Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach

Charles Desjardins and Brahim Chaib-draa, *Senior Member, IEEE*

***Abstract*—Recently, improvements in sensing, communicating, and computing technologies have led to the development of driver-assistance systems (DASs). Such systems aim at helping drivers by either providing a warning to reduce crashes or doing some of the control tasks to relieve a driver from repetitive and boring tasks. Thus, for example, adaptive cruise control (ACC) aims at relieving a driver from manually adjusting his/her speed to maintain a constant speed or a safe distance from the vehicle in front of him/her. Currently, ACC can be improved through vehicle-to-vehicle communication, where the current speed and acceleration of a vehicle can be transmitted to the following vehicles by intervehicle communication. This way, vehicle-to-vehicle communication with ACC can be combined in one single system called cooperative adaptive cruise control (CACC). This paper investigates CACC by proposing a novel approach for the design of autonomous vehicle controllers based on modern machine-learning techniques. More specifically, this paper shows how a reinforcement-learning approach can be used to develop controllers for the secure longitudinal following of a front vehicle. This approach uses function approximation techniques along with gradient-descent learning algorithms as a means of directly modifying a control policy to optimize its performance. The experimental results, through simulation, show that this design approach can result in efficient behavior for CACC.**

***Index Terms*—Autonomous vehicle control, cooperative adaptive cruise control (CACC), neural networks, policy-gradient algorithms, reinforcement learning (RL).**

I. Introduction

RECENT improvements in sensing, communicating, and computing technologies have led to the development of driver-assistance systems (DASs). Such systems aim at assisting drivers by either providing warning to reduce crashes or performing some of the control tasks to relieve a driver from repetitive and boring tasks. Thus, a DAS can replace a driver's decisions and actions in some routines, without possible errors, which can lead to accidents, while achieving more regulated and smoother vehicle control [1]. As a consequence of this approach, we can cite the following three advantages: 1) more comfort for drivers; 2) increased traffic capacity; and 3) energy and environmental benefits.

According to Piao and McDonald [1], DASs encompass the following three aspects: 1) adaptive cruise control (ACC) and collision warning and avoidance (CWA); 2) legal aspects; and 3) implementation aspects. ACC aims at relieving a driver from manually adjusting his/her control to achieve a safe cruise, whereas CWA aims at reducing rear-end collisions by emitting suitable warnings. Legal aspects study, in general, the legal framework and market introduction of a DAS by analyzing its different functions. Finally, the implementation of a DAS is a complex task that encompasses a large variety of technologies, user preferences, and government policies.

Both ACC and CWA can be categorized into the following two types: 1) autonomous systems and 2) cooperative systems. For an *autonomous* system, the vehicle control mechanism only needs the information gathered by the sensors of the vehicle. On the other hand, *cooperative* systems require communication with adjacent vehicles or transportation infrastructure. Communication can be from vehicle to vehicle (V2V) or from the road to a vehicle (R2V). With V2V communication, a group of equipped vehicles can form a "virtual" network that is linked together by wireless ad hoc communication. With R2V, communication is achieved through different technical approaches, e.g., visible light, optical beacons, or the 5.9-GHz Dedicated Short-Range Communications (DSRC) standard [2]. Undoubtedly, the next evolution for DASs will come through the integration of modern wireless communication technologies as a means of sharing information between neighboring vehicles. Using such systems will enable vehicles to gather extended information about their surroundings, e.g., acceleration, heading, yaw rate, and even related to the driving intentions of other vehicles.

Analytical solutions to control problems as ACC or cooperative adaptive cruise control (CACC) are often elusive because of nonlinear dynamics and high-dimensional state spaces. Linearization, in general, does not help much in this case, and it would be better to investigate new approaches, particularly reinforcement learning (RL), which do not need the knowledge of the Markov decision process (MDP) that sustains it. In this paper, we consider a RL approach to the CACC approach that uses policy search, i.e., directly modifying the parameters of a control policy based on observed rewards. We have opted for a policy-gradient method, because unlike other RL methods, it scales very well to high-dimensional systems. The advantages of policy-gradient methods are numerous [3]. Among the most important approaches, it is required that the policy

Manuscript received February 23, 2009; revised June 1, 2010, October 28, 2010, and April 20, 2011; accepted May 5, 2011. Date of publication June 16, 2011; date of current version December 5, 2011. This work was supported in part by the AUTO21 Network of Centres of Excellence, which is an automotive research-and-development program that focuses on issues related to the automobile in the 21st century. AUTO21 (www.auto21.ca) is a member of the Networks of Centres of Excellence of Canada Program. The Associate Editor for this paper was N. Zheng.

C. Desjardins was with the Department of Computer Science and Software Engineering, Laval University, Quebec City, QC G1K 7P4, Canada. He is now with Genetec Inc., Montreal, QC H4S 2A4, Canada (e-mail: desjardins@damas.ift.ulaval.ca).

B. Chaib-draa is with the Department of Computer Science and Software Engineering, Laval University, Quebec City, QC G1K 7P4, Canada (e-mail: chaib@damas.ift.ulaval.ca).

Digital Object Identifier 10.1109/TITS.2011.2157145

representation can be chosen such that it is meaningful for the task, i.e., the domain knowledge can easily be incorporated. This approach, in general, leads to fewer parameters in the learning process compared to other RL methods. Moreover, there exists a variety of different algorithms for policy-gradient estimation in the literature, most of which are sustained with strong theoretical foundations. Finally, policy-gradient methods can be used model free and can therefore also be applied to problems without analytically knowing task and reward models.

Consequently, this paper proposes a policy-gradient algorithm for CACC, where this algorithm repeatedly estimates the gradient of the value with respect to the parameters, using information observed during policy trials, and then adjusts the parameters in the "uphill" direction.

This paper is organized as follows. Section II reviews the ACC, from autonomous to cooperative systems. Section III presents RL, a practical approach for the adaptive control. Section IV details the vehicle architecture and vehicle simulator. Section V briefly exposes the theoretical aspects behind the policy-gradient learning framework that sustains our RL. Section VI presents the experiments that we have conducted and gives the results achieved by the resulting controller.

II. Adaptive Cruise Control: From Autonomous to Cooperative Systems

A. Evolution of Cruise Control

New technology that actively intervenes and controls car driving may have a very great effect on comfort, safety, and traffic flow. ACC and CWA are representative of this technology, and currently, ACC is becoming widely available in passenger cars. The objective of ACC consists of automatically maintaining safe cruise driving, thus relieving a driver from manually performing a repetitive and boring task. When driving in free traffic, the ACC system holds a preset speed similar to any conventional cruise control system. When, on the other hand, a driver has to follow another vehicle, the system automatically maintains a desired time gap from the preceding vehicle [4]. An ACC can be conceived through an autonomous approach (using ranging sensors) or a cooperative approach (using V2V and/or R2V communication). Ranging sensors, e.g., radars or lasers, are generally used to measure the range and the rates of this range from the preceding vehicle. In general, ACC systems switch off when the speed is less than 30 km/h, because most of these systems are developed for highway traffic. In fact, the autonomous ACC based on ranging sensors has limited anticipatory capabilities, because it is impossible to react to what happens in front of the immediate predecessor.

Equipping the vehicles with V2V or R2V to ensure a cooperative approach started with the advent of intelligent transport systems (ITSs) in the 1980s. However, large-scale research and development of cooperative systems have only taken off in recent years. Early investigations in Europe were undertaken in the context of the PROMETHEUS project [5], where a 57-GHz communication system was developed to achieve an intervehicle communication system. Around 2000, the CHAUFFEUR project [5] was developed for trucks so that they can follow each other with a prerequired spacing. At the same time, the CarTALK project 2000 [6] was developed as a DAS that uses intervehicle communication. Currently, several projects on cooperative systems are ongoing, particularly under the Sixth Framework Program of the European Commission, e.g., Corporative Vehicle-Infrastructure Systems (CVIS) [7] and Cooperative Systems for Intelligent Road Safety (COOPERS) [8].

In Japan, one main program that is devoted to cooperative systems concerned the use of the differential global positioning system (DGPS) and V2V communication for the coordination of vehicle control. Some of this program, called Supersmart Vehicle Systems (SSVSs), was demonstrated in Smart Cruise 21 Demo 2000 [9]. During this demo, some facets of another main program, called Assist Highway Systems Research Association (AHSRA), was also demonstrated. AHSRA defined the following three levels of development of cooperative systems: 1) information to the driver (AHS-i); 2) control assist for the driver (AHS-c); and 3) fully automated operations (AHS-a).

In the U.S., the Cooperative Vehicle–Highway Automation Systems (CVHAS) Project was launched in 2000, with the main objective of providing the driver with control assistance. It is based on information about the vehicle's driving environment that we can obtain by communication through V2V or R2V. In this project, great efforts have been put into the deployment of advanced V2V and vehicle–infrastructure communications [10].

B. Related Work on ACC

As related work to ACC, we can cite the works of Hoffman *et al.* [11] and Davis *et al.* [12]. Hoffman *et al.* have developed a system for hybrid adaptive control (HACC) on high-speed roads, designed as a combination of a radar-based ACC and visual perception. The authors have shown that the combination of radar and vision leads to a system with an enhanced performance, which can jointly handle several tasks using a common knowledge base.

For their part, Davis *et al.* [12] have studied the flow of traffic composed of vehicles that are equipped with ACC using simulation. In their simulation, ACC vehicles are modeled by linear dynamical equation with string stability. By doing so, perturbations due to changes in the velocity of the lead of a platoon do not cause jams.

More recently, a vehicle controller for stop-and-go (SG) ACC has been proposed [13]–[15]. The main feature of this controller is that there is adaptation to a user-preset speed and, if necessary, speed reduction to keep a safe distance from the vehicle ahead in the same lane of the road, regardless of the speed. The extreme case is the SG operation, in which the lead car stops, and the vehicle at the rear must do the same. The authors of this approach did real experiments on two mass-produced Citroën Berlingo electric vans, in which all the actuators have been automated to achieve humanlike driving. In this context, the input information is acquired by a real-time kinematic phase-differential global positioning system (GPS) and wireless local area network links. The experimental results show that unmanned vehicles behave very similarly to human-driven cars and are very adaptive to any kind of situation at a

broad range of speeds, thus raising the safety of the driving and allowing cooperation with manually driven cars.

A complete comprehensive review of the development of ACC systems has recently been issued; see [16] for more details.

C. Related Work on CACC

In the context of CACC systems, de Bruin *et al.* [17] gave an overview on how such a system can be designed. They suggested using the following four techniques:

1) a positioning system;
2) a world model;
3) a controller system;
4) an inter vehicle communication system that allows including preview information from vehicles further in front.

Their first test results show that such a system enables anticipatory breaking actions, which means that upstream vehicles do not have to brake as severely when a downstream vehicle brakes, compared with the case without intervehicle communication.

In the same context, Naus *et al.* [18] have worked on a setup for CACC where the feasibility of the actual implementation is one of the main objectives. To this end, they have considered communication with the directly preceding vehicle only, as opposed to communication with multiple preceding vehicles or with a designated platoon leader. In addition, the communication has been implemented as a feedforward signal so that, if such a communication is not present, the standard ACC functionality will be available. To our knowledge, this approach has not been tested.

The communication aspect of CACC has particularly been studied by van Eenennaam *et al.* [19]. They proposed a channel busy-time model to evaluate the solution space of a vehicular-beaconing system designed to communicate information that is both vital and sufficient for vehicular traffic applications. These authors have identified that the solution space is 3-D because of the following three aspects: 1) the number of nodes (or vehicles); 2) the beacon generation rate of the nodes; and 3) the size (or duration) of a beacon message. Based on the channel busy-time model, the authors derived boundaries and a range of parameters, within which the beaconing system can be adapted to meet the requirements of the CACC system.

Finally, van Arem *et al.* [20] have studied the impact of CACC on traffic-flow characteristics. The authors have examined to what extent CACC can contribute to better traffic-flow performance. To this end, the authors performed simulations with data measured on a four-lane Dutch highway with a bottleneck due to a lane drop. Their simulation results indicate that CACC can improve traffic-flow performance. However, to what extent depends heavily on the traffic-flow conditions on the highway stretch and the CACC penetration rate. The traffic flow particularly improves in conditions with high traffic volume and when high fractions of the vehicle fleet are CACC equipped. Under these conditions, more vehicles can participate in a CACC platoon, resulting in reduced time gaps and improved string stability.

III. Reinforcement Learning as an Adaptive Control System

Most of the projects on CACC have relied on the classic control theory to develop autonomous controllers. However, ongoing research from the machine-learning community has yielded promising theoretical and practical results for the resolution of control problems in uncertain and partially observable environments, and it would be opportune to test it on CACC. One of the first research efforts to use machine learning for autonomous vehicle control was Pomerleau's autonomous land vehicle in a neural network (ALVINN) [21], which consisted of a computer vision system, based on a neural network, that learns to correlate observations of the road to the correct action to take. This autonomous controller drove a real vehicle by itself for more than 30 mi.

To our knowledge, Yu [22] was the first researcher to suggest using RL for steering control. According to Yu, using RL allows control designers to eliminate the need for external supervision and also to provide continuous learning capabilities. RL is a machine-learning approach that is shown as the adaptive optimal control of a process P, where the controller (called agent) interacts with P and learns to control it. To this end, the agent learns behavior through trial-and-error interactions with P. It then perceives the state of P, and it acts to maximize the *cumulative return* that is based on a real-valued reward signal, which comes from P after each action. Thus, RL involves modifying the control policy (which associates an action a to a state s) based on responses from the environment, and consequently, it is closely related to *adaptive control*, a family of successful control techniques held in high regard in the control systems community [23].

As aforementioned, Yu [22] was the first researcher to introduce RL to steering control. More precisely, he proposed to take vision sensor input (road boundary images) and generate steering control input while maintaining the vehicle moving within road boundaries. In addition to the usual neural learning using supervised data, the system uses the RL mode to eliminate the need for external supervision while, at the same time, providing the system with continuous learning ability similar to human driving practice. Road following has also been investigated by Se-Young *et al.* [24] using RL and vision. Through RL, the control system indirectly learns the vehicle–road interaction dynamics, the knowledge of which is essential to stay on the road in high-speed road tracking.

Mixing supervised learning and RL, Moriarti *et al.* [25] have proposed an approach that generates the lane-selection strategies through trial-and-error interactions with the traffic environment. The authors evaluated their approach through simulations and found that, compared with both a selfish strategy and the standard "yield to the right" strategy, their smart cars maintained speed close to the desired speeds of their drivers while making fewer lane changes.

The Forbes's work [26] has been directed toward obtaining a vehicle controller using instance-based RL. To this end, the work utilized stored instances of past observations as values estimates for controlling autonomous vehicles that were extended to automobile control tasks. An extensible simulation

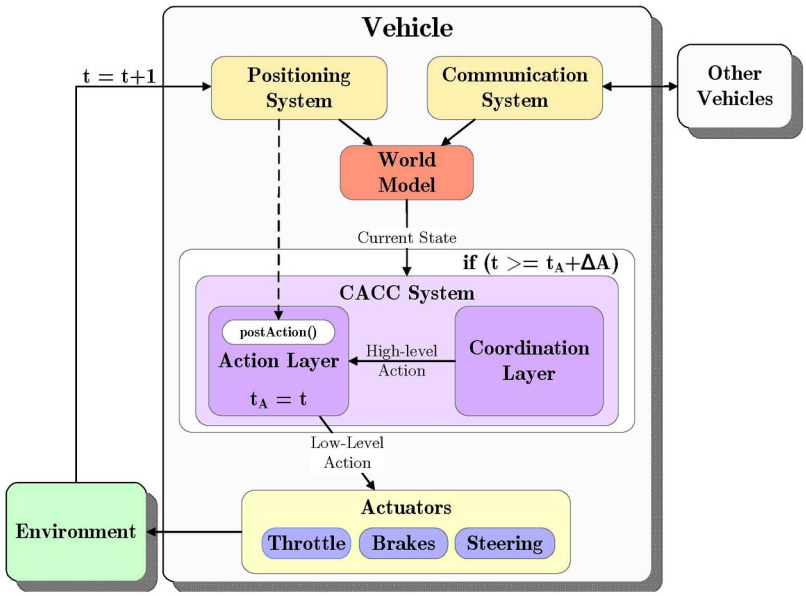

Fig. 1. Overview of the vehicle architecture with its CACC system.

environment and a hierarchical control architecture for autonomous vehicles have been implemented. In particular, the controllers derived from this architecture were evaluated and improved in the simulator until they took into account difficult traffic scenarios in a variety of (simulated) highway networks. This approach is, however, limited to the memory length, which can very rapidly grow when we deal with a realistic application.

More recently, Ng *et al.* [27] have proposed an adaptive control system using gain scheduling learned by RL. By doing so, they somehow kept the nonlinear nature of vehicle dynamics. This approach performs better than a simple linearization of the longitudinal model, which may not be suitable for the entire operating range of the vehicle. The performance of the proposed controller at specific operating points shows accurate tracking of both velocity and positions in most cases. When the adaptive controller is deployed in a convoy or a platoon, the tracking performance is less smooth. In particular, as the second car attempts to track the leader, slight oscillations result. This oscillation is passed onto the cars following, but as we move farther in the platoon, the oscillations decrease, implying stability. Thus, this approach is more convenient for platooning control than the CACC, because in this later case, it engenders slight oscillations.

Thus, although some attempts have been directed toward lateral and longitudinal control using RL, no researcher has particularly used RL for controlling CACC. This paper attempts to fill this gap.

IV. Vehicle Architecture and Simulator

A. Vehicle Architecture

We have designed system architecture around a vehicle architecture in connection with its environment encompassing other vehicles. Fig. 1 illustrates this architecture by showing, in particular, that the *CACC system* has two separate layers. First, the *coordination layer* is responsible for the selection of "high-level actions," e.g., lane changing or secure vehicle following. It was created for experiments, which were conducted at the Decision Adaptation Multiagents (DAMAS) Laboratory by Laumônier [28], on learning multiagent 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. In our case, the action layer aims at elaborating actions according to a policy-gradient estimation represented under the form of a backpropagation neural network, as explained in Section V-C. Notice that this paper focuses only on learning a policy for the secure longitudinal vehicle-following behavior, and consequently, it ignores the presence of the *coordination layer* in the simulator's control loop. For more details about this specific layer, see [29] and [30].

Accurate simulation also requires the use of an environment perception mechanism. To address this simulation need, we designed two modules to handle information gathering from the neighborhood of vehicles. First, we designed a *positioning system* that is sustained by a "sensor manager" in charge of all sensors embedded on a vehicle. Among existing sensing solutions, front-vehicle sensors are particularly interesting, because several of these sensors are currently built by third-party manufacturers and are embedded in vehicles for ACC applications. These sensors all work in a similar fashion by employing either radar or laser technology to sense the area in front of a vehicle. Their specifications are usually described by their reach, their angle of observation, and their update (refresh) rate. The European Commission Intelligent Vehicles (IV) project Preventive and Active Safety Applications to Improve Traffic Safety (PReVENT) explains how such sensors work in their state of the art on that technology [31], whereas its companion report [32] 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 m,

a beam angle of 8°, and a sensor refresh rate of 10 Hz. Finally, our front laser implementation does not include the notion of sensor noise or delay.

The other important module for the management of vehicle perceptions is the *communication system*, which is used to handle each car's intervehicle communication system. When vehicles decide to transmit information to other vehicles, they pass the messages through their communication system. For protocols that specify how the reception of messages is affected by factors such as communication range, researchers use parameters such as delay and packet loss. These parameters are applied by the global communication system to every transmitted message. These messages are then accordingly transferred to each receiving vehicle's communication system. In our case, we have made the assumption of the availability of a communication channel with a messaging delay of 100 ms, which is a value that has been considered acceptable for future safety-critical systems [33]. The transmission range has been evaluated to 100 m, because this value is consistent with the IEEE 802.11b wireless local area network (LAN) communication protocol used for intervehicle messaging [34]. As aforementioned, we did not consider noise in the communication channel or packet loss; the messages are received exactly as they were sent.

The *world model*, which represents a snapshot of the environment at the moment considered, elaborates the current state and sends it to the CACC system, which, in turn, deliberates on the low-level action to send to the *actuators*.

Finally, note that the focus of our experiments was not on the integration and implementation of *exact* specifications and behavior of vehicle systems. However, the approach that we developed is still a precise model that is also adequate for our needs related to environment perception. Of course, it is interesting, in future work, to bring this framework even closer to reality by integrating more accurate sensor and communication models that can include precise simulation.

B. Vehicle Simulator

Designing an autonomous longitudinal vehicle controller using RL implies running numerous experiments in a simulator. To build such a simulator, we have adopted the following three requirements.

1) The behavior of the simulated vehicles should be as close as possible to the real vehicles.
2) The simulator should be flexible enough to embed a RL engine.
3) The simulator should run "faster than real time" 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 RL algorithms. Thus, to address our needs for a precise simulation tool in which we can embed an autonomous vehicle control system based on RL, we have built a microscopic vehicle simulator. The microscopic simulation approach has been justified by the fact that it focuses on more or less accurately modeling the behavior of each simulated vehicle, depending on the required precision. This condition contrasts with macroscopic vehicle simulation methods, which, in general, put emphasis on equations that globally model the traffic conditions. Macroscopic simulators are mainly used to simulate a large number of vehicles for traffic-flow studies [35].

Another important characteristic of our simulator lies in its discrete-time approach. This condition means that simulations are divided into time steps Δt, which represent a certain amount of simulated time that occur 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 to update the position of the objects for the following time step. Because the computation of these updates does not rely on actual time but, rather, on the simulation time step value Δt, this method has the advantage of efficiently simulating "faster than real time."

We have used a time step value of 10 ms (corresponding to a 100-Hz update frequency), a value required for accurate physics modeling. However, we designed our control loop so that we can decouple the refresh rate of the different modules from the simulator's main loop. Thus, sensor, communication, and learning modules updated at different frequencies. For example, we can update sensors and communication at every 100 ms while taking learning decisions at every 250 ms.

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 computing their resulting position, velocity, and acceleration.

For a computationally efficient simulation, we relied on dynamics based on the "single-track" model, as described by Kiencke *et al.* [36]. This model features a nonlinear approximation technique that takes particular care in the description of the wheel model and the vehicle's longitudinal and lateral motion. Although 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 shown as one), the model is still quite precise, because it integrates the notions of friction and 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 [37]. By taking as inputs the pressure on the acceleration and brake pedals, the driveline model first computes the engine torque and transmits it to the automatic transmission module, which then calculates the resulting drive torque. Finally, the drive torque is transformed to obtain the wheels' angular acceleration that can be used as input by the wheel model. Because our driving agents take direct action on the vehicle's acceleration and brake pedals, the presence of a complex nonlinear driveline whose behavior is close to a real vehicle is a valuable characteristic for us, because it gives much more realism to the definition of our problem. For a complete description of the modules that are part of our vehicle dynamics engine, see [36]–[38].

V. Controller Design Using Policy-Gradient Methods

A. RL: A Brief Introduction

In RL, the agent-learner observes the state s (where $s \in S$, and S is the finite set of states) and takes an action a that can depend on s. The action a causes the environment to change its state from s to s' according to the probability $p_{ss'}(s)$. Note that the action set of the agent is assumed to be finite, and it is allowed to depend on the current state, i.e., the action set is $A(s)$. In these conditions, after taking action $a \in A(s_t)$, the agent receives a reinforcement signal under a real value form $r_{s,a}(s')$ that depends on the action a taken and the successor state s'. The environment that sustains RL is typically formulated as a finite-state MDP, and RL algorithms are consequently highly related to dynamic programming. The Markov property beside the MDP requires that the probability distributions on the successor state and on the rewards depend on the current state and action only. In other words, the distributions do not change when additional information on past states, actions, and rewards is considered.

In RL, at each time step t, the agent perceives its state $s_t \in S$ and the set of possible actions $A(s_t)$; it chooses and executes an action $a \in A(s_t)$ and then receives the new state s_{t+1} and the reward r_{t+1}. The aim of the agent is to find, through its interactions, a policy $\pi : S \rightarrow A$ that maximizes the cumulative reward $R = r_0 + r_1 + \cdots + r_n$ for an MDP with a terminal state or the quantity $R = \sum_t \gamma^t r_t$ for an MDP without terminal states (note that $0 \leq \gamma \leq 1$ and is the discount factor that promotes recent rewards). The expectation of the cumulative reward $V^\pi(s) = E[R|s_0 = s]$ is defined as the "value" of the state s with respect to π. There exists a stationary and deterministic policy π^* for which $V^{\pi^*}(s)$ is optimal for every state s. This policy π^* is termed the *optimal policy*. Sometimes, it would be more useful for an agent to learn a Q-value that learns an action-value representation instead of the value $(V(s))$ in a state. The idea is to learn a Q-function, which is a prediction of the return associated with each action $a \in A$ in each state. This prediction can be updated with respect to the predicted return of the next state visited $Q(s_t, a_t) \leftarrow r_t + \gamma V(s_{t+1})$.

Because the overall aim of the system is to maximize the payoffs received, the current estimate of $V(s_t)$ of a state is given by $\max_{a \in A} Q(s_t, a)$. In these conditions, the previous value of $Q(s_t, a_t) \leftarrow r_t + \gamma V(s_{t+1})$ becomes

$$Q(s_t, a_t) \leftarrow r_t + \gamma \max_{a \in A} Q(s_{t+1}, a).$$

One major issue of classic RL 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, because it rapidly grows 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 RL 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 means of solving control problems that has gained particular interest in recent years has been the use of policy-gradient methods. This class of algorithms modifies the parameters of a stochastic policy in the direction that optimizes the expected reward. These algorithms have can work with continuous-state variables, because the policy that they use is based on a value function approximator. However, these approaches are different, because they do not rely on accurately approximating the value function but, rather, on modifying it to improve the current policy. Their advantage over classic RL methods that use function approximation is that policy-gradient algorithms are guaranteed to converge (although only to a local maximum). Of course, such abilities to handle high-resolution continuous environments are prerequisites for efficiently solving the problem of autonomous vehicle control considered here.

Another important aspect of policy-gradient methods that justifies our choice comes from the fact these methods are model-free approaches. This condition means that these methods can learn an efficient behavior by direct interaction with the environment. This characteristic is also important in the context of our problem, because 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. Because it is quite difficult to solve such a control problem using analytical mathematics with precision, approaches that learn to control a system by direct interaction offer an obvious advantage, because 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 [39].

Policy-gradient algorithms have been applied to various control problems, e.g., autonomous helicopter flight [40], robot locomotion [41], [42], and unmanned underwater vehicle control [43]. In this paper, we show that this approach can also be efficient for the resolution of the autonomous longitudinal vehicle control problem.

For these reasons, we selected the policy-gradient algorithm proposed in [44], because it is well adapted to solve the problem of finding an ACC control policy in our complex vehicle simulator. This online algorithm can update the weights of the policy using an estimation of the gradient at each iteration. In fact, this algorithm is designed to handle partially observable MDPs, but to solve our problem, we assumed full observability over the environment.

B. Estimation of the Policy Gradient

Policy-gradient methods rely on the computation of the gradient of a policy's performance according to its parameters. However, because the exact calculation of this gradient is only possible when the transition probabilities are known [44], researchers have, instead, focused on estimating it through simulation.

In the case where the policy is under the form $\pi_{\vec{\theta}}(x_t, a_t)$, denoting the probability that the agent chooses action a_t in state s_t, the performance ρ of such a policy according to its parameters[1] $\vec{\theta}$ corresponds to the expected reward $r(X)$, which can be obtained through a trajectory X [44]. The following equation reflects that

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

This expectation can be expressed as a sum, over all possible trajectories, of the reward obtained through a trajectory X weighted by the probability $q_{\vec{\theta}}(X)$ of observing this trajectory $\rho(\vec{\theta}) = \sum_X r(X) q_{\vec{\theta}}(X)$.

As we can see, this probability is parameterized by weights $\vec{\theta}$, because it depends not only on the environment's transition probabilities but also on the choices made using the stochastic parameterized control policy. Its gradient is then expressed by the following equation: $\nabla_{\vec{\theta}}\rho(\vec{\theta}) = \sum_X r(X)\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)$.

By multiplying the right term of this equation by $(q_{\vec{\theta}}(X)/q_{\vec{\theta}}(X))$, we see that the gradient calculation corresponds to

$$\nabla_{\vec{\theta}}\rho(\vec{\theta}) = \sum_X r(X)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} q_{\vec{\theta}}(X) \tag{1}$$

$$= E\left[r(X)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)}\right]. \tag{2}$$

Finally, the expectation that is reflected by (2) can be estimated by averaging over N trajectories as

$$\hat{\nabla}_{\vec{\theta}}\rho(\vec{\theta}) = \frac{1}{N}\sum_{i=1}^{N} r(X_i)\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X_i)}{q_{\vec{\theta}}(X_i)}. \tag{3}$$

Under the MDPs framework, the probability $q_{\vec{\theta}}(X)$ of observing a specific trajectory X corresponds to 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} in this trajectory, as reflected by the following equation:

$$q_{\vec{\theta}}(X) = p(x_0)p_{\vec{\theta}}(x_1|x_0)p_{\vec{\theta}}(x_2|x_1)\ldots p_{\vec{\theta}}(x_T|x_{T-1}) \tag{4}$$

$$= p(x_0)\Pi_{t=0}^{T-1} p_{\vec{\theta}}(x_{t+1}|x_t). \tag{5}$$

However, and as noted earlier, these transitions depend both on the environment's dynamics and the policy used for control. As a result, it is possible to rewrite (5) as

$$q_{\vec{\theta}}(X) = p(x_0)\Pi_{t=0}^{T-1} p(x_{t+1}|x_t)\pi_{\vec{\theta}}(x_t, a_t) \tag{6}$$

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.

This result can be used to replace the trajectory's probability $q_{\vec{\theta}}(X)$ in (3), because it is possible to show that the ratio $(\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)/q_{\vec{\theta}}(X))$ corresponds to a sum, over the trajectory, of ratios $(\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t, a_t)/\pi_{\vec{\theta}}(x_t, a_t))$, as given by the following:

$$\frac{\nabla_{\vec{\theta}} q_{\vec{\theta}}(X)}{q_{\vec{\theta}}(X)} = \sum_{t=0}^{T-1}\frac{\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}. \tag{7}$$

Thus, and as suggested by the policy-gradient name, we end up differentiating the policy's representation $\pi_{\vec{\theta}}(s, a)$ according to its parameters $\vec{\theta}$. This method reflects an important aspect of policy-gradient algorithms, because it renders possible the optimization of the policy's performance without having any knowledge of the system's dynamics. Further details about this transformation are given in [41], [45], and [46].

Now that we have differentiated the policy's representation according to its parameters, (7) can recursively be accumulated over the course of a trajectory using an intermediate variable $\vec{z}_t$. This variable, which is called the eligibility trace, contains a sum of gradients in the space of the policy parameters that result from each transition and represents the influence of each weight in the previous decisions, i.e.,

$$\vec{z}_{t+1} = \beta\vec{z}_t + \frac{\nabla_{\vec{\theta}}\pi_{\vec{\theta}}(x_t, a_t)}{\pi_{\vec{\theta}}(x_t, a_t)}. \tag{8}$$

Using the same technique, rewards can also recursively be accumulated through a trajectory using an intermediate variable r_t. With these two new variables, the gradient estimator of (3) can now be expressed by

$$\hat{\nabla}_{\vec{\theta}}\rho(\vec{\theta}) = \frac{1}{N}\sum_{i=1}^{N} r_i\vec{z}_i \tag{9}$$

where r_i and $\vec{z}_i$, respectively, denote the reward and the eligibility trace cumulated over trajectory i. This equation also shows that the rewards are distributed to each weight according to their influence in the decision. Finally, the estimation of the gradient can be used to update the policy parameters.

We have briefly described the general policy-gradient framework; see [45], [47], and [48] for more details.

C. Policy-Gradient Estimation Through a Backpropagation Neural Network

The OLPOMDP algorithm, as proposed in [39] and [49] and prolonged by [45], is an online algorithm based on the policy-gradient principle described in the previous section. Instead of estimating the gradient over a certain number of state transitions T, the online approach modifies the weights of the policy representation at each step. This stochastic gradient algorithm, as presented in the algorithm in Fig. 2, has been shown to converge to local optima [50].

This algorithm works as follows. After initializing the parameters vectors $\vec{\theta}_0$, the initial state x_0, and the eligibility trace $z_0 = 0$, the algorithm iterates T times. After each iteration, the parameters' eligibility z_t is updated according to the policy-gradient approximation. The discount factor $\beta \in [0, 1]$ increases or decreases the agent's memory of past actions.

[1]The parameters vector θ represents all the modifiable parameters of the chosen function approximation representation, e.g., the weights that are adjusted during learning in a neural network.

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** $\vec{\theta}_T$.

Fig. 2. Policy-gradient algorithm (adopted from [45]).

The immediate reward r_{t+1} and the learning rate α allow for computing the new vector of parameters $\vec{\theta}_{t+1}$. Finally, the new parameters allow for modifying the current policy, making it closer to a final policy (local optima) that represents a solution to our problem.

The most important operation of this algorithm is the computation of the gradient of the policy $\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)$. Consequently, policy-gradient algorithms consider the use of stochastic policies, because they are usually easily differentiable. The use of neural networks for policy representation is a particularly appropriate technique that offers significant advantages. First, because the network's task is to return action probabilities according to the current state features, these networks can easily be extended with an additional layer that uses the soft-max function to compute these probabilities.

The soft-max function is particularly useful in the policy-gradient context, because it easily meets the differentiability requirements of the algorithm presented in Fig. 2. This function is expressed as follows:

$$\pi_\theta\left(x_t, a_t^i\right) = \frac{e^{Q_{\vec{\theta}}\left(x_t, a_t^i\right)}}{e^{Q_{\vec{\theta}}\left(x_t, a_t^1\right)} + \cdots + e^{Q_{\vec{\theta}}\left(x_t, a_t^m\right)}}. \tag{10}$$

In fact, this function generates the action selection probabilities using their Q-values $Q_{\vec{\theta}}(x, a)$, which correspond to the outputs of the neural network before their soft-max exponentiation, and thus, they depend on the parameters $\vec{\theta}$. In this case, $\nabla_\theta \pi_\theta = \nabla_Q \pi \nabla_\theta Q_\theta$.

Using the neural network, step 10 of algorithm 2 can be rewritten as

$$z_{t+1} = \beta z_t + \frac{\nabla_Q \pi}{\pi_\theta} \nabla_\theta Q_\theta. \tag{11}$$

Then, we can differentiate the soft-max function and evaluate $\nabla_Q \pi$. To do so, we have two cases to evaluate. In the first case, we have to compute the partial derivative of the policy $\pi(x_t, a_t^i)$ according to the value of the selected action $Q(x_t, a_t^i)$: $(\partial \pi_{\vec{\theta}}(x_t, a_t^i) / \partial Q_{\vec{\theta}}(x_t, a_t^i))$.

Next, we have to calculate all the partial derivatives of the policy according to the values of all the other actions $Q(x_t, a_t^j)$ (where $a_t^i \neq a_t^j$) that were not selected: $(\partial \pi_{\vec{\theta}}(x_t, a_t^i) / \partial Q_{\vec{\theta}}(x_t, a_t^j))$.

This case leads to the following result:

$$\frac{\partial \pi_{\vec{\theta}}\left(x_t, a_t^i\right)}{\partial Q_{\vec{\theta}}(x_t, a_t^j)} = \begin{cases} \pi_{\vec{\theta}}\left(x_t, a_t^i\right)\left(1 - \pi_{\vec{\theta}}\left(x_t, a_t^i\right)\right), & \text{if } a_t^i = a_t^j \\ -\pi_{\vec{\theta}}\left(x_t, a_t^i\right)\pi_{\vec{\theta}}\left(x_t, a_t^j\right), & \text{if } a_t^i \neq a_t^j. \end{cases}$$

Of course, to compute policy-gradient $\nabla_{\vec{\theta}} \pi_{\vec{\theta}}(x_t, a_t)$, we need to differentiate the policy according to all of its weights. This condition can easily be achieved using the standard backpropagation algorithm, which is another reason that neural networks are appropriate for use with policy-gradient algorithms. Backpropagation is used to backpropagate the soft-max gradient through the network to obtain the gradient of each weight. However, the policy-gradient algorithm defines the eligibility (local gradient of the weights for a single step) as $(\nabla_Q \pi_{\vec{\theta}}(x_t, a_t) / \pi_{\vec{\theta}}(x_t, a_t))$. As a result, the soft-max gradient divided by $\pi_{\vec{\theta}}(x_t, a_t)$ is the actual value to be backpropagated, and this value is expressed as follows:

$$\frac{\nabla_Q \pi_{\vec{\theta}}\left(x_t, a_t^i\right)}{\pi_{\vec{\theta}}\left(x_t, a_t^i\right)} = \begin{cases} 1 - \pi_{\vec{\theta}}\left(x_t, a_t^i\right), & \text{if } a_t^i = a_t^j \\ -\pi_{\vec{\theta}}\left(x_t, a_t^j\right), & \text{if } a_t^i \neq a_t^j. \end{cases} \tag{12}$$

If we refer to (11), it remains to evaluate $\nabla_\theta Q_\theta$. This expression corresponds to the derivative of the activation function associated with each neuron of the output layer. Note that we do not backpropagate the gradient of an error measure; instead, we back propagate the soft-max gradient of this error.

To sum up, the algorithm represented in Fig. 2 works as follows. First, we initialize the parameters β, T, θ_0, and $\vec{z}_0$. At every iteration, the parameters gradient z_t is updated. According to the immediate reward received $r(x_{t+1})$, the new parameter vector θ_{t+1} is calculated, and the current policy π_{t+1} followed the next iteration, getting closer to $t \to T$ to a final policy π_T.

The next section shows how the policy-gradient estimation can be used to learn a longitudinal vehicle controller.

VI. Experiments and Results

This section details our efforts in designing an autonomous CACC system [33] that integrates both sensors and intervehicle communication in its control loop to keep a secure longitudinal vehicle-following behavior. To achieve this approach, we used the policy-gradient method that we described in the previous section to learn a vehicle control by direct interaction with a complex simulated driving environment. This section presents the driving scenario considered, details the learning simulations, and evaluates the performance of the resulting policies.

A. Experiments

1) Learning Task: The learning task considered here corresponds to a Stop&Go (SG) scenario. This type of scenario is interesting, because it usually happens on urban roads. It has been used by several researchers for the development of autonomous controllers and the assessment of their efficiency and effects on the traffic flow [15],

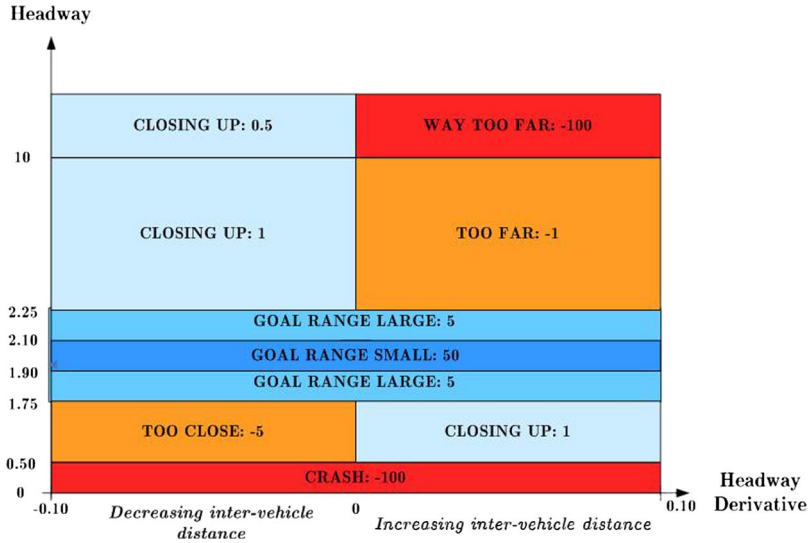

Fig. 3. Reward function.

[51], [52]. In our case, a leading vehicle that starts 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 s, and then accelerates back to its original 20 m/s cruising velocity. In this case, the learning vehicle's task was to learn to follow the leading vehicle while keeping a specific desired range of 2 s. These simulations were executed in our vehicle simulator presented in Section IV.

2) State and Action Spaces: For the definition of the states, we have considered the following three state variables: 1) the headway (Hw); 2) the headway derivative (ΔHw); and 3) the front-vehicle acceleration (Acc). Headway (also called the "range") refers to the distance in time from a front vehicle and is calculated as follows:

$$Hw = \frac{(\text{Position}_{\text{Leader}} - \text{Position}_{\text{Follower}})}{\text{Velocity}_{\text{Follower}}}. \tag{13}$$

This measurement is standard for intervehicle spacing (see [15], [53], and [54]) that has the advantage of being dependent on the current velocity of the follower. This state representation is also interesting, because it is independent of the velocity of the front vehicle (i.e., the leader). Thus, a behavior learned using these states will generalize to all the possible front-vehicle velocities.

The headway derivative (also called the "range rate") (ΔHw) contains valuable information about the relative velocity between the two vehicles and is expressed by

$$\Delta Hw = Hw_t - Hw_{t-1}. \tag{14}$$

It indicates whether the vehicles have been moving closer to or farther from each other since the previous update of the value. Both the headway and the headway derivative are provided by a simulated laser sensor, as detailed in Section IV. Although we consider continuous values, we restrict the size of the state space by bounding the value of these variables to specific intervals that offer valuable experience to learn vehicle-following behavior. Thus, we bounded the possible values of headway from 0 s to 10 s, whereas the headway derivative was bounded from -0.1 s to 0.1 s.

The acceleration (Acc), which is obtained through wireless intervehicle communication, is another important state variable of our system. Similar to other variables, the acceleration values were bounded to a particular interval, ranging from -2 m/s^2 to 2 m/s^2.

Finally, the action space contains the following three actions: 1) a braking action $(B100)$; 2) a gas action $(G100)$; and 3) a "no-op" action $(NO-OP)$. The state and action space of our framework can formally be described as follows:

$$S = \{Hw, \Delta Hw, Acc\} \tag{15}$$

$$A = \{B100, G100, NO-OP\}. \tag{16}$$

3) Reward Function: The success of the learning task depends on the reward function used by the agent, because this function is mostly used by the learning algorithm to direct the agent in areas of the state space where it will gather the maximum expected reward. Evidently, the reward function must be designed to give positive reward values to actions that get the agent toward the safe intervehicle distance to the preceding vehicle (see Fig. 3).

Because the secure intervehicle distance should be around 2 s (a common value in industrialized countries' legislation [33]), we decided that a large positive reward should be given when the vehicle enters the zone that extends at ± 0.5 s from the headway goal of 2 s. We also considered a smaller zone at ± 0.1 s from the safe distance, where the agent receives 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 very far from the safe distance or when it is very close to the preceding vehicle. To reduce learning times, we also use a technique called reward shaping [55], which directs the exploration of the agent by giving positive rewards to actions that make the agent progress along a desired trajectory through the state space (i.e., by giving

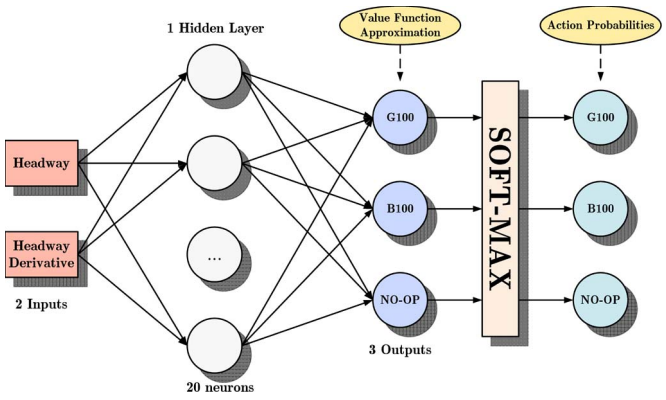

Fig. 4. Neural network architecture policy representation.

TABLE I
OLPOMDP PARAMETERS FOR THE SINGLE-TRACK ACC LEARNING TASK

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

positive rewards when the vehicle is very far but gets closer to its front vehicle).

4) Algorithm: We have used the policy-gradient algorithm, as presented in Section V-C, to solve the aforementioned SG. The neural network that corresponds to the policy representation for the problem that we consider is shown in Fig. 4.

The parameters that must be set for the algorithm include β, which describes the bias-variance tradeoff, α, which refers to the learning rate, and T, which refers to the number of iterations of the algorithm. The parameters were empirically set to the values shown in Table I. The number of iterations was at most of 2 500 000 (5000 episodes of 500 steps), but in practice, this number was slightly lower, because some episodes were reset when the agent was very close from its preceding vehicle. Evaluated over ten learning simulations, we obtained an average number of iterations of 2 202 718.

In our case, we have opted for an artificial neural network with two inputs for state variables (the headway and the headway derivative), one hidden layer of 20 neurons, and one output layer of three neurons (1 for the value of each action). The threshold function used for the hidden-layer neurons was a sigmoid function, whereas the output-layer nodes used a linear function. This network architecture is adopted to function approximation, because it has been shown that networks with one hidden and one output layer using, respectively, sigmoidal and linear transfer functions can approximate any given function with a precision that depends on the number of neurons on the hidden layer [56]. Because 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.

B. Results

1) Learning: Due to the stochastic nature of the policy-gradient algorithms, ten learning simulations that result in ten different control policies have been executed. Results of the learning process are given in Fig. 5, which gives the minimum, average, and maximum reward sum obtained through our learning simulations.

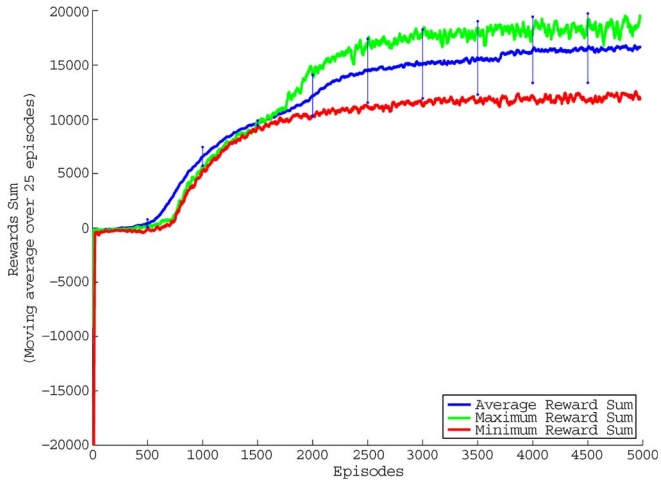

Fig. 5. Rewards sum for ten learning simulations of 5000 episodes.

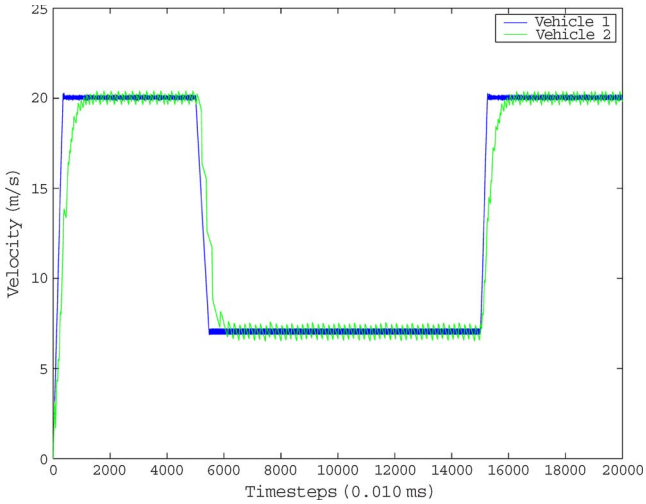

Fig. 6. Vehicle velocities for a single-track CACC simulation.

2) Execution: After the learning phase, the policy that yields the highest reward sum (shown in green in Fig. 5) has been selected and was tested in the same scenario that was used for learning. Other results are presented in Figs. 6–8. They show, respectively, the velocities of both vehicles during the simulation, their accelerations, and the headway of the follower.

3) Discussion: The headway result of the follower, as shown in Fig. 8, indicates that, when the front vehicle is braking, the follower can keep a secure distance by using the learned policy. During this time interval, the headway of the follower oscillates close to the desired value of 2 s (approximately 1.85 s). Note that this oscillatory behavior is due to the small number of discrete actions that we consider here.

From timesteps 5500 to 5600, however, it can be observed that CACC 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

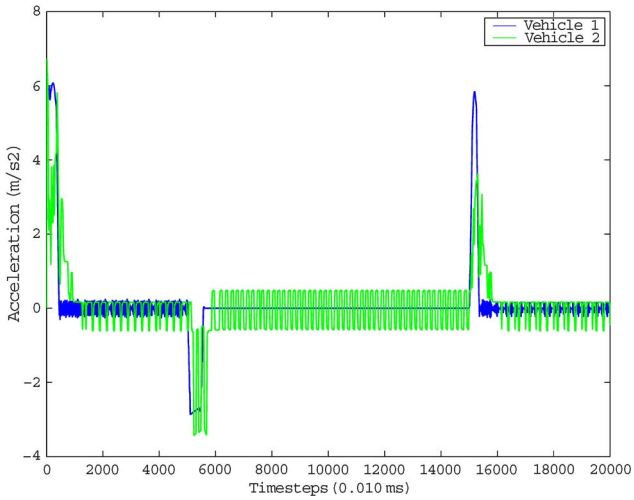

Fig. 7. Vehicle accelerations for a single-track CACC simulation.

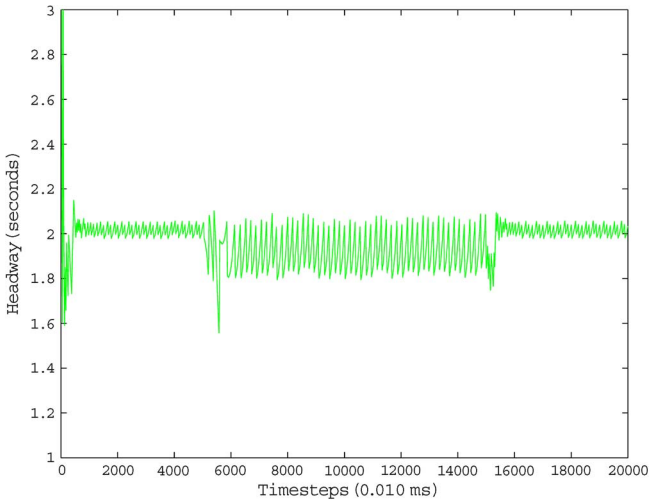

Fig. 8. Headway of the following vehicle for a single-track CACC simulation.

observes a constant velocity (acceleration of 0 m/s^2) of the leader and accordingly selects actions. In reality, at this time, the follower still goes a little faster than the front vehicle (as shown in the velocity profile in Fig. 6). As a result, the follower has a tendency to get closer to the front vehicle, because it uses "no-op" actions, although it should still be braking for a small amount of time.

The CACC behavior obtained is also interesting when looking at the acceleration of the follower. Indeed, Fig. 7 shows that CACC does not need to use much harder braking than the leader (around -3 m/s^2). This behavior is interesting, because it shows that there is no amplification of accelerations or decelerations, which would result, with several 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 enables the learning of a better control policy.

C. Improvements and Comparisons

We have also lead experiments on ACC to see how CACC can be compared with it. We have experimented the SG scenario in the context of ACC, where there is no communication and, consequently, no acceleration Acc in the state. In this case, states and actions are simply given as follows: 1) $S = \{Hw, \Delta Hw\}$, and 2) $A = \{B100, G100, NO - OP\}$. The reward function is identical to the previous function presented in Fig. 3. We also used the policy-gradient algorithm estimated through a neural network backpropagation, as was the case with CACC.

TABLE II
HEADWAY AND HEADWAY ERROR VALUES FOR THE SINGLE-TRACK ACC, CACC, AND CACC$^+$ (i.e., CACC WITH REALISTIC SENSORS DELAYS) SIMULATIONS

	ACC	CACC	CACC$^+$
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
RMS Hw Error	0.135	0.086	0.066

Table II compares ACC and CACC by summarizing their performance using the corresponding headway Hw and headway error of their execution on the same SG scenario. This table is based on the root-mean-square (RMS) error of Hw, which is a particularly interesting measure, because it gives more weight to values far from the desired safe distance. As shown in Table II, CACC is generally closer to the desired goal region than ACC. This condition confirms that the acceleration signal was successfully integrated in CACC.

To be closer to real-life values, we have also simulated the learned CACC policy with realistic sensor delays ($^+$). For this simulation, we increased the sensor refresh rates at higher frequencies than the 4 Hz (250 ms) used for CACC. Our intuition was that, with a refresh rate of 10 Hz (100 ms), we would observe a more precise control of our vehicle. This condition was, indeed, confirmed by results, as shown in Table II. As we can see, the amplitude of the oscillations of the headway, velocity, and accelerations of CACC$^+$ are smaller than CACC. Similarly, CACC$^+$ offers better performances at staying within the desired safe distance.

Note that these results have been obtained in "ideal" situations, where the flow traffic is tidy. It would be interesting to simulate untidy situations and see how RL reacts to them. To do so, we have approached this type of situation by an increase of the frequency of SG; the more that the number of SG is higher with a short time between two successive SG, the more that the traffic flow is untidy. Our preliminary results show that, if the time between two successive SG is less than 2 s and the SG is repeated at least three times, then the control becomes completely unstable after the second SG. In this case, the best approach is to take off the cruise control; in this type of intense SG, it is not useful, because humans are more adapted to this sort of situation.

Finally, it is interesting to measure the performances of our controller against other similar autonomous controllers. Unfortunately, this case is difficult to achieve, because our research work is entirely done through simulation, and we have made several assumptions to simplify the problem (e.g., no sensor noise and a simplified communication protocol that does not consider several factors such as packet loss), which would have

to be lifted to compare our controller with other controllers. Moreover, while being precise, our vehicle dynamics model is still an approximation of real vehicular dynamics.

VII. Conclusion

This paper has proposed a novel design approach to obtain an autonomous longitudinal vehicle controller. To achieve this condition, a vehicle architecture with its ACC subsystem has been presented. With this architecture, we have also described the specific requirements for an efficient autonomous vehicle control policy through RL and the simulator in which the learning engine is embedded.

A policy-gradient algorithm estimation has been introduced and has used a backpropagation neural network for achieving the longitudinal control. Then, experimental results, through simulation, have shown that this design approach can result in efficient behavior for CACC.

Much work can still be done to improve the vehicle controller proposed in this paper. First, it is clear that some modifications to the learning process should be made to improve the resulting vehicle-following behavior. Issues related to the oscillatory behavior of our vehicle control policy can be addressed by using continuous actions. This case would require further study to efficiently implement this approach, because it brings additional complexity to the learning process. Once the oscillatory behavior of the RL approach has been addressed, it would be profitable to compare it with a control obtained by the traditional proportional–integral–derivative (PID) controllers.

Some elements with regard to our simulation framework can also be improved, with the ultimate goal of having an even more realistic environment through which we can make our learning experiments. In fact, an important aspect to consider would be to integrate a more accurate simulator for sensory and communication systems. This way, we can eliminate some of our current assumptions, e.g., the absence of sensor and communication noise. This condition would make the learning process more complex, but the resulting environment would be much closer to real-life conditions.

Our controller can also be completed by an autonomous lateral control system. Again, this approach can be done using RL, and a potential solution is to use a reward function in the form of a potential function over the width of a lane, similar to the current force feedback given by the existing lane-keeping assistance system. This reward function will surely direct the driving agent toward learning an adequate lane-change policy. The lateral control system may be completed by a situation assessment for automatic lane-change maneuvers, as proposed by Schubert *et al.* Finally, the integration of an intelligent vehicle coordination system for collaborative decision making can transform our system into a complete DAS.

References

[1] J. Piao and M. McDonald, "Advanced driver assistance systems from autonomous to cooperative approach," *Transp. Rev.*, vol. 28, no. 5, pp. 659–684, Sep. 2008.

[2] *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*, ASTM E2213-03, 2003.

[3] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, May 2008.

[4] P. Fancher, Z. Bareket, and R. Ervin, "Human-centered design of an ACC with braking and forward-crash-warning system," *Vehicle Syst. Dyn.*, vol. 36, no. 2, pp. 203–223, 2001.

[5] D. Augello, "Description of three PROMETHEUS demonstrators having potential safety effects," in *Proc. 13th Int. Tech. Conf. Exp. Safety Vehicles*, 1993, pp. 4–7.

[6] C. Bonnet and H. Fritz, "Fuel consumption reduction experienced by two promote chauffeur trucks in electronic tow bar operation," in *Proc. 7th World Congr. Intell. Transp. Syst.*, 2000.

[7] E. Commission, *CVIS: Cooperative Vehicle Infrastructure Systems*. [Online]. Available: http://www.cvisproject.org/en/home.htm

[8] E. Commission, *COOPERS: Cooperative Systems for Intelligent Road Safety*. [Online]. Available: http://www.coopers-ip.eu/

[9] S. Tsugawa, "An introduction to Demo 2000: The cooperative driving scenario," *IEEE Intell. Syst.*, vol. 15, no. 4, pp. 78–79, Jul. 2000.

[10] K. Petty and W. Mahoney, "Enhancing road weather information through vehicle infrastructure integration," *Transp. Res. Rec.*, no. 2015, pp. 132–140, 2007.

[11] U. Hofmann, A. Rieder, and E. Dickmanns, "Radar and vision data fusion for hybrid adaptive cruise control on highways," *Mach. Vis. Appl.*, vol. 14, no. 1, pp. 42–49, Apr. 2003.

[12] L. Davis, "Effect of adaptive cruise control systems on traffic flow," *Phys. Rev. E: Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 6, p. 066 110, Jun. 2004.

[13] J. E. Naranjo, C. Gonzàlez, J. Reviejo, R. Garcia, and T. de Pedro, "Adaptive fuzzy control for intervehicle gap keeping," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 2, pp. 132–142, Sep. 2003.

[14] J. E. Naranjo, C. Gonzàlez, T. de Pedro, R. Garcia, J. Alonso, M. A. Sotelo, and D. Fernandez, "Autopia architecture for automatic driving and maneuvering," in *Proc. IEEE ITSC*, Toronto, ON, Canada, Sep. 2006, pp. 1220–1225.

[15] J. E. Naranjo, C. Gonzàlez, R. Garcia, and T. de Pedro, "ACC +stop&go maneuvers with throttle and brake fuzzy control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 213–225, Jun. 2006.

[16] L. Xiao and F. Gao, "A comprehensive review of the development of adaptive cruise control systems," *Vehicle Syst. Dyn.*, vol. 48, no. 10, pp. 1167–1192, 2010.

[17] D. De Bruin, J. Kroon, R. Van Klaveren, and M. Nelisse, "Design and test of a cooperative adaptive cruise control system," in *Proc. IEEE Intell. Vehicles Symp.*, 2004, pp. 392–396.

[18] G. Naus, R. Vugts, J. Ploeg, M. Van de Molengraft, and M. Steinbuch, "Towards on-the-road implementation of cooperative adaptive cruise control," in *Proc. 16th World Congr. Exhib. Intell. Transp. Syst. Serv.*, Stockholm, Sweden, 2009.

[19] M. van Eenennaam, W. Wolterink, G. Karagiannis, and G. Heijenk, "Exploring the solution space of beaconing in VANETs," in *Proc. IEEE VNC*, 2010, pp. 1–8.

[20] B. Van Arem, C. Van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 429–436, Dec. 2006.

[21] D. Pomerleau, "Neural network vision for robot driving," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA: MIT Press, 1995.

[22] G. Yu and I. Sethi, "Road following with continuous learning," in *Proc. Intell. Vehicles Symp.*, 1995, pp. 412–417.

[23] F. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Third Quarter, 2009.

[24] S. Oh, J. Lee, and D. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Trans. Veh. Technol.*, vol. 49, no. 3, pp. 997–1005, May 2000.

[25] D. Moriarty and P. Langley, "Distributed learning of lane-selection strategies for traffic management," Daimler-Benz Res. Technol. Center, Palo Alto, CA, Tech. Rep. 98-2, 1998.

[26] J. R. Forbes, "Reinforcement learning for autonomous vehicles," Ph.D. dissertation, Univ. California, Berkeley, 2002.

[27] L. Ng, C. Clark, and J. Huissoon, "Reinforcement learning of dynamic collaborative driving—Part I: Longitudinal adaptive control," *Int. J. Vehicle Inf. Commun. Syst.*, vol. 1, no. 3, pp. 208–228, 2008.

[28] J. Laumonier, "Méthodes d'apprentissage dans le cadre de la coordination multiagent: Application au transport intelligent," Ph.D. dissertation, Univ. Laval, Québec, QC, Canada, 2008.

[29] C. Desjardins, P.-L. Grégoire, J. Laumônier, and B. Chaib-draa, "Architecture and design of a multilayered cooperative cruise control system," in *Proc. SAE World Congr.*, Detroit, MI, Apr. 2007.
[30] C. Desjardins, J. Laumônier, and B. Chaib-draa, "Learning agents for collaborative driving," in *Multiagent Architectures for Traffic and Transportation Engineering*. Hershey, PA: IGI-Global, 2008.
[31] T. Strobel, A. Servel, C. Coue, and T. Tatschke, *Compendium on Sensors—State of the Art of Sensors and Sensor Data Fusion for Automotive Preventive Safety Applications*, PReVENT Consortium, Tech. Rep., 2004. [Online]. Available: http://www.prevent-ip.org/download/deliverables/ProFusion/PR-13400-IPD-040531-v10-Compendium_on_Sensors.pdf
[32] T. Strobel and A. Servel, *Sensor Data Sheets—State of the Art of Sensors and Sensor Data Fusion for Automotive Preventive Safety Applications*, PReVENT Consortium, Tech. Rep., 2004. [Online]. Available: http://www.prevent-ip.org/download/deliverables/ProFusion/PR-13400-IPD-040531-v10-Sensor_Data_Sheets.pdf
[33] W. R. Bishop, *Intelligent Vehicle Technology and Trends*. Norwood, MA: Artech House, 2005.
[34] Y. Günter and H. P. Gromann, "Usage of wireless LAN for inter-vehicle communication," in *Proc. IEEE Conf. Intell. Transp. Syst.*, 2005, pp. 296–301.
[35] P. A. Ehlert, "The agent approach to tactical driving in autonomous vehicle and traffic simulation," M.S. thesis, Knowl. Based Syst. Group, Delft Univ. Technol., Delft, The Netherlands, 2001.
[36] U. Kiencke and L. Nielsen, *Automotive Control Systems: For Engine, Driveline and Vehicle*. New York: Springer-Verlag, 2000.
[37] X. Huppé, "Guidance et commande longitudinale d'un train de voitures adaptées aux conditions routières et climatiques canadiennes," M.S. thesis, Univ. Sherbrooke, Sherbrooke, QC, Canada, 2004.
[38] S. Hallé, "Automated highway systems: Platoons of vehicles viewed as a multiagent system," M.S. thesis, Univ. Laval, Québec, QC, Canada, 2005.
[39] H. Kimura, M. Yamamura, and S. Kobayashi, "Reinforcement learning in POMDPS with function approximation," in *Proc. 14th ICML*, 1997, pp. 152–160.
[40] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proc. IEEE ICRA*, Seoul, Korea, May 2001, pp. 1615–1620.
[41] T. Field, "Policy-gradient learning for motor control," M.S. thesis, Victoria Univ. Wellington, Wellington, New Zealand, 2005.
[42] N. Kohl and P. Stone, "Policy-gradient reinforcement learning for fast quadrupedal locomotion," in *Proc. IEEE ICRA*, New Orleans, LA, May 2004, pp. 2619–2624.
[43] A. El-Fakdi, M. Carreras, and P. Ridao, "Direct gradient-based reinforcement learning for robot behavior learning," in *Proc. 2nd ICINCO*, Sep. 2005, pp. 225–231.
[44] J. Baxter and P. Bartlett, "Infinite-horizon policy-gradient estimation," *J. Artif. Intell. Res.*, vol. 15, no. 4, pp. 319–350, Jul. 2001.
[45] J. Baxter, P. Bartlett, and L. Weaver, "Experiments with infinite-horizon policy-gradient estimation," *J. Artif. Intell. Res.*, vol. 15, no. 1, pp. 351–381, Jul. 2001.
[46] J. Peters and S. Schaal, "Policy-gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. IROS*, Los Alamitos, CA, Oct. 2006, pp. 2219–2225.
[47] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, May 1992.
[48] P. Marbach, "Simulation-based optimization of Markov decision processes," Ph.D. dissertation, Lab. Inf. Decision Syst., Mass. Inst. Technol., Cambridge, MA, Sep., 1998.
[49] H. Kimura, M. Yamamura, and S. Kobayashi, "Reinforcement learning by stochastic hill climbing on discounted reward," in *Proc. 12th ICML*, 1995, pp. 295–303.
[50] P. Bartlett and J. Baxter, "Stochastic optimization of controlled partially observable Markov decision processes," in *Proc. 39th IEEE Conf. Decision Control*, 2000, vol. 1, pp. 124–129.
[51] R. Hallouzi, V. Verdult, H. Hellendorn, P. L. Morsink, and J. Ploeg, "Communication based longitudinal vehicle control using an extended Kalman filter," in *Proc. IFAC Symp. Adv. Automot. Control*, Salerno, Italy, 2004.
[52] D. de Bruin, J. Kroon, R. van Klaveren, and M. Nelisse, "Design and test of a cooperative adaptive cruise control," in *Proc. IEEE Intell. Vehicles Symp.*, Parma, Italy, Jun. 2004, pp. 392–396.
[53] Z. Bareket, P. S. Fancher, H. Peng, K. Lee, and C. A. Assaf, "Methodology for assessing adaptive cruise control behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 3, pp. 123–131, Sep. 2003.
[54] M. Richardson, D. Corrigan, P. King, I. Smith, P. Barber, and K. J. Burnham, "Application of a control system simulation in the automotive industry," in *Proc. IEE Semin. Tools Simul. Model.*, London, U.K., 2000, pp. 8/1–8/13.
[55] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. 16th ICML*, 1999, pp. 278–287.
[56] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.
[57] R. Schubert, K. Schulze, and G. Wanielik, "Situation assessment for automatic lane-change maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 607–616, Sep. 2010.
[58] J. Laumônier and B. Chaib-draa, "Partial local friendQ multiagent learning: Application to team automobile coordination problem," in *Proc. Can. AI*, *Lecture Notes in Artificial Intelligence*, Jun. 2006, pp. 361–372.

Charles Desjardins received the degree in software engineering from Laval University, Quebec City, QC, Canada, in 2006, from which he also received the Master's degree (with honors) in computer science.

His research focused on applying machine learning techniques to the autonomous vehicle control systems problem. Since graduating in 2008, he has been a software engineer with Genetec Inc., a leader in Internet protocol video surveillance software solutions located in Montreal, QC.

Brahim Chaib-draa (SM'03) received the Diploma degree in computer engineering from the Ecole Supérieure d'Electricité, Paris, France, in 1978 and the Ph.D. degree in computer science from the Université du Hainaut-Cambrésis, Valenciennes, France, in 1990.

In 1990, he joined the Department of Computer Science and Software Engineering, Laval University, Quebec City, QC, Canada, where he is currently a Professor and the Group Leader of the Decision for Agents and Multiagent Systems Group. He is the author or a coauthor of several technical publications. He is on the Editorial Boards of *Computational Intelligence* and the *International Journal of Grids and Multiagent Systems*. His research interests include reasoning under uncertainty, machine learning, and multiagent systems.

Dr. Chaib-draa is a member of the Association for Computing Machinery and the Association for the Advancement of Artificial Intelligence and a Senior Member of the IEEE Computer Society. He is on the Editorial Board of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS.