CAR PLATOONS SIMULATED AS A MULTIAGENT SYSTEM

S. Hallé, B. Chaib-draa and J. Laumonier
Département d'informatique et de génie logiciel - Université Laval
Sainte-Foy, QC, Canada, G1K 7P4
e-mail: {halle,chaib}@iad.ift.ulaval.ca

March 7, 2003

Abstract

Collaborative driving is an important sub-component of Intelligent Transportation Systems ITS and it strives to create vehicles that are able to cooperate in order to navigate through urban traffic by using communications. In this paper, we address this issue by putting emphasis on the simulation of a platoon of cars considered as more or less autonomous software agents. To do that, we propose a hierarchical architecture based on three layers (guidance layer, management layer and traffic control layer) which can be used for simulating a centralized platoon (where a head vehicle-agent coordinates other vehicle-agents by applying its coordination rule) or a decentralized platoon (where the platoon is considered as a team of vehicle-agents trying to maintain the platoon). Such hierarchical architecture is sustained by a simulator that we describe in details. Finally we present our first results concerning the first step of our project and which only focuses on the first level (autonomous longitudinal control) where only the relative distance and speed of the cars are actively controlled.

Keywords
Automated Highway System, Intelligent Transport System, Collaborative Driving, Multiagent Driving Simulation

1 Introduction

Traffic volume is increasing everywhere in the world. To address that, we generally build more and more roads in order to meet the increasing traffic volume. In fact, simply adding new roads is no longer the best solution because of the limited land areas. An alternative is to develop techniques that increase existing roads capacity and existing transportation systems capacity [GL00]. This policy focuses on building fewer lane-miles, while investing in Intelligent Transportation Systems (ITS) infrastructure. It is shown that ITS may provide potential capacity improvements as high as 20 percent [Sto01]. ITS can be seen as a complex set of technologies that are derived from information and computer technologies, as well as applied to transport infrastructure and vehicles [LL02]. We can cite as ITS components: advanced/transportation management, advanced transportation information system, and commercial vehicle operations. Among these components, there are sub-components such as automobile collision avoidance and electronic guidance system. These sub-components are generally sustained by some individual technologies as: electronic sensors, wire and wireless communications, computer software and hardware, GPS, GIS, etc. The main objectives of ITS include: reduce environmental impacts, enhance safety, reduce congestion, etc.

Collaborative driving is an important sub-component of ITS and it strives to create vehicles that are able to cooperate in order to navigate through urban traffic by using communications. Simulation of driving agents as part of an ITS will enable us to develop and test this technology, while making appropriate amelioration to it. As canadian specific highways, climat and laws will be simulated, our simulation results will help us to prove the benefits of ITS for canadian roads. In this paper, we address this issue by putting emphasis on the simulation of a platoon of cars considered as more or less autonomous agents.

2 Platoons of Collaborative vehicles

Collaborative driving is a research domain that aims to create automated vehicles which collaborate in order to navigate through traffic. In this sort of driving, one generally form *a platoon*, that is a group of vehicles whose actions on the road are coordinated by the means of communications. The first vehicle of a platoon is called the platoon leader and its role is to manage the platoon and guide it on the road. Our work comes within this framework and is a part of the Auto21 project [Aut03], a member of the Canadian Networks of Centres of Excellence, studying the automobile of the 21st century within three levels of system functionality, examined in parallel. In the first level (autonomous longitudinal control), only the relative distance and velocity of the cars will be actively controlled in a type of generalized and distributed "cruise control system", although drivers will still steer their vehicles.

In the second level of complexity (semi-autonomous longitudinal-lateral control), the relative lateral and longitudinal motion of each vehicle, relative to the one preceding it, will be autonomously controlled, all the way up to the first "lead car", in a form of generalized car-train with a specially equipped lead car and trained driver. Many such lead cars would co-exist in a given urban center, each with its own generic destination, much like a conventional train or bus, but with the added freedom of "getting off the train with your car".

In the level-3 version (fully autonomous longitudinal-lateral control), the addition of cooperative steering, using the road and the telematic infrastructure as a guide for absolute motion control, will provide autonomous road-following capabilities. All three levels of functionality will be addressed during the conceptual design phase. This hierarchical decomposition will define the various achievement levels that will guide the evolution of the project. This will guarantee that a functional system will result from the long-term project.

This paper presents our preliminary ideas on how we want to address the cooperative driving in the context of this project. For that, we have taken the road which consists to view the collaborative driving as a group of agents more or less autonomous which try to drive without collision while communicating with each other. To achieve such a system, we propose a hierarchical architecture that we now describe in details.

3 Hierarchical Architecture for Collaborative Driving

3.1 ITS Components

Our architecture was inspired by Tsugawa's architecture [TST00] and other ideas coming mainly from the PATH's project [Pro03]. The resulting architecture has three major layers: guidance layer, management layer and traffic control layer, as indicated in figure 1.

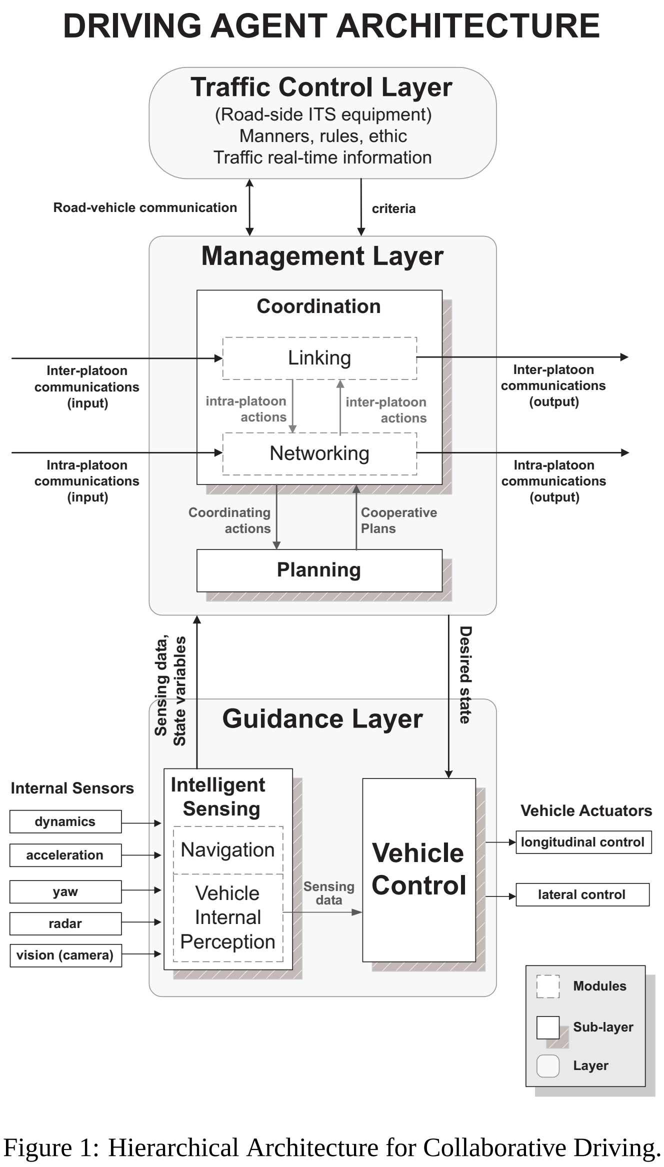

Figure 1: Hierarchical Architecture for Collaborative Driving.

The *guidance layer* has as functions to sense the conditions and states ahead and around the vehicle and to activate the longitudinal or the lateral actuators. For the sensing systems, inputs come from sensors for speed, acceleration, raw rate, machine vision, etc. This layer also outputs sensing data and vehicles state variables to the vehicle guidance layer and then receives steering and vehicles velocity commands from the same guidance layer. These considerations have lead us to divide this layer in *intelligent sensing* and *vehicle control* sub-layers as depicted in figure 1.

The *management layer* determines the movement of each vehicle under the cooperative driving constraints with the data from (a) the guidance layer, (b) neighboring vehicles through the inter-vehicle communication, (c) the leader vehicle through the specific inter-vehicle communication leader-to-vehicles, (d) the traffic control layer through the road-vehicle communication. To determine the movement of each vehicle under the cooperative constraints, this layer needs to reason on the place of the vehicle in the platoon when this platoon remains the same (intra-platoon coordination), and its place in a new platoon when this platoon changes (inter-platoons coordination). The first type of coordination is handled by the *networking* sub-layer and the second by the *linking* sub-layer. Generally, the task of the *linking* sub-layer is to communicate with the traffic control layer to receive suggestions on actions to perform. Resulting from this suggestion the agent's linking module will try to coordinate inter-platoon actions like: join, split and lane-change. This layer is also responsible of maintaining a safe inter-platoon distance which will also define a desired velocity and inter-vehicle spacing for platoon members. This will be maintained using the *networking module*, which is responsible of the intra-platoon coordination and thus, the platoon formation. Finally, the management layer should also maintain a platoon formation plan, a task which is devoted to the *planning* sub-layer.

The *traffic control layer* is a roads-side system composed of infrastructure equipments like sign boards, traffic signals and the road-vehicle communications as well as a logical part including: social laws, social rules, weather-manners and other ethics (more specific to Canada), etc.

The implementation of those layers will be done using the two major architectures we are studying for the platoon grouped driving agent system, which are a centralized and a decentralized architecture.

3.2 How It can be used for Platoons

3.2.1 Centralized Platoons

First, centralized architectures for a driving agent system will be presented. A centralized architecture can be defined either for the multi-agent interactions within a platoon or for the interactions between each platoons. Centralized architectures imply that a head agent will coordinate other agents by applying its coordination rules. As shown in figure 1, the intra-platoon coordination is done using the *networking module*, while the *linking module* is used for inter-platoon coordination, in an architecture inspired by [TST00]. In this way, each coordination level can be implemented with a different model without disturbing the other.

At the platoon level: The focus is on the intra-platoon coordination, which is centralized through the platoon's leader, representing the head vehicle. Thus, the leader is the one who plans for the whole platoon and then prescribes inter-vehicle spacing for every member of its platoon. In this centralized model, split and merge task within a platoon are handled by the leader's networking sub-layers which commands inter-vehicles spacing to create or close a gap between vehicles. In this way, the merge and split tasks are "transparent" to the follower agents that do not have direct contacts with each others but only with the leader. The leader agent will also try to define behaviors for each platoon member in collaboration with the *networking* sub-layer. Such relations between behaviors and the networking layer is useful to coordinate split and merge tasks or plans that would be more specific to a vehicle, in a peer to peer interaction. At last, the task of a follower consists in maintaining a prescribed constant spacing from the preceding vehicle.

At the Highway System level: The centralization of inter-platoon coordination is done using a road-side *traffic control layer* as shown on top of figure 2, which was inspired by a road-side system used in the PATH project [Ban21]. This controller is in charge of a specific part of the highway where it will regulate its traffic. Thus, it receives real-time information about vehicles density and velocity profile on the road and determines the entry flow rate for a specific neighborhood and guides platoons' velocity and maximum platoons' size. In the figure 2 representation, platoon 1 (P1) and 2 (P2) send informations about their formation to the traffic controller using their *link layer*, in an asynchronous call. After it's deliberation, the traffic control layer will send back a guide for the platoon velocity or a task command, which is explained next. Thus, the traffic control also determines the required actions by vehicles (when applicable) at every lane in the form of lane change, merging or splitting task directions. When the traffic control layer comes to a consensus, i.e., an agreement on a specific task to be executed, it has to coordinate on that task. In the case of figure 2, the traffic layer will command a higher velocity to the front (L1, F1, F2) of platoon 1 and a lower velocity to the back (F3, F4) , creating a merging space. This is followed by a command to platoon 2 to steer right until it is aligned with follower 2.

3.2.2 Decentralized Platoons

As an alternative to the previous approach, we consider a decentralized approach where the definition of modules' roles presented in the introduction of section 3.2.1 remains the same but their implementation is different for a decentralized coordination. Thus, the definition of a leader is pretty much absent and the existence of a *traffic control layer* is not required.

At the platoon level: In a decentralized platoon perspective, team-work theories might be used to maintain the platoon while giving each agent the same autonomy. The theory on multi-agents teamwork, explained in [PT02b], can be applied to our

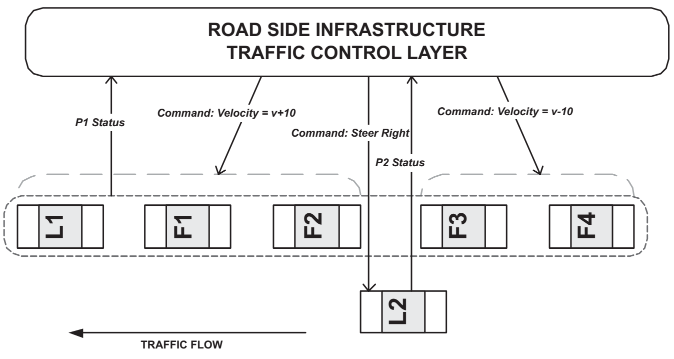

Figure 2: Centralized Decision Making Using a Traffic Control Layer.

platoon, where it is considered as a team and vehicle drivers as agents. Using this approach, the leader is seen as the team captain and the followers are assigned roles within the platoon. Each vehicle keeps its autonomy as there are local and team goals, but the platoon benefits from everyone's knowledge and capabilities. Tambe's COM-MTDP model [PT02a] presents a coordination architecture to support cooperative plans within platoon members allowing thus to maintain a flexible, persistent platoon formation. Theory on cooperative planning is defined in [GK96] where the definition of a Full Shared Plan (FSP) can be used to plan a platoon formation that will respect a given inter-vehicle spacing and velocity. But its most useful application is to coordinate merge and split tasks that require different vehicles' expertise, defined by their position in the platoon. Inspired by a distributed model, well known in networking, the teamwork for agents has the same benefits on a computing level, in addition to a powerful knowledge sharing.

At the Highway System level: To create a decentralized architecture for our inter-platoon coordinations, we plan to consider mobile agents. Those agents create a merging interface through each vehicle's linking layer and optimize the communications by minimizing the round trips usually necessary to come to mutual intentions between platoons. Therefore, a mobile agent acting on the behalf of a platoon would coordinate itself with others using less bandwidth, since its coordination is done locally. In our application, a mobile agent carries information on the leader and its platoon formation relative to the current type of coordination. The mobile agent coordination model we are considering is a *blackboard* model for indirect coordinations [GC97]. Applying this model to collaborative driving, the previous organisation in neighborhoods will be kept. Each neighborhood has its own blackboard which resides on the biggest, previously elected, platoon. Platoons' representatives will locally coordinate themselves on this platoon blackboard in an asynchronous way. This is presented in figure 3 where a mobile agent representing platoon 2 (M.A.2) is sent to the blackboard situated on platoon 1. There, it coordinates itself with platoon 1 representative (M.A.1) using information from other agents that passed by, stored on the blackboard . From their coordination, M.A.1 will command it's platoon velocity changes to create a merging space. Then, M.A.2 comes back to platoon 2 and commands to steer right when merging space is available. Thus, mobile agents benefit our communications as they can coordinate themselves with available agents and leave information for upcoming ones on the blackboard without having to wait for others.

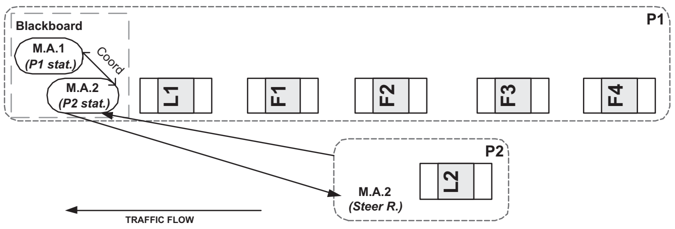

Figure 3: Decentralized Decision Making Using Mobile Agents.

4 Simulation

In the previous section, the architecture for a vehicle driving agent was defined and going from theory to practice, it's implementation in a simulated environment will be defined in this section. First the simulator's engine, around which vehicle shapes and dynamics are simulated will be explained. Then the agent framework used to implement our driving agent architecture will be defined in section 4.2 with the tools and developing guidelines it offers.

4.1 Automated Highway System Simulator

We now present the Automated Highway System (AHS) simulator developed as part of our multiagent approach. Similar to traffic simulators like Carnegie Mellon's SHIVA [SHT97], our simulator, called HESTIA (a screen shot of it is presented in figure 4), aims a lower level of vehicle simulation, as its main purpose is to create an environment for Intelligent Transport Systems (ITS) developing and testing. To do so, it simulates a highway environment, with vehicles represented as 3D shapes, which are using simulated dynamics and sensors to retrieve information from the environment, as shown in figure 5. Here, the driver agent presented in section 3 uses sensors, controller and a communication module to interact with the simulated environment. Shapes, data and dynamics modules are also used to complete a *vehicle package* that interacts with other vehicles in the 3D environment. Before we explain the simulator's components in details in the following subsections, let us describe its simulator's modelling language.

Agents' environment: The agents' environment was built from JAVA 3D^{TM}technology, which brings a 3D environment which is easy to use and in that our agents can easily evolve. First, the 3D environment offers a very pleasant user interface that is completely manageable. 3D objects can then be dynamically altered, assigned different behaviors and picked on their contour in real-time. Using different object behaviors we can easily represent different type of road conditions and their resulting friction, as well as an object's behavior on such events as a collision.

Sensors: Using the previous environment, JAVA 3D^{TM} enables us to implement sensors by using their *picking* tool on 3D objects. This tool can provide informations about a vehicle's

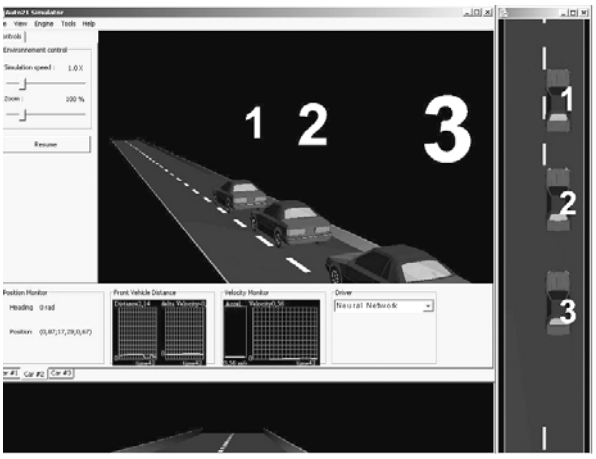

Figure 4: Vehicles platoon formation in HESTIA 3D Simulator.

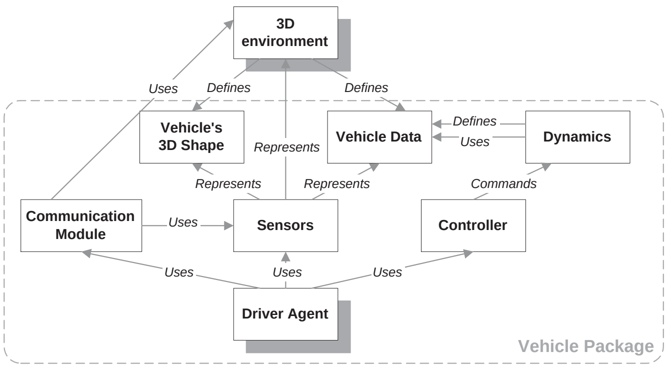

Figure 5: HESTIA Vehicle Simulation Environment.

surrounding objects' distances detected in the 3D environment. Since vehicles have a *pickable* behavior, other vehicles' sensors detect them and can then access available information about the sensed vehicle. Those sensors return a very precise information about the sensed object's shape which can then be altered considering the sensor's specifications. Notice that the simulator also plays the role of an internal-sensor by providing information based on the vehicle's specifications and dynamic information, as the engine status for example. These informations are taken from figure 5's *Vehicle Data* box .

Dynamics: The dynamics engine simulates the vehicle's dynamics using data on the vehicle's engine, shape and tires, from the *Vehicle Data* box and the road conditions from the *3D environment* box. These informations are transformed, using the current steering and throttle level, in angular forces applied on the vehicle. Putting all those forces together gives us the current acceleration, speed, heading and position. The latter heading and position are sent to the 3D engine to change the vehicle's appearance.

4.2 Simulator's Multi-Agent Framework

Over the previously defined simulator is the agent framework, which was also developed in JAVA^{TM}. This framework is based on a set of tools required by agents to evolve in an automated highway environment. We already defined the tools to *sense* on the environment, as part of the 3D engine, and to *act* on the environment, as part of the dynamics engine, wihch are the two principal properties that define an agent, as presented in [RN95]. Thus, starting from a simulator that responds to an agent's basic needs, the framework was completed using JACK Intelligent Agents^{TM}, a JAVA based programming language that extends agent-oriented concepts. To complete this strong framework, our own communications module was developed , so we could control and simulate different communication devices as used in platoons (radio or others). The global view of this framework is presented in figure 6 where the agent architecture from section 3 is implemented in accordance with the following agent framework, using an interface with the environment formed by sensors and controller.

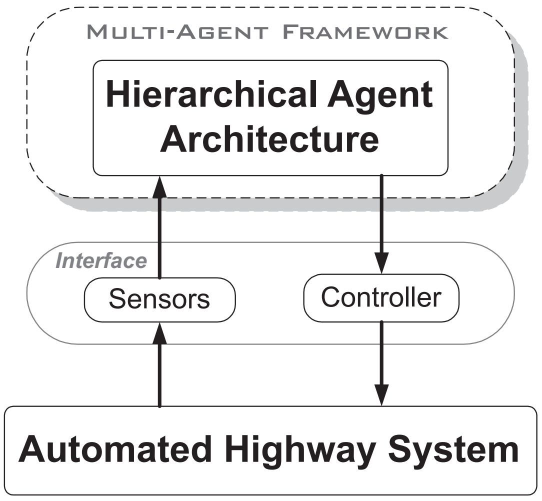

Figure 6: Global view on the simulator's multi-agent framework.

4.2.1 Agent Modelling Language

The agent modelling language used as part of this AHS simulation framework is mainly based on JACK, a programming language related to the BDI agent model [RG95]. Useful tools from IBM's ABLE have been added to this framework forming thus a powerful building toolkit. This toolkit has been completed with an inter-agent communication module which is related to our AHS simulator.

JACK Intelligent Agents^{TM}extends six major agent-oriented concepts: agents, capabilities, events, plans, knowledge bases, and resource and concurrency management. It even includes a model for agent teams programming. All of these concepts are implemented using five main classes that are extended for specific usage: plan, event, beliefset, capability, and task manager. The first three will be described, making reference to their usage in our simulator.

A plan is used to define a sequence of actions that an agent can do when an event occurs. In our case, plans will be used in the *Planning sub-layer* of figure 1, mostly for tasks like merging, splitting and lane changes. Different plans will then be declared for different event types, and plans handling the same events will be differentiated using *relevance* and *context* criterions defined in JACK language's documentation. Moreover, plans are closely related to event types and definitions, since they occur when an event arise. JACK offers a number of event models for different needs, represented as: *internal stimuli*, *external stimuli*, and *motivations*. These events will then be used in multi-agents communication, mostly for intra-platoon coordination. In addition events will arise from the lower levels of our agent architecture in figure 1, as the sensors will inform planners about sensed vehicles or situations. Then, to determine an agent's environmental context, Jack provides a data structure called a *Beliefset*, which enables our agents to collect information on traffic, platoon formation, and platoon members and to infer on it.

In addition to the BDI extension, JACK provides an extension to support Team Oriented programming, called SimpleTeam. Team Oriented programming is a sort of agent oriented programming where agent collaboration is specified from the abstract viewpoint of the group as a whole [Cob02]. This extension supports shared plans and enables us to easily implement the theory on platoons represented as teams, presented in section 3.2.2.

ABLE, the complementary toolkit to our agent framework, was created for agent building purpose and it offers a variety of agent's artificial intelligence tools in the way of Java Beans. For the moment, neural network tools from Able librairy have been used as part of our agent guidance layer to create a learning brake and accelerator controller presented in section 5.2.

4.2.2 Inter-Agents communications

The inter-agents communications are related to the environment simulator and to the multi-agent framework. Thus, to simulate communications within an AHS environment, we should take into account: the environment properties, the agents' vehicle properties and the communication system. This is done by the communication module in collaboration with the communication manager which is in charge of dispatching the messages (as shown in figure 7). The communication module, part of the simulated vehicle shown in figure 5, is used by agents to communicate between them. These communications are done in order to coordinate themselves or to share results or information. The messages are then sent by the agent's communication module to the communication manager which dispatches them using the specification of the currently simulated communication system. Depending on the these specifications and the message type launched by the agent, the manager determines the agent(s) that is(are) in the right to receive the message, and send it within an appropriate simulated time. For example, a radio communication simulator will determine agents in the radius of the current broadcast, using the 3D engine, and will send the message to those agents in a radio communication interval.

5 First Experiments

We are currently in the first phase of this project devoted to collaborative driving and this phase focuses only on the first level (autonomous longitudinal control) where only the relative distance and velocity of the cars are actively controlled. The longitudinal control consists in finding the acceleration or braking of a vehicle, called the following vehicle, in order

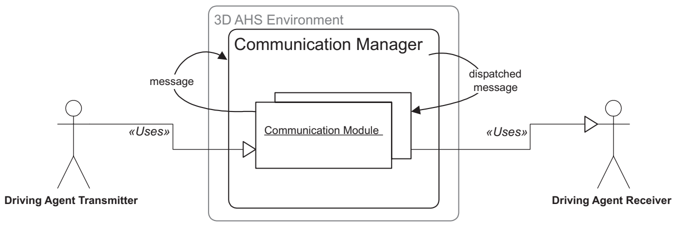

Figure 7: Agent Communications Simulation.

that it follows another vehicle, called the preceding vehicle, with security and without delay. To achieve this first phase, the guidance layer of our arhcitecture, presented in figure 1, was implemented using Tsugawa's longitudinal control function [TKT+01]. More precisely, we have experimented two versions: The first one is the direct implementation of the function, whereas the second is the evaluation of the function by a neural network. Both methods use the simulated laser sensor as external sensor to evaluate the inter-vehicle distance and the difference of velocity between both vehicles. Moreover, an internal sensor is used to get the current velocity.

5.1 Using a Longitudinal Control Function

A longitudinal control function calculates the commanded velocity of the following vehicle according to the current velocity of the preceding one, the relative velocity and the inter-vehicle distance. It includes also a minimal inter-vehicle distance. The commanded velocity is given by the following :

$$v_c = v_p + k_1(-\delta_v) + k_2(L_r - L_m)$$

with
v_c : the commanded velocity of the following vehicle.
v_p : the current velocity of the preceding vehicle.
$k_1 = \frac{m_1|L_r - L_m|}{|L_r|}$, and $k_2 = m_2 k_1$
δ_v : the relative velocity between the following vehicle and the preceding vehicle. If d_v is positive, the both vehicles are getting closer.
L_r : the minimal inter-vehicle distance.
L_m : the measured inter-vehicle distance.
m_1, m_2 : control gains. The tests show that their sign must be the opposites to give correct results. One of the best solution is $m_1 = 1$ and $m_2 = -1$. Some more precise tests must be performed to have a better solution.

The figure 8 shows the result of such a function: the commanded velocity is calculated according to the relative velocity and the inter-vehicle distance. In this example, we assume that the preceding vehicle velocity is equal to zero. It shows that as the distance with the precedent vehicle gets higher, the acceleration of the current vehicle gets higher. Moreover, the closest the following vehicle gets with the precedent one, the more important is its braking. We can see in the figure 8, that the commanded velocity is zero when the distance is equal to the minimal inter-vehicle distance (one meter in this example). This means the following vehicle velocity must be the same as the preceding vehicle to keep the minimal distance.

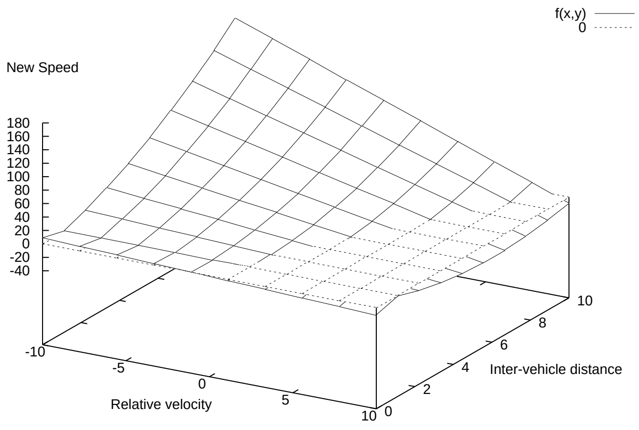

Figure 8: The new velocity evaluation function.

However, since the agent can only act on the vehicle's acceleration and braking, the new acceleration must be calculated by the velocity derivative. Obviously, as we can see in figure 8, this function calculates a theoretical commanded velocity necessary to keep the minimal inter-vehicle distance. That is why, in some case, the velocity seems to be too high but this function is evaluated several times per second and the next calculated velocity will be smaller.

5.2 Evaluation of the Longitudinal Control Function by a Neural Network

The second implementation is based on a neural network which evaluates the previous function. The learning and testing data have been generated with the following parameters : from 0 to 25 m/s with a step of 0.5 m/s for the velocity of the following vehicle, from -10 to 10 m/s with a step of 0.5 m/s for the difference of velocity and from 0 to 10 meters with a step of 0.5 m for the inter-vehicle distance.

Although the neural network is just a function evaluation, it reacts appropriately to the situations. However, since it is an evaluation, it will not work as well as the function in the extreme cases. Despite this weakness, one strength of the neural network is the capacity to learn from real situation.

The longitudinal control has been implemented with an evaluation function based on a robotic control and a neural network trained with simulation data. These approaches gave very satisfactory results for following vehicles, part of the centralized platoon approach. The next step will focus on the coordination, moving through a decentralization of the platoon.

6 Conclusion

This paper presented a complet simulation environment based on agents driving vehicles in platoon formation. This kind of simulator offers many advantages, both on a developing side and a marketing side. First, simulations enable us to test implementation of different driving agents models without using real cars and roads. Different simulation scenarios can be easily tested and are totaly mangeable from vehicle types to road conditions. The results it gives are very reliable and help us to choose from different theories and to easily implement learning algorithms. On a marketing point of view, the simulations of an ITS infrastructure based on the canadian highway system help us to prove the benefits of this technology on the traffic flow, car accidents, highway capacity and more. Specific to the canadian climat, the environment simulated in HESTIA will demonstrate the application of ITS to snowy roads and the type of infrastructure it needs to be reliable.

As we presented different centralized and decentralized models for cooperative driving in platoon formations, there is a lot of work to come to implement and test the best theories. The first step will be intra-platoon coordination, where a centralized leader will be compared to team formations. The many simulations to come will tell us which degree of autonomy should be given to each vehicle member of a platoon and the best strategies to take on merge and split platoon tasks. Many scenarios will be taken into account using the simulator, to make sure our agents will react well to every eventuality.

7 Acknowledgments

This work has been carried out as part of the Automobile of the 21st Century (Auto21) project supported by the Government of Canada through the Networks of Centres of Excellence (NCE).

References

[Aut03] Auto'21. http://www.damas.ift.ulaval.ca /projets/auto21/fr/index.html. Technical report, 2003.

[Ban21] S. Vahdati Bana. Coordinating automated vehicles via communication. Ucb-its-prr-2001-20, Univeristy of California, Berkeley, September 20021.

[Cob02] M. Coburn. Simpleteam technical brief. Technical report, Agent Oriented Software, February 2002.

[GC97] F. Zambonelli G. Cabri, L. Leonardi. Coordination in mobile agent applications. Technical report no. dsi-97-24, Dipartimento di Scienze dell'Ingegneria Universit di Modena, October 1997.

[GK96] B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

[GL00] S. Ghosh and T. Lee. *Intelligent Transportation Systems: New Principles and Architectures*. CRC Press, 2000.

[LL02] W. Lin and H. Leung. Comparison of vehicle detectors used in intelligent transportation systems. Technical report, NCE, Auto'21, 2002.

[Pro03] PATH Project. http://www.path.eecs.berkeley.edu/. Technical report, 2003.

[PT02a] D. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of AI Research (JAIR)*, 16:389–423, 2002.

[PT02b] D. Pynadath and M. Tambe. Team coordination among distributed agents: Analyzing key teamwork theories and models. In *Proceedings of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems*, 2002.

[RG95] A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[RN95] S. J. Russell and P. Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 1st edition, January 1995.

[SHT97] R. Sukthankar, J. Hancock, and C. Thorpe. Tactical-level simulation for intelligent transportation systems. *Journal on Mathematical and Computer Modeling*, 1997. (To appear).

[Sto01] R. R. Stough. *Intelligent Transportation Systems: Cases and Policies*. Edward Elgar Pub. LTD: Cheltenham, UK/Northampton, MA, USA, 2001.

[TKT$^+$01] S. Tsugawa, S. Kato, K. Tokuda, T. Matsui, and H. Fujii. A copperative driving system with automated vehiclues and inter-vehicle communications in demo 2000. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2001.

[TST00] S. Kato T. Sakaguchi, A. Uno and S. Tsugawa. Cooperative driving of automated vehicles with intervehicle communications. In *Procs. IEEE Intelligent Vehicles Symposium 2000*, pages 516 – 521, Dearbsorn, MI, USA, October 2000.