View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

Design of a Hybrid Controller for Autonomous Vehicles Driving on Automated Highways

Alain Girault

To cite this version:

Alain Girault. Design of a Hybrid Controller for Autonomous Vehicles Driving on Automated Highways. RR-4286, INRIA. 2001. inria-00072301

HAL Id: inria-00072301

https://hal.inria.fr/inria-00072301

Submitted on 23 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Design of a Hybrid Controller for Autonomous Vehicles Driving on Automated Highways

Alain Girault

N° 4286

THÈME 4

ISSN 0249-6399 ISRN INRIA/RR--4286--FR+ENG

RHÔNE-ALPES

Design of a Hybrid Controller for Autonomous Vehicles Driving on Automated Highways

Alain Girault

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet BIP

Rapport de recherche n° 4286 — — 31 pages

Abstract: In this paper we address the problem of the hybrid control of autonomous vehicles driving on automated highways. Vehicles are autonomous, so they do not communicate with each other nor with the infrastructure. Two problems have to be dealt with: A vehicle driving in a single-lane highway must never collide with its leading vehicle; and a vehicle entering the highway at a designated entry junction must be able to merge from the merging lane to the main lane, again without any collision. To solve these problems we propose to equip each vehicle with a hybrid controller, consisting of several continuous control laws embedded inside a finite state automaton. The automaton specifies when a given vehicle must enter the highway, merge into the main lane, yield to other vehicles, exit from the highway, and so on. The continuous control laws specify what acceleration the vehicle must have in order to avoid collisions with nearby vehicles. By carefully designing these control laws and the conditions guarding the automaton transitions, we are able to demonstrate three important results. Our first result states the initial conditions guaranteeing that a following vehicle never collides with its leading vehicle. Our second result extends the first one to a lane of autonomous vehicles. Our third result states that if all the vehicles are equipped with our hybrid controller, then no collision can ever occur and all vehicles either merge successfully or are forced to drop out when they reach the end of their merging lane. Finally we show the outcome of a highway microsimulation conducted with the SHIFT hybrid simulator developed at UC Berkeley by the PATH group.

Key-words: Autonomous vehicle, automated highway, intelligent cruise control, hybrid controller.

This work started when the author was a postdoc researcher at UC Berkeley and California PATH.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Conception d'un contrôleur hybride pour véhicules autonomes roulant sur des autoroutes automatiques

Résumé : Nous nous intéressons dans cet article au problème du contrôle hybride de véhicules autonomes roulant sur des autoroutes automatisées. Les véhicules sont autonomes, c'est-à-dire qu'ils ne communiquent ni les uns avec les autres ni avec l'infrastructure. Deux problèmes doivent être résolus : un véhicule roulant sur une voie simple d'autoroute ne doit jamais entrer en collision avec son véhicule leader ; de plus un véhicule arrivant sur l'autoroute à une des entrées doit être capable de s'insérer depuis sa voie d'accélération vers la voie principale, là encore sans qu'il y ait de collision. Afin de résoudre ces deux problèmes, nous proposons d'équiper chaque véhicule d'un contrôleur hybride, constitué de plusieurs lois de commande continues enfouies dans un automate d'états fini. L'automate indique à quel moment un véhicule donné doit entrer sur l'autoroute, s'insérer dans la voie principale, quitter l'autoroute... Les lois de commande continues indiquent quelle accélération le véhicule doit avoir pour éviter les collisions avec ses véhicules voisins. En spécifiant soigneusement ces lois de commande continues ainsi que les conditions gardant les transition de l'automate, nous démontrons trois résultats importants. Notre premier résultat établit les conditions initiales sous lesquelles un véhicule suiveur n'entre jamais en collision avec son véhicule leader. Notre second résultat étend le premier à une file de véhicules autonomes. Notre troisième résultat établit que si tous les véhicules sont équipés de notre contrôleur hybride, alors aucune collision ne peut se produire sur l'autoroute, et que tous les véhicules ou bien réussissent à s'insérer avec succès, ou bien sont obligés de quitter la voie d'accélération quand ils en atteignent la fin. Enfin, nous présentons les résultats de la microsimulation d'une autoroute, programmée avec le simulateur hybride SHIFT développé par l'équipe PATH de l'Université de Berkeley.

Mots-clés : Véhicule autonome, route automatisée, cruise control intelligent, contrôleur hybride.

1 Introduction

1.1 The Need for Automated Traffic Control

Passenger safety and traffic congestion are growing problems in several urban corridors. Two approaches can be used to alleviate these problems: construction of more lanes on congested highways or use of emerging technologies for more automated traffic control. The first approach, constructing more lanes or highways, is rapidly becoming untenable because of lack of right of ways, complexity of highway design and layout, costs and environmental considerations. Hence several metropolitan transportation agencies are turning towards higher levels of automation technologies to address these problems.

Different automation strategies have been proposed in [23]. Typically they are classified according to their distribution of intelligence attributes. Here, "intelligence" refers to the techniques used to govern the flow of traffic. At one end of the spectrum, all automation intelligence is concentrated within individual vehicles—leading to the *autonomous* scenario. In the autonomous case, vehicles incorporate enhanced sensing, computation and actuation technologies to improve vehicle control functions such as lane keeping, headway keeping and velocity tracking. This system is known as AICC, acronym for "Autonomous Intelligent Cruise Control". At the next level, individual vehicles cooperate by communicating selected state information to each other—leading to the *cooperative* scenario. In the cooperative case, vehicles can coordinate their activities even when they are not within each other's sensor ranges. The communication may be directly from vehicle to vehicle or it may be mediated by the transportation infrastructure. Finally, vehicles may form tightly coordinated platoons by communicating more detailed state information at a greater frequency—leading to the *platooning* scenario.

In this paper we are focusing on the autonomous case. Vehicles are equipped with a controller and with sensors that give information about the nearby vehicles both in the main lane and in the neighbouring lanes: gap, relative speed,... Based on the data provided by these sensors, the controller of each vehicle must decide whether to brake, accelerate, merge into the traffic, or yield, while avoiding accidents. All decisions must be taken *locally*. In particular, there is no coordination layer like in the cooperative scenario. The purpose of this article is to design a controller that achieves this goal.

1.2 Hybrid Control

Two aspects must be dealt with by the vehicle controller:

- The continuous control of the vehicle, which divides into longitudinal and lateral control. The former consists of control laws for accelerating and braking when following a vehicle, when yielding to neighbouring vehicles, and so on. The latter consists of control laws for turning, merging, and so on. The job of the continuous control is to decide *how* to accelerate, brake, turn, merge,...
- The discrete control of the vehicle, whose job is to decide *when* to start accelerating, braking, merging, yielding,... Usually it is specified as a finite state automaton, but many other equivalent notations can also be used.

Because of this intrinsic duality between the continuous part and the discrete part, such controllers are called *hybrid*. It should be possible to design a purely continuous controller, meaning that the differential equations controlling the movements of the vehicle would take care of both how and when to brake, accelerate, turn, merge,... but we think that designing a hybrid controller is easier than designing a purely continuous one.

In this paper we are presenting an hybrid controller for autonomous vehicles under AICC. As a first approach, we only focus on the *longitudinal* control. Yet, we show that our approach is flexible and that our controller can be easily adapted to include also the lateral control. Finally, since we are focusing on the autonomous scenario, we adopt a spacing policy with *constant time headway* for our longitudinal control law, i.e., the time between two successive vehicles must be constant.

Our vehicle model is a very simple one, i.e., acceleration control and two-dimension kinematic. One of our objectives is to simulate a whole highway, with a number of vehicles above 1000 at the same time. Therefore a too complex model would prevent us from running such simulations.

1.3 Related Work

Work related to our problem can be divided into three parts: the longitudinal control of automated vehicle and the asymptotic stability analysis; the merging strategies for an automated vehicle willing to change lane; and the hybrid control of automated vehicles.

1.3.1 On Longitudinal Control

Work on asymptotic stability analysis of dynamic systems with saturation constrains include the work of Ye, Michel, and Hou [35] and of Åström and Brufani [4]. However these works deal with linear and second order dynamic systems, so their results are not transposable to our problem.

Other work on AICC with a constant time headway has been reported by Ioannou, Ahmed-Zaid, and Wuh in [17], and by Daviet and Parent in [10]. In these papers the kinematic model is more complex than ours, including for instance a dynamic model of the engine. The authors perform extensive simulations to tune up the parameters of their control laws. However, no stability results are given.

In [16], Ioannou and Chien also study the constant time headway policy. The authors propose a longitudinal control law for an AICC system, where the desired inter-vehicle distance is of the form $\lambda_2 v_i + \lambda_3$, where λ_2 and λ_3 are constants, and v_i is the following vehicle speed. Then they compare their automatic controller with several human control models and show that it suppresses totally the accordion effect [30], leads to smoother traffic flows, and larger traffic flow rates, due to the shorter inter-vehicle safety spacing and the elimination of human delays and large reaction time.

In [18], Ioannou and Xu further study the constant time headway policy. The acceleration of the following vehicle is achieved by controlling both the throttle and the brake. The engine, the transmission, and the drive-train are also included in the model. The first studied system consists of a leading vehicle and a following vehicle. The proposed controller guarantees the stability of a platoon of vehicles. Gains that ensures the asymptotic stability are proposed. Yet, this is valid only in the non-saturated case. When the acceleration is bounded, the authors do not say whether or not the system is still stable. The second studied system consists of a platoon of vehicles, each still separated by a constant time headway. A controller is proposed that guarantees the platoon stability (i.e., no accordion effect), provided that all the vehicles operate around some constant speed with only small variations.

In [33], Swaroop, Hedrick, Chien, and Ioannou design two longitudinal control laws, the first one with a constant spacing objective, and the second one with a constant time headway objective. Their vehicle model is based on [9] and is more complex than ours. They consider a platoon of vehicles driving along a single straight lane, i.e., with no lane changing. The constant spacing control law involves communication between the vehicles (acceleration of the leading vehicle is broadcasted to all other vehicles) and ensures string stability. The author compare the lane capacity vehicles/lane/hour and show, as expected, that the constant spacing law is always better than the constant time headway one.

Finally, there is also a lot of work on the constant spacing policy. See for instance [34, 26] for the most recent articles. Such work takes place in Automated Highway Systems (AHS) where the vehicles travel in platoons. This increases a lot the flow capacity of highways, but on the other hand it requires costly infrastructures, for instance ramps, mettering, communications... In particular, such systems use a coordination layer to coordinate the actions of all the vehicles: merging, exiting, splitting a platoon, joining a platoon...

1.3.2 On Merging Strategies

In [20], Kanaris, Kosmatopoulos, and Ioannou propose a merging strategy involving, for the merging vehicle, first braking or accelerating to make its headway gap large enough; then braking or accelerating in order to make its longitudinal velocity equal to the velocity of the destination lane. The important assumption is that, with the exception of the merging vehicle, the velocities of all the other vehicles in the originating and destination lanes must be *constant*. This is not the case in our study, so we cannot reuse their results.

In [19], Jula, Kosmatopoulos, and Ioannou address the problem of collision avoidance during lane-changing maneuvers. Like [20], they assume that, with the exception of the merging vehicle, the velocities of all the other vehicles in the originating and destination lanes must be constant.

In [28], Ran, Leight, and Chang present an AHS system, consisting of a single main lane, and an entry lane made of an entering ramp followed by a merging portion. Their merging logic involves targeting a gap, adjusting to it, and merging. Gaps are created on the main lane following orders issued by the infrastructure. If no gap is available, the vehicle stops on the entry ramp, and then proceeds when a gap is finally available.

In [22], Lu and Hedrick address the problem of longitudinal control for automated vehicle merging in an automated highway system. They generate a reference trajectory for the merging vehicle based on the position and speed of the vehicles in the main lane, so that the merging vehicle will form a platoon with them. This trajectory is based on information provided by the coordination layer, and further requires the aimed the platoon in the main lane to split in order to create a suitable gap for the merging vehicle.

1.3.3 On Hybrid Control

The AHS system presented by Ran, Leight, and Chang in [28] has a hybrid logic. The discrete logic part deals with the communications between the infrastructure and the automated vehicles. For instance, gaps are created on the main lane following orders issued by the infrastructure. The continuous logic part deals with platoon following, platoon forming, merging. No global correctness property is proved on this hybrid controller.

In [27], Rajamani, Tan, Law, and Zhang describe a longitudinal and a lateral control systems for automated vehicles. These two control systems are then integrated in the form of an hybrid system: each vehicle has several supervisors modelled as finite state machines, depending on which maneuver it intends to complete. In the discrete states, the desired acceleration of the vehicle is determined by the appropriate differential equations. Again, no global correctness property is proved on this hybrid controller.

In [14] and [15], Godbole, Lygeros, Sastry, Horowitz and Varaiya describe in details the five-layer architecture developed at PATH for AHS. In particular, the coordination layer communicates and coordinates with peer vehicles and select one maneuver to be executed; the regulation layer executes maneuvers such as join, split, lane change, and so on. The first one is modelled as a finite state machine while the second one is modelled as feedback laws based on linear models. Hence their composition is a hybrid model.

1.4 Paper Outline

The paper is organised as follows. In Section 2 we present briefly hybrid automata and we introduce the language SHIFT developed at UC Berkeley and California PATH. Then, in Section 3, we present a discrete controller that implements a merging and yielding protocol for autonomous vehicles driving on automated highways. In Section 4 we design and study two longitudinal control laws: the goal of the first one is to make the vehicle reach a desired speed in the absence of other vehicles in front of it, while the goal of the second one is to make the vehicle follow a leading vehicle without colliding with it. We also prove the asymptotic stability of the second control law. Then in Section 5 we put together the results of Sections 3 and 4 to design the complete controller hybrid automaton. We prove that if all the vehicles on a highway are equipped with this hybrid controller, collisions can never happen. Finally, in Section 6, we present micro-simulation results that were obtained with our hybrid controller for a highway case study.

2 Hybrid Automata

Hybrid systems are systems where the continuous and discrete dynamics coexist. There has been a lot of research in this field during the past ten years. See for instance the recent special issue [3] or the LNCS "Hybrid Systems Computation and Control" conference series. In this section we present briefly a model commonly used to specify hybrid systems, namely *hybrid automata*. Then we present the SHIFT language for describing and simulating dynamic networks of hybrid automata.

2.1 Definition

There exist several models for describing hybrid systems, one of the most convenient being *hybrid automata*. A hybrid automaton is a finite state automaton with a set of continuous variables. The system spends

time in the states, therefore allowing the continuous variables to evolve, and performs discrete transitions between states. The continuous dynamics associated with each discrete state is typically specified as a set of differential equations. At each discrete transition, there are guards and assignments on the continuous variables. Guards are typically inequalities on the continuous variables. A transition can be fired only if its guard is satisfied. When this happens, the continuous variables are updated according to the transition's assignments. Additionally, a hybrid automaton can have invariants associated with its discrete states, typically specified as inequalities on the continuous variables. Finally, the hybrid state of a hybrid automaton consists both of its discrete state and the current values of its continuous variables.

Figure 1 is a classical example of a hybrid automaton describing a thermostat. It has two discrete states, `on` and `off`, and one continuous variable, x, which is the temperature of the water. In the `on` state the heater is functioning so the temperature increases, while in the `off` state the temperature decreases. The transitions between the `on` and `off` states are discrete, depending whether the temperature hits the lower threshold m or the upper threshold M. Finally, K_1 and K_2 are two constants.

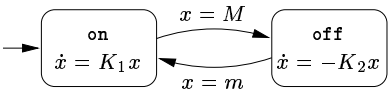

Figure 1: An example of hybrid automaton.

There exist several semantics regarding the way transitions are fired. Because the transition guards and the state invariants are inequalities, the instant when a transition can be fired ranges within an interval. So it can be specified that transitions are *urgent*, in which case a transition must be fired as soon as its guard allows it to; or it can be specified that transitions are *late*, in which case a transition must be delayed until the last moment when the source state invariant still holds; or it can be specified that transitions can be fired at any time within their legal interval.

Finally, over the years, researchers have defined and studied several variants of hybrid automata, depending on the type of the continuous dynamics. For instance in *time automata*, the continuous dynamics are of the form $\dot{x} = 1$. In *linear hybrid automata*, the continuous dynamics are of the form $\dot{x} = k$ where k is a constant. And in *rectangular hybrid automata*, the continuous dynamics are of the form $\dot{x} \in [a, b]$ where a and b are two constants.

2.2 The SHIFT Language

The SHIFT hybrid systems simulation language [11, 31] has been developed at UC Berkeley and California Partners for Advanced Transit and Highways (PATH) partners. It is both a simulation language and a runtime system. It allows the specification of *dynamic* networks of hybrid automata. For this reason, it is ideally suited to automated highway simulations.

A SHIFT program consists of a set of *types*, each describing a generic hybrid automaton. A type has a set of member variables, a set of discrete states, a set of differential and algebraic equations associated with each discrete state, and a set of exported events that are triggered by discrete transitions.

Instances of these types are called *components*: they can be created and destroyed dynamically, and can even synchronise their transitions dynamically with other components. Components can be created at the start of the program, or later by other existing components.

For instance a vehicle source can be specified to create cars with a uniform or random distribution, with an initial speed in a given interval, and so on. Then the dynamic synchronisation mechanism is useful when several vehicles have to interact.

Below is the SHIFT source code of the thermostat shown in Figure 1:

```
type thermostat {
  state
    continuous number x;
    number m := 18;
    number M := 22;
    number K1 := 0.5;
    number K2 := 0.5;
  discrete
    on {x' = K1 * x;},
    off {x' = (-K2) * x;};
  transition
    on -> off {} when x >= M,
    off -> on {} when x <= m;
}
global
  thermostat t := create(thermostat, x:= 15);
```

The keyword `type` indicates a type declaration: here we declare the type `thermostat`. After the keyword `state` are the member variable declarations: `continuous` means that the variable `x` evolves with time, and thus it can appear in the left-hand part of the flow equations. After the keyword `discrete` are the discrete state declarations. To each discrete state is associated a set of flow equations between braces. After the keyword `transitions` are the discrete transition declarations. To each discrete transition is associated a guard after the keyword `when`, and possibly a set of exported events between braces. These are triggered when the corresponding transition is fired and are used to synchronise several components. Finally after the keyword `global` are the global variable declarations: here there is only the creation of the component `t` of type `thermostat`.

SHIFT has also a notion of class hierarchy: a *subtype* can inherit member variables from its parent type. For instance, one can specify a generic `vehicle` type, with subtypes `car` and `bus`.

SHIFT comes with a runtime environment: it implements the continuous time integration, with a Runge-Kutta variable step integrator, as well as the dynamic synchronisation and transition firing mechanism. Because of this variable step integrator, the continuous variables are actually discretised. This is why the guards in the `thermostat` program are inequalities instead of equalities.

Finally, SHIFT has been extensively used for performing various kinds of simulations, including air traffic control, automated underwater vehicles (AUV), helicopter control, and of course automated highway design and simulation [15, 2].

3 Design of the Discrete Event Controller

3.1 Controller Objectives

In this section we show how we design a discrete controller for autonomous vehicles driving on automated highways. The highways we consider have only one main lane, but with several entry and exit lanes. We call *junctions* the highway portions where there is both the main lane and an entry or exit lane. Therefore the goals of such a controller are:

- Whatever the lane the vehicle is driving in, if there is a vehicle in front of it, then the controller must make sure that the vehicle does not collide with it. The controller must also ensure that the vehicle does not exceed the speed limit, and if possible that the vehicle reaches the desired speed corresponding to its lane.
- When the vehicle is driving in the main lane at an entry junction, the controller must decide when the vehicle must yield to vehicles driving in the entry lane.
- When the vehicle is driving in the main lane at an exit junction, the controller must decide when the vehicle must exit the highway. Exiting means that the vehicle moves from the main lane to the exit lane.

- When the vehicle is driving in the entry lane, the controller must decide when the vehicle must begin to merge into the main lane traffic. Merging means that the vehicle moves from the entry lane to the main lane.

We will present in Section 6 the results of our micro-simulations. For this purpose, we define the following counters: `nb_miss` is the number of vehicles that failed to merge, and `nb_coll` is the number of collisions.

3.2 Assumptions

For the sake of simplicity, we assume that the speed limit and the desired speed are equal and are the same in all lanes. Throughout the paper, we note v_{max} this nominal velocity, and we assume it is finite:

Assumption 1 *The maximal and desired velocity in all lanes is v_{max}. It is finite.*

Also the acceleration of each vehicle is bounded by the technology bounds $[a_{min}, a_{max}]$. We take $a_{min} = -0.5g = -4.905\ ms^{-2}$ and $a_{max} = 0.2g = 1.962\ ms^{-2}$ (with $g = 9.81\ ms^{-2}$). These figures are reasonable with the current technology for autonomous vehicles. They are for instance inferior to those given by Ioannou and Chien in [16], where, based on [12] and personal communications with engineers from Ford Motor Company, they state that $a_{max} = 0.4g$ and $a_{min} = -0.8g$.

Assumption 2 *The acceleration of each vehicle is bounded by the technology bounds $[-4.905\ ms^{-2}, 1.962\ ms^{-2}]$.*

Finally, as we have said in Section 1.2, we only focus on the longitudinal control of the vehicles. Therefore, we assume that the lateral speed of each vehicle is always equal to $0\ ms^{-1}$, except of course when the vehicle is merging or exiting. We adopt a simplified model of the lane-changing maneuvers: we assume that the lateral speed is equal to $-1\ ms^{-1}$ during the whole merging maneuver, and to $1\ ms^{-1}$ during the whole exiting maneuver. The lateral position of each vehicle is always measured from left to right, with the origin set at the border of the main lane. Since the highway lanes are 4 meters wide, these figures imply a changing lane time of 4 seconds from the center of the source lane to the center of the destination lane. For realistic lane changing trajectories for autonomous vehicles, the reader can refer to [7, 21, 24, 25, 27].

Assumption 3 *When merging, the lateral speed is equal to $-1\ ms^{-1}$. When exiting, the lateral speed is equal to $1\ ms^{-1}$. Otherwise, the lateral speed is always equal to $0\ ms^{-1}$.*

3.3 Highway Layout

Figure 2 shows the overall layout of an example of highway, approximately modelled after the Katy Corridor (Interstate Highway 10) of the Houston metropolitan region. This highway has three entry lanes and three exit lanes. Of course these parameters can be changed at will for new case studies.

Figure 2: Highway overview.

These figures are kept in the two following arrays, `entry_position` and `exit_position`:
Figure 3 shows the layout of each entry junction.

`entry_position`

number	1	2	3
position	0 m	2640 m	7680 m

`exit_position`

number	1	2	3
position	960 m	8160 m	9600 m

Table 1: Entry and exit positions.

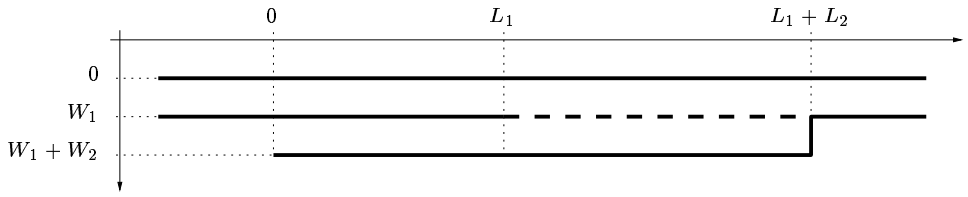

Figure 3: Layout of the entry junction.

L_1 = Distance between the beginning of the entry lane and the first point where a vehicle could possibly merge.

L_2 = Distance between the first point and the last point where a vehicle could possibly merge.

W_1 = Width of the main lane at the entry junction.

W_2 = Width of the entry lane.

Figure 4 shows the layout of each exit junction.

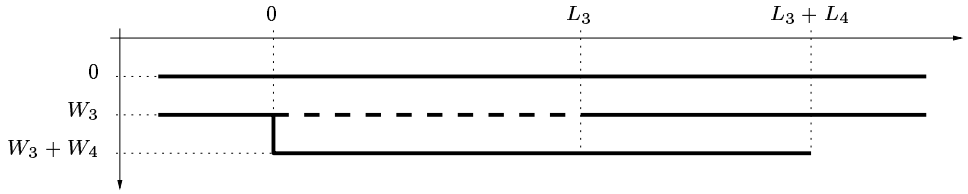

Figure 4: Layout of the exit junction.

L_3 = Distance between the first point and the last point where a vehicle could possibly exit.

L_4 = Distance between the last point where a vehicle could possibly exit and the end of the lane.

W_3 = Width of the main lane at the exit junction.

W_4 = Width of the exit lane.

The layout of the three entry and exit junctions is summarized in the following tables:

entry junction	
L_1	240 m
L_2	480 m
W_1	4 m
W_2	4 m

exit junction	
L_3	480 m
L_4	240 m
W_3	4 m
W_4	4 m

Table 2: Figures of the entry and exit junctions.

3.4 Sensing Capabilities

We assume that each vehicle is equipped with sensors that allow it to maintain a pointer towards the nearby other vehicles. When a vehicle is driving in a one-lane portion of the highway, we adopt the notations described in Figure 5. Here the concerned vehicle is represented by a thick rectangle and the neighbour vehicles are represented by thin rectangles.

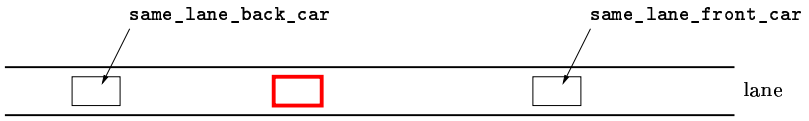

Figure 5: Neighbour vehicles in a one-lane highway portion.

We assume that the sensors the vehicles are equipped with have a limited range equal to d_{max}. For instance, whenever the `same_lane_front_car` is be too far away (or when there is no vehicle in front at all), the pointer `same_lane_front_car` is undefined. These sensors allow the vehicle to know at any time the gap and the relative velocity between itself and the nearby vehicles that are within range. The vehicle also knows its own current velocity.

Similarly, when a vehicle is driving in the main lane of a two-lane portion of the highway, we adopt the notations described in Figure 6. When the vehicle is driving in the side lane, the figure is symmetric.

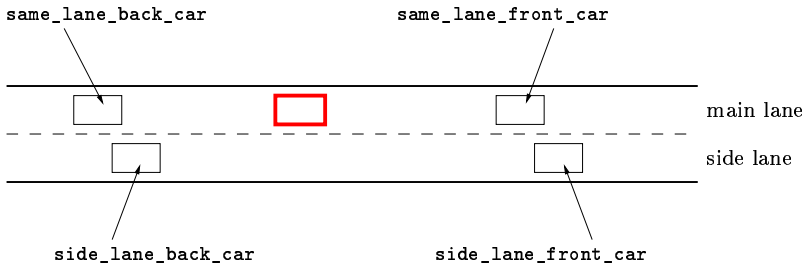

Figure 6: Neighbour vehicles in a two-lane highway portion.

Assumption 4 *The sensor range is d_{max}; it is finite. Whenever a vehicle is within sensor range, a pointer is kept at it, and the gap and the relative velocity are known. The vehicle also knows its own current velocity.*

3.5 Design of the Controller

All the vehicles start at the beginning of the entry lane at one of the three entry junctions with an initial speed equal to v_{init}, and stop at the end of the exit lane at one of the three exit junctions. The junction at which a vehicle must exit is known. Of course, a vehicle entering the highway at the second entry junction can only exit at the second or third exit junction. In concrete terms, we define for each vehicle:

- `entry_junction_number` as the number of the entry junction where the vehicle enters the highway;
- `exit_junction_number` as the number of the exit junction where the vehicle must exit the highway.

Therefore, each vehicle goes through the following phases, each of which is embedded in a distinct discrete state of the hybrid controller:

1. An `accelerate` phase while it is in the entry lane but cannot yet merge (i.e., $x < L_1$; for the sake of simplicity, we write L_1 instead of `entry_position[entry_junction_number]` $+ L_1$); it starts at a speed equal to v_{init} and tries to reach v_{max}, of course without colliding with its possible `same_lane_front_car`; its lateral speed is equal to 0 ms^{-1};
2. A `merge` phase where it tries and hopefully succeeds to merge from the entry lane to the main lane (i.e., $L_1 < x < L_1 + L_2$); here it must drive at v_{max} wherever possible, without colliding either with its possible `same_lane_front_car` or with the possible vehicles `side_lane_back_car` and `side_lane_front_car` between which it is merging; its lateral speed is equal to -1 ms^{-1};
3. A `drop-out` phase where it exits the highway because it was unable to merge before $L_1 + L_2$;
4. A `cruise` phase where it drives in the main lane, *without* any vehicle within sensor range in the side lane (when there is one); it must drive at v_{max} wherever possible, without colliding with its possible `same_lane_front_car`; its lateral speed is equal to 0 ms^{-1};
5. A `yield` phase where it drives in the $[L_1, L_1 + L_2]$ portion in the main lane, *and* there is a vehicle in the entry lane in front of it and within sensor range; it must drive at v_{max} wherever possible, without colliding either with its possible `same_lane_front_car` or with the possible merging `side_lane_front_car`; its lateral speed is equal to 0 ms^{-1};
6. An `exit` phase where it goes from the main lane to the exit lane of its assigned exit junction (i.e., $x < L_3$); it must drive at v_{max} wherever possible, without colliding either with its possible `same_lane_front_car` or with its possible `side_lane_front_car`; its lateral speed is equal to 1 ms^{-1};
7. An `end` phase where it exits the highway (i.e., $L_3 < x < L_3 + L_4$); again it must drive at v_{max} without colliding with its possible `same_lane_front_car`; its lateral speed is equal to 0 ms^{-1};
8. A `collision` phase where a vehicle suddenly exits the highway, with a lateral speed of 2 ms^{-1}. In any of the above phases, whenever there is a collision between two vehicles, we increment the `nb_coll` counter and both vehicles go into this `collision` phase.

We choose to control the longitudinal movements of the vehicle with its *acceleration*. This acceleration is bounded by Assumption 2. Two approaches can be found in the literature. For instance, No, Chong, and Roh also use a longitudinal acceleration control law [26], while Ioannou and Xu use a throttle and brake control [18]. At INRIA, experiments are performed with several CYCAB electric vehicles [6, 29], with acceleration control laws.

We use two functions for computing the acceleration of the vehicle. Each corresponds to a continuous control law which will be presented in Section 4. To specify and use these functions, we adopt the following notation: the longitudinal position of the vehicle is noted x, its longitudinal velocity $\dot{x}$, and its acceleration $\ddot{x}$. Also, whenever `same_lane_front_car` is defined, its longitudinal position is noted z_1, its velocity $\dot{z_1}$, and its acceleration $\ddot{z_1}$. Similarly, whenever `side_lane_front_car` is defined, its position is noted z_2, its velocity $\dot{z_2}$, and its acceleration $\ddot{z_2}$. Whenever `same_lane_back_car` is defined, its position is noted z_3, its velocity $\dot{z_3}$, and its acceleration $\ddot{z_3}$. Whenever `side_lane_back_car` is defined, its position is noted z_4, its velocity $\dot{z_4}$, and its acceleration $\ddot{z_4}$. Finally, the lateral position of the vehicle is noted y and its lateral velocity $\dot{y}$. For both functions, if the result is negative, then the vehicle decelerates, while if it is positive, the vehicle accelerates:

1. The first function is used when both `same_lane_front_car` and `side_lane_front_car` are undefined. It computes the acceleration needed to reach a given desired velocity v_d from the current speed of the vehicle v. We call it the "velocity" acceleration law and note it $\gamma_v(v, v_d)$.

2. The second function is used when `same_lane_front_car` and/or `side_lane_front_car` are defined. It computes the acceleration needed to stay behind `same_lane_front_car` and/or `side_lane_front_car` without colliding with them. We call it the "follow" acceleration law and note it $\gamma_f(v, g, v_f)$, where v is the velocity of the vehicle, g is either the gap with `same_lane_front_car` (i.e., $z_1 - x$) or with `side_lane_front_car` (i.e., $z_2 - x$), and v_f is either the velocity of `same_lane_front_car` (i.e., $\dot{z_1}$) or of `side_lane_front_car` (i.e., $\dot{z_2}$). Without entering into the details, the function decides whether the vehicle is too close or too far from the front vehicle according to v, g, and v_f. If the vehicle is too close, γ_f returns a negative result, with a smaller gap resulting in a stronger deceleration. If the vehicle is too far, γ_f returns a positive result, with a larger gap resulting in a stronger acceleration.

With these two functions, we compute the acceleration of the vehicle in the following way:

1. In the `accelerate` phase, if `same_lane_front_car` is undefined, then $\ddot{x} = \gamma_v(\dot{x}, v_{max})$, else $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}))$. Through its sensors, the vehicle is able to know the value of the gap $z_1 - x$. However, only the velocity difference $\dot{z_1} - \dot{x}$ is known, and not the value of $\dot{z_1}$, which must be computed by adding the vehicle velocity to the measured velocity difference.

   When $x = L_1$, the vehicle proceeds to the `merge` phase.

2. The `merge` phase is the difficult part of the controller. The usual strategy found in the literature involves, for the vehicle willing to merge, targeting a gap between two successive vehicles of the main lane, managing to align to this gap, and if successful actually merging. For instance, this is the strategy adopted by [20]. The difficulty lies in the choice of the gap and in the computation of the acceleration required to align to it. In work on AHS, it is different because there is a coordination layer sending orders to the vehicles in the main lane, for instance to create a suitable gap for a vehicle willing to merge. This is for instance the case of [22], where after merging, the vehicle must join with a target platoon driving in the main lane. Although the basic objective for us is the same (first align and then merge), we propose here a totally different strategy, based on the *cooperation* of the vehicles in the main lane. Since we do not use communication, we cannot rely on a coordination layer, thus this cooperation has to be implemented by the discrete controller of each vehicle. In concrete terms, we divide the `merge` phase into two sub-phases, `align-to-gap` and `go-to-main`:

   (a) In the `align-to-gap` sub-phase, if `same_lane_front_car` and `side_lane_front_car` are defined, then $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}), \gamma_f(\dot{x}, z_2 - x, \dot{z_2}))$. If `same_lane_front_car` and/or `side_lane_front_car` are undefined, then the corresponding term in the min function disappears. Therefore the vehicle behaves exactly as if `side_lane_front_car` was in its own lane! Similarly, `side_lane_back_car`, if it is defined, must *yield* to the vehicle, that is, behave as if its `side_lane_front_car`, which is the vehicle whose acceleration we are currently computing, was in its own lane (see the `yield` phase below). This double yielding will *mechanically* increase the length of the gap between the `side_lane_back_car` and the `side_lane_front_car`, therefore allowing the vehicle to merge between them!

   If the speed difference between the vehicles in the entry lane and the main lane is too great, or if, when the vehicle enters the $[L_1, L_1 + L_2]$ portion, its `side_lane_front_car` is too close, and accordingly the gap $z_2 - x$ too small, then the vehicle will brake as hard as he can in order to stay behind. Because of Assumption 2, the acceleration is bounded and $z_2 - x$ can become negative. In this case, the gap is missed and the vehicle will take as new `side_lane_front_car` the `same_lane_front_car` of its former `side_lane_front_car`, if it exists, and as new `side_lane_back_car` its former `side_lane_front_car`. The other vehicles will update their pointers accordingly. As a result, if a gap is missed, then the vehicle will try to align to the next gap.

   When some safety condition involving the vehicle, its `side_lane_front_car`, and its possible `side_lane_back_car` is satisfied, the vehicle can proceed in the `go-to-main` sub-phase. If this condition is not fulfilled when $x = L_1 + L_2$, then the merging is impossible: the vehicle goes to the `drop-out` phase. This safety condition remains to be defined.

(b) The `go-to-main` sub-phase is split in two parts. The first part is when the vehicle is in the entry lane, and the second part is when the vehicle is in the main lane. In the entry lane, if `same_lane_front_car` and `side_lane_front_car` are defined, then $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}), \gamma_f(\dot{x}, z_2 - x, \dot{z_2}))$. If `same_lane_front_car` and/or `side_lane_front_car` are undefined, then the corresponding term in the min function disappears. Meanwhile, $\dot{y} = -1\ ms^{-1}$. As soon as the vehicle crosses the limit between the entry and the main lanes (i.e., $y = W_1$), its pointers `side_lane_front_car` and `same_lane_front_car` are exchanged. The pointers of the other neighbour vehicles are updated accordingly. In the main lane, the acceleration is the same as before, except that the pointers have changed. Its lateral speed remains equal to $-1\ ms^{-1}$ until the vehicle reaches the middle of the main lane (i.e., $y = W_1/2$); at this moment, the vehicle goes to the `cruise` phase.

If the vehicle is unable to move from the entry lane to the main lane before $L_1 + L_2$, then it goes to the `drop-out` phase.

3. In the `drop-out` phase, we increment the `nb_miss` counter and remove the vehicle from the simulation.

4. In the `cruise` phase, the vehicle must avoid collisions with its `same_lane_front_car`, if it exists. Therefore $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}))$. If `same_lane_front_car` is undefined, then the γ_f term of the min function disappears. When the vehicle enters the $[L_1, L_1 + L_2]$ portion of any entry junction, if there is a vehicle within sensor range in the entry lane, then it goes to the `yield` phase and this vehicle becomes its `side_lane_front_car`. Otherwise there is no vehicle to yield to, so it remains in the `cruise` phase. When it enters the $[0, L_3]$ portion of its assigned exit junction, it goes to the `exit` phase.

5. In the `yield` phase, the vehicle must yield to its `side_lane_front_car` while avoiding collisions with its `same_lane_front_car`, if they exist. Therefore $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}), \gamma_f(\dot{x}, z_2 - x, \dot{z_2}))$. If `side_lane_front_car` and/or `same_lane_front_car` are undefined, then the corresponding term of the min function disappears. With this acceleration, the vehicle behaves exactly as if the `side_lane_front_car` was in its own lane! Our micro-simulations results, presented in Section 6, show that this passive cooperation between vehicles in the main lane and in the entry lane ensures that all vehicles can merge safely. When $x = L_1 + L_2$ or when `side_lane_front_car` is not defined anymore, it goes to the `cruise` phase. The `side_lane_front_car` pointer can become undefined for several reasons: the vehicle can be outside the sensor range, or it can merge into the main lane, or it can be overtaken.

6. We treat the `exit` phase exactly like the `merge` phase, by considering that the vehicle must merge from the main lane to the exit lane. The only difference is that there will be no `side_lane_back_car`. Concretely, we divide the `exit` phase in two sub-phases, `prepare-exit` and `go-to-exit`. We do not detail them as they are similar respectively to `align-to-gap` and `go-to-main`. The only differences are that the safety condition (which also remains to be defined) guarding the `prepare-exit` to `go-to-exit` transition involves only the vehicle and its possible `side_lane_front_car`, and that in both states the lateral speed is $\dot{y} = 1\ ms^{-1}$. As soon as the vehicle crosses the limit between the main and the exit lanes (i.e., $y = W_3$), its pointers `side_lane_front_car` and `same_lane_front_car` are exchanged. The pointers of the other neighbour vehicles are updated accordingly. In the exit lane, we do not care anymore of the possible `side_lane_front_car`, so the acceleration is $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}))$. Its lateral speed remains equal to $1\ ms^{-1}$ until the vehicle reaches the middle of the exit lane (i.e., $y = W_3 + W_4/2$); at this moment, the vehicle goes to the `end` phase.

7. In the `end` phase, if `same_lane_front_car` is undefined, then $\ddot{x} = \gamma_v(\dot{x}, v_{max})$, else $\ddot{x} = \min(\gamma_v(\dot{x}, v_{max}), \gamma_f(\dot{x}, z_1 - x, \dot{z_1}))$. When $x = L_3 + L_4$, the vehicle is removed from the simulation.

8. In the `collision` phase (reached whenever $z_1 - x$ or $x - z_3$ becomes negative), we distinguish two cases. The first case is when the vehicle has been hit at the back by its `same_lane_back_car`, i.e., $x - z_3 < 0$; we brake as hard as possible and reach the side of the highway as fast as possible: so $\ddot{x} = -4.905\ ms^{-2}$ and $\dot{y} = 2\ ms^{-1}$. The second case is when the vehicle has hit the back of its `same_lane_front_car`,

i.e., $z_1 - x < 0$; again $\ddot{x} = -4.905\ ms^{-2}$ and $\dot{y} = 2\ ms^{-1}$; we also set $\dot{x}$ to $\dot{z}_1$ at the time of the inpact because we assume that both vehicles are crunched together. This goes on until the vehicle leaves the highway (i.e., $y = W_1$ or W_3 in a one-lane portion, $y = W_1 + W_2$ or $W_3 + W_4$ in a two-lane portion).

3.6 Hybrid Automaton

Figure 7 shows the controller automaton as described in the previous subsection. The initial state is `accelerate`, and the terminal states are `end`, `collision`, and `drop-out`. All the transitions leading to `collision` have the same guard: $z_1 - x < 0$ or $x - z_3 < 0$.

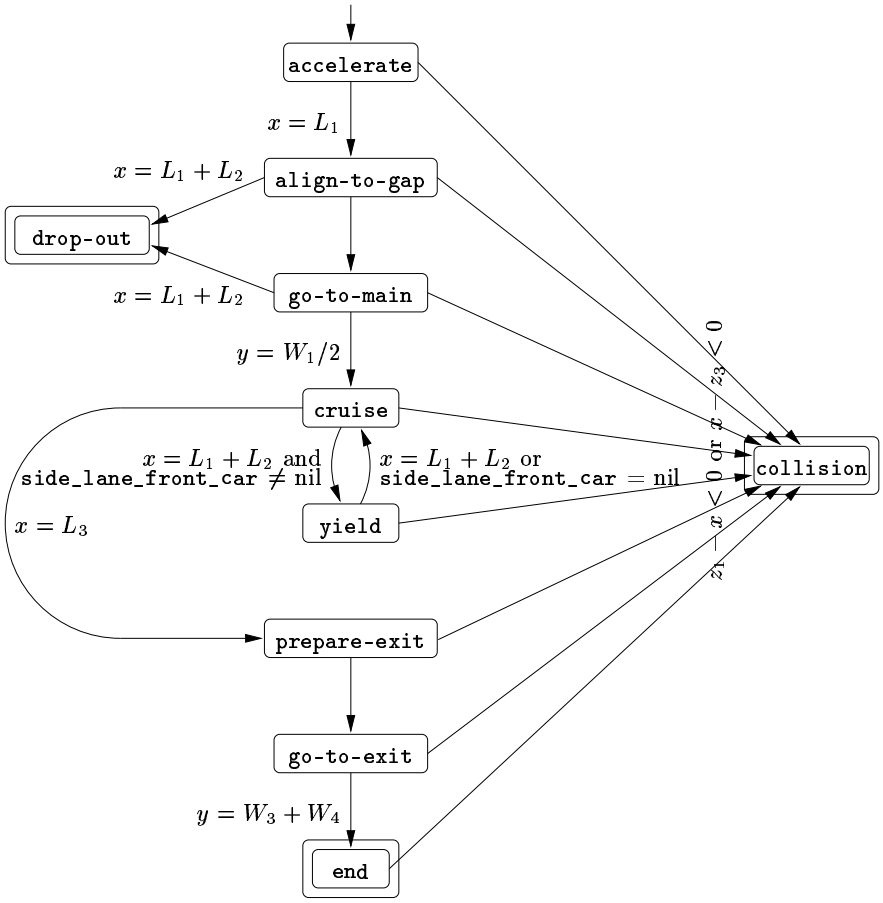

Figure 7: Controller automaton.

This controller has been implemented in the SHIFT language [11] (see Section 2.2). We will present simulation results in Section 6.

Finally, the guards of the initial transition, of the transition from `align-to-gap` to `go-to-main`, and of the transition from `prepare-exit` to `go-to-exit` remain to be defined. These guards must ensure that the newly created of merged vehicle will not cause a collision. We will do this in Section 5, after having studied the continuous part of our hybrid controller in the next Section.

This approach is similar to [15]. Here the hybrid controller is implemented as the composition of the coordination layer, modelled as a finite state machine, and the regulation layer, modelled as feedback laws based on linear models. The coordination layer's correctness is verified using software verification tools. Moreover, the regulation layer's consistency is proved using game theory, and these results are used to initiate a maneuver only when it can be safely completed, much like the conditions guarding our transitions.

3.7 Other Merging Strategies

As we have shown in Section 3.5, our merging strategy is embedded in the `align-to-gap` and `yield` phases. The principle is that any vehicle in the main lane must yield to its `side_lane_front_car`, while at the same time, any vehicle in the entry lane must yield to its `side_lane_front_car`. This double yielding creates a gap for the vehicle willing to merge. This proceeds until the gap is big enough and it is safe to merge.

It is perfectly possible to define and implement other merging strategies. Changing the merging strategy of the implemented controller is very easy to do thanks to the SHIFT language. Note that in our controller, the merging strategy is *disconnected* from the safety conditions which are embedded in the transition guards. Therefore, changing the merging strategy does not modify the safety of the whole controller.

Finally, the criteria for evaluating a merging strategy are, ranked by decreasing importance: the absence of collisions, the absence of vehicle drop-out, and the smoothness of the traffic. In Section 6, we will evaluate these criteria for our own merging strategy, based on the simulation results.

4 Design of the Acceleration Control Laws

In this section, we design and study the two acceleration control laws needed by our hybrid controller, the "velocity" acceleration law γ_v and the "follow" acceleration law γ_f. Part of this section has been published in the IEEE Conference on Decision and Control [13].

The goal of the first control law is to make the controlled vehicle reach a desired speed in the absence of vehicles in front of it. The goal of the second one is to make the controlled vehicle follow a vehicle in front of it without colliding with it. We study analytically this control law and we prove its asymptotic stability, and we end with some simulation results. Specifically, we prove that under some initial conditions, the vehicles never collide. We first conduct this analysis in the non-saturated case, and then in the saturated case.

There exist three basic classes of such control laws depending on the spacing policy, i.e., the desired distance between the leading and the following vehicle [32]:

1. Constant *spacing*: such systems are most suitable for use at very short spacings, for instance when vehicles are grouped into a platoon. However, this requires inter-vehicle communication of the acceleration of the leading vehicle.

2. Constant *time headway*: such systems use only information derived from sensors, and they maintain spacing proportional to the velocity v. The time headway is the time needed for the leading vehicle to travel the distance between his front bumper and the rear bumper of the following vehicle.

3. Constant *safety factor*: such systems are designed so that the instantaneous stopping of the leading vehicle does not cause the following vehicle to collide with it. Typically, they maintain spacing proportional to v^2.

The second class (constant time headway) is the more suitable to our application. For this reason, we chose an acceleration law that maintains a constant time headway. The vehicles are autonomous, i.e., they do not communicate with each other and sensor range is limited. Thus, at any time a vehicle knows which vehicles are in front of and behind it. The longitudinal speed of each vehicle within sensor range is also known. Each vehicle has its own controller that computes the current longitudinal acceleration. The kinematic model is a basic 2-dimensional model.

4.1 Design of the "Velocity" Acceleration Law

The design of this acceleration control law is easy. We present it here for the sake of completeness. Recall that the desired velocity is v_{max} (see Assumption 1 page 8). The position of the vehicle is x, its velocity $\dot{x}$, and its acceleration $\ddot{x}$. We neglect the internal dynamic of the vehicles and choose to control the following vehicle with its acceleration:

$$\ddot{x} = u(x, \dot{x}, v_{max}) \tag{1}$$

The solution of the ordinary differential equation (1) is the function $x(t)$. We choose a classic proportional control:

$$\ddot{x}(t) = u(t) = \mu(v_{max} - \dot{x}) \tag{2}$$

where the gain μ is of dimension s^{-1}. Now let $\varepsilon = \dot{x} - v_{max}$: Equation (2) becomes:

$$\dot{\varepsilon}(t) = -\mu\varepsilon(t)$$

This yields:

$$\varepsilon(t) = Ce^{-\mu t} \Longrightarrow \dot{x}(t) = v_{max} + Ce^{-\mu t}$$

The initial condition for $t = 0$ gives:

$$\dot{x}(0) = v_0 = v_{max} + C \Longrightarrow C = v_0 - v_{max}$$

Therefore, we finally get:

$$\dot{x}(t) = v_{max} + (v_0 - v_{max})e^{-\mu t}$$

The control law $u(t)$ ensures that the vehicle reaches the desired velocity v_{max}. Now what happens if the acceleration of the vehicle must remain within the technology bounds of Assumption 2? In this case, we apply the following saturated command:

$$\begin{aligned} u(t) &= \mu(v_{max} - \dot{x}(t)) \\ \ddot{x}(t) = \tilde{u}(t) &= \text{if } u(t) > 1.962\ ms^{-2} \text{ then } 1.962\ ms^{-2} \\ &\quad \text{else if } u(t) < -4.905\ ms^{-2} \text{ then } -4.905\ ms^{-2} \\ &\quad \text{else } u(t) \end{aligned}$$

Since the desired velocity v_{max} is finite, the saturated control law $\tilde{u}(t)$ also ensures that the vehicle reaches v_{max}.

Remark 1 *In this paper we choose to neglect the internal dynamics of the vehicles. One solution to take it into account would be to design a two-level continuous controller, the upper level deciding what acceleration should be applied to a vehicle, and the lower level deciding what throttle and brake inputs are needed to achieve this desired acceleration [27].*

4.2 Design of the "Follow" Acceleration Law

4.2.1 State Variables of the System

We now consider a system consisting of one leading vehicle and one following vehicle, driving along a single-lane straight highway. The behaviour of the leading vehicle is a priori unknown. Its position is $z(t)$, its velocity $\dot{z}(t)$, and its acceleration $\ddot{z}(t)$. We assume that the acceleration of the leading vehicle is a piecewise continuous function. That way we have $\ddot{z}(t) = \frac{d\dot{z}}{dt}$ and $\dot{z}(t) = \int \ddot{z}dt$.

Assumption 5 *$\ddot{z}(t)$ is at least piecewise continuous.*

The following vehicle drives right behind the leading vehicle and tries to avoid collisions. Its position is x, its velocity $\dot{x}$, and its acceleration $\ddot{x}$. Like in the previous section, we neglect the internal dynamic of the vehicles and choose to control the following vehicle with its acceleration:

$$\ddot{x} = u(x, \dot{x}, z, \dot{z}) = u(x, \dot{x}, t) \tag{3}$$

The solution of the ordinary differential equation (3) is the function $x(t)$. We assume that the following vehicle is always *behind* the leading vehicle:

Assumption 6 $\forall t,\ x(t) \leq z(t)$.

We further assume that both vehicles are moving *forward* and are never *immobile*, and that the speed of the leading vehicle is *bounded* by the given maximal velocity v_{max}:

Assumption 7 $\forall t,\ \dot{x}(t) > 0$ *and* $v_{max} > \dot{z}(t) > 0$.

We assume nothing about the jerks $\dddot{z}$ and $\dddot{x}$. However, the acceleration of the leading vehicle must remain within technology bounds ($g = 9.81\ ms^{-2}$):

Assumption 8 $\forall t,\ a_{min} = -0.5g = -4.905\ ms^{-2} \leq \ddot{z}(t) \leq a_{max} = 0.2g = 1.962\ ms^{-2}$.

The desired time headway will be noted h. It has a direct influence on the maximal density that can be accommodated by an automated highway [8]. Intuitively, for a given density and nominal speed, it is possible to compute the average gap between two successive vehicles. In [1, 2], we have reported extensive simulations of an automated highway which show that this gap divided by the nominal speed is the time headway that will not cause congestion. For these highway simulations, we chose a desired time headway h equal to 0.6 s, which is perfectly feasible with the current technology of sensors, actuators, and computing systems.

4.2.2 Non Linear Controller

The aim of the control law is to try to keep a constant time headway. We thus compute the time headway, i.e., the gap divided by the vehicle speed:

$$\text{time headway} = \frac{z(t) - x(t)}{\dot{x}(t)}$$

We then compute the ratio of the time before impact by the desired time headway h. This ratio $r(t)$ is a pure number:

$$r(t) = \frac{z(t) - x(t)}{h\dot{x}(t)} \tag{4}$$

Because of Assumptions 6 and 7, $r(t)$ is in the range $[0, +\infty)$. It measures the correct positioning of the following vehicle relatively to the leading vehicle. A value between 0 and 1 means that the following vehicle is too close to the leading vehicle and must brake. A value greater than 1 means that the following vehicle is too far from the leading vehicle and can accelerate. A value equal to 1 means that the following vehicle is exactly at the desired time headway from the leading vehicle. Keeping a constant time headway amounts to keeping $r(t)$ equal to 1. So the output of our system will be: $y(t) = r(t)$, with 1 as our set point.

We wish to perform output regulation with 1 as our set point. The output dynamics is:

$$\begin{aligned}
\dot{y}(t) \ = \ \dot{r}(t) \ &= \ \frac{h\dot{x}(t)(\dot{z}(t) - \dot{x}(t)) - h\ddot{x}(t)(z(t) - x(t))}{h^2\dot{x}^2(t)} \\
&= \ \frac{\dot{z}(t) - \dot{x}(t)}{h\dot{x}(t)} - \frac{\ddot{x}(t)}{\dot{x}(t)} \cdot \frac{z(t) - x(t)}{h\dot{x}(t)} \\
&= \ \frac{\dot{z}(t) - \dot{x}(t)}{h\dot{x}(t)} - \frac{\ddot{x}(t)}{\dot{x}(t)} \cdot r(t) \\
&= \ \frac{\dot{z}(t) - \dot{x}(t) - h\ddot{x}(t)r(t)}{h\dot{x}(t)}
\end{aligned} \tag{5}$$

The equilibrium is reached if $\dot{y}(t) = \dot{r}(t) = 0$ and $y(t) = r(t) = 1$, and the control at the equilibrium is $\hat{u}(t)$. From Equation (5), this implies $\dot{z}(t) - \dot{x}(t) - h\hat{u}(t) = 0$ and therefore we have:

$$\hat{u}(t) = \frac{\dot{z}(t) - \dot{x}(t)}{h} \tag{6}$$

We finally add a feedback term to our control u:

$$\begin{aligned}
\ddot{x}(t) = u(t) \ &= \ \hat{u}(t) + \lambda(r(t) - 1) \\
&= \ \frac{\dot{z}(t) - \dot{x}(t)}{h} + \lambda(r(t) - 1)
\end{aligned} \tag{7}$$

where the gain λ is of dimension ms^{-2}. If there exists t_0 such that $r(t_0) = 1$, the controller given by Equation (7) is the same as the one given by Equation (6), which keeps $r(t)$ equal to 1 by construction. In other words, we have the following first proposition:

Proposition 1 *If* $\exists t_0 : r(t_0) = 1$, *then* $\forall t, r(t) = 1$.

The expanded form of Equation (7) is:

$$\ddot{x}(t) = \frac{\dot{z}(t) - \dot{x}(t)}{h} + \lambda \frac{z(t) - x(t)}{h\dot{x}(t)} - \lambda \tag{8}$$

This controller is non-linear. We could have chosen instead a linear controller. However, such a linear controller would not satisfy Proposition 1.

4.3 Asymptotic Stability Analysis

A first approach to study the stability of our system would be to compute a differential equation on $r(t)$ and to integrate it. From equations (4) and (7), we obtain after some computations:

$$\begin{array}{l} \lambda + \lambda r(t)^2 - 2\lambda r(t) - 2h\dot{r}(t)^2\dot{z}(t) + 2\lambda h^2\dot{r}(t)^2 + \ddot{z}(t) \\ - 3\lambda h^2 r(t)\dot{r}(t)^2 + 3\lambda h\dot{r}(t) - 5\lambda h r(t)\dot{r}(t) + h\dot{r}(t)\ddot{z}(t) \\ + 2\lambda h r(t)^2\dot{r}(t) + \lambda h^2 r(t)^2\ddot{r}(t) - \dot{r}(t)\dot{z}(t) - 2r(t)\ddot{z}(t) \\ + r(t)^2\ddot{z}(t) + h r(t)\ddot{r}(t)\dot{z}(t) - h\ddot{r}(t)\dot{z}(t) - \lambda h^2 r(t)\ddot{r}(t) \\ + r(t)\dot{r}(t)\dot{z}(t) - h r(t)\dot{r}(t)\ddot{z}(t) \;=\; 0 \end{array}$$

This differential equation is not integrable. Our idea is therefore to study analytically the $r(t)$ function rather than trying to solve it. We first study the variations of this function. We then prove that it converges towards 1 when $t \to +\infty$.

4.3.1 Study of the Variations of $r(t)$

Under our assumptions, $r(t)$ is continuous and derivable, and its first derivative is also continuous. From Equation (5), we get:

$$\begin{aligned} \dot{r}(t) &= \frac{\dot{z}(t) - \dot{x}(t) - h\ddot{x}(t)r(t)}{h\dot{x}(t)} \\ &= \frac{\dot{z}(t) - \dot{x}(t) - \lambda h(r(t) - 1) - r(t)(\dot{z}(t) - \dot{x}(t))}{h\dot{x}(t)} \\ &= \frac{(\dot{z}(t) - \dot{x}(t))(1 - r(t)) + \lambda h r(t)(1 - r(t))}{h\dot{x}(t)} \\ &= \frac{(1 - r(t))(\dot{z}(t) - \dot{x}(t) + \lambda h r(t))}{h\dot{x}(t)} \\ &= \frac{(1 - r(t))(\ddot{x}(t) + \lambda)}{\dot{x}(t)} \end{aligned} \tag{9}$$

Now let us study the variations of $r(t)$. According to Equation (9) and Assumption 7, we have the following lemma on the sign of the first derivative of r:

Lemma 1 *If* $\forall t, r(t) < 1$ *(resp.* $r(t) > 1$*), then* $\forall t, \ddot{x}(t) > -\lambda \Leftrightarrow \dot{r}(t) > 0$ *(resp.* $\dot{r}(t) < 0$*).*

Lemma 1 allows us to state the following proposition on the variations of r:

Proposition 2 *If* $\forall t, \ddot{x}(t) > -\lambda$, *then:*

- $\forall t, r(t) \in [0, 1)$ *and* r *is strictly increasing, or*

- $\forall t,\ r(t) = 1$, *or*
- $\forall t,\ r(t) \in (1, +\infty)$ *and* r *is strictly decreasing.*

This is an important result since it establishes that, under the condition $\ddot{x}(t) > -\lambda$, $r(t)$ evolves in the good direction: If the following vehicle is too close, i.e., $r(t) < 1$, then $r(t)$ increases, while if the following vehicle is too far away, i.e., $r(t) > 1$, then $r(t)$ decreases. Moreover, $r(t)$ never oscillates around the value 1.

4.3.2 Study of the Limit of $r(t)$

We know by assumption that the speed of the leading vehicle is bounded (Assumption 7). The following lemma is on the boundedness of the following vehicle speed:

Lemma 2 *If* $\exists t_0$ *such that* $\forall t \geq t_0,\ \ddot{x}(t) > -\lambda$, *then* $\dot{x}(t)$ *is upper bounded.*

Proof For any $t \geq t_0$, we have:

$$\begin{aligned}
-\lambda < \ddot{x}(t) &\Longrightarrow -\lambda < \frac{\dot{z}(t) - \dot{x}(t)}{h} + \lambda(r(t) - 1) \\
&\Longrightarrow 0 < \frac{\dot{z}(t) - \dot{x}(t)}{h} + \lambda r(t) \\
&\Longrightarrow 0 < \dot{z}(t) - \dot{x}(t) + \lambda h r(t) \\
&\Longrightarrow \dot{x}(t) < \dot{z}(t) + \lambda h r(t) \\
&\Longrightarrow \dot{x}(t) < v_{max} + \lambda h r(t)
\end{aligned}$$

We then have two exclusive cases depending on $r(t)$:

1. $\forall t \geq t_0,\ r(t) > 1$. Since $r(t)$ is decreasing, $\forall t \geq t_0,\ r(t) \leq r(t_0)$. Thus we have $\forall t \geq t_0$, $\dot{x}(t) < v_{max} + \lambda h r(t_0)$.
2. $\forall t \geq t_0,\ r(t) \leq 1$. Thus we have $\forall t \geq t_0,\ \dot{x}(t) < v_{max} + \lambda h$. □

This lemma allows us to prove the following proposition on the asymptotic limit of the ratio $r(t)$ when t grows to infinity:

Proposition 3 *If* $\exists t_0$ *such that* $\forall t \geq t_0,\ \ddot{x}(t) > -\lambda$, *then* $\lim_{t \to +\infty} r(t) = 1$.

Proof In the $r(t) \in [0, 1)$ case, since $r(t)$ is increasing and bounded, it has a limit, say l. Similarly, in the $r(t) \in (1, +\infty)$ case, since $r(t)$ is decreasing and bounded, it has a limit, say L. Now let us prove that $l = 1$. Since $\lim_{t \to +\infty} r(t)$ is finite, it is an asymptotic limit and therefore $\lim_{t \to +\infty} \dot{r}(t) = 0^+$. According to Equation (9), we have three cases:

1. $\lim_{t \to +\infty}(1 - r(t)) = 1 - l = 0 \Longrightarrow l = 1$.
2. $\lim_{t \to +\infty}(\ddot{x}(t) + \lambda) = 0 \Longrightarrow \lim_{t \to +\infty} \ddot{x}(t) = -\lambda \Longrightarrow \lim_{t \to +\infty} \dot{x}(t) = -\infty$. According to Assumption 7, $\dot{x}(t) > 0$, so this case is impossible.
3. $\lim_{t \to +\infty} \dot{x}(t) = +\infty$. According to Lemma 2, $\dot{x}(t)$ is upper bounded, so this case is impossible.

Since cases 2 and 3 are impossible, $l = 1$. The proof that $L = 1$ is similar. □

This is a major result since it establishes that, under the condition $\ddot{x}(t) > -\lambda$, the system is asymptotically stable, and therefore that the following vehicle never collides with the leading vehicle. In the next section, we establish initial conditions under which the invariant condition $\ddot{x}(t) > -\lambda$ is fulfilled.

4.3.3 Study of the Condition $\ddot{x}(t) > -\lambda$

Through this section, we suppose $r(t) \in [0, 1)$. The case where $r(t) \in (1, +\infty)$ is analogous. The following propositions states that the *invariant* condition on the following vehicle acceleration $\ddot{x} > -\lambda$ is satisfied if the *invariant* condition on the leading vehicle acceleration $\ddot{z} > -\lambda$ and the *initial* condition on the following vehicle acceleration $\ddot{x}(t_0) > -\lambda$ for some t_0 are satisfied.

Proposition 4 *If $\exists t_)$ such that $\ddot{x}(t_0) > -\lambda$ and $\forall t \geq t_0,\ \ddot{z}(t) > -\lambda$, then $\forall t \geq t_0,\ \ddot{x}(t) > -\lambda$.*

Proof Without loss of generality, we start at time $t_0 = 0$. Our hypotheses are $\ddot{x}(0) > -\lambda$, $\forall t,\ \ddot{z}(t) > -\lambda$, and $\forall t,\ r(t) < 1$. We distinguish two exclusive cases, depending whether $\ddot{x}(t) \geq \ddot{z}(t)$ or $\ddot{x}(t) \leq \ddot{z}(t)$ over an interval $[0, \delta]$ where $\ddot{z}(t)$ is continuous:

1. Suppose that $\forall t \in [0, \delta],\ \ddot{x}(t) \geq \ddot{z}(t)$. By hypothesis, $\forall t \in [0, \delta],\ \ddot{z}(t) > -\lambda$. Thus $\forall t \in [0, \delta],\ \ddot{x}(t) > -\lambda$.

2. Suppose that $\forall t \in [0, \delta],\ \ddot{x}(t) \leq \ddot{z}(t)$. Let us prove that $\dot{r}(t) > 0$ over $[0, \delta]$. Equation (9) is:

$$\dot{r}(t) = \frac{(1 - r(t))(\ddot{x}(t) + \lambda)}{\dot{x}(t)}$$

   By hypothesis, $1 - r(0) > 0$ and $\ddot{x}(0) + \lambda > 0$. By Assumption 7, $\dot{x}(0) > 0$, so $\dot{r}(0) > 0$. Suppose that $\dot{r}$ is not strictly positive over $[0, \delta]$: Then $\exists \varepsilon > 0$ such that $\dot{r}(\varepsilon) = 0$ and $\forall t \in [0, \varepsilon),\ \dot{r}(t) > 0$. Now $\dot{r}(\varepsilon) = 0$ implies that $\ddot{x}(\varepsilon) = -\lambda$. According to Equation (7), $\dddot{x}$ exists:

$$\dddot{x}(t) = \lambda \dot{r}(t) + \frac{\ddot{z}(t) - \ddot{x}(t)}{h} \tag{10}$$

   By hypothesis, $\forall t \in [0, \delta],\ \ddot{z}(t) - \ddot{x}(t) \geq 0$, therefore, we have $\forall t \in [0, \varepsilon),\ \dddot{x}(t) > 0$. Hence, $\ddot{x}$ is strictly increasing over $[0, \varepsilon)$. Since $\ddot{x}$ is continuous, $\ddot{x}$ is strictly increasing over $[0, \varepsilon]$ and $\ddot{x}(\varepsilon) > \ddot{x}(0)$. But by hypothesis, $\ddot{x}(0) > -\lambda$, hence $\ddot{x}(\varepsilon) > -\lambda$. So we have $\ddot{x}(\varepsilon) = -\lambda > -\lambda$ which is a contradiction. Therefore, there does not exist $\varepsilon > 0$ such that $\dot{r}(\varepsilon) = 0$ and $\forall t \in [0, \varepsilon),\ \dot{r}(t) > 0$. We conclude that $\forall t \in [0, \delta],\ \dot{r}(t) > 0$, and according to Lemma 1, $\forall t \in [0, \delta],\ \ddot{x}(t) > -\lambda$. □

The consequence of Proposition 4 is that to avoid collisions, one should design the acceleration control law with the gain λ greater than the opposite of the lower bound of the leading vehicle acceleration. In mathematical terms: $\lambda > -a_{min}$.

4.3.4 Asymptotic Stability Theorem

Theorem 1 (asymptotic stability) *If $\exists t_0$ such that $\ddot{x}(t_0) > -\lambda$ and $\forall t \geq t_0,\ \ddot{z}(t) > -\lambda$, then one of the three following holds:*

- *$\forall t \geq t_0,\ r(t) \in [0, 1)$, r is strictly increasing, and $\lim_{t \to +\infty} r(t) = 1$, or*
- *$\forall t \geq t_0,\ r(t) = 1$, or*
- *$\forall t \geq t_0,\ r(t) \in (1, +\infty)$, r is strictly decreasing, and $\lim_{t \to +\infty} r(t) = 1$.*

Proof By combining Propositions 2, 3 and 4. □

This formally proves that our control law given by Equation (7) is asymptotically stable, under the initial condition $\ddot{x}(0) > -\lambda$ and the invariant condition $\ddot{z}(t) > -\lambda$. If these conditions are fulfilled, the vehicles never collide.

4.4 Study of the Bounds

We now address the problem of saturation by technology bounds. We know that the acceleration of the leading vehicle is bounded (Assumption 8). In Section 4.4.1, we prove that under some initial condition the acceleration of the following vehicle is also bounded. In Section 4.4.2, we generalise our results to a lane of vehicles.

4.4.1 Bounds on $\ddot{x}(t)$

The following proposition is very much like Theorem 1. It requires the same invariant condition on $\ddot{z}$ and initial condition on $\ddot{x}$, plus an additional initial condition on $\frac{\dot{z}-\dot{x}}{h}$:

Proposition 5 *If* $\forall t \geq t_0,\ -\lambda < a_{min} \leq \ddot{z}(t) \leq a_{max}$, *and* $a_{min} \leq \ddot{x}(t_0) \leq a_{max}$, *and* $a_{min} \leq \frac{\dot{z}(t_0)-\dot{x}(t_0)}{h} \leq a_{max}$, *then we have* $\forall t \geq t_0,\ a_{min} \leq \ddot{x}(t) \leq a_{max}$.

Proof We only prove the proposition when $r(t_0) < 1$. Cases $r(t_0) = 1$ and $r(t_0) > 1$ are analogous. Without loss of generality, we take $t_0 = 0$:

$$r(0) < 1 \Longrightarrow r(0) - 1 < 0 \Longrightarrow \ddot{x}(0) < \frac{\dot{z}(0) - \dot{x}(0)}{h}$$

Our hypotheses are $a_{min} \leq \ddot{x}(0) < \frac{\dot{z}(0)-\dot{x}(0)}{h} \leq a_{max}$, and $\forall t,\ -\lambda < a_{min} \leq \ddot{z}(t) \leq a_{max}$. We distinguish two exclusive cases, depending whether $\dddot{x}(t) \geq \dddot{z}(t)$ or $\dddot{x}(t) \leq \dddot{z}(t)$ over an interval $[0, \delta]$ where $\dddot{z}(t)$ is continuous:

1. Suppose that $\forall t \in [0, \delta],\ \dddot{x}(t) \geq \dddot{z}(t)$:

   - By hypothesis, $\forall t \in [0, \delta],\ \ddot{z}(t) \geq a_{min}$. Thus $\forall t \in [0, \delta],\ \ddot{x}(t) \geq a_{min}$.
   - $\forall t \in [0, \delta], \dddot{x}(t) \geq \dddot{z}(t) \Longrightarrow \forall t \in [0, \delta], \ddot{z}(t) - \ddot{x}(t) \leq 0 \Longrightarrow \dot{z}(t) - \dot{x}(t)$ is decreasing over $[0, \delta]$. Hence $\forall t \in [0, \delta], \frac{\dot{z}(t)-\dot{x}(t)}{h} \leq \frac{\dot{z}(0)-\dot{x}(0)}{h}$ which is by hypothesis smaller than a_{max}. Moreover, $r(t) < 1 \Longrightarrow \lambda(r(t) - 1) < 0$, so:

     $$\forall t \in [0, \delta],\ \lambda(r(t) - 1) + \frac{\dot{z}(t) - \dot{x}(t)}{h} \leq a_{max}$$

     Therefore $\forall t \in [0, \delta], \ddot{x}(t) \leq a_{max}$.

2. Suppose that $\forall t \in [0, \delta],\ \dddot{x}(t) \leq \dddot{z}(t)$:

   - By hypothesis, $\forall t \in [0, \delta],\ \ddot{z}(t) \leq a_{max}$. Thus $\forall t \in [0, \delta],\ \ddot{x}(t) \leq a_{max}$.
   - From to Theorem 1, $\forall t \in [0, \delta], \dot{r}(t) > 0$. By hypothesis $\forall t \in [0, \delta], \ddot{z}(t) - \ddot{x}(t) \geq 0$. So according to Equation (10), $\forall t \in [0, \delta], \dddot{x}(t) > 0$, which implies that $\ddot{x}(t)$ is strictly increasing over $[0, \delta]$. Hence $\forall t \in [0, \delta], \ddot{x}(t) \geq \ddot{x}(0)$ which is by hypothesis greater than a_{min}. So $\forall t \in [0, \delta], \ddot{x}(t) \geq a_{min}$.

In both cases, $\forall t \in [0, \delta], \ddot{x}(t) \in [a_{min}, a_{max}]$. □

4.4.2 Lane Stability

We now generalise our results to an arbitrary lane of n vehicles. Vehicle i is following vehicle $i-1$ and has the state variables x_i, $\dot{x}_i$, and $\ddot{x}_i$. By definition, $r_i = \frac{x_{i-1}-x_i}{h\dot{x}_i}$. We define the following predicates on the vehicles:

- $\textsc{Inv}_i(t_0) \triangleq \forall t \geq t_0,\ -\lambda < a_{min} \leq \ddot{x}_i(t) \leq a_{max}$
- $\textsc{InitAcc}_i(t_0) \triangleq a_{min} \leq \ddot{x}_i(t_0) \leq a_{max}$
- $\textsc{InitVel}_i(t_0) \triangleq a_{min} \leq \frac{\dot{x}_{i-1}(t_0) - \dot{x}_i(t_0)}{h} \leq a_{max}$

Theorem 2 (lane stability) *If* $\exists t_0$ *such that* $\textsc{Inv}_1(t_0)$ *and* $\bigwedge_{i=2}^{n}(\textsc{InitAcc}_i(t_0) \wedge \textsc{InitVel}_i(t_0))$, *then* $\bigwedge_{i=1}^{n} \textsc{Inv}_i(t_0)$.

Proof From Theorem 1 and Proposition 5, we have for any i: $\textsc{Inv}_i(t_0) \wedge \textsc{InitAcc}_{i+1}(t_0) \wedge \textsc{InitVel}_{i+1}(t_0) \Longrightarrow \textsc{Inv}_{i+1}(t_0)$. Therefore, we have by induction on i: $\bigwedge_{i=1}^{n} \textsc{Inv}_i(t_0)$. □

4.5 Computation of the "Velocity" and "Follow" Accelerations

As a conclusion, we compute the acceleration γ_v as follow, v_d being the desired velocity of the current lane, and v the velocity of the vehicle:

$$\begin{aligned} u &= \mu(v_d - v) \\ \gamma_v(v, v_d) &= \text{if } u > 1.962 \text{ then } 1.962 \\ &\quad \text{else if } u < -4.905 \text{ then } -4.905 \\ &\quad \text{else } u \end{aligned} \tag{11}$$

And we compute the acceleration γ_f as follow, v_f being the velocity of the leading vehicle, v the velocity of the vehicle, g the gap between both vehicles, and h the desired time headway:

$$\begin{aligned} u &= \frac{v_f - v}{h} + \lambda\Big(\frac{g}{hv} - 1\Big) \\ \gamma_f(v, g, v_f) &= \text{if } u > 1.962 \text{ then } 1.962 \\ &\quad \text{else if } u < -4.905 \text{ then } -4.905 \\ &\quad \text{else } u \end{aligned} \tag{12}$$

5 Design of the Hybrid Controller

To complete the design of our desired hybrid controller, there remains to chose the guard of the discrete initial transition, of the transition from `align-to-gap` to `go-to-main`, and of the transition from `prepare-exit` to `go-to-exit`. The initial guard must ensure that the newly created vehicle will not cause a collision in the entry lane, while the second (resp. third) guard must ensure that the merging (resp. exiting) vehicle will not cause a collision in the main (resp. exit) lane.

For the initial guard, we use the initial conditions of Theorem 2, where i is the newly created vehicle and $i-1$ is its `same_lane_front_car`. We call t_c the time at which the vehicle is created by the vehicle source. $\textsc{Inv}_{i-1}(t_c)$ is satisfied thanks to Assumption 2. Thus the guard of the initial transition must be $\textsc{InitAcc}_i(t_c) \wedge \textsc{InitVel}_i(t_c)$.

For the guard from `align-to-gap` to `go-to-main`, we use the initial conditions of Theorem 2, where i is the merging platoon, $i-1$ is its `side_lane_front_car`, and $i+1$ is its `side_lane_back_car`, as shown in Figure 8:

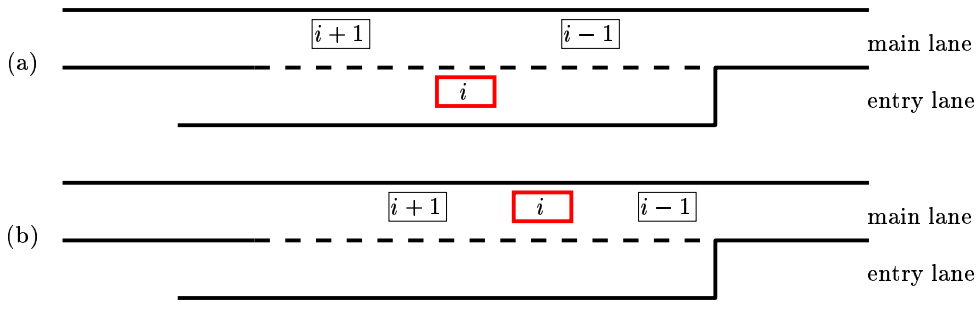

Figure 8: (a) Before the merging of vehicle i; (b) After the merging.

We call t_m the merging time. $\textsc{Inv}_{i-1}(t_m)$ and $\textsc{Inv}_i(t_m)$ are satisfied thanks to Assumption 2. Thus the guard of the discrete transition from `align-to-gap` to `go-to-main` must be $\textsc{InitVel}_i(t_m) \wedge \textsc{InitAcc}_i(t_m) \wedge \textsc{InitVel}_{i+1}(t_m) \wedge \textsc{InitAcc}_{i+1}(t_m)$.

$\textsc{Inv}_{i-1}(t_m) \wedge \textsc{InitVel}_i(t_m) \wedge \textsc{InitAcc}_i(t_m)$ will ensure that the vehicle i does not collide with vehicle $i-1$, while $\textsc{Inv}_i(t_m) \wedge \textsc{InitVel}_{i+1}(t_m) \wedge \textsc{InitAcc}_{i+1}(t_m)$ will ensure that the vehicle $i+$ does not collide with vehicle i. Since we are concerned with collisions, we only use the $\geq a_{min}$ part of the inequalities. Indeed, $\ddot{x} \not\leq a_{max}$ means that the newly created/merged vehicle is too far away from the vehicle in front, which is never a problem as far as safety is concerned.

The guard from `prepare-exit` to `go-to-exit` is similar except that it only involves the `side_lane_front_car`. This guard is $\textsc{InitVel}_i(t_e) \wedge \textsc{InitAcc}_i(t_e)$, where i is the exiting vehicle and t_e is the exiting time.

For each guard, the acceleration in $\textsc{InitAcc}_i(t)$ is the "raw" one, i.e., not the saturated one:

$$\ddot{x}(t_c) = \frac{\dot{z_1}(t_c) - \dot{x}(t_c)}{h} + \lambda\Big(\frac{z_1(t_c) - x(t_c)}{h\dot{x}(t_c)} - 1\Big)$$

$$\ddot{x}(t_m) = \frac{\dot{z_2}(t_m) - \dot{x}(t_m)}{h} + \lambda\Big(\frac{z_2(t_m) - x(t_m)}{h\dot{x}(t_m)} - 1\Big)$$

$$\ddot{z_4}(t_m) = \frac{\dot{x}(t_m) - \dot{z_4}(t_m)}{h} + \lambda\Big(\frac{x(t_m) - z_4(t_m)}{h\dot{z_4}(t_m)} - 1\Big)$$

$$\ddot{x}(t_e) = \frac{\dot{z_2}(t_e) - \dot{x}(t_e)}{h} + \lambda\Big(\frac{z_2(t_e) - x(t_e)}{h\dot{x}(t_e)} - 1\Big)$$

Figure 9 shows the resulting complete controller automaton. Thanks to Theorems 1 and 2, this hybrid controller ensures that there will *never* be a collision between the vehicles. However, it remains possible that a vehicle will be dropped out because it will be unable to safely merge before the end of its entry lane.

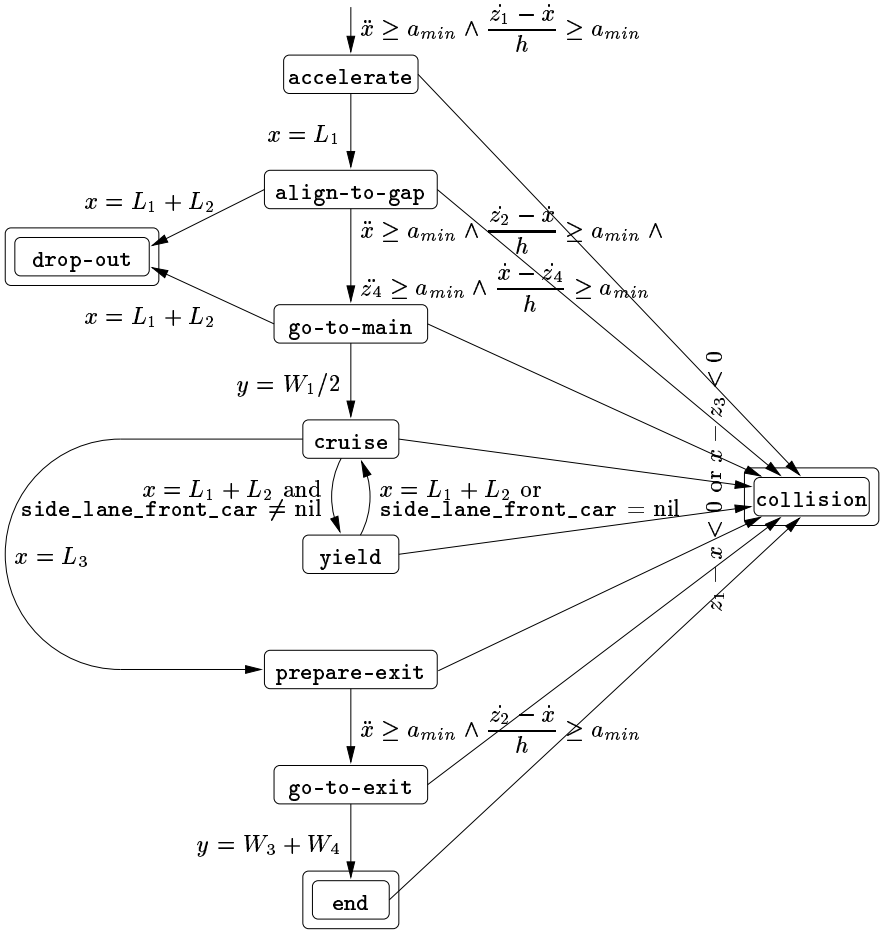

Figure 9: Final controller automaton.

This allows us to state the following highway stability theorem:

Theorem 3 (highway stability) *If all the vehicles driving along the highway of Figure 2 are controller with the hybrid controller of Figure 9, then there is no collision.*

Proof The highway of Figure 2 contains one main lane, three entry lanes, and three exit lanes. All the vehicles are created in either one of the entry lanes. Each vehicle i actually enters its entry lane at time t_i when it performs the initial transition into the `accelerate` state. This transition is guarded by $\textsc{InitVel}_i(t_i)$ and $\textsc{InitAcc}_i(t_i)$. When the vehicle 1 is created, there is no vehicle in front of it, so its acceleration is γ_v which satisfies Assumption 8: $\textsc{Inv}_1(t_1)$ is thus true. Therefore, by Theorem 2, $\textsc{Inv}_i(t_i)$ is true $\forall\ i \in [1, n]$. From this, by Theorem 1, $\forall\ i \in [2, n+1]$, there is no collision between vehicle i and its leading vehicle $i-1$. This ensures that there are no collisions in any of the three entry lanes.

Then, each vehicle j enters in the main lane at time t_j when it performs the `go-to-main` to `cruise` transition. Let t'_j be the time at which the vehicle j performs the `align-to-gap` to `go-to-main` transition. This last transition is guarded by $\textsc{InitVel}_j(t'_j)$ and $\textsc{InitAcc}_j(t'_j)$. At the first entry junction, since there are no vehicles in the main lane, the vehicles merge freely. Everything happens as if there was a single lane of vehicles, so Theorem 1 applies directly. At the two other entry junctions, we distinguish two phases: In the first phase there are not yet any vehicles in the main lane and the reasoning is the same as for the first entry junction, while in the second phase there are vehicles in the main lane. During the second phase, once the vehicle j is in the `align-to-gap` state, the vehicles in the main lane (in the `yield` state) consider j to be also in the main lane for the computation of their acceleration control law, even though the collisions are not

yet real ones at this stage. Then, once in the `go-to-main` state, by Theorem 2, $\text{INV}_j(t'_j)$ is true $\forall\ j \in [1,n]$. From this, by Theorem 1, $\forall\ j \in [2,n+1]$, there is no collision between vehicle j and its leading vehicle $j-1$. This ensures that there are no collisions in the main lane.

At either of the three exit junctions, an exiting vehicle k enters the exit lane at time t_k when it performs the `go-to-exit` to `end` transitions. Let t'_k be the time at which the vehicle performs the `prepare-exit` to `go-to-exit` transition. This last transition is guarded by $\text{INITVEL}_k(t'_k)$ and $\text{INITACC}_k(t'_k)$. At each exit junction, since there are no vehicles in the exit lane prior to the junction, the vehicles exit freely. Everything happens as if there was a single lane of vehicles, so Theorem 1 applies directly.

As a result, there are no collisions in the whole highway. □

Now we have claimed in Section 3.7 that other merging strategies can be designed and that our controller can be easily modified to implement them. More important, the result stated in Theorem 3 remains valid independently of the merging strategy chosen. Indeed this result rests only on the continuous control laws and on the conditions guarding the transitions that cause a vehicle to enter a new lane. And neither of these belong to the merging strategy.

6 Micro-Simulations

We present in this section the results of the microsimulations performed with the SHIFT hybrid system simulator [11] (see Section 2.2). The highway layout was given in Figure 2. There are three sources of vehicles, one at each entry junction.

6.1 Traffic Conditions and Travel Demand

The traffic conditions we have considered were given by the Houston Metropolitan Transit Authority [5]. They are presented in Table 3.

	Entry 1	Entry 2	Entry 3
Average flow	2000/hr	1000/hr	1000/hr
Inter-arrival time distribution (s)	Uniform [1.3,2.3]	Uniform [3.1,4.1]	Uniform [3.1,4.1]
Exit distributions	Exit 1 5% Exit 2 24% Exit 3 71%	Exit 1 0% Exit 2 25.6% Exit 3 74.4%	Exit 1 0% Exit 2 25.6% Exit 3 74.4%
Initial speed	11 ms^{-1}	22 ms^{-1}	22 ms^{-1}

Table 3: Traffic conditions and travel demand.

The intervals of the uniform laws are computed by dividing 3600 seconds (one hour) by the number of vehicles to create. For entry 1, it gives one vehicle every 1.8 second. Then the creation interval is between 0.5 second before and after this time. For entry 1, it thus gives the interval [1.3,2.3].

Table 3 also indicates the probability that a vehicle will leave the highway at exit i if it has not left already. For instance, a vehicle arriving at entry 2 has 25.6% chances of leaving at exit 2 and 74.4% at exit 3. Of course, since exit 1 is *before* entry 2, the vehicle cannot leave at exit 1, hence the 0%.

To each entry, we attach a vehicle source, in charge of generating the required number of vehicles, with the correct inter-vehicle time. Below is the SHIFT source code of our vehicle source. The `random` function returns a random number in the range $[0,1]$. We have used the `drand48` function from the standard C library on SOLARIS 8. The `source` hybrid automaton has two discrete states, `initial` and `produce`. The `initial` to `produce` transition creates the first vehicle, and the `produce` to `produce` transitions create the subsequent vehicles. Note that when the `produce` to `produce` transition guard is evaluated, the `new_vehicle` member variable still contains the previously created vehicle. This guard is the $\text{INITVEL}_i(t_c) \wedge \text{INITACC}_i(t_c)$ condition (see Section 5). To evaluate these conditions, each `Vehicle` outputs its longitudinal position `gxp` and its longitudinal speed `xDot`. The longitudinal position of `source` is kept in its own `gxp` member variable.

```
type Source {
  state
    continuous number duration := 0;
    number period;
    number vehicles_created := 0;
    number gxp;
    number initial_speed;
    array(Sink) sinks := [nil, nil, nil];
    array(number) sinks_percentages := [0, 0, 0];
    Vehicle new_vehicle := nil;
  discrete
    initial {duration' = 1;},
    produce {duration' = 1;};
  transition
    initial -> produce {}
      when duration >= period - 0.5 + random()
      define {
        number r := random();
        Sink new_sink := if r <= sinks_percentages[0] then sinks[0]
                         else if r <= sinks_percentages[1] then sinks[1]
                         else if r <= sinks_percentages[2] then sinks[2]
                         else nil;
      }
      do {
        duration := 0;
        vehicles_created := vehicles_created + 1;
        new_vehicle := create(Vehicle, source := self, sink := new_sink);
      },
    produce -> produce {}
      when duration >= period - 0.5 + random()
           and (xDot(new_vehicle) - initial_speed) / time_headway >= min_acc
           and (xDot(new_vehicle) - initial_speed) / time_headway
               + headway_param * ((gxp(new_vehicle) - gxp)
               / (time_headway * initial_speed) - 1) >= min_acc
      define {
        number r := random();
        Sink new_sink := if r <= sinks_percentages[0] then sinks[0]
                         else if r <= sinks_percentages[1] then sinks[1]
                         else if r <= sinks_percentages[2] then sinks[2]
                         else nil;
      }
      do {
        duration := 0;
        vehicles_created := vehicles_created + 1;
        new_vehicle := create(Vehicle, source := self, sink := new_sink);
      };
}
```

Then, a `source` is created by the following code:

```
global
  number time_headway := 0.6,
  number headway_param := 7,
  number g := 9.81,
  number max_acc := 0.2 * g,
  number min_acc := -0.5 * g,
  Source source1 := create(Source,
                           period := 3.6,
                           gxp := 2640,
                           initial_speed := 22,
                           sinks := [sink1, sink2, sink3],
                           sinks_percentages := [0.00, 0.256, 1.0]);
```

This vehicle source design can easily be modified to adopt for instance an exponential law, or a deterministic law for debugging purpose.

6.2 Results

We have run our micro-simulation during 60 minutes. Each vehicle was controlled by the hybrid automaton of Figure 9 and the acceleration control laws of equations (11) and (12). The λ parameter was set to 7 ms^{-2} (which is, as required by Proposition 4, greater than $-a_{min} = 4.905\, ms^{-2}$), μ to 7 ms^{-2}, and the time headway to 0.6 seconds.

As expected, the simulation ran without a single collision. The following table summarises the statistics at the end of the simulation:

	maximal merging distance	merged vehicles	dropped out vehicles
Entry 1	153.6 m	2085	0
Entry 2	168.8 m	1020	0
Entry 3	185.9 m	1021	0

Table 4: Microsimulation statistics after 60 minutes.

We have claimed in Section 3 that the criteria for evaluating a merging strategy are, ranked by decreasing importance: the absence of collisions, the absence of vehicle drop-out, and the smoothness of the traffic. We have proved in Section 2 that there can never be any collision (Theorem 3). Then our simulation results show that there are no vehicle drop-out. There remains to evaluate the smoothness of the traffic. Figure 10 shows the velocity profile in the main lane during the whole simulation. The merge portion of the second and third entry junction correspond respectively to the intervals [2880 m, 3360 m] and [7920 m, 8400 m]. The minimal velocity in these merge portions is 21.11 ms^{-1}, which is 24.6% less than the nominal velocity of 28 ms^{-1}, so this velocity profile is rather smooth. More important, there is very traffic little congestion, and the congestion does not propagate *outside* the entry junctions. In other words, the velocity reduction upstream and downstream the entry junction is 0%. It is interesting to note that our results are comparable to those of [28], where the upstream and downstream velocity reductions is also 0%, for comparable flow and desired velocities[1], but with a constant spacing policy (i.e., platoons of vehicle).

[1] Main lane desired velocity = 30.5 ms^{-1}, entry lane desired velocity = 23 ms^{-1}, and main lane flow = 2500 vehicles per hour

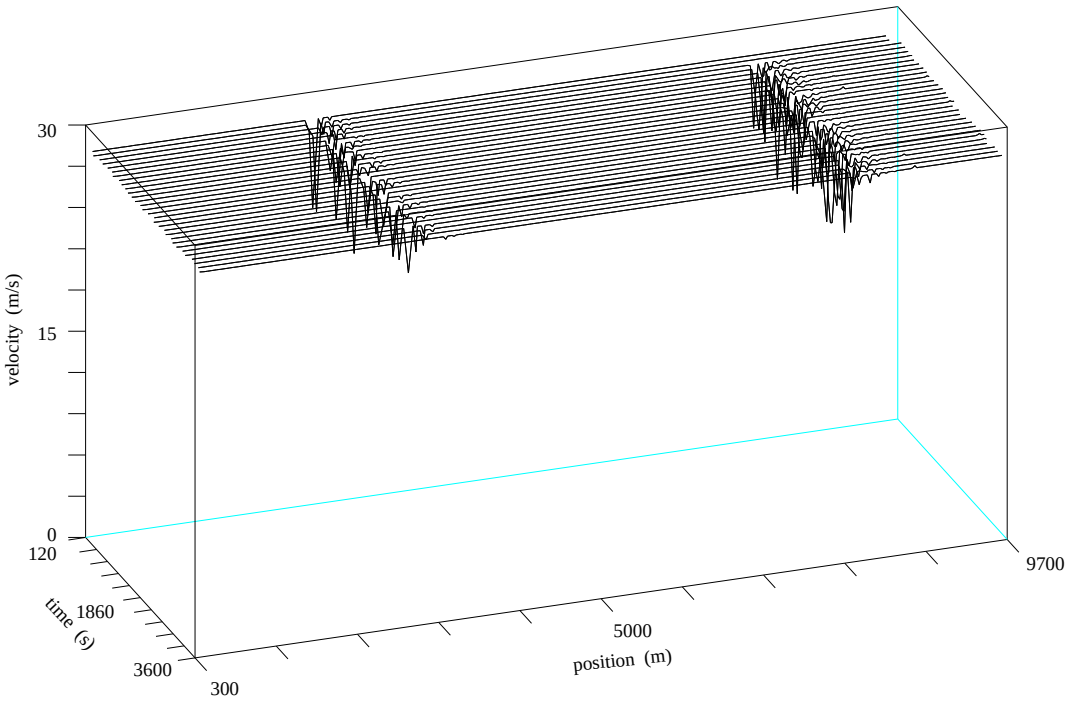

Figure 10: Velocity vs position in the main lane.

Finally, the maximal merging distance is only 185.9 m. This is very low compared to the length of actual merge lanes. Moreover, this figure is stable during the simulation. In fact, for entry 2, it is already 168.8 m after only 3 minutes. And for entry 3, it is already 185.9 m after only 6 minutes.

7 Conclusion and Future Research

We have presented in this paper a hybrid controller for autonomous vehicles driving on automated highways. This controller is in the "Autonomous Intelligent Cruise Control" class (AICC). The need for such intelligent autonomous controllers is motivated by the passenger safety as well as traffic congestion. Then we have advocated that designing a *hybrid* controller, where the discrete and continuous control parts are interconnected, is easier and safer than designing a purely continuous controller, where the differential equations controlling the movements of the vehicle would have to take care of both how and when to brake, accelerate, turn, merge,...

We have first presented the discrete control part, in the form of a finite state automaton which implements a merging and yielding protocol for our autonomous vehicles. Then we have studied and analysed two longitudinal continuous control laws: the first one is used to reach a desired speed in the absence of vehicles in front of it, while the second one is used to follow a vehicle in front of it without colliding with it. For both control laws, we have studied first the unsaturated case, and then the saturated case where the acceleration must remain within some technology bounds. We have proved two key results. First, a vehicle with the "follow" control law can never collide with its front vehicle, provided that some initial conditions are satisfied. Second, there can be no collisions in a lane of such vehicles provided that the same initial conditions are satisfied when each single vehicle is added to the lane.

Based on these two results, we have designed a complete hybrid controller, and we have proved that if all vehicles driving on the highway are equipped with it, there can never be collisions, even at the merging junctions. The only "bad" thing that can occur is that a vehicle can fail to merge and be subsequently dropped-out.

Finally, we have presented some microsimulation results, based on a highway design and traffic demand provided by the Houston Metropolitan Transit Authority. This microsimulation was programmed with

the SHIFT language for describing and simulating dynamic networks of hybrid automata, developed at UC Berkeley in the PATH group. These results show the practicability of our hybrid controller. The simulation was run for 60 minutes, as expected without a single collision, but also without a single dropped-out vehicle. Moreover, the maximal merging distance is only 185.9 m, which is very low compared to the length of the actual highways merging lanes. Finally, the velocity profile in the main lane was found to be very smooth, and proved not only that there was very little traffic congestion at the entry junctions, but also that this congestion did not propagate outside the entry junctions. This is one more advantage of our hybrid controller.

Now future research on the continuous control part could involve, for instance, studying the influence of jerk limitation, adopting a realistic lateral control law (see [7, 21, 24, 25, 27]), or coupling our longitudinal acceleration control law with other control laws deciding what throttle and brake inputs are needed to achieve the desired acceleration (see [27]). Concerning the discrete part, we could divide the merging portion (L_2 portion of Figure 3) into two portions: vehicles in the main lane would only be allowed to yield in the second portion. The merging strategy would then involve first targeting a gap, aligning to it, and adjusting the speed in the first portion, and then increasing this gap and actually merging in the second portion. We could then compare this strategy with ours in terms of efficiency, that is, the maximal merging distance and the traffic congestion.

Acknowledgements

The author would like to thank Prof. Darbha Swaroop (Texas A&M University) for his useful input on lane changing in AHS.

References

[1] M. Antoniotti, A. Deshpande, and A. Girault. Microsimulation analysis of a hybrid system model of multiple merge junction highways and semi-automated vehicles. In *IEEE International Conference on Systems Man and Cybernetics, SMC'97*, Orlando, FL, October 1997. IEEE.

[2] M. Antoniotti, A. Deshpande, and A. Girault. Microsimulation analysis of multiple merge junctions under autonomous ahs operation. In *IEEE Conference on Intelligent Transportation Systems, ITSC'97*, Boston, MA, November 1997. IEEE.

[3] P. Antsaklis. Hybrid systems: Theory and applications. *Proceedings of the IEEE*, 88(7), July 2000. Special Issue.

[4] K.J. Åström and S. Brufani. Manual control of an unstable system with a saturating actuator. In *IEEE Conference on Decision and Control, CDC'97*, pages 964–965, San Diego, CA, December 1997. IEEE.

[5] Houston Metropolitan Transit Authority. Houston metro corridor data, 1997.

[6] G. Baille, P. Garnier, H. Mathieu, and R. Pissard-Gibollet. The INRIA Rhône-Alpes CYCAB. Technical Report 0229, INRIA, Rocquencourt, France, April 1999.

[7] W. Chee and M. Tomizuka. Lane change maneuvers for AHS applications. In *International Symposium on Advanced Vehicle Control, AVEC'94*, Tsukuba-shi, Japan, 1994.

[8] C.C. Chien. *Intelligent Vehicle Highway Systems (IVHS): Advanced Vehicle Control System.* PhD Thesis, University of Southern California, 1994.

[9] D. Cho and J.K. Hedrick. Automotive powertrain modelling for control. *Journal of Dynamic Systems, Measurement, and Control*, 111(4), December 1989.

[10] P. Daviet and M. Parent. Platooning techniques for empty vehicle distribution in the PRAXITÈLE project. In *4th Mediterranean Symposium on New Directions in Control and Automation*, Maleme, Grece, June 1996.

[11] A. Deshpande, A. Göllü, and L. Semenzato. The SHIFT programming language for dynamic networks of hybrid automata. *IEEE Trans. on Automatic Control*, 43(4):584–588, April 1998. Research Report UCB-ITS-PRR-97-7.

[12] R.E. Fenton. A headway safety policy for automated highway operations. *IEEE Trans. on Vehicular Technology*, VT-28, 1979.

[13] A. Girault and S. Yovine. Stability analysis of a longitudinal control law for autonomous vehicules. In *IEEE Conference on Decision and Control, CDC'99*, Phœnix, USA, December 1999. IEEE. Research Report INRIA 3498.

[14] D.N. Godbole, J. Lygeros, and S. Sastry. Hierarchical hybrid control: A case study. In *Proceedings of the Conference on Decision and Control, CDC'94*, pages 1592–1597, Orlando, USA, December 1994. IEEE.

[15] R. Horowitz and P. Varaiya. Control design of an automated highway system. *Proceedings of the IEEE*, 88(7):913–925, July 2000.

[16] P. Ioannou and C.C. Chien. Autonomous intelligent cruise control. *IEEE Trans. on Vehicular Technology*, 42:657–672, November 1993.

[17] P.A. Ioannou, F. Ahmed-Zaid, and D.H. Wuh. A time headway autonomous intelligent cruise controller: Design and simulation. Research Report UCB-ITS-PWP-94-07, University of Southern California, April 1994.

[18] P.A. Ioannou and Z. Xu. Throttle and brake control systems for automatic vehicle following. *IVHS Journal*, 1(4):345–377, 1994.

[19] H. Jula, E.B. Kosmatopoulos, and P.A. Ioannou. Collision avoidance analysis for lane changing and merging. *IEEE Trans. on Vehicular Technology*, 49(6):2295–2308, November 2000.

[20] A. Kanaris, E.B. Kosmatopoulos, and P.A. Ioannou. Strategies and spacing requirements for lane changing and merging in automated highway systems. Research Report, Department of Electrical Engineering, University of Southern California, Los Angeles, USA, 1998.

[21] Y. Kanayama and B.I. Hartman. Smooth local path planning for autonomous vehicles. In *IEEE International Conference on Robotics and Automation, ICRA'89*, pages 1265–1270, May 1989.

[22] X.-Y. Lu and K.J. Hedrick. Longitudinal control algorithm for automated vehicle merging. In *IEEE Conference on Decision and Control, CDC'00*, Sydney, Australia, December 2000. IEEE.

[23] NAHSC milestone 2 report (Task C2: Downselect system configuration and workshop #3). Internal NAHSC document, 1996.

[24] V.K. Narendran and J.K. Hedrick. Autonomous lateral control of vehicles in an automated highway system. *Vehicle System Dynamics*, 23(4):307–324, 1994.

[25] W. Nelson. Continuous-curvature path for autonomous vehicles. In *IEEE International Conference on Robotics and Automation, ICRA'89*, pages 1260–1264, May 1989.

[26] T.S. No, K.-T. Chong, and D.-H. Roh. A Lyapunov function approach to longitudinal control of vehicles in a platoon. *IEEE Trans. on Vehicular Technology*, 50(1):116–124, January 2001.

[27] R. Rajamani, H.-S. Tan, B.K. Law, and W.-B. Zhang. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Trans. on Control Systems Technology*, 8(4):695–708, 2000.

[28] B. Ran, S. Leight, and B. Chang. A microscopic simulation model for merging control on a dedicated-lane automated highway system. *Transportation Research Part C: Emerging Technologies*, 7(6):369–388, 1999.

[29] S. Sekhavat and J. Hermosillo. The CYCAB robot: A differentially flat system. In *IEEE Intelligent Robots and Systems, IROS'00*, Takamatsu, Japan, November 2000. IEEE.

[30] S. Sheikholelslam and C.A. Desoer. A system level study of the longitudinal control of a platoon of vehicles. *Journal of Dynamic Systems, Measurement, and Control*, 1991.

[31] SHIFT web page: www-shift.eecs.berkeley.edu.

[32] S.E. Shladover. Review of the state of development of advanced vehicle control systems (AVCS). *Vehicle System Dynamics*, 24:551–595, 1995.

[33] D. Swaroop, J.K. Hedrick, C.C. Chien, and P. Ioannou. A comparison of spacing and headway control laws for automatically controlled vehicles. *Vehicle System Dynamics*, 23(8):597–626, November 1994.

[34] D. Swaroop, J.K. Hedrick, and S.B. Choi. Direct adaptative longitudinal control of vehicle platoons. *IEEE Trans. on Vehicular Technology*, 50(1):150–161, January 2001.

[35] H. Ye, A.N. Michel, and L. Hou. Stability theory for hybrid dynamical systems. *IEEE Trans. on Automatic Control*, 43(4):461–474, April 1998.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399