mercedes-Benz (2008). Mercedes-benz s-class ebrochure. Online available from: http://www.mbusa.com/media/downloads/main/models/eBrochure/sclass.pdf, accessed May 5th, 2008.

Microsoft (2008). Online available from: http://msdn.microsoft.com/en-us/directx/default.aspx, accessed May 15, 2008.

Moriarty, D. and Langley, P. (1998). Learning cooperative lane selection strategies for highways. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 684–691.

Morsink, P., Hallouzi, R., Dagli, I., Cseh, C., Schäfers, L., Nelisse, M., and de Bruin, D. (2003). Cartalk 2000: Development of a cooperative adas based on vehicle-to-vehicle communication. In *10th World Congress and Exhibition on Intelligent Transport Systems and Services*, Madrid, Spain.

Naranjo, J. E., Gonzàlez, C., de Pedro, T., Garcia, R., Alonso, J., Sotelo, M. A., and Fernandez, D. (2006a). Autopia architecture for automatic driving and maneuvering. In *Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1220–1225, Toronto, Canada.

Naranjo, J. E., Gonzàlez, C., Garcia, R., and de Pedro, T. (2006b). Acc+stop&go maneuvers with throttle and brake fuzzy control. *IEEE Transactions on Intelligent Transport Systems*, 7(2):213–225.

Naranjo, J. E., Gonzàlez, C., Reviejo, J., Garcia, R., and de Pedro, T. (2003). Adaptive fuzzy control for inter-vehicle gap keeping. *IEEE Transactions on Intelligent Transport Systems*, 4(2):132–142.

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

Ng, A. Y., Harada, D., and Russell, S. J. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 278–287, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ng, A. Y. and Jordan, M. I. (2000). Pegasus: A policy search method for large mdps and pomdps. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 406–415, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ng, A. Y., Kim, H. J., Jordan, M., and Sastry, S. (2004b). Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems 16 (NIPS 16)*.

Ünsal, C., Kachroo, P., and 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.

Panwai, S. and Dia, H. (2005a). Comparative evaluation of car following behaviour. *IEEE Transactions on Intelligent Transport Systems*, 6(3):314–325.

Panwai, S. and Dia, H. (2005b). Development and evaluation of a reactive agent-based car following model. In *Proceedings of the Intelligent Vehicles and Road Infrastructure Conference (IVRI' 05)*, pages 1–7, Melbourne, Australia.

Panwai, S. and Dia, H. (2007). Neural agent car-following models. *IEEE Transactions on Intelligent Transport Systems*, 8(1):60–70.

PATH (2008). Partnership for transit and highways (path). Online available from: http://www.path.berkeley.edu, accessed July 9th, 2008.

Pendrith, M. D. (2000). Distributed reinforcement learning for a traffic engineering application. In *the fourth international conference on Autonomous Agents*, pages 404 – 411.

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

Peters, J. and Schaal, S. (2006). Policy-gradient methods for robotics. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, Los Alamitos, CA, USA.

Pomerleau, D. (1995). Neural network vision for robot driving. In Arbib, M., editor, *The Handbook of Brain Theory and Neural Networks*.

Pyeatt, L. D. and Howe, A. E. (1998). Learning to race: Experiments with a simulated race car. In Cook, D. J., editor, *Proceedings of the Eleventh International Florida Artificial Intelligence Research Symposium Conference*, pages 357–361, Sanibel Island, FL. AAAI Press.

Rajamani, R. (2005). *Vehicle Dynamics and Control*. Springer-Verlag, New York, NY. ISBN = 0-137-90395-2.

Rajamani, R. and Zhu, C. (1999). Semi-autonomous adaptive cruise control systems. In *Proceeding of the American Control Conference*, pages 1491–1495.

Randlov, J. and Alstrom, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference (ICML'98)*. MIT Press.

Raza, H. and Ioannou, P. (1997). Vehicle following control design for automated highway systems. Technical Report UCB-ITS-PRR-97-2, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

Richardson, M., Corrigan, D., King, P., Smith, I., Barber, P., and Burnham, K. J. (2000). Application of a control system simulation in the automotive industry. In *IEE Seminar on Tools for Simulation and Modeling*, pages 8/1–8/13, London, UK.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. pages 318–362.

Russell, S. J. and Norvig, P. (2002). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition. ISBN = 0-137-90395-2.

Schrank, D. and Lomax, T. (2007). The 2007 urban mobility report. Technical report, Texas Transportation Institute at Texas A&M University.

Schulze, M. (2007). Cordis. chauffeur tr 1009: Summary of chauffeur project,. http://www.cordis.lu/telematics/tap_transport/research/projects/chauffeur.html. Accessed December 21, 2007.

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

Sengupta, R., Xu, Q., Mak, T., and Ko, J. (2004). Ad-hoc medium access control protocol design and analysis for vehicle safety communications. Technical Report UCB-ITS-PRR-2004-34, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

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

Shladover, S., VanderWerf, J., Miller, M. A., Kourjanskaia, N., and Krishnan, H. (2001). Development and performance evaluation of avcss deployment sequences to advance from today's driving environment to full automation. Technical Report UCB-ITS-PRR-2001-18, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

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

Statistics Canada (2006). *Canadian Vehicle Survey: Annual.* Government of Canada. Catalogue No. 53-223-XIE.

Strobel, T. and Servel, A. (2004). Sensor data sheets - state-of-the-art of sensors and sensor data fusion for automotive preventive safety applications. Technical report, PReVENT Consortium. http://www.prevent-ip.org/download/deliverables/ProFusion/PR-13400-IPD-040531-v10-Sensor_Data_Sheets.pdf.

Strobel, T., Servel, A., Coue, C., and Tatschke, T. (2004). Compendium on sensors - state-of-the-art of sensors and sensor data fusion for automotive preventive safety applications. Technical report, PReVENT Consortium. http://www.prevent-ip.org/download/deliverables/ProFusion/PR-13400-IPD-040531-v10-Compendium_on_Sensors.pdf.

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

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA. ISBN = 0-262-19398-1.

Sutton, R. S., Barto, A. G., and Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, 12(2):19–22.

Sutton, R. S., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NIPS 12)*, pages 1057–1063, Cambridge, MA. MIT Press.

Tambe, M. and Zhang, W. (2000). Toward flexible teamwork in persistent teams: extended report. *Journal of Autonomous Agents and Multi-agents Systems, special issue on Best of ICMAS 98*, 3:159–183.

TNO (2008). The netherland's organization for scientific research (tno). Online available from: http://www.tno.nl, accessed July 9th, 2008.

Transport Canada (2006). Canadian motor vehicle traffic collision statistics: 2005. Technical report. TP 3322, Cat. T45-3/2005.

Transport Canada (2007). The cost of urban congestion in canada. Technical report.

Tsitsiklis, J. N. and Van Roy, B. (1996). Feature-based methods for large scale dynamic programming. *Machine Learning*, 22(1-3):59–94.

Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

Tsugawa, S. (2005). Issues and recent trends in vehicle safety communication systems. *LATSS Research*, 29:7–15.

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

Vahidi, A. and Eskandarian, A. (2003). Research advances in intelligent collision avoidance and adaptive cruise control. *IEEE Transactions on Intelligent Transportation Systems*, 4(3):143–153.

VanderWerf, J., Shladover, S., and Miller, M. A. (2004). Conceptual development and performance assessment for the deployment staging of advanced vehicle control and safety systems. Technical Report UCB-ITS-PRR-2004-22, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

VanderWerf, J., Shladover, S. E., Miller, M. A., and Kourjanskaia, N. (2002). Effects of adaptive cruise control systems on highway traffic flow capacity. *Transportation Research Record*, 1800:78–84.

Varaiya, P. (1993). Smart cars on smart roads : Problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207.

Visser, R. (2005). Co-operative driving on highways. Master's thesis, Knowledge Centre Applications of Integrated Driver Assistance (IDA), University of Twente.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 538–545, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Wilson, R. E. (2001). An analysis of gipps's car-following model of highway traffic. *IMA Journal of Applied Mathematics*, 66(5):509–537.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley and Sons, Ltd., Chichester, West Sussex, England. ISBN = 0-471-49691-X.

Xu, Q., Hedrick, K., Sengupta, R., and VanderWerf, J. (2002). Effects of vehicle-vehicle/roadside-vehicle comunication on adaptive cruise controlled highway systems. In *Proceesings of the IEEE Conference on Vehicular Technology*.

Xu, Q., Mak, T., Ro, J., and Sengupta, R. (2005). Medium access control protocol design for vehicle-vehicle safety messages. Technical Report UCB-ITS-PWP-2005-04, California Partners for Advanced Transit and Highways (PATH), University of California at Berkeley.

Appendix A

Trajectory Probability Ratio Transformation

The following section demonstrates that we can replace the ratio of trajectory probabilities

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)}$$

with a sum of policy gradient ratios

$$\sum_{t=0}^{T} \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}$$

To simplify the demonstration, we use property A.1 to replace the original ratio by the log derivative (as seen in Equation A.2).

$$\nabla_x \log f(x) \quad = \quad \frac{\nabla_x f(x)}{f(x)} \tag{A.1}$$

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} \quad = \quad \nabla_{\vec{\theta}} \log q_{\vec{\theta}}(X) \tag{A.2}$$

Then, as detailed in Section 3.3.1, we can replace the trajectory probability $q_{\vec{\theta}}(X)$ by a product of all individual transition probabilities and action selection probabilities:

$$q_{\vec{\theta}}(X) \quad = \quad p(x_0)\Pi_{t=0}^{T-1} p(x_{t+1}|x_t, a_t)\pi_{\vec{\theta}}(x_t, a_t) \tag{A.3}$$

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} \quad = \quad \nabla_{\vec{\theta}} \log \left(p(x_0)\Pi_{t=0}^{T-1} p(x_{t+1}|x_t, a_t)\pi_{\vec{\theta}}(x_t, a_t)\right) \tag{A.4}$$

Then, using property A.5, we can develop equation A.4 as follows.

$$\log(ab) = \log a + \log b \tag{A.5}$$

$$\begin{aligned}
&= \nabla_{\vec{\theta}}(\log p(x_0) + \log \Pi_{t=0}^{T-1} p(x_{t+1}|x_t, a_t)\pi_{\vec{\theta}}(x_t, a_t)) \\
&= \nabla_{\vec{\theta}}(\log p(x_0) + \log p(x_1|x_0, a_0)\pi_{\vec{\theta}}(x_0, a_0) + \dots \\
&\qquad + \log p(x_T|x_{T-1}, a_{T-1})\pi_{\vec{\theta}}(x_{T-1}, a_{T-1})) \\
&= \nabla_{\vec{\theta}} \log p(x_0) + \sum_{t=0}^{T-1} \nabla_{\vec{\theta}} \log \left(p(x_{t+1}|x_t, a_t)\pi_{\vec{\theta}}(x_t, a_t)\right) \\
&= \nabla_{\vec{\theta}} \log p(x_0) + \sum_{t=0}^{T-1} \nabla_{\vec{\theta}} \left(\log p(x_{t+1}|x_t, a_t) + \log \pi_{\vec{\theta}}(x_t, a_t)\right) \\
&= \nabla_{\vec{\theta}} \log p(x_0) + \sum_{t=0}^{T-1} \nabla_{\vec{\theta}} \log p(x_{t+1}|x_t, a_t) + \sum_{t=0}^{T-1} \nabla_{\vec{\theta}} \log \pi_{\vec{\theta}}(x_t, a_t)
\end{aligned} \tag{A.6}$$

Of course, all the gradient terms that are not dependent on parameters $\vec{\theta}$ can be removed from this equation as their derivative is null. As a result, we obtain

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} = \sum_{t=0}^{T-1} \nabla_{\vec{\theta}} \log \pi_{\vec{\theta}}(x_t, a_t)$$

Finally, using the inverse transformation of A.1, we finally arrive to:

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} = \sum_{t=0}^{T-1} \frac{\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}$$

Thus, we proved that the trajectory probability ratio can be replaced by a sum of policy gradient ratios for each transition of a trajectory.

Appendix B

Soft-Max Policy Gradient Derivation

This section will show how we compute the gradient of the soft-max function used for stochastic action selection. First, remember that this function is defined as

$$\pi_{\theta}\left(s, a_i\right) = \frac{e^{Q_{\vec{\theta}}(s,a_i)}}{e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}} \tag{B.1}$$

This function generates the action selection probabilities using their values $Q_{\vec{\theta}}(s,a)$. In the case we have considered in Section 3.3.2, these values correspond to the outputs of the neural network before their soft-max exponentiation, and thus, they depend on the parameters $\vec{\theta}$.

To differentiate the soft-max function, we will need to use the following equations:

$$\frac{de^x}{dx} = e^x \tag{B.2}$$

$$\frac{d\left[\frac{f(x)}{g(x)}\right]}{dx} = \frac{f^{'}\left(x\right)g\left(x\right) - f\left(x\right)g^{'}\left(x\right)}{g\left(x\right)^2} \tag{B.3}$$

To find the gradient of the soft-max function, we have to differentiate the function, evaluated for state s and selected action a_i, according to value $Q_{\vec{\theta}}(s,a)$ of all actions $a \in \mathcal{A}$. In other words, we are looking for a vector of partial derivatives (the gradient) of the policy according to all the action values. This vector is illustrated by Equation B.4.

$$\frac{\partial \pi_{\vec{\theta}}\left(s, a_i\right)}{\partial Q_{\vec{\theta}}(s,\vec{a})} = \left[\begin{array}{cccc} \frac{\partial \pi_{\vec{\theta}}(s,a_i)}{\partial Q_{\vec{\theta}}(s,a_1)} & \frac{\partial \pi_{\vec{\theta}}(s,a_i)}{\partial Q_{\vec{\theta}}(s,a_2)} & \ldots & \frac{\partial \pi_{\vec{\theta}}(s,a_i)}{\partial Q_{\vec{\theta}}(s,a_m)} \end{array}\right] \tag{B.4}$$

Clearly, we have two cases to evaluate. In the first case, we have to compute the partial derivative of the policy $\pi(s, a_i)$ according to the value of the selected action $Q(s, a_i)$

$$\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_i)}$$

Next, we also have to calculate all the partial derivatives of the policy according to the values of all the other actions $Q(s, a_j)$ (where $a_i \neq a_j$) that were not selected.

$$\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_j)}$$

Let us observe both cases:

1. If $a_i = a_j$

$$\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_i)} = \frac{\partial \left(\frac{e^{Q_{\vec{\theta}}(s,a_i)}}{e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}} \right)}{\partial Q_{\vec{\theta}}(s, a_i)}$$

Using formula B.3, we can rewrite this as

$$\begin{aligned}
\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_i)} &= \frac{e^{Q_{\vec{\theta}}(s,a_i)} \left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right) - e^{Q_{\vec{\theta}}(s,a_i)} e^{Q_{\vec{\theta}}(s,a_i)}}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right)^2} \\
&= \frac{e^{Q_{\vec{\theta}}(s,a_i)} \left(\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right) - e^{Q_{\vec{\theta}}(s,a_i)} \right)}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right)^2} \\
&= \frac{e^{Q_{\vec{\theta}}(s,a_i)}}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right)} \cdot \frac{\left(\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right) - e^{Q_{\vec{\theta}}(s,a_i)} \right)}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right)}
\end{aligned}$$

Remembering the definition of the soft-max function, we can transform this equation into

$$\begin{aligned}
&= \pi_{\vec{\theta}}(s, a_i) \cdot \frac{\left(\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right) - e^{Q_{\vec{\theta}}(s,a_i)} \right)}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)} \right)} \\
&= \pi_{\vec{\theta}}(s, a_i) \cdot \left(1 - \pi_{\vec{\theta}}(s, a_i) \right)
\end{aligned}$$

2. If $a_i \neq a_j$

$$\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_j)} = \frac{\partial \left(\frac{e^{Q_{\vec{\theta}}(s,a_i)}}{e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}} \right)}{\partial Q_{\vec{\theta}}(s, a_j)}$$

Like preceedingly, we use formula B.3, to rewrite as

$$\begin{aligned}\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_j)} &= \frac{0 \cdot \left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}\right) - e^{Q_{\vec{\theta}}(s,a_i)} e^{Q_{\vec{\theta}}(s,a_j)}}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}\right)^2} \\ &= -\frac{e^{Q_{\vec{\theta}}(s,a_i)}}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}\right)} \cdot \frac{e^{Q_{\vec{\theta}}(s,a_j)}}{\left(e^{Q_{\vec{\theta}}(s,a_1)} + \ldots + e^{Q_{\vec{\theta}}(s,a_m)}\right)}\end{aligned}$$

By the definition of the soft-max function, we finally obtain

$$= \quad -\pi_{\vec{\theta}}(s, a_i) \cdot \pi_{\vec{\theta}}(s, a_j)$$

To summarize, the differentiation of the soft-max policy according to the action values $Q(s, a_j)$ gives the following result:

$$\frac{\partial \pi_{\vec{\theta}}(s, a_i)}{\partial Q_{\vec{\theta}}(s, a_j)} = \begin{cases} \pi_{\vec{\theta}}(s, a_i)\left(1 - \pi_{\vec{\theta}}(s, a_i)\right) & \text{if } a_i = a_j, \\ -\pi_{\vec{\theta}}(s, a_i)\, \pi_{\vec{\theta}}(s, a_j) & \text{if } a_i \neq a_j. \end{cases}$$