Optimization for fast and safe trajectories

Thierry Gruber

Kongens Lyngby 2013
M.Sc.-2013-115

DTU Compute
Technical University of Denmark
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk IMM-M.Sc.-2013-115

Abstract

The subject of this dissertation is optimizing the trajectory of a car on a highway. The goal is to give decisions to the driver about what speed to adopt and when to change lane in order to satisfy their desire for speed but still ensure safety.

A simulation framework has been programmed so that the car of study can evolve in an environment that simulates real situations. Within this simulation model, the goal is to find an optimal trajectory for the car of study. Optimal trajectories can be retrieved offline, where time backtracking is possible: the A* algorithm has been implemented to perform this search. But the aim is also to find an online algorithm to assist the driver in real-time: an agent-oriented algorithm has been programmed for a real-time assistance algorithm. Only safety and time has been taken into account for the optimality criterion; for the offline optimal search, safety is considered as a constraint and time is optimized. Making safety constraints vary for several optimal searches provides a range of optimal solutions that can be used as benchmarks to assess the performance of the online algorithm.

The implemented programs work and are computationally achievable. The optimal searches provide a various range of trajectories which enable to choose the best trade-off between time and safety. The agents have shown to take good decisions that make the trajectory safe and fast for basic tests. But when the scenario gets more complex, they may take irrational decisions. Nevertheless, this thesis shows the feasibility of such a program and provides a first flexible framework to be improved.

Preface

This thesis was prepared in fulfilment of the requirements for acquiring a M.Sc. in Computer Science and Engineering at the Danmarks Tekniske Universitet. It has been undertaken at the Centre for Accident Research & Road Safety of Queensland (CARRS-Q) in Kelvin Grove, Australia, between June 3^{rd} 2013 and November 8^{th} 2013. It has been supervised at CARRS-Q by Grégoire Larue and Andry Rakotonirainy, and from DTU by Niels Kjølstad Poulsen.

The thesis deals with the optimization on time and safety of the trajectory of a car on a highway. It presents the state-of-the-art, an implemented simulation model, the analysis of this model and the results of the conducted scenario tests.

Kelvin Grove, 17-November-2013

Thierry Gruber

Acknowledgements

The work underlying this thesis was supported by the Centre for Accident Research & Road Safety of Queensland (CARRS-Q) and the Danmarks Tekniske Universitet (DTU).

I would like to thank Grégoire for having set up this internship. He was very trustful and let me work autonomously, still he kept me on the right track, giving invaluable advices and feedback.

Thank you Andry for having supervised from a higher level, given the broad directions to take and being always opened to discussions.

Niels was thousands of kilometres away, though he kept track of my work and provided me good insights on control theory, although the internship did not end up focusing on that part.

I would like to thank all the people from CARRSQ who made my stay there an interesting and enjoyable experience.

Contents

Chapter 1

Introduction

The first section presents the motivation of the problem and why the driving task is crucial in our society. The second section will discuss about what aspects of this problem are dealt with and what goal is meant to be achieved. The third section describes the methodology and the structure of the present report.

1.1 The driving task

1.1.1 Types of transports

Transporting people and good has been an increasingly essential and challenging human need. Today the world is organized under a mobile paradigm thus transport needs to be efficiently handled. New approaches of transport are explored to ensure its sustainability in a fast-paced changing world, especially considering speed, safety, comfort and environmental concerns. One can distinguish three main kinds of transports [Van12]: mass transport, individual transport and digital transport. Mass transport refers to systems that are able to carry a large amount of people, such as buses, trains and ships. Individual transport, instead, corresponds to smaller systems that are able to transport one or a few passengers, such as cars, trucks and bicycles. The latter

has been exponentially growing since the popularization of the internet. It can be an alternative to some physical transports such as shown by videoconferencing or distance education. However, it cannot replace the inherent need of physical transport; instead physical transport is predicted to keep growing as digital technologies have indeed a counterpart, trigger more physical transactions and contacts and need physical infrastructures. Mass transport is safer and more environmental-friendly than individual transport. Nevertheless individual transport is preferred to mass transport: in the EU in 2010, 72.4% of passenger-kilometres are travelled by car and 72.5% of tonne-kilometres for inland good transport were travelled by truck. Indeed mass transport is logistically inefficient for individual purposes as it regularly halts, and has specific routes that may not deserve the targeted destination.

Whereas bicycle is flexible, environmental-friendly and favoured by a growing number of cities such as Copenhagen or Amsterdam, it does not suit to all people and longer distances. Individual transport systems are the most favoured especially for their flexibility and comfort and despite their risk. But risk is high and figures show that mortality on roads is serious. As given by the Australian Automobile Association (AAA) [Ass13], the economic cost of road crashes in Australia is $27 billion each year. The World Health Organization (WHO) estimated that in 2010, 1.2 million people were killed worldwide in road crashes and 50 million people were injured [Org10]. Still according to WHO, projections indicate that these figures will increase by about 65% over the next 20 years unless major changes are made in prevention and road safety. 95% of all accidents on roads are caused by drivers' errors such as miscalculation, drowsiness and driver impairments (for instance alcohol or speeding) [Shi78]. By getting safer and more efficient, intelligent systems seem a good trade-off between flexible individual transport systems and safe, efficient mass transport systems. Still, one must keep in mind that there could be other problems triggered by use of such techniques (for instance software-related) and they may not be suitable for all situations – for instance if everybody uses his car in town, it may trigger some traffic congestion.

1.1.2 Towards autonomous vehicles

Vehicle automation has been increasingly developed through Advanced Driver Assistance Systems (ADAS) for the automotive industry for half a century now. Shaout et al. (2011) [SCA11] traced the history of such systems. One of the earliest and most remarkable modern embedded systems was the Apollo Guidance Computer created at MIT in 1966 and used by NASA in late 1960s on the Apollo Space Program. Since then research in this field has kept increasing

and been largely used in the automotive industry. These systems are naturally real-time and multitasking. They are getting more numerous and complex in order to face the crucial need of improving safety on roads. Below is a list of the most common systems that exist and are put in mass production nowadays. It is not meant to be exhaustive but rather to show how diverse these systems are. Dates when the systems were developed in the automotive industry are given in parenthesis and may differ from the date when the systems were put in mass production.

- (Adaptive) Cruise Control (1960s): automatic control of the speed of the vehicle
- Autonomous Parking Assistance Systems (1990s): help the driver in manoeuvring into parking spaces
- Precrash Systems (2000s): detect and alert the driver to imminent accidents
- Drowsiness Detection Systems (2006): detect sleeping-patterns, alert the driver and in emergency cases apply braking
- Blindspot Information Systems (2011): detect and alert the driver to presence of objects in areas they cannot see without turning head
- Lane Departure Warning System (2001): detect and alert the driver to unintended lane change and may steer to correct trajectory

Driving assistance has been greatly improved for several decades and is still a key research topic in different domains of study such as control theory or artificial intelligence applied to driving and road safety.

1.2 Optimization for safe and fast trajectories

1.2.1 Problem statement

Finding optimal trajectories is relevant to achieve fast and safe travels. However an algorithm that suits to every driving situation is complex to implement. There are actually different approaches depending on the situation, as shown by the different designs and structures of the cars of the Stanford teams for different DARPA Challenges, organized by the Defense Advanced

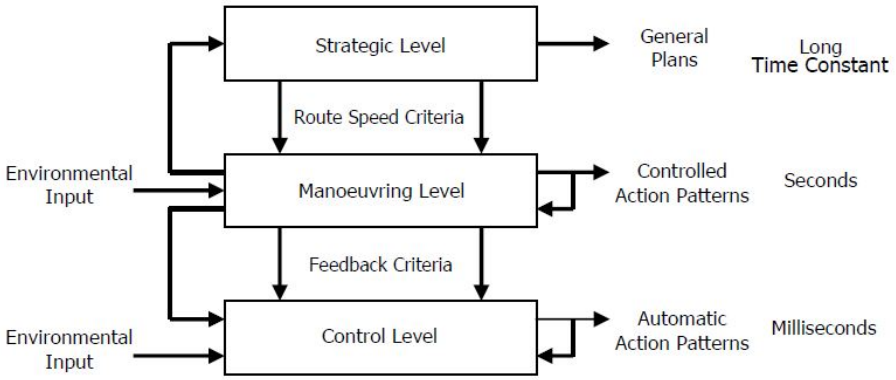


Figure 1.1: The hierarchical structure of the road user task (Michon, 1986) 1.1

Research Projects Agency (DARPA): Junior for the urban situation (Montemerlo et al., 2009) [MBB+09] and Stanley for the desert challenge (Thrun et al., 2006) [TMD+06]. This problem focuses on the particular environment of a highway; it is simpler than urban situations as sudden changes are less likely to occur. So the aim of this thesis is to find an optimal trajectory for a car on a highway of several lanes.

Based on the model of Michon (1986) [Mic86], given on Figure 1.1, the assistance can be provided at three different levels of skills and control.

- The higher level is the strategic level: it defines the general planning stage of the trip. Path optimization is done based on traffic information, roadmap configuration and given goals of time and speed. As an example, Junior from the DARPA Challenge [MBB+09] comports a global path planner to fulfil the goals of this level.
- The medium level is the tactical level where the driver handles the prevailing circumstances by manoeuvring controls for short laps of time: the driver deals with the direct environment around the vehicle for up to some hundreds meters. At this level the driver can choose to change lane, to adapt speed and also to meet the general goal given by the higher level, such as risk acceptance. Taking safe and intelligent decisions is crucial to suggest to the driver a trajectory he accepts that respects traffic and safety rules.
  - As an example, Stanley from the DARPA Challenge [TMD+06] was

equipped with smart computer vision systems and a global map of the trajectory. Its main task was to apply lateral offsets and adapt its speed to a fixed-based trajectory that was initially planned. By intelligently choosing these parameters, Stanley could avoid obstacles on the road at high speed and achieve fast progress along the course.

- The lower level is the control model where there is no decision but automatic action patterns. It directly interacts with the actuators and the time laps for such changes are in milliseconds. Both Stanley and Junior were equipped with such systems. Such methods enable very short-term tasks such as lane keeping.

The highway is supposed to be straight and infinitely long, which occult the need of a complex high strategic level. The car targets a given abscissa $Xgoal$. It is equivalent to being in the curvilinear lane coordinate system. This system is generally used for such problems; for instance the car of the Stanford team who won the DARPA challenge uses this coordinate system. The transformation from one base to the other is not orthogonal and thus does not keep angles and distances, however for highways where curvature values are low (under $1/500$ m^{-1}) errors introduced by these transformations are assumed to be negligible with respect to perception and control errors. But for higher curvature these assumptions are not valid [Van12]. Such cases are outside the scope of this thesis. At the higher level, only goals concerning risk acceptance, depending on the driver's profile, will be given. Control is supposed to be perfect so the lower level is not considered either. Other driving inherent tasks such as perception are supposed to be perfect. Therefore this problem focuses on the medium level where tactical choices have to be taken to compute an efficient trajectory. For this thesis, optimality takes into account time and safety and depends on driver's profiles. It is assumed in the simulations that drivers follow exactly the instructions from the software.

1.2.2 Research methodology

The goal was initially to find optimal trajectories considering that all speed and positions of all the other vehicles were known in advance for the whole trip. It is a first simplified step; however useful since optimal trajectories cannot be computed in real-time. Since this goal has been achieved and the assumption of knowing everything in advance was extremely strong, the goal has been pushed further in computing an "intelligent" trajectory online. What "intelligent" means will be described in details later. The implementation aims at showing feasibility of such a system by providing an adapted simulation framework, and provides a first simple model to find a trajectory and to assess it. The

car of interest evolves in an environment with other vehicles: it is unrealistic to predefine the speeds and positions of the other vehicles as they should react to changes in the environment, such as vehicles' movements, particularly the controlled car. Hence these vehicles have been equipped with a car-following model to simulate basic car behaviour. Optimal trajectories computation has been adapted to this study framework; they are computed offline and will be used as a benchmark to measure the performance of the "intelligent" online computation. Performance measurement of the online algorithms will be based on a series of test scenarios. All the models have been programmed and integrated together from scratch.

Notations The variable t is the time in seconds. The variables referring to the car of study have a subscript $_i$ and this car is called the *ego car*, also the driver of this car is called the *ego driver*. The variables referring to the car directly in front and the one directly behind the controlled car have respectively a subscript $_{ft}$ and $_{bk}$. The variables referring to any car have no subscript. The position is given by x in meters and refers to the front of the car, the speed or velocity is given by v in meters per second, the acceleration is given by a in meters per seconds squared. v is not to be mistaken with v_{md} which is the maximum desired speed of a driver, or v_{max} which is the maximum speed allowed on the road. The headway is the distance between a car and the car directly in front: $h_{k,ft}(t) = x_{ft}(t) - L_{ft} - x_k(t)$ for the vehicle k. The lane ID of the car is given by the variable l. The constants L and m respectively refers to the length and the mass of the vehicle. Every driver has a constant reaction time τ. g and b are respectively the maximum and minimum acceleration that the vehicle is capable to reach. Other constants and variables will be detailed when being introduced. For the object-oriented implementation, classes have the prefix HW for HighWay.

1.3 Structure of the thesis

To begin with this thesis reviews the state-of-the-art and the considered techniques. Then it presents the structure of the models used and the algorithms to perform optimal trajectories search and online "intelligent" assistance. Finally in the results section, the implemented simulation framework and the algorithmic techniques are discussed based on simulation tests and models are compared in order to measure the performance of the "intelligent" algorithms on a series of test scenarios.

Chapter 2

State-of-the-art

This chapter first introduces the existing models that describe the driving components and their interactions. In the second section techniques for trajectory optimization are presented.

2.1 Existing models for Driver and Vehicle

This topic has been investigated and a review presented by Rakotonirainy et al. (2007) [GRGN07]. This section first presents driver models, which relates to decision making and risk assessment. Then longitudinal models are presented: they model the speed or the acceleration of the vehicle depending on the vehicle directly in front. So they simulate vehicle behaviour on a single lane, which is needed for this problem. Finally other models are presented; they are mainly related to the control layer. Those models will be briefly presented not to restrict the literature to the exclusive scope of the conducted implementation but also to enable the reader to go further by providing the basis for future improvements.

2.1.1 Driver models

2.1.1.1 Decision making by risk assessment

Driving is a complex task and the driver is exposed to a certain risk at any moment. There are two types of risk according to Glaser (2011) [GVGM11]):

- objective risk: refers to the probability of being involved in a crash and the severity of that potential crash.
- subjective risk: refers to the driver's own assessment and depends on his personality

Objective risk According to Glaser (2011) [GVGM11], a common approach for assessing objective risk is to consider two criteria:

- the probability that the event occurs
- the severity of the resulting situation in case the event occurs

The latter has been extensively studied and researchers accept broadly the common criterion of Energy Equivalent Speed (EES) (Mills et al., 1984 [MH84], Zeidler et al., 1985 [ZSS85]). It expresses the deformation energy between two vehicles colliding and is given by $EES = \frac{2 \cdot m_i}{m_i + m_{ft}} \cdot (v_i(t_{crash}) - v_{ft}(t_{crash}))$ where t_{crash} is the time at which the collision occurs. The higher the EES is, the more serious the crash is. Based empirically on data, a relation between EES and probability of injuries have been given by Glaser et al. (2011) [GVGM11] and is shown on Figure 3.1 of Section 3.1.1.1. It relates the EES to the probability of a moderate injury ($MAIS_{>2}$, Maximum Abbreviated Injury Scale).

The former criterion can be given by different relevant indicators, that help assessing how dangerous a situation is. The headway is the distance between the ego car and the car directly in front: $h_{i,ft}(t) = x_{ft}(t) - L_{ft} - x_i(t)$. The smaller the headway is, the more dangerous the situation is. However, this indicator only can be highly misleading because danger depends also on the speed of the vehicles. A headway of 10 meters is probably very dangerous on a highway situation but can be safe in a city-driving situation. In the study of Vogel (2002) [Vog02], two indicators are used. The first one is the time headway

δT. It is measured by taking the difference of time between two vehicles reaching the same position x: $\delta T_{i,ft}(x) = t_{ft}(x) - t_i(x)$, with $t_k(x)$ the time at which the vehicles k reaches the abscissa x. Vogel explains that short time headway potentially generate dangerous situations therefore can be taken into account to assess risk. He discusses also about the Time To Collision (TTC) criterion: it is defined by the time that would pass until a crash occurs with a car in front if both cars keep the same speeds. It is infinite if the car in front is faster, otherwise the TTC is: $TTC_i(t) = \frac{h_{i,ft}(t)}{v_i(t) - v_{ft}(t)}$. TTC actually indicates with more precision the occurrence of a dangerous situation since speed is taken into account.

Subjective risk Ranney (1994) [Ran94] gives a review of the models of driving behaviours: some of the most cited ones are the *Zero-Risk Model*, the *Risk Avoidance*, the *Task Capability* and the *Risk Homeostasis*. Summalla and Naataneen (1988) [Sum88] suggest that the driver targets zero risk of crash, whereas Svenson et al. (1985) [SFM85] assume that the driving task does not involve risk assessment, and that hazardous situations only make the driver takes risk into account for avoiding collisions. Fuller (2005) [Ful05] even claims that risk of collision is not relevant and instead task difficulty is taken into account in the decision-making process. According to Wilde (2001) [Wil01], a driver accepts a certain risk threshold which varies between individuals. Factors taken into account would be time, fuel, cost of a fine and expected crash cost: this concept is called *Risk Homeostasis*. It is a common concept for risk assessment on which the team of Rakotonirainy has based their work [GRGN07]. I will consider a model based on this concept as it is more general, flexible and able to take into account diverse driver's profiles. It could be easily adapted to other concepts, by modifying the limits of risk and standardizing profiles.

2.1.1.2 Driver's profiles

Driver's profiles is a concept related to *Risk Homeostasis*. They determine patterns of driving behaviour and classify drivers depending on their most common behaviour on the road. Risks thresholds can be associated to profiles. Inspired by the social theory work of Abric (2003) [Abr03] and based on questionnaires on driver's preferred attitudes and mindsets, the French project LAVIA on Intelligent Speed Adaptation (ISA) has derived three types of profiles [LS11]. These profiles are represented by the risk the driver is willing to take as a function of the potential cost that this risk could trigger, as explained by Rakotonirainy [GRGN07]. These models are very simple since a driver may have different risk assessments depending on several factors, such as drowsiness or even her mood. But they aim at giving a classification and general behaviours,

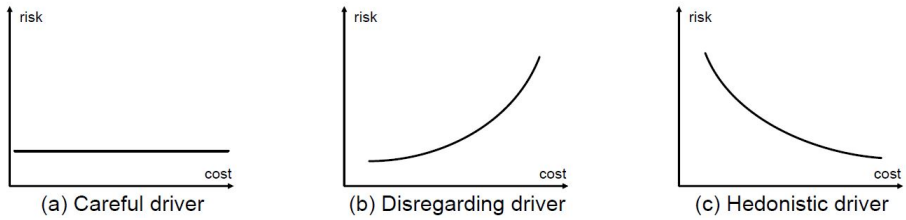


Figure 2.1: Risk functions of different driver profiles

rather than providing a very accurate description of risk assessment. In order to use this model, cost and the evaluation of risk need to be clearly defined. These profiles are shown on Figure 2.1 and are defined in this way:

(a) Careful driver: the driver maintains a constant level of risk independent from the cost. For example, if the vehicle is blocked by a slow car in front that makes the cost of the trip increases (as the time for the trip will be longer), the driver will not take more risky decisions in order to overtake and gain time.

(b) Disregarding driver: the driver accepts higher risk thresholds as the cost increases. In the example described previously, the driver would consider more risky decisions to overtake, achieve higher speed and reduce the cost of the trip.

(c) Hedonistic/Pragmatic driver: conversely, this driver accepts higher risk when the cost is low but reduces the risk threshold as cost increases. For example, this driver will go at high speed on a highway but would decelerate when approaching speed radar as the cost is potentially higher (the driver is likely to get a fine).

2.1.1.3 Other characteristics

Homeostatic and circadian processes represent sleep regulation of the human body: they model driver's tiredness over time (Daan et al., 1984 [DBB84], Borbet al., 1999 [BA99]). It could be taken into account to adapt the system to the driver's attention, and adapt his characteristic (such as risk acceptance level and reaction time) in function of his tiredness. However these are more complex details that are not required by the scope of this study and they will not be taken account in the model.

2.1.2 Longitudinal Control Models

For our purpose, models on single lane are relevant as lateral position is discretized with lanes. Some of the most cited one are presented below. A study has been conducted to select an appropriate longitudinal model for simulation in the implemented program: it is presented in the last paragraph of this section.

2.1.2.1 Newell's model

Newell is one of the first to suggest a non-linear model for velocity-headway relations in 1961 [New61]. He based his theory upon the work of Gazis et al. (1959) [GHP59]: they state that the non-linearity is necessary at least to represent the steady-state relation between average velocity and average headway, which was well-known for being non-linear. He suggests the formula:

$$v_i(t+\tau_i) = v_{md,i}(1 - e^{-\frac{\lambda_i}{v_{md,i}} \cdot h_{i,ft}(t)})$$

with λ_i a parameter to be fixed. He explains that there is no special motivation for it except that it has approximately the correct shape and is reasonably simple.

2.1.2.2 Gipps' model

Gipps's model (1981) [Gip81] calculates a safe speed with respect to the preceding vehicle based on limits of the performance of the driver and the vehicle. Gipps gives two inequalities to model the behaviour of the car. The first constraint ensures that the speed will not exceed its driver's desired speed by limiting the acceleration when the speed is too close to the desired speed. This inequality is purely descriptive and has been elaborated to fit empirical observations. It prevails when the vehicle in front is too far.

$$v_i(t+\tau_i) \leq v_i(t) + 2.5 \cdot g_i\tau_i(1 - \frac{v_i(t)}{v_{md,i}})\sqrt{0.025 + \frac{v_i(t)}{v_{md,i}}}$$

The second constraint relates to braking. The vehicle should be able to stop if the car in front performs an emergency braking. From physical equations, considering the reaction time of the driver and the maximum decelerations of the cars, the second inequation on the speed is given by:

$$v_i(t+\tau_i) \leq -b_i\tau_i + \sqrt{(\tau_i b_i)^2 + b_i \cdot h_{i,ft}(t) - v_i(t)\tau_i + \frac{v_{ft}(t)^2}{b_{ft}}}$$

This model has shown to mimic well the behaviour of real traffic, although it is idealistic as no crashes would occur in such traffic simulation.

2.1.2.3 Intelligent Driver Model (IDM)

The Intelligent Driver Model has been presented by Treiber et al. (2000) [THH00]. It also expresses velocity as function of headway and desired speed. The tendency to accelerate on a free road is captured by the term $g_i \cdot (1 - (\frac{v_i(t)}{v_{md,i}})^\gamma)$ and the tendency to brake with deceleration while getting closer to the vehicle in front is expressed by the term $-g_i(\frac{{d_{i,ft}}^*(t)}{d_{i,ft}(t)})^2$ in the formula:

$$a_i(t+\tau_i) = g_i \cdot (1 - (\frac{v_i(t)}{v_{md,i}})^\gamma - (\frac{{d_{i,ft}}^*(t)}{d_{i,ft}(t)})^2)$$

γ is an adjustment parameter, $d_{i,ft} = x_{ft} - x_i$ is the spacing between the front of the two cars and ${d_{i,ft}}^*$ the desired spacing given by ${d_{i,ft}}^* = v_i(t) \cdot \tau_i + L_j + max(0, (\frac{{v_i}^2(t)}{2b_i} - \frac{{v_j}^2(t)}{2b_j}))$. It varies dynamically and captures the concept of the second Gipps' condition. This distance is the minimum safe distance for the car to be able to stop in case of an emergency braking, as detailed in Section B.1.1 (Appendix B).

2.1.2.4 Field Theory's model

Ni (2013) [Ni13a] based his work on precedent models, included all the ones detailed previously, and derived a "unified model" from them. He considered the driver-vehicle-environment system from a physical point of view and made an analogy with forces in physics. In his model the acceleration varies positively with a "flow gravity" force, making the driver go forward along the road, and negatively with a resistance force (due to the desired speed) and a repulsive force from the car in front. The equivalence of the force derived from physics can be for instance a stress in the driver's mind which makes them change their action and so the motion of the car. His model agrees with the concepts detailed above, and is actually the same as the intelligent driver model but the repulsive force is exponential instead of quadratic:

$$a_i(t+\tau_i) = g_i \cdot (1 - \frac{v_i(t)}{v_{md,i}} - e^{\frac{{d_{i,ft}}^*(t) - d_{i,ft}(t)}{d_{i,ft}(t)}})$$

with the same notation as the IDM. According to Ni, this model gives a good ratio quality/complexity but tends to predict small estimates of acceleration.

2.1.2.5 Other approaches

Other models exist for longitudinal control and tackle the problem from different points of view. Brackstone (1999) [BM99] gives an historical review and Ni (2013) [Ni13b] refers to other models to elaborate his unified theory. Particularly, another approach which has not been evoked is the rule-based one. Kosonen (1999) [Kos99] has elaborated one in which depending on the situation, rules give a convenient speed to target. These rules are based on the desired of speed, the headway and the previous states.

2.1.2.6 Evaluation of Longitudinal Models

Test simulations have led to an evaluation and a comparison of Newell's, Gipps', IDM, and Field Theory models; this is presented in Appendix A. These models are tested on three scenarios:

- acceleration on a free road from standstill until a desired speed
- deceleration until standstill to stop before an obstacle
- car-following on highway conditions (high speed)

Results have been compared to empirical data for acceleration and deceleration performance. For the car-following performance, it has been checked whether safe distances were respected. As explained in Appendix A, Newell's model lacks of complexity while Gipps' and Ni's models give good performance but they are all outperformed by the IDM. As a result IDM will be used in the simulation program.

2.1.3 Other models

Other models are necessary for a more complex and accurate description of the vehicle. They are not used in the model of this thesis but they are given here as tracks to go further:

Bicycle model A simple and broadly accepted vehicle model is the bicycle model (Peng and Tomizuka, 1990) [PT90]. The lateral dynamics are compressed and the 4-wheels vehicle is simplified to a 2-wheels one: one front wheel and one

rear wheel. There is neither pitch nor roll dynamics and the vehicle is assumed to have planar motion.

Lateral models The lateral motion of the vehicle is given by a set of equations. There are different formulations of lateral models (for example kinematic and dynamics) depending on what situation it is (lower or higher speeds) and if simplifying assumptions are reasonable (for instance, slip angles of the wheels at zero for lower speed). This has been studied and detailed by Wong (2001) [Won01], Milliken (2003) [MKMM03] and different situations have been described by Rajamani (2011) [Raj11]. Abstract representations related to driver's mind has been suggested by Ni (2013) [Ni13c]: he uses potential fields to describe the 'mental force' that makes the driver keeps lane.

Cognitive model This model of driver behaviour is within a framework with underlying psychological theories based upon a two-way control paradigm (Salvucci, 2006) [Sal06]. Driver's perception is reduced to a near point and a far point. The steering angle and acceleration given as input by the driver are derived from these parameters.

Optimal control models Kleinman et al. (1970) [KBL70] have pioneered work in this topic, especially to develop the response characteristics of the human operator due to an external noise input that disturb the driving environment.

Lane-changing models Gipps provides a pioneer model to simulate lane-changing decisions (1986) [Gip86], but it focuses on urban traffic situations. Treiber et al. (2007) have extended the IDM model considering multi-lanes scenarios and suggests a complex model that can be adapted to a more various set of scenarios.

These models are outside the scope of this thesis since lateral position is discretized with lane IDs, which supposes that the vehicle keeps lane perfectly.

2.2 Trajectory optimization

Trajectory optimization is a very wide topic. A vast literature exists on such algorithms and provides techniques for different purposes. For this thesis, optimization will be based on time and safety. Some of these algorithms provide optimal solutions but they require strong assumptions and are costly, while other algorithms provide suboptimal but computationally achievable solutions.

2.2.1 Optimal trajectory search

2.2.1.1 Algorithms for optimal search

Finding an optimal trajectory supposes that the whole space of search is known. For the driving problem, dynamic objects are present on the map: as a result the space includes a time dimension. As a consequence real-time algorithms cannot guarantee to find an optimal trajectory. Here are two ways for retrieving an optimal trajectory for this problem:

- supposing that the trajectories of the other cars are known and fixed: the whole space of search is known exactly. Any scenario could be retrieved *a posteriori* and the algorithm be ran to find optimal trajectories. However this is a very strong assumption for this problem as the other cars should react to other vehicles' movements and may have different behaviours if the ego car has a different behaviour. It is unrealistic and the optimal solution retrieved would not be meaningful.
- supposing that the other cars follow a simulation model and run the algorithm within this simulation frame only. This approach is more complex as the environment varies and depends on the actions of the car. It cannot be run on any scenario *a posteriori*. Instead, initial conditions are given, and the algorithm runs within a simulation model: for example other cars are modelled with IDM. It is more realistic, therefore this approach has been chosen for the implementation.

Discretization Given a simulation model, finding such an optimal trajectory is relevant as it enables to compare and evaluate the performance of a real-time algorithm. Russell and Norvig [RN09] give a review of AI techniques, including search problems. Most of them suit discrete problems but the driving problem is continuous as most of real problems. This task is complex, time is part of

search space and constraints and rules about vehicle and driver and traffic need to be respected. As a result, continuous techniques such as gradient-based, linear programming or convex optimization are not adapted. One way to avoid this issue is to discretize the problem. Sampling-based techniques are appropriate to adapt the problem but optimality may be lost depending on the technique. However, by choosing appropriate discretization steps, the algorithm can compute the solution in a reasonable time and give an "acceptable" suboptimal solution. "Acceptable" means that errors are sufficiently low to be considered negligible; it will be assessed when defining the model of the implementation.

There are different ways to tackle the problem for discretization when the space is at least two-dimensional ((x,y) coordinates), such as cell decomposition, exact cell decomposition or skeletonization. But the driving environment is particular because longitudinally-oriented; besides for this thesis lateral positions are discretized on lanes. Environment sampling-based techniques are particularly adapted for the driving task as Vanholme explains [Van12] while referring to many projects based on it, including Stanley, the Stanford Robot that won the DARPA Challenge [TMD+06]. Longitudinal speed is discretized which gives a set of possible trajectories at any moment for each lane. The discretization step needs to be well chosen so that the algorithm has a reasonable running time but still outputs an acceptable suboptimal solution.

2.2.1.2 Description of A*

For discrete optimal search problems, the A* algorithm is optimal if the heuristic is admissible and consistent, and only more accurate/informed heuristic can improve search. Other adapted algorithms such as IDA* could be used at a low extra time cost if memory usage needs to be kept low. This thesis focuses on the structure of A* as it is the standard algorithm for such search. The A* algorithm is a best-first search extended from Dijkstra's algorithm. It finds a least-cost path from an initial node to one goal node. Its strength and advantage over Dijkstra's is the use of heuristics that estimates the distance from a given node to the closest goal node. The representation of a node is atomic: a node is reduced to its associated cost. The cost f of a node n is given by

$$f(n) = g(n) + h(n)$$

with $g(n)$ the effective path cost from the initial node to the node n, and $h(n)$ the heuristic cost of the node n.

Heuristics A heuristic enables informed search to decide better what branches to follow first, by evaluating the cost to a goal node. The heuristic needs to be admissible so that A* effectively computes the shortest path. A heuristic h is admissible if it never overestimates the cost of reaching a goal: it is optimistic. For every node n, $h(n) \leq h^*(n)$ with $h^*(n)$ the cost of the shortest path from n to a goal node. If the heuristic h is admissible and consistent, A* is the optimal best-first search algorithm as it finds an optimal solution by expanding fewer nodes. A heuristic h is consistent if for every edge (x, y) of the graph, where d denotes the cost of that edge, $h(x) \leq d(x, y) + h(y)$; in other words the path cost is estimated in an incremental way without taking any step back. If $h(g) = 0$ with g a goal node, an induction on the length of the best path from a node to the goal node proves that a consistent heuristic is also admissible. Only the heuristic could be improved to achieve a better performance algorithm. However, with a more complex heuristic, the algorithm will take more time to compute the successors of a node: that is why a trade-off is needed between the accuracy of the estimation and its computation cost.

Comparison to uninformed search The branching factor b is the average number of successors nodes derived from a given node, and d the depth of the optimal solution. The complexity of breadth-first search is $O(b^d)$. This is also the worst-case performance of A*. However if the heuristic h is close enough to the optimal heuristic h^* (if $|h(x) - h^*(x)| = O(log(h^*(x)))$), the complexity is polynomial in d [Pea84]. That is why the A* algorithm can be a lot faster but also requires a good heuristic to make it performs better than a simple breadth-first search.

2.2.2 Online algorithms for suboptimal solution

For a real-time assistance, online algorithms need to be considered. They provide suboptimal solutions but the wide range of techniques enables to select appropriate ones for a specific problem.

2.2.2.1 Locally optimal search

Vanholme (2012) [Van12] gives a review of such algorithms; all of them output locally (sub)optimal solutions, considering only a short time length for the trajectory computation. They are of two kinds: sampling-based or direct. The former discretizes the solution space and select a best solution whereas the latter directly outputs a trajectory and leaves out the evaluation step.

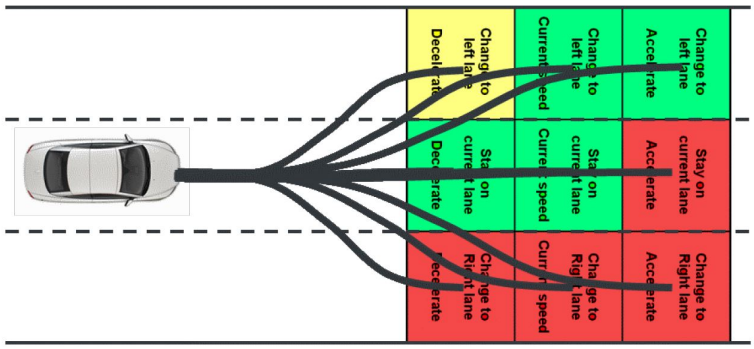


Figure 2.2: Manoeuvre grid representation

Environment-sampling-based An environment-based technique discretizes longitudinal and lateral speed, computes several trajectories, evaluates them based on a defined metrics and selects the best one. In the grid algorithm of Glaser et al. (2011) [GVGM11], 3 possible manoeuvres are possible per lane: braking, keeping the same speed or accelerating. Solutions at a given time result in 9 possibilities which are represented on the manoeuvre grid on Figure 2.2.

Expert systems They are direct systems based on if-else rules. More elaborated versions are based on fuzzy logic as in the HAVEit project [Van12] or probabilities, using Bayesian Networks. Running times are low and they are more suited to accept traffic rules and vehicles constraints. Besides they are intuitive to understand and are based on human reasoning. However, the number of rules should be high in order to capture most of driving situations and this is not suited to unexpected behaviours. All situations cannot be considered and it could result in hazardous decisions; moreover it is focused on the very present and expert systems are not made for tactical level decisions.

2.2.2.2 Towards more intelligent search

Locally optimal search techniques are not satisfying as they do not take into account the big picture of the problem: they are blinkered to the very present moment. More intelligent search techniques have been developed for a few decades.

Probability-based techniques In the driving task, many elements are uncertain, as they depend on behaviours of the other cars. Uncertainty is inherent to the driving task and thus is crucial to a safe and reliable system. Probabilities over possible environment configurations or action outcomes are a way to take into account uncertainty. Some of the driver/vehicle models, including longitudinal, lateral and driver's models have been integrated together to target a consistent model for the driving task, including probability-based techniques. For instance, Cheng and Fujioka (1997) [CF97] propose a hierarchical drivel model whose safety judgment is based on fuzzy logic. Fuzzy reasoning is close to human thinking because truth values of booleans are not binary but range in degree from 0 to 1. These techniques are based on a human built logic and are not able to learn which make the model not suitable to adapt to new situations and various environments. Another technique is Bayesian networks [SSW10]: it is a data structure that incorporate probabilities and particularly dependencies among variables. Such structures have been used for direct systems. It is a relevant technique which is based on model conceptually understandable because built by human and moreover designed to be able to learn and adapt. However, these models are able to deliver decision for a particular task at a given moment. They are not designed to planning and thinking at a higher level. They could be used inside a more complex model able to plan and take decisions for specific tasks at various tactical levels.

Planning algorithms Planning algorithms take the problem from a higher perspective and elaborate a plan of actions in order to achieve one's goal. In classical planning, the world is described with factor represented states which enables a richer description than the atomic one in search problem, where the state is reduced to a cost. Planning algorithms may suit because they are targeting the tactical layer this thesis aims to tackle. However they are generally used for problems such as scheduling the operations of spacecraft, factories or military campaigns, in which there are many different defined tasks to accomplish and available resources to consider. They are more suited for offline computation or in environment where objects are static. In the driving task, the world is unpredictable and continuously evolving, which makes planning a sequence of tasks in advance a tough challenge: decisions cannot be taken too much time in advance. Besides only a few "planning" decisions need to be taken; instead the acceleration needs to be calculated in real time. Nevertheless, getting a step higher and trying to see further than an immediate decision is what a driver actually does and this concept of rational and planned decisions is intrinsic to the driving task. Such behaviour is realizable by agent-oriented programming.

2.2.2.3 Agent-oriented search

The concept of rational agent is widely used in artificial intelligence. An agent perceives its environment through sensors and acts upon that environment through actuators. A human driver is a human agent who meets these requirements: a driver senses the road environment with his eyes, ears and other organs and acts through his hands, legs and so on. A robotic driver can have cameras and other sensors and acts in the environment by actuating motors to help driving the car and display/tell a message to assist the driver. Between perception and action, the agent needs to think and make right choices. It needs to be rational and to do the 'right thing'. Evaluating this 'right thing' is made by performance measure; metric needs to be well designed in order to properly assess agent's efficiency. A rational agent should choose actions that are expected to maximize the performance measure given its percept sequence, metric for the measure and the built-in knowledge it has. An agent is built on an agent program that implements the agent function from percepts to action, and run on devices that are the architecture. In the driving problem the architecture is integrated computing devices to the cars with physical sensors and actuators. Agents programs can be based on different principles.

BDI model The Belief-Desire-Intention model is a common paradigm to describe components of an agent. It is used in many fields including psychology, economics, philosophy or artificial intelligence. The agents have three main concepts:

- Beliefs: information the agent has about the world. They can be wrong.
- Desires: things that an agent would like to achieve. They can conflict.
- Intentions: things that an agent has committed to achieving. More concrete than desires, they cannot conflict, are possible and persistent. They are usually based on a utility measure (internal performance measure).

Figure 2.3 shows the program design of such an agent. Beliefs are based upon percepts and desires possibly re-evaluated upon them. Intentions are derived from beliefs and desires and a plan is built in order to achieve these intentions. The agent executes actions of the plan and impacts on the environment. Then it gets new percepts and updates its beliefs. The program needs to be built so that plans are wisely elaborated (no need to plan too much in advance), possibly repaired (if something goes wrong, need contingency plans). Thus

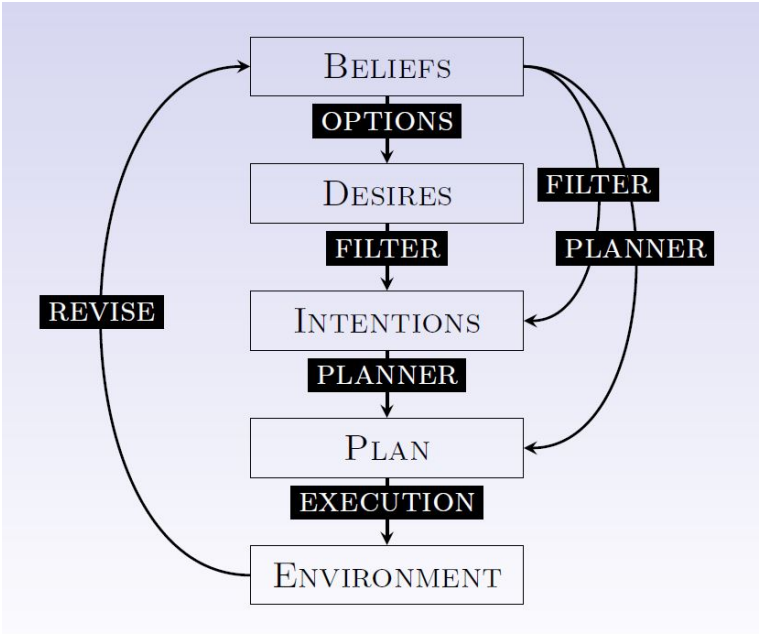


Figure 2.3: BDI model (AI course of DTU, 2012)

intentions need to be reconsidered but neither too seldom nor too often. An efficient trade-off needs to be elaborated. Although this paradigm is common, it comports its limitations: for example this model lacks of learning abilities, and three attitudes may not be sufficient to fully describe a rational agent. Other more flexible and universal models have been suggested, especially by Russell and Norvig.

Modern classes of intelligent agents Russell and Norvig [RN09] group agent programs in five classes that are nowadays the references for agent-oriented programming:

Simple reflex agent This agent selects an action based on the current percept and condition-action rules. It is extremely simple but therefore lacks intelligence. A little bit of non-observability can trigger serious issues.

Model-based agent One solution is to keep track of the part of the world it cannot see now by using percept sequence (history of percepts) and a model of

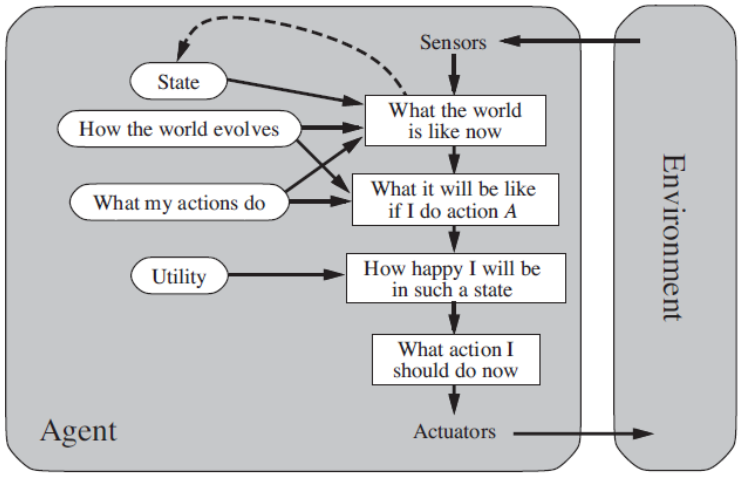


Figure 2.4: A model-based, utility-based agent (Russell and Norvig, 2009 [RN09])

the environment in order to keep an internal state of the world even in case of partial observability. For instance if the car in front brakes and blinks light, at a given time t, braking lights can be off but the agent has previously noticed that the car was breaking and keeps this information in memory. Or this agent would know that a car which is overtaking is supposed to keep higher speed. These are called model-based agents.

Goal-based agent However an agent may have to take decisions based on a given goal at moment t. A goal-based agent is conceptually different as future considerations are involved and thus a reasoning phase is needed to choose what kind of action will help achieving the goal. The concept of goal is important in the driving task: the destination targeted can vary and thus decisions (at a junction for example) cannot be given by condition-rules and require instead higher level considerations. Nevertheless goal-based agents lack of some kind of performance measure ability since different ways can lead to one destination; some of them may be very long, or hazardous; a rational agent would choose its actions according to these considerations.

Utility-based agent Utility-based agent overcome this problem and use utility functions as internal performance measures. Such a function is rational if it is in accordance with the external actual performance measure. This agent maximizes the expected utility of the action outcomes. It sounds intuitive and

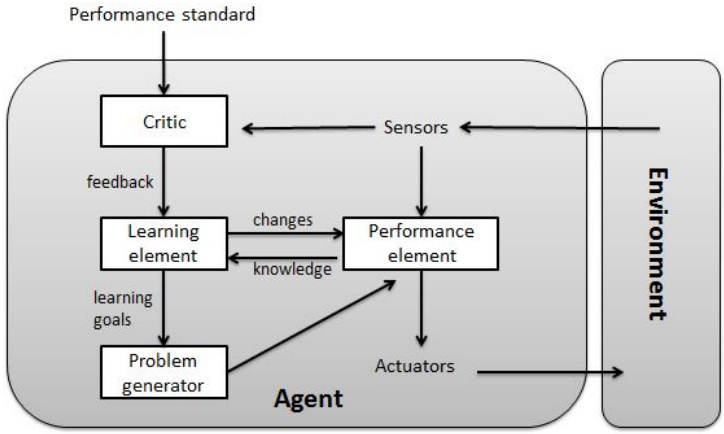


Figure 2.5: A learning agent (Russell, 2009 [RN09])

simple but the challenge is to find good metrics, representation and reasoning. Figure 2.4 shows such an agent: it uses a model of the world, along with a utility function. However, such an agent cannot be autonomous because is entirely dependent on the built-in knowledge and model that the agent designer has programmed.

Learning agent Autonomy is crucial for an agent in order to adapt to different environment and unforeseen events. Partial or incorrect prior knowledge should be corrected so that the agent can act better. This is done through learning: a learning agent is able to evaluate its performance, gets feedback from a critic component based on performance standard. Its structure is shown on figure Figure 2.5. For example, reinforcement learning enables the agent to adopt best strategy: a problem generator component makes it try new strategies; decisions are taken then the agent rewards good decisions and gives penalty to bad ones – according to performance measure and standard. Therefore the agent is getting better over time, preferring decisions that were assessed as appropriate.

The utility-based agent suits to the requirements of the problem of study. It simulates human behaviour based on a high level of decision-making, considering performance measure. Moreover it could be augmented by a learning structure which makes the agent capable of adapting to various environments and situations, and also achieving autonomy and getting better over time. However in this thesis no study has been conducted to find good metrics for performance measure. Therefore only a model-based agent will be used. It could be improved

into a model, utility-based agent with metrics defined. The goal is simple as the agent needs to go as fast as possible, therefore the goal-structure of the agent is trivial and the model will not focus on this part.

Chapter 3

Description of the model and the algorithms

This chapter describes the implemented algorithms. The first section defines the problem particularly its task environment, the driving rules to respect and how these aspects are treated in the implementation. Then the second section describes the agent-oriented online algorithm and the two implemented agents and their decision process. The third section explains how the A* algorithm is performed for the optimal search, and discusses about criteria of optimality.

3.1 Description of the integrated model

The integrated model built for this thesis is based on concepts and techniques described previously that are relevant to this problem for the ego car. Models are also used for the simulation of the other vehicles. As explained in Section 2.2.2.3, agent-oriented programming suits this problem. Therefore the online algorithm that has been implemented is based on this paradigm. The optimal search is based on the structure of the online search. The simulation structure aims at being flexible, respecting driving rules and being realistic enough to get meaningful results. Characteristics of this simulation model will be assessed in section 4.1.

3.1.1 Task Environment of the problem

Russell and Norvig [RN09] advocate to study first the task environment or PEAS (for Performance, Environment, Actuators, Sensors) before designing a problem related to Artificial Intelligence (A.I.) and particularly agents. This section focuses on this approach.

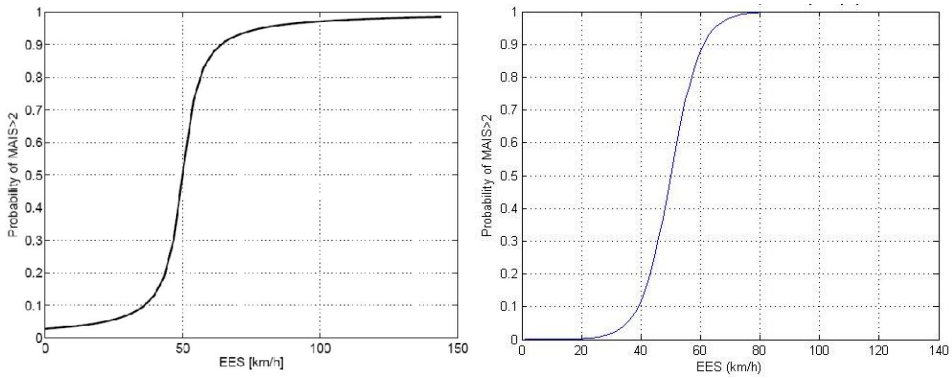
3.1.1.1 Performance Measure

Performance measure enables to assess how good the system is according to given criteria. It defines metrics and desirable qualities of the system. For example in this problem, the trajectory aims at being:

- legal: speed limits respected, lane limits respected, etc.
- safe: in compliance with traffic rules (safety distances, possibly keep lower lane etc.)
- fast: minimize the time t to reach a given location (abscissa $Xgoal$)

Other criteria could be taken into account, such as minimizing fuel consumption, comfort or wear and tear, but the program focuses on the above issues only. In this problem, the goal is to reach a given abscissa $Xgoal$. Only trajectories that, according to the decision process, should be legal and safe are considered. Since lateral position is discretized and given only by lane IDs, cars always keep lane and so only longitudinal control is relevant. Safety of a trajectory can be assessed with the criteria detailed in Section 2.1.1.1. For the MAIS estimation, there is not any explicit formula in the paper of Glaser, neither in the literature, but the function $p : EES \rightarrow MAIS_{>2} = \frac{1}{1+e^{-0.2(EES-50)}}$ fits reasonably well the graphs Glaser presents [GVGM11]: the shape is similar, a probability of 0.5 is given for $EES = 50\ km/h$, below 40 km/h the probability is under 0.1 and above 60 km/h the probability is above 0.9, as shown on Figure 3.1. It is considered sufficient in the scope of this study.

At each moment, a virtual scenario is launched to assess risk: the car in front brakes hardly and the ego car reacts after a reaction time to this change and brakes hardly as well. In case there is no crash, the distance is retrieved between the two cars when both are stopped. In case there is a crash, the EES and the probability of an injury $MAIS_{>2}$ are retrieved. These information, together with the safety indicators detailed in section 2.1.1.1 are retrieved considering the car in front, as a given car is only responsible for keeping safe distances with



(a) (b)

Figure 3.1: EES and $MAIS_{>2}$ scale for Glaser (a) and this model (b)

the car directly in front and not the one behind. However, when overtaking or getting back the driver must keep a safe distance with the car on the new lane hence in this case the criteria are retrieved both for the car in front and the car behind. Two solutions have been considered to take these criteria into account:

- fixing a threshold limit and determine the percentage of time that the situation is not considered as dangerous for a criterion, or
- retrieving the most extreme value of each of these criteria.

The second option might be overly stringent however it is the safest as high risks can be detected. Therefore this option has been chosen. However, it is still interesting to have an overview of the danger during the whole trajectory. A trajectory could be safe 99% of the time and extremely dangerous at one precise moment, whereas another trajectory could never be very dangerous but maintain a moderately high level of risk. Both of these trajectories are potentially dangerous. So the percentage of time safe distances are respected is also computed as it gives a risk indicator of the whole trajectory. Safe distances will be dependent on driver's profile (subjective risk) and explained further in paragraph 3.1.2.3. However, safety distances are programmed as constraints (cf. Section 3.2.2) which means that they should almost (almost, because of the uncertainty of predictions) be respected according to the agent's point of view. But a distance assessed by "safe" by a risk taker may actually not be. Therefore for the performance measure, safe distance will be evaluated from a standard driver's point of view, which agrees with the approach from Russell and Norvig [RN09]: "As a general rule, it is better to design performance measures

according to what one actually wants in the environment, rather than according to how one thinks the agent should behave". Also, one should notice that since perfect lane keeping is supposed, only rear-end collision avoidance is considered: there is no lateral model, not any lateral risk evaluation criterion. To sum up, these are the relevant criteria retrieved to assess the risk of a trajectory:

- $sd_{\%}$: percentage of time that safe distances are respected (from a standard driver's point of view)
- TTC_{min}: lowest TTC during the trajectory, in s
- δT_{min}: lowest time headway during the trajectory, in s
- h_{min}: lowest headway during the trajectory, in m
- EES_{max}: highest EES during the trajectory, in $m.s^{-2}$ (if a crash occurs in a virtual scenario)
- $p(MAIS_{>2})_{max}$: highest probability of injury $MAIS_{>2}$ (if a crash occurs in a virtual scenario)
- d_{min}: lowest distance with the car in front (in a virtual scenario, if a crash never occurs)

TTC are considered safe for values larger than 5 s by Vogel [Vog02]. As explained in Section 3.1.2.3, time headway on highways should be above $2-3$ seconds, which corresponds approximately to 70 meters of headway. EES is considered high for values above 11 m/s, which is the point at which the sigmoid increases suddenly from 0.1% of injury to 0.9% for 17 m/s. But if there are non-NaN (EES_{max}, $p(MAIS_{>2})_{max}$) values retrieved, it means that a crash could occur which is criterion for a risky trajectory.

In order to assess if a trajectory is better than another, one should define metrics, probably depending on driver's profile. For instance it could be based on fuzzy logic to mimic human reasoning. Also, one could give a performance score to decisions based on these indicators, which would enable the agent to learn through reinforcement learning techniques for instance. Which is better: to be in a moderate danger during a long time or very safe all the time except during a few seconds when the car is in a very risky situation? How many seconds are worth winning by taking risk? The performance measure will highly depend on the driver's profile. However it has not been implemented in this thesis. The purpose was to retrieve relevant indicators that enable comparisons, as a first step for feasibility purposes. A study dedicated to this performance measure is needed to improve evaluation of the model and prepare the learning stage.

3.1.1.2 Environment

The environment is a highway with a fixed number of lanes, a fixed speed limit and it is infinitely long. There is not any sign on the road and not any external object that could distract or disturb the driver. The road conditions are supposed to be normal. These are simplifying assumptions for a first model in order to focus on the algorithm itself rather than geometry considerations. Other cars are equipped with an agent structure and as such, they are able to react to their close environment. For a basic simulation, they are equipped with an IDM as this model has shown the best behaviour among longitudinal models in Appendix A. They do not change lane for simplification purposes. Each car is given characteristics (as detailed in Section 3.1.2.1), initial conditions (physical characteristics: position, speed, acceleration) and a maximum desired speed v_{md}. Each IDM car has a driver with a reaction time τ. However, any agent implementation can be put in any car of the simulation. Still following Russell and Norvig's approach [RN09], environment's characteristics will be detailed below. They relate to environment's criteria such as observability, determinism or continuity.

Partially observable The observability is an important criterion for the design of the problem. In this model it is partially observable as the whole environment is not known, moreover some information such what the other drivers are planning to do are not known. There could be noise and inaccuracy in data, even missing data, but it is not taken into account for this model as what is known is supposed to be known with a perfect accuracy for simplification purposes. This partial observability is represented by the limited field of the sensors (detailed in Section 3.1.1.4) and dealt with in the decision process of the agents, which is detailed in Section 3.2.2.

Stochastic environment The ego car is deterministic as it is supposed that the car reacts exactly to the input: there is one outcome from an action decided by the agent. But after an action, the next overall state of the environment is not determined as it does not depend only on the ego car but on other car's behaviours as well. Several outcomes can occur from a given state, each of them could be given a probability of occurrence. Thus the environment is stochastic. This characteristic is somehow related to the unobservability of what the other drivers are planning to do, and is taken into account in the decision process of the agents. However in this model, there is no probability assignment for possible outcomes that can occur from a given state, it is simplified as explained in Section 3.2.2.

Single Agent Each car can be considered as an agent for this problem, which would make it a multi-agent problem. However the purpose of this thesis is not to design an algorithm that makes them cooperate to optimize the overall traffic. In the near future all the cars should not be able to communicate and automated cars will share road structures with standard cars. Therefore it is relevant to focus on a single car problem. The problem is single-agent and the behaviours of the other cars are modelled with IDM.

Sequential Decisions at time t will affect future decisions. Short-term actions trigger long-term consequences: for example the decision to change lane. Acting rationally is considering long term consequences and taking high level decisions, so it confirms that an agent structure is adapted to this problem.

Dynamic This problem is clearly not static as the other cars keep moving - and also the ego car - while the agent is deliberating. Decisions are taken very frequently in this dynamic environment; between two decisions, the same acceleration is kept. When there is not any percept, speed is maintained. In this model, the only moment when there is no percept is at the moment of a lane change.

Continuous and discrete variables Most of the variables of the problem are continuous. However data are sampled every time step dt and are therefore discretized. Lateral position is discretized with lane IDs. In the process selecting the appropriate acceleration, as explained in Section 3.2.2, a range of speeds are tested which discretizes speed output possibilities.

Known The environment is known which means that the agent knows what will be the outcome of its action (acceleration makes the car goes faster for instance). In real life, road conditions could be difficult (snow for instance) and the agent would need to learn how the car reacts to steering angle and acceleration. But road conditions are supposed to be standard so in this model there is no need to adapt to different environment conditions.

3.1.1.3 Actuators

The actuators for a car are the pedals and the steering wheel which make it move on the road. Since lateral movement is discretized with lane IDs, the actions to be taken contain two decisions:

- acceleration/deceleration (continuous)
- lane changing
  - overtake, or
  - keep lane, or
  - get back

3.1.1.4 Sensors

A driver can perceive and evaluate the distances with other vehicles, and also their speed or acceleration. Perception may be incomplete as there are blind spots and other vehicles or obstacles can obstruct the field of vision. There exist distance sensors, speed sensors (Doppler radar for example) that can supplement these data; acceleration could be derived from speed sensors. For this model, it is simplified and the car is able to perceive the distance to and the speed of the vehicles directly in front and behind which are located on the left lane, on the same lane and on the right lane, within 200 meters. This value corresponds to the frontal area covered by the radar of Stanley [TMD+06]. The radar could cover non-adjacent lanes but it would depend on obstacles on the road (a car could obstruct vision): for simplification purposes, only adjacent lanes are considered in perception. A more complex model could consider the other lanes for more informed decisions.

3.1.2 Driving rules

Actions to be taken must respect several sets of rules, considering system limits, human characteristics and traffic rules.

3.1.2.1 System rules

Since this model aims at tactical decisions, control is not part of it so the vehicle's model is very simplified. The car model is even simpler than a bicycle model. It goes straight on a lane and can change lane in one time step. Characteristics of a car are:

- m: mass, in kg

- L: length, in m
- g: maximum acceleration, in $m.s^{-2}$
- b: maximum deceleration (absolute value), in $m.s^{-2}$
- v_{cmax}: maximum possible speed, in $m.s^{-1}$

The acceleration chosen for an action must be in the interval $[-b, g]$. Also the action is for $[t, t+dt]$: the resulted speed at $t+dt$ cannot exceed the maximum desired speed v_{md} and *a fortiori* the maximum speed allowed v_{max} - since for legal reasons, $v_{md} \leq v_{max}$.

3.1.2.2 Human rules

Human rules set the characteristics the system should comport in order to adapt to and interact with human beings. They especially focus on the human-machine interface [Van12]. In this model the driver is supposed to react exactly to what the machine decides and this interface is not part of the simulation. Still, the algorithm has been implemented to take into account human drivers. In this model, a driver has four characteristics:

- τ: reaction time, in s
- θ: risk taking trait from -1 (very careful) to 1 (high risk taker)
- γ: ability and willingness to perform manoeuvres from -1 (very low) to 1 (very high)
- μ: willingness to go fast, from -1 to 1 (wants to go respectively from 20% slower than v_{max} to v_{max})

θ can be lower than -1 or larger than 1 (up to 3 for this model, as explained later) but in that cases behaviours are extreme. For this model, μ can actually be decreased down to -9 which is equivalent to targeting a null speed. But most of drivers would have a μ value between -1 and 1.

Drivers' profiles It is unclear how to properly define a cost measure as it depends on various criteria such as time, potential crash cost (on vehicles, road infrastructure, injuries...), potential fine or comfort. Therefore the simplest

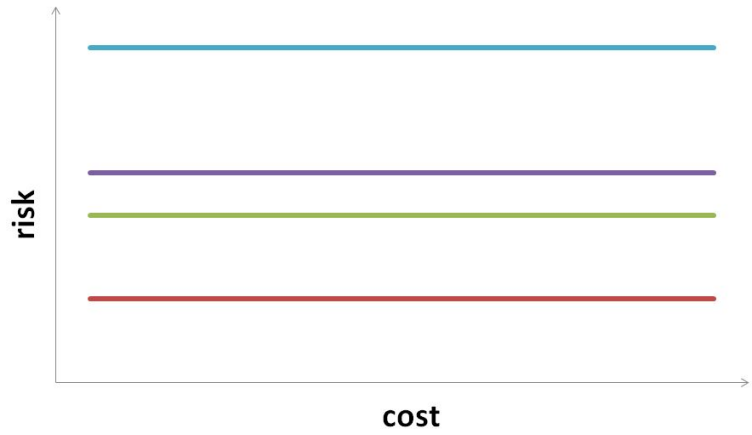


Figure 3.2: Different cautious driver profiles

model for driver profile has been chosen in order to provide a first model depending on driver's characteristics: all the drivers have a *careful driver* profile, as shown on Figure 2.1. However the risk threshold varies and depends on driver's characteristics detailed above: θ, μ are taken into account but not γ as explained in Section 3.2.1.1.

On Figure 3.2 several driver profiles are displayed. The blue one has high θ and μ values while the red one is very careful: negative θ and desired speed below the maximum speed allowed. Depending on their profiles, drivers will accept or not more hazardous trajectories and higher speeds. These limits are detailed in Section 3.2.2.

3.1.2.3 Traffic rules

Traffic rules need to be respected to ensure legal and safe trajectories. Lanes are always kept due to lateral discretization, which avoid cars to disrespect rules related to it in this model. Driving should be on the right-most lane in this model. Undertaking (overtaking on a lower lane) is allowed in Australia: this rule has been implemented for this model. To adapt to a country where it is prohibited, the algorithm could be modified. For instance, speed could be upper-bounded by the speeds of cars in front on higher lanes. Also safe distances must be respected to ensure safe travels.

Safety distances The safe distance given by Gipps (1981) [Gip81] and described in Section 2.1.2.2 is used. Furthermore, the IDM model which gave the best performance in tests (cf. Appendix A) is based on this safety distance. This distance is given by $d_{i,ft}{}^*(t) = v_i(t) \cdot \tau_i + L_j + max(0, (\frac{{v_i}^2(t)}{2b_i} - \frac{{v_j}^2(t)}{2b_j}))$. It takes system rules into account: a driver would have time to perform an emergency braking at any time, as the headway is always long enough to let the driver react and then brake until standstill complete stop, as detailed in Section B.1.1 of Appendix B. It is a consistent safety distance for a long term simulation: if all the cars respect it, there would not be any accident except unexpected circumstances (something blocks suddenly the road for example). However, as the driver acts after a reaction time τ, the agent needs to provide information in advance and needs to predict other cars' trajectories. Those predictions may lack precision and safe distance could be underestimated. Moreover, this distance gives a limit and could trigger short headway and stress on the driver. A general rule of thumb in many countries is to let a time headway of 2 to 3 seconds with the car in front (for example in Queensland, Australia [Gov10]). These two criteria are relevant and characterize safe distances on highways. Therefore, in the model a position is considered not safe if at least one of these criteria is not respected. The latter depends on the parameter θ_j of driver j: at each moment the agents take a decision to respect $\frac{h_{i,ft}(t)}{v_i(t)} > 3 - \theta_j$.

3.1.3 Other precisions about the model

3.1.3.1 Design of classes

The program is object-oriented. Most of the more important classes are given in Section 3.2.1 on Figure 3.4, as they are related to the agent. Specific structures for the optimal solutions algorithm will be described in Section 3.3. Other general precisions are given in this section.

Physical Characteristics Physical characteristics are associated to a car in the environment with hash maps that enable to retrieve these characteristics from an agent or the object at a given location, as shown on Figure 3.4. Such an object stores:

- the abscissa x of its front point
- the longitudinal speed v
- the longitudinal acceleration a

- the lane l
- the $xCell$ corresponding to x
  - the environment is divided into cells of a given length for computation purposes for the optimal search, as explained in Section 3.3. For example if the cell length is 5 meters and $x = 16\ m$, $xCell = 3$.

Scenario A scenario initializes the simulation. It gives the number of lanes, the maximum speed allowed, the list of agents and their initial physical characteristics. Then these agents evolve in the environment according to their implementation model.

Simulation Result The simulation result is created at the beginning and updated during the simulation for the online algorithm, and retrieved at the end for the optimal search. It stores the history of states of the environment and computes the trajectory's criteria given in Section 3.1.1.1 *a posteriori*.

3.1.3.2 Visual representation of trajectories

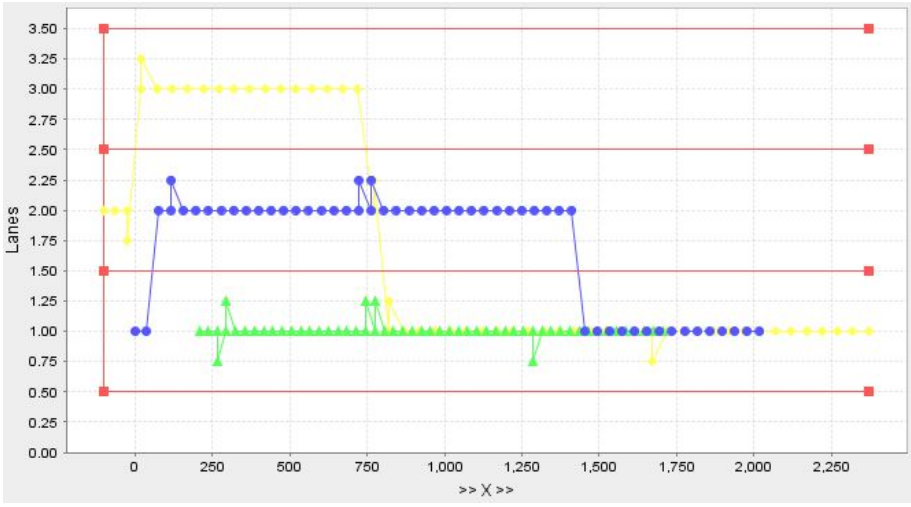


Figure 3.3: Trajectory representation

Since the information about performance measure retrieved from a trajectory does not fully describe it, graphs are derived to represent trajectories. They will

be used when presenting results in order to show the trajectory. The abscissa is x in meters, cars travel from left to right. The ordinate is the lane index, lane borders are represented in red. The lowest lane (coded as 1) is the lane that cars are supposed to keep if not overtaking. A coloured line represent a trajectory and each point a position at a given time t multiple of dt. The ego car is traced in blue on the example on Figure 3.3. If the ego car changes lane at time t, a doubloon will be created on the graph for the other trajectories with a point below the position at time t (for instance on the figure, for $t = 3$). If another car changes lane at time t, a doubloon will be created on the graph for all the trajectories with a point above the position at time t (for instance on the figure, for $t = 4$ the yellow car overtake). These visual tricks are very handy to have an overall visualization of the trajectories on only one image. The visualization is good for short-time scenarios but should be improved for longer ones or scenarios with a high number of cars, otherwise there are too many overlaps and it is hard to distinguish trajectories and understand what happens.

3.2 Online agent-oriented algorithm

This section describes how the online algorithm works. As explained previously, it is agent-oriented and uses the structure already described. The agent structure has been based on the work of Ravi Mohan in collaboration with Russell and Norvig (2009) [RN09]: they provide a code architecture for such problems and appropriate interfaces for the objects.

3.2.1 Agent structure

The class diagram on Figure 3.4 shows the agent's structure. The agent perceives the environment through the method *getPerceptSeenBy* acting on the current environment. A new *HWEnviroment* is created and stored in the list *state_history*, which is the percept sequence: the history of data position and speed of all the cars around. This information gathering helps the agent to choose which action to perform, as explained in Section 3.2.3. A driver agent corresponds to a car and a human driver. It implements the interface *DriverAgent*, which is abstract and is a generalization of the classes *Greedy_HWAgent* and *BDI_HWAgent*. They are detailed below.

3.2.1.1 Greedy Agent

The *Greedy Agent* has a straightforward strategy. It evaluates the maximum speed the car can reach, below the maximum desired speed of the driver v_{md}, on each lane, and chooses the fastest solution. If the agent can go as fast as possible on two lanes, the decision that targets the lower lane will be chosen. As a result, it minimizes time locally until the next time step. A penalty could be added for changing lanes but it has not been implemented to keep the model simple. A particular study should be conducted to evaluate an adequate penalty to take this into account. In that case the attribute γ of the agent would be used to make the penalty varying depending on the driver. γ is not used in this model.

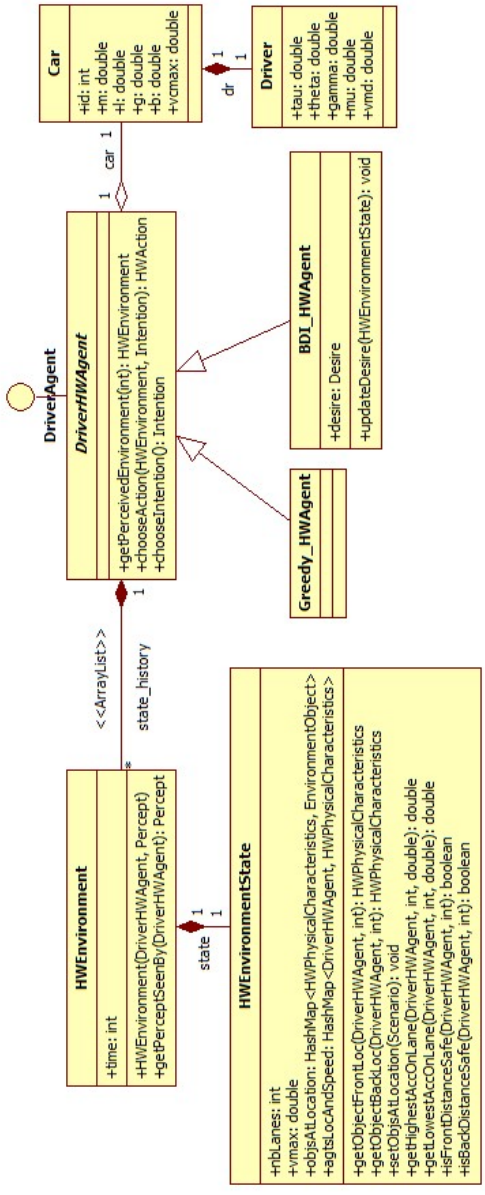


Figure 3.4: Class Diagram centred on the Agent

3.2.1.2 BDI Agent

A Belief-Desire-Intention (BDI) Agent (cf. Section 2.2.2.3) with a simple intelligence has been implemented to provide a more rational behaviour than the one of the *Greedy Agent*. Its characteristics are:

- Beliefs: the history of states of the world he knows through his percepts.
- Desires: 4 different desires have been programmed:
  - *keepLane*: if it wants to keep the same lane
  - *overtake*: if it wants to overtake the car in front
  - *keepOvertaking*: if it wants to keep overtaking a car. As soon as it starts overtaking, its desire switches automatically to *keepOvertaking*. Then it may want to overtake a car and get to a higher lane again; hence *keepOvertaking* and *overtake* are two distinct desires.
  - *getBack*: if it wants to get back on a lower lane, to finish overtaking
- Intentions: In this model desires and intentions are the same: intentions are directly derived from desires.

In this model, the desire to go fast is implicit, as the highest possible acceleration on each lane is computed. In a more complex model, it could conflict with *getBack* if the number of lanes on the highway is reduced from 3 to 2 and if there is a car on a lower lane for example. In that case the intention derived from *goFast* and *getBack* could be *keepLaneAndReduceSpeed*. Hence desires would be different from intentions in such a model. However the model of this thesis is very simple for a first attempt to model a *BDI Agent*. The desire is updated every time step, based on previous percepts. The scheme of *updateDesire()* is:

```
if frontCarIsSlow() and thereIsAHigherLane()
  desire= overtake;
else if thereIsALowerLane() and noCarInFrontOnLowerLane()
  desire= getBack;
else if desire == overtake || desire == keepOvertaking
  desire= keepOvertaking;
else
  desire= keepLane;
end if
```

A car in front is supposed to be slow if it goes at less than 90% of the maximum desired speed of the ego driver. There is no particular motivation for this threshold as the purpose of this implementation is to show a different behaviour but not provide a convenient and usable program for a real application. In this model, only the last percept is taken into account. It would be interesting to build a more intelligent reasoning taking into account the history of percepts, and also to elaborate more the reasoning. For example, it would be interesting to make the agent adapt its speed to achieve the desire he wants (instead of always targeting a maximal speed) as suggested in the previous section. The implemented behaviour aims at showing that with the concept of overtaking implemented in the agent's structure, it can perform better than a *Greedy Agent* in some situations. Lane changing decision is based on a few information gathering: the decision model is simple for a first integrated model.

3.2.2 Decision process

The aim of a decision is to give an acceleration to take at time t and keep it until $t+dt$, when a new decision will be taken. It is equivalent to target a speed $v(t+dt)$ supposing that the acceleration is kept constant during this time lapse. These choices of acceleration should give a trajectory that is safe and satisfies the agent's intentions.

3.2.2.1 Model-based but not utility-based agents

This decision process is more complex than a reflex agent's one. It is not limited to the only information gathered as the decision is not based directly on the position and speed of the other cars as retrieved by the sensors. Instead, the agent reacts to what the environment is predicted to be one step further predicting the location of other cars at that time. Therefore the agent is model-based. In order to be a utility-based agent also, the decision process for choosing what action to perform should be based on an internal utility function in agreement with the performance measure. Further studies are required to evaluate an appropriate utility function. The implemented agents, the *Greedy Agent* and the *BDI Agent*, are not utility-based. Instead, it has been implicitly programmed that the agent must go fast and safely. The former has been done by selecting the highest possible acceleration on each lane when evaluating possible actions. Then the choice of the action depends on the agent: for the *Greedy Agent*, best is to go fast. For the *BDI Agent*, best is to achieve its desires (go fast and change lane if intended and possible). The latter has been programmed as constraints. A given acceleration is possible if the agent predicts that it will keep a safe

distance until next time step. As a result, any action that would not respect it, according to the model, is never given to the driver. Besides, for both agents, contingency plans are considered; in case the intention cannot be satisfied, it would mean that safe distance with the car directly in front is too low for any acceleration even $-b$. In that case an emergency braking is considered and the car decelerates at $-b$ on the same lane.

3.2.2.2 Decision process scheme

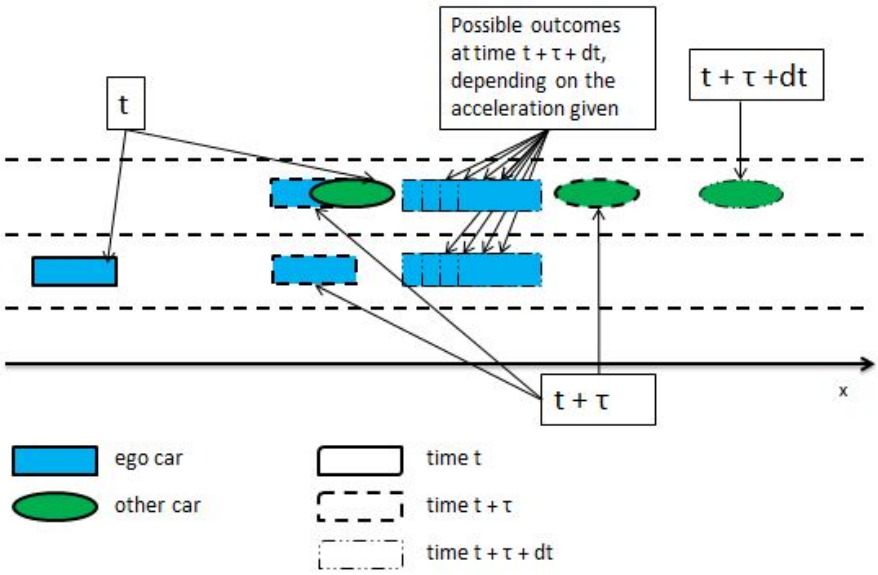


Figure 3.5: Visualization of predictions for the Decision Process

The driver has a reaction time τ. Therefore the agent needs to give information to him at time τ second before the driver actually acts. As the environment is dynamic, the agent needs to predict at time t what the environment will be like at time $t + \tau$. The agent has a built-in model to make it predict the future environment state. As shown on Figure 3.5 two predictions are made at time t:

- time $t + \tau$: the environment state at the moment the agent will take the action
- time $t + \tau + dt$: the environment state for the other cars at the next time step

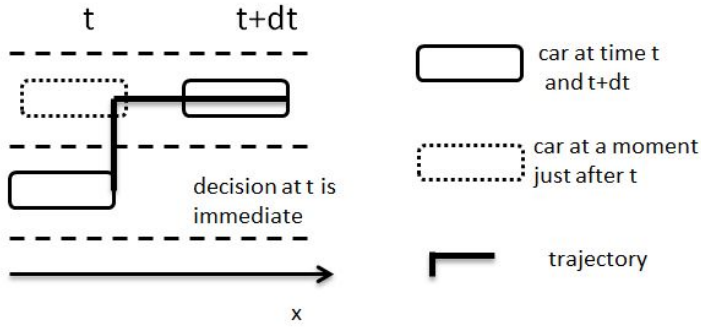


Figure 3.6: Movement between t and $t + dt$

In the implementation, at time t, for each driver agent j, *chooseAction* is invoked on the driver agent$_j$, with its perceived environment at time t and the intention of the agent j as parameters. The agent j decides to take an action that the driver j will do at time $t + \tau_j$. This action is supposed instantaneous as shown on Figure 3.6. Therefore if the driver changes lane, distances must be safe on the current lane and the new lane: that is why positions on all adjacent lanes are considered for the ego car on Figure 3.5.

The decision scheme is given by the activity diagram on Figure 3.7. Based on these predictions, and depending on his intention and profile, the agent takes an appropriate decision to give to the driver. For instance, if the agent intents to go as fast as possible, Take Fast-intended decision is chosen on Figure 3.7. The end of the decision process for this particular intention is given by the activity diagram on Figure 3.8. The process is very similar for other intentions, and even simpler, as preferred targeted lanes are put first on the list (the higher lane for overtaking for example). Being on a lane l on predicted state at time $t + \tau$ must be safe to consider an action on that lane. Then, several accelerations are considered from $t+\tau$ to $t+\tau+dt$ as shown on Figure 3.5, and the highest possible one according to safety conditions is kept for that lane. These accelerations are considered from the minimum of g and the acceleration that targets the desired maximum speed, down to the maximum of b and the acceleration that targets a null speed. It is discretized as, due to the form of the equation, there is no direct formula giving the targeted acceleration. As a result the acceleration output is not exactly the highest (or lowest) possible. However discretization is made on speed with a sampling of $dv = 1\ m.s^{-1}$, which results in a negligible error as detailed in Section B.1.2 of Appendix B.

In these predictions, the car is predicted to be at one specific position at time $t + \tau$ and $t + \tau + dt$. Another way would be to give a probability distribution

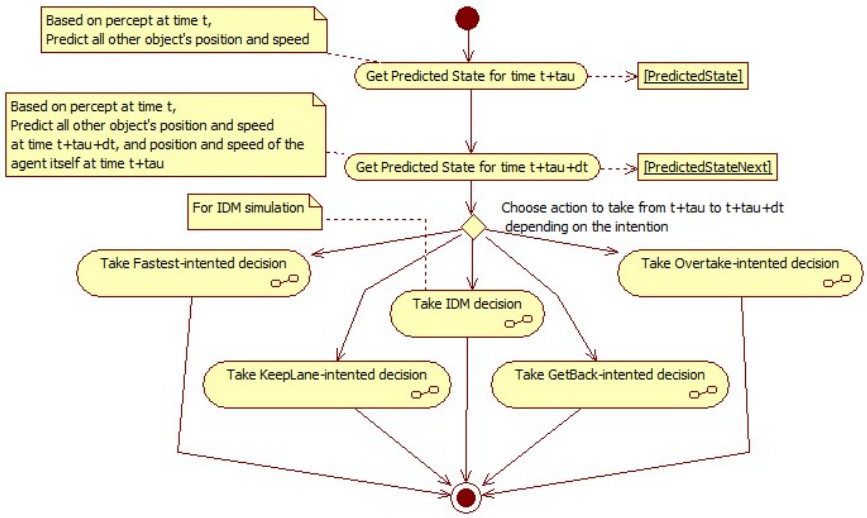


Figure 3.7: Decision Process Scheme

on the road. Based on this probability a decision could be assessed to be more or less dangerous. This design would be very different from the one used in the model because not based on a threshold. Possible techniques to do so are segmented cones, as suggested by Greene et al. (2011) [GLR+11], or Kalman filters. However it would require further studies and evaluation of risk, and that measure could be integrated in the evaluation of the utility function. As explained previously, the focus has not been made on this part for this thesis. Therefore a threshold-based model is used.

Prediction Model The prediction model that has been implemented supposes that the other cars keep the same lane and speed. It is very basic because based only on the last percept. Based on a history of percepts, one could implement a more sophisticated model in order to reduce errors. Even if predictions are made on short term (maximum of $\tau + dt$ which is approximately 1-1.5 s in this model), if a car j has a constant speed and then performs an emergency braking at $-b_j = -8\ m.s^{-2}$ during 1.5 seconds, the resulted errors on his position is 12 m which can be dangerous. But sudden changes are taken into account with the safety distance of Gipps: respecting this distance gives time for the driver to perform an emergency braking if needed. Therefore, supposing that the cars keep the same speed seems reasonable for a first simple model.

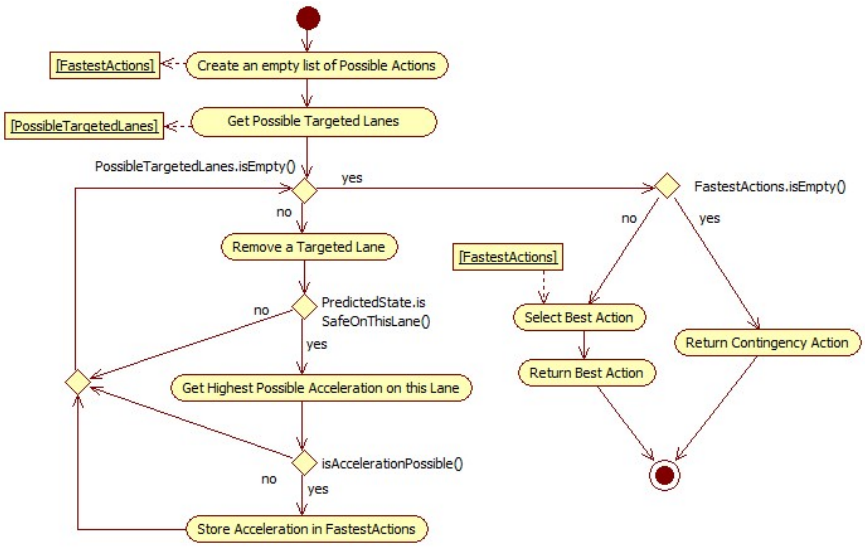


Figure 3.8: End of Decision Scheme for Fastest action

3.2.3 Online Algorithm

For the online algorithm, every agent acts in real time based on its percepts and implemented intelligence. After initialization, the program enters a loop in which the agents act every time step. This loop is described by the sequence diagram on Figure 3.9. A time limit has been fixed to avoid infinite loop in case of failure. While the goal is not achieved - or the time limit not passed - every agent chooses an action according to its percepts, taking into account the reaction time of its associated driver. At time t, the current environment is perceived with *getPerceptSeenBy*, *chooseIntention* retrieves the intention of the agent and an action is chosen, that will be done at time $t + \tau$ by the driver. For example, a *Greedy Agent* will always have the intention to go the fastest possible, while a *BDI Agent* may want to overtake or keep lane for instances, depending on the perceived environment. Finally the agent j acts in the current environment, doing the action the agent had decided τ second before.

In case the agent j has changed lane between t and $t - \tau_j$, its percepts have not been updated on the new lane and so the decisions taken between t and $t-\tau_j$ are not valid. In that case the driver keeps lane and speed, as shown on Figure 3.10. Implementation details are provided in Section B.2.1 of Appendix B. The action is supposed to be instantaneous once the decision is taken, in other words the

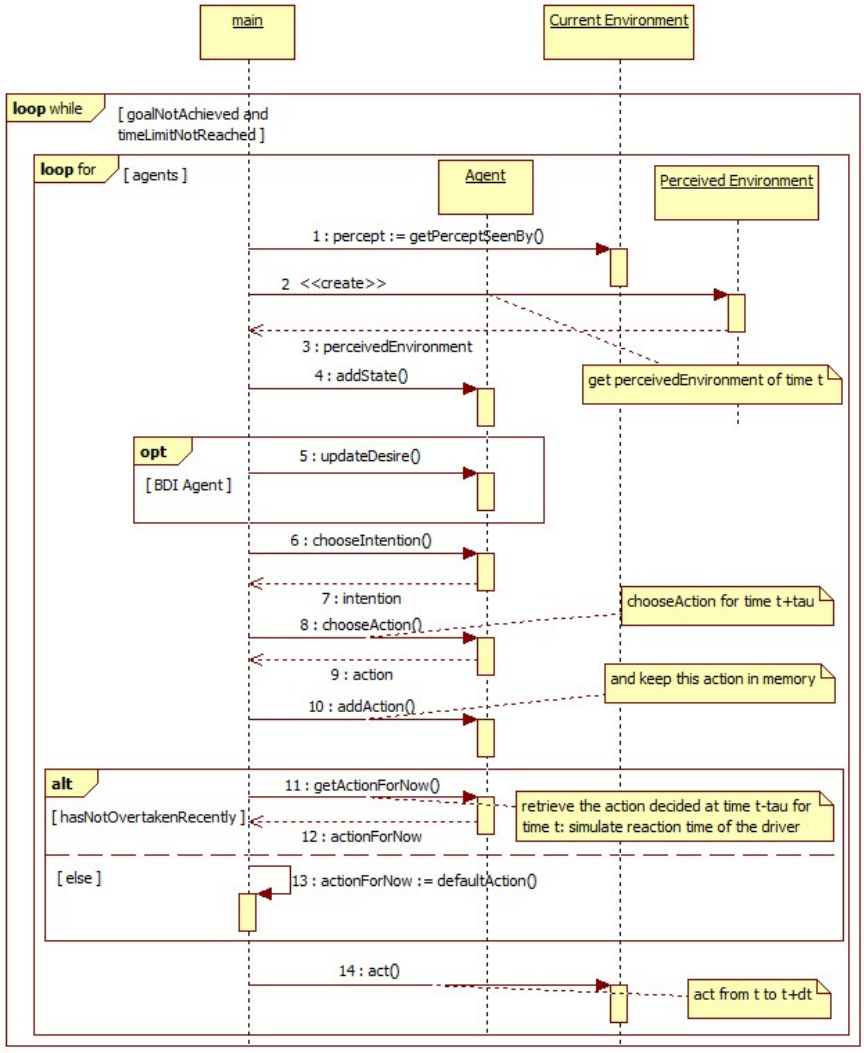


Figure 3.9: Sequence Diagram: Online loop over time

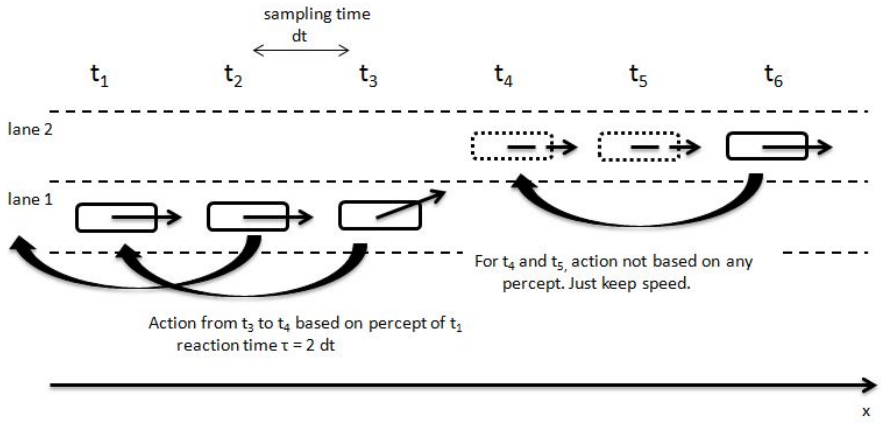


Figure 3.10: Perception-Action example

lane change and the acceleration is immediate after t, as shown on Figure 3.6. The agents are acting simultaneously in the current environment. Finally, it is checked whether the goal is achieved; if not, the loop goes on. This is done every dt (sampling time), and all the tau_j are multiple of dt, so that the times of perception and action match the time steps. During this process, a complete state history is stored so that relevant information about the trajectory can be computed later, as explained is Section 3.1.3.1.

3.3 Optimal search

As explained in Section 2.2.1, the A* algorithm is the fastest algorithm to find an optimal solution. This optimal search will be used offline, within the simulation framework, to assess performance of the online algorithm.

3.3.1 Problem and algorithm definitions

3.3.1.1 Discretization

The time is discretized by sampling time every dt second. The sampling technique used is environment-based sampling, as explained in Section 2.2.1.1. At time t corresponds an environment state with physical characteristics, including the abscissa x, associated to objects (only cars so far), and between two time steps the acceleration is supposed to be constant. The problem is defined by:

- the list of driver agents
- initial conditions from which the initial node is constructed
- a goal: reach a given abscissa $Xgoal$ for this thesis

A node is defined by:

- t_n: the number of time steps to reach the node. $t_n = \frac{t}{dt}$ with t the effective time in seconds and dt the sampling time
- *state* a node state that stores
  - the corresponding environment
  - a list of percepts associated to each agent at this moment
- *parent*: the parent node from where this node was derived
- *action*: the action from the parent node to this node
- *pathCost*: $f(n) = g(n) + h(n)$ with f, g and h detailed in section 3.3.1.2

A node also stores other information needed to wait for getting new percepts when changing lane: this is detailed in Section B.2.1 of Appendix B. It also stores the number of time the car has changed lane to overtake, in order to avoid a "ping pong" effect on lanes: it will be explained in Section 4.2.4.1.

3.3.1.2 Cost and heuristic

The cost function of a A* algorithm is described in Section 2.2.1.2.

Effective path cost The effective path cost $g(n)$ of a node n is given by the time spent so far: $g(n) = t_n \cdot dt$. To avoid a "ping pong" effect, a penalty associated to lane changing has been added and it will be explained in Section 4.2.4.1.

Heuristic cost The heuristic cost $h(n)$ associated to node n is the time for the ego agent to reach $Xgoal$ if the driver goes at her maximum desired speed v_{md}. It is consistent since it can only increase with distance. Therefore it is also admissible. It underestimates the time to get from a given node n to a goal node. However, it does not take into account possible obstacles on the road which makes the information hardly useful if the car is blocked by slower cars in front. If the ego car has to decelerate because of a slow car, it cannot reach v_{md} and a large part of the graph will be explored. The search will be almost equivalent to an uninformed search until the car is able to travel at v_{md}, which can induce a large complexity, in the worst case $O(b^d)$ as explained in Section 2.2.1.2. Nevertheless it seems hard to find a better heuristic without need of costly computation. Indeed, foreseeing slow cars and giving a better estimation of the maximum speed at each moment for the car would be equivalent to solving the problem. Due to these computation's problems, adapted variations of the algorithm can be implemented, making the algorithm lose in complexity but gain in computation time. It will be detailed in the next section.

3.3.1.3 Algorithm implementation

The basic A* algorithm is a tree search. However, it can be turned into a graph search in order to avoid considering several times a same state. To do so, a hash set of visited nodes is stored. A node is considered already visited if the car has the same physical characteristic x, v and l at that moment t. Therefore a same state is not visited twice and the complexity is improved at a low memory cost. The pseudo-code of the graph-search A* algorithm is:

```
path = aStarSearch(scenario) {
  queue=PriorityQueue<Node>;
  initialNode = Node(scenario);
  visitedNodes = HashSet<Node>;
  queue.add(initialNode);
```

```
  visitedNodes.add(initialNode);

  while (!queue.isEmpty())
    node = queue.poll();
    if (node.isGoal())
      return node.getPath();
    end if

    successors = node.expand();
    for n in successors
      if not visitedNodes.contains(n)
        queue.add(n);
        visitedNodes.add(n);
      end if
    end for
  end while

return null;
}
```

Successors When expanding a node, all the actions of the other cars are accurately predicted by the algorithm. To do so, the same process as the online search loop is simulated for each agent in order to retrieve what action they would take in that situation. Which enables the optimal search to know precisely where the other cars will be at time $t + dt$ and then choose a trajectory that perfectly respects safety criteria. Complexity exponentially increases with the branching factor b, then b needs to be kept low in order to perform the computation search. It has been chosen to keep 3 possible accelerations for each lane, as the manoeuvre grid model of Glaser suggests (cf. Section 2.2.2.1). 3 trajectories per lane seems the minimum to consider as it enables the car to keep the same speed, accelerate or decelerate. Considering less trajectories may cut much better solutions: assuming that a car cannot decelerate, or cannot keep speed is too strong. However, considering more trajectories would lead to high complexity as it is exponential in b. On a lane l, if the highest possible acceleration is $a_2 > 0$ and the lowest possible acceleration is $a_1 < 0$, then every acceleration in between is possible. For the optimal search, a_1, 0 and a_2 are kept. In case both highest and lowest acceleration are either positive or negative, a_1, $\frac{a_1}{a_2}$ and a_2 are kept to have a range of 3 possibilities in that case as well. Therefore from a position the car is able to perform 9 different actions, which provides a range of possibilities to consider various trajectories.

Optimality of A* The algorithm is guaranteed to find an optimal solution. But as explained in Section 2.2.1.2, A* is polynomial only if the heuristic respects $|h(x) - h^*(x)| = O(log(h^*(x)))$ with h^* the actual cost. It is not the case here as if there are slow cars, the ego agent may be blocked and search a large part of the graph. As a result, it may take a lot of time to compute a solution. To face this problem, the problem can be relaxed: the algorithm will not find an optimal solution but it will be able to compute a solution faster. Three possible solutions have been considered to relax the problem and make it computationally possible:

1. Increase dt to decrease the depth of the best solution
2. Decrease the number of successors per nodes
3. Use hybrid A*

Worst-case complexity of A* is $O(d^b)$ with b the branching factor (number of successors for a node, here 9 maximum) and d the depth of the best solution. The first solution divide d by 2 and the complexity is then $O((\frac{d}{2})^b) = O(\frac{d^b}{2^b}) = O(d^b)$: the gain is not substantial and the order of magnitude is the same. For the second solution, reducing its possible acceleration to 3 possible ones at each state is already a strong assumption. Reducing it to 2 would mean that the car is either not able to accelerate, decelerate or keep the same speed: which is too strong. Hence it has not been considered as the loss of optimality may be huge if the car has less than 3 options per lane. The third solution is to use hybrid A*: so far, a node is considered already visited if the ego agent has the same physical characteristics x, l and v at the same time step t. Instead, a node can be considered already visited if the coordinates had already been reached (only x, l and t are considered). It can be even more simplified, by discretizing space. Instead of considering every possible x, the road is represented as a grid of cells. Each cell has a length $xCell$ and a node is considered visited if the car has already reached a cell $(xCell, l)$ at time t. The larger $xCell$ is, the more A* loses in optimality. This adapted algorithm is called hybrid A*. This technique has been used to face complexity issues. $xCell$ has been set to $\lambda \cdot dt$ meters with $\lambda = 5\ m.s^{-1}$. As a result, successors of a node where the speed difference is less than $5\ m.s^{-1}$ are considered equivalent, and generally if a node n_p corresponding to the cell and time step of a given node n has already been expanded, the node n is not expanded. It avoids considering different similar solutions such as (*accelerate, accelerate, decelerate*) and (*accelerate, decelerate, accelerate*) with a margin of $2.5\ m$ for $dt = 0.5$. It has been set to 5 since $dt = 0.5$ s and $2.5\ m$ it is negligible compared to safe gap distances on highways (at least 30 meters), and it seems reasonable though not too restrictive to consider at time t that two solutions with a difference of $5\ m.s^{-1}$ are equivalent - reminded that only three accelerations are considered at time t. This technique may be

used if optimal solutions cannot be found because of a too large complexity of the initial A* algorithm in particular scenarios.

3.3.2 Application of the algorithm

The solution given by A* is considered optimal in this problem if the path cost if the lowest possible. Path cost takes only time into account, and safety is used as a constraint. It is interesting to know if by changing the safety criterion (lower headway allowed) a large gain of time can be achieved. A driver could accept a more hazardous trajectory if the resulted gain of time is appreciable. To do so, the A* algorithm can be launched with several combinations of θ and μ values for the ego agent. Several optimal solutions will be computed that will enable to consider several kind of trajectories, from a more hazardous but fast trajectory to a safer but slower trajectory. As explained previously further studies should be conducted to define metrics and assess automatically what trajectory is better for a driver. In the tests of this thesis, it will be discussed what trajectories seem best and they will be compared to the trajectory computed by the online search, based on an analysis of the trajectories' criteria retrieved. In case the scenario triggers a too high complexity, the extensions of the model explained above enable to get suboptimal trajectories.

Chapter 4

Results and Analysis

This section presents the analysis of the model and the results of the conducted studies. The first part explains what criteria are to be assessed to validate the model and its compliance to what is expected from such a framework. In the second part, the optimal search method will be discussed and tested. In the third part, online agent's behaviours will be tested, compared to each other and to optimal solutions.

4.1 Model's validation

This model has been described in Chapter 3. It has been implemented in order to provide a consistent study framework to study intelligent behaviours of agents. It must have flexibility to enable several tests to be conducted, and the implementation must also enable the model to be improved with other features. Also it must respect rules: the system must be ethically valid in order to be accepted for driving assistance.

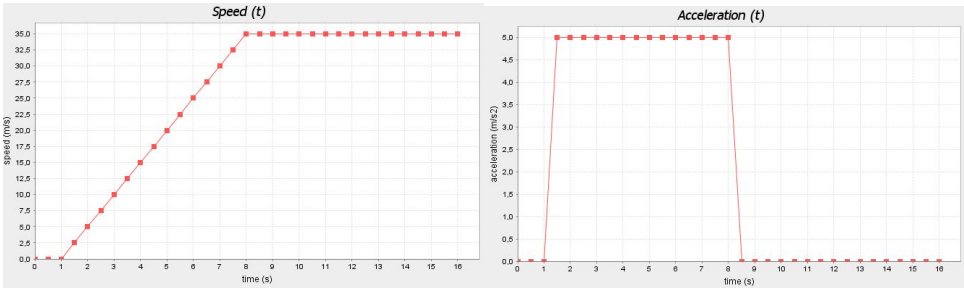
4.1.1 Rules to be respected

The implementation should be in accordance with several sets of rules. Vanholme [Van12] gives a review of the rules that must be respected, according to consensus such as the Vienna Convention on Road Traffic.

4.1.1.1 Traffic rules

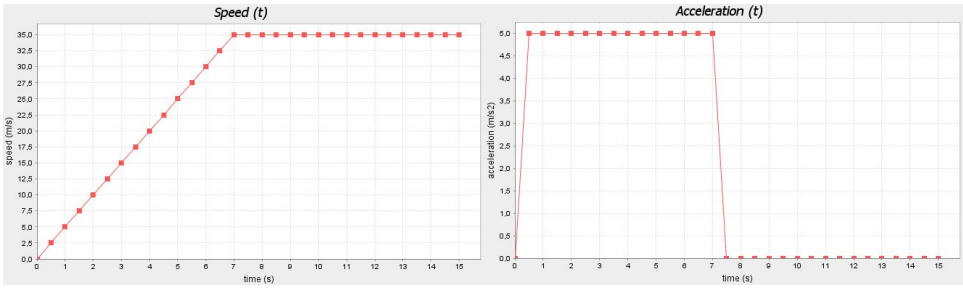
Traffic rules should be respected so that users can safely share roads. This section enumerates the main rules that have should be taken into account and explains whether the algorithm comply with them. Tests presented in this section have been run with the online algorithm; same results would apply to the optimal search as evaluation of safety and acceptance for a trajectory is based on the same rules. If not specified, the parameters for a car are $m = 1800$ $m.s^{-1}$, $L = 4.5$ m, $g = 5$ $m.s^{-1}$ and $b = 8$ m/s^{-2} and for a driver $\tau = 1$ s, $\theta = 0$ and $\mu = 1$, and the agent is a *Greedy Agent*. If not specified, results are the same for the *BDI Agent* as its behaviour only differ for overtaking-related decisions. On a single lane scenario, decisions are the same for both agents. Simulation tests have been run for the optimal search corresponding to the same (θ,μ) parameters. Speed limit is set to 35 $m.s^{-1}$. Tables sum up the trajectory evaluation criteria detailed in Section 3.1.1.1.

Respect speed limits In order to respect safe limits, the parameter μ of the driver needs to be inferior to 1 (maximum desired speed of the driver is $v_{md} = v_{max}$ in that case), since the agent will regulate its speed based on v_{md}. If $\mu > 1$, the speed limits may not be respected. However if $\mu \leq 1$ they are respected as shown by the scenario test below. In this acceleration scenario test, the road is free and the agent targets the speed $v_{md} = v_{max} = 35$ $m.s^{-1}$. Figure 4.1 shows its speed and acceleration as functions of time. Speed never exceeds the maximum speed allowed; the driver is supposed to follow exactly the instructions given by the agent, so the agent knows at time t what acceleration to give for time $t + tau$ in order to respect this limit until time $t + tau + dt$ (cf. Section 3.2.2). The optimal search computes a similar trajectory, that differs only at the beginning because no reaction time is considered, as shown on Figure 4.2. One should notice that the acceleration from standstill is not realistic, as explained in Appendix A. But the purpose is to go as fast as possible, so the acceleration is maximal to reach as fast as possible the maximum speed.



(a) (b)

Figure 4.1: Speed and Acceleration, Acceleration test (agent)

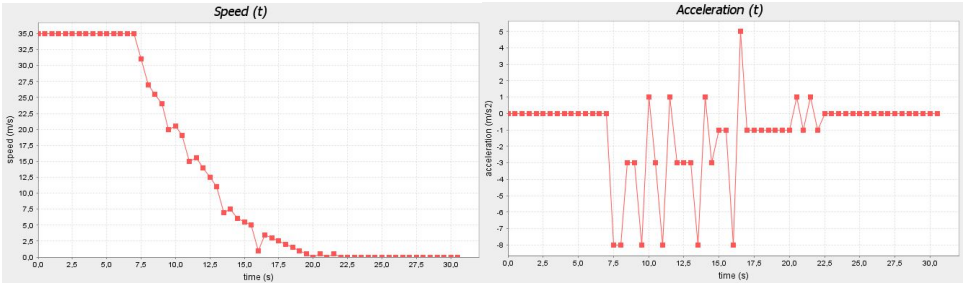


(a) (b)

Figure 4.2: Speed and Acceleration, Acceleration test (optimal)

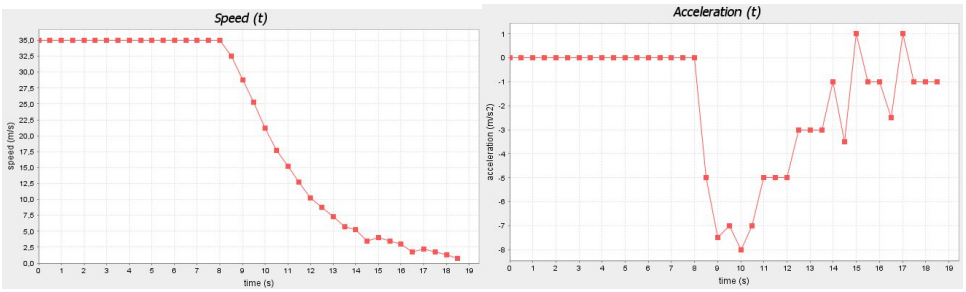
Keep safe distance The agent should be able to brake until standstill- which is keeping safe distance with a stopped car - and also to keep safe distance in a traffic situation to be able to perform a safe emergency braking if needed.

In the braking scenario test, there is a car stopped 400 m in front of the initial position of the ego car and the ego car travels at the desired speed of the ego driver. Figure 4.3 shows its speed and acceleration. The agent manages to stop. At the end of the simulation, the car stands at 0.75 meters from the stopped car, after a smooth braking that starts around 15 seconds before full stop of the ego car and about 150 meters before the obstacle. This is a realistic and desired behaviour as explained in Appendix A. Results do not depend on θ since the vehicle in front is stopped and Gipps' criterion is the same for any θ in this model. For this test, the goal of the optimal search has been set to target the abscissa which is 0.75 meter behind the stopped car, so that the algorithm can find a solution. Results are similar as shown on Figure 4.4.



(a) (b)

Figure 4.3: Speed and Acceleration, Braking test (agent)



(a) (b)

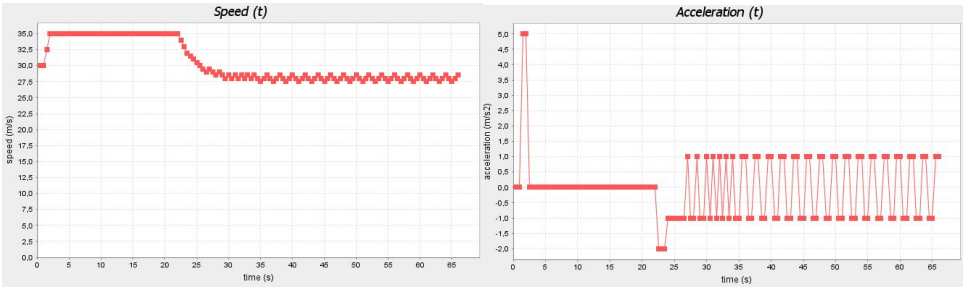
Figure 4.4: Speed and Acceleration, Braking test (optimal)

In the car-following scenario test, there is a car 250 m in front of the initial position of the ego car, they are traveling both at 30 $m.s^{-1}$ but the ego agent desires to travel at the maximum speed allowed ($\mu = 1$) while the car in front desires to travel at a speed 20% lower ($\mu = -1$). Table 4.1 sums up the trajectory's criteria, and Figure 4.5 shows its speed and acceleration. The driver was standard in this simulation, so decisions were taken in order to respect safe trajectories. As a result, safe distances are respected 100% of the time. The lowest tolerated δT (time headway) was 3 s and throughout the trajectory the lowest is $\delta T_{min} = 3.07$ s which is in accordance with the driver's risk taking trait θ. The TTC is high (always superior to 15 s) as standard critical values for TTC are 5 s: the car adapts his speed well in order not to trigger a potential dangerous situation. Therefore, if the car in front braked hard at any moment, a crash would never occur (NaN values for EES_{max} and $p(MAIS_{>2})_{max}$) and in the worst case the ego car would stop at standstill at about $d_{min} = 40$ m before the car in front. This behaviour is safe. It is very similar for the optimal search, which is normal as the agent aims at each moment to go the fastest possible: if

there is one lane only, it is optimal.

Results	
time	66.0 s
#lane changes	0
$sd_{\%}$	100.0 %
TTC_{min}	15.04 s
δT_{min}	3.07 s
h_{min}	85.83 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	39.82 m

Table 4.1: Results for the Car-following test (agent, $\theta = 0$)



(a) **(b)**

Figure 4.5: Speed and Acceleration, Car-following test (agent, $\theta = 0$)

However, as θ increases, the driver accepts lowest headway and thus takes more dangerous decisions. Similar simulation tests have been run with $\theta = 1$ (lowest tolerated $\delta T = 2\ s$) and $\theta = 3$ (lowest tolerated $\delta T = 0\ s$). For a safe application, θ should not be higher than 1 as a minimum of 2 seconds of time headway is advocated by governments (for example in Queensland [Gov10]). The simulation is run with $\theta = 3$ for test purposes. Speed and acceleration profiles are similar in all cases. However, trajectories' results differ as shown on Figure 4.2 ($\theta = 1$) and Figure 4.3 ($\theta = 3$).

As soon as the ego car is close enough to the car in front, it keeps approximately the same headway, which is unsafe from a standard driver's perspective (taken into account to assess if a distance is safe in the results, as explained in Section 3.1.1.1): from that moment safe distances are never respected. However, if the car in front was braking at its maximum deceleration until standstill, a crash

Results	
time	64.5 s
#lane changes	0
$sd_{\%}$	34.0 %
TTC_{min}	13.55 s
δT_{min}	2.05 s
h_{min}	57.33 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	17.08 m

Table 4.2: Results for the Car-following test (agent, $\theta = 1$)

Results	
time	64.0 s
#lane changes	0
$sd_{\%}$	34.0 %
TTC_{min}	13.55 s
δT_{min}	1.31 s
h_{min}	36.83 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	2.2 m

Table 4.3: Results for the Car-following test (agent, $\theta = 3$)

would never occur even for $\theta = 3$, where in the worst case the ego car would stop at standstill 2 meters before the car in front. This is due to the Gipps' safety criteria which does not depend on θ. Therefore decisions are taken in order that the headway is never smaller than the limit given by Gipps, which results in physically sound safe distances, as detailed in Section B.1.1 of Appendix B.

Be responsible for the gap with the car directly in front The driver is responsible only for the gap with the car directly in front, excepts when he changes lane. It has been taken into account and when taking a decision, the car considers only the distance with the car directly in front - except when it changes lane and in that case both gaps, front and backwards, are checked if acceptable. It will be verified in simulation test of Section 4.3: decisions will still be safe even when changing lane. As a result, all EES values are NaN in the results tables.

Keep a lower lane when possible This criterion has been taken into account for both agents, as explained in Section 3.2.1. It will be verified in simulation test of Section 4.3 where both agents will be compared on more complex scenarios.

These two last rules will be tested in Section 4.2 for the optimal search and Section 4.3 for the agents as compliance to traffic rules depends on driver's behaviour, therefore on the intelligence implemented and not the model itself.

4.1.1.2 System rules

Cars are systems with physical limits. It must be programmed as constraint in the program as it is not possible to exceed these limits. Cars must comply with the following rules:

- Acceleration should be comprised between $-b$ and g
- Speed should be inferior to the maximum possible speed of the car
- Speed should be superior to the minimum possible speed of the car
- Perception should be limited
- Trajectories should be physically feasible for the car

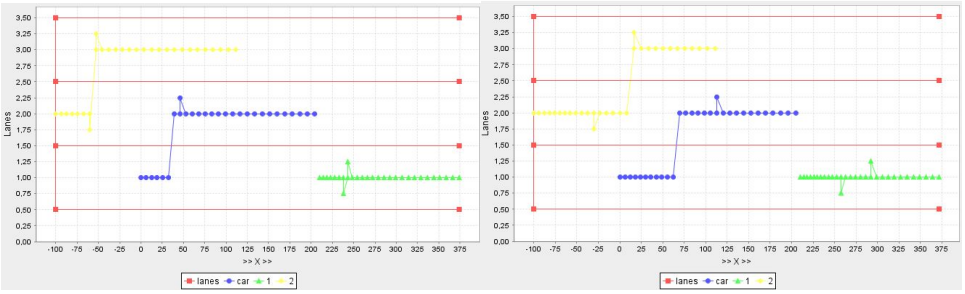
Acceleration is bounded, thus the algorithm can only output accelerations that are in accordance with the car's characteristics. Speed is also upper bounded in the model and cannot exceed v_{cmax}: if the acceleration of an action is too high, it is re-adjusted to target the maximum possible speed. It is also lower bounded: the minimum possible speed is theoretically negative but for simplification purposes, negative speeds are not consider on highways for this problem. Perception of the agent in the online simulation is limited by the characteristic of the sensors and is detailed in Section 3.1.1.4. But the lateral movement has not been modelled. It triggers issues for trajectories to be physically possible. A car is considered to be on a given lane: the model considers that the car changes lane instantaneously. This simplification was made to focus on the decisions to take rather than finding the perfect suitability of the trajectory for the car. For a usable application the solution must be realistic and realizable by a physical car but it exists smoothing trajectory techniques that could be used, such as the one for Stanley [TMD+06]. It was not within the scope of this thesis.

4.1.1.3 Human rules

Human rules describe especially the relation and interface between the system and the human driver. In case of fully automated driving, the human driver can choose what style of driving to take (similar to driving profiles) and whether to turn on the automated driving mode. In case of assisted driving, the system must let a degree of freedom to the human driver and takes into account that the reaction from the human driver may not be conform to what was planned. In this thesis, simulation is run in a fully automated driving mode as the driver reacts exactly to what is decided by the agent - which enables to focus on the decision process. Still, the system should adapt to:

- Different possible driving profiles
- Reaction time of drivers (either human reaction/execution or system execution time)

These rules are respected thanks to the flexibility of the model. Different driving profiles can be set in driver's characteristics: it will be tested in Section 4.2 as several trajectories are proposed for the optimal search, which differ from the characteristics of the driver set up. Reaction time can vary and is also taken into account, as shown on Figure 4.6 where $dt = 0.2\ s$.



(a) (b)

Figure 4.6: Reaction time of 0 s (a) and 1 s (b)

In this scenario the blue car overtakes the green car. The yellow car reacts that there is a slower vehicle in front and changes lane in order to overtake it after a reaction time. As explained by Figure 3.6 in Section 3.2.3, the car acts just after the displayed point. Figure 4.6 shows this behaviour: the model can adapt to different reaction time and each driver has his own reaction time.

4.1.2 Desired features

4.1.2.1 Flexibility

The program must be flexible in order to enable various possible simulations and also to be improved. The object-oriented architecture of the program enables easy changes to some particular components. For example, safety thresholds or system limits can be changed from one car to another. As another example, the simulation framework uses IDM models to simulate other cars' behaviours. Using another model just requires to change the agent's type. For instance, if all the IDM models are changed into *Greedy Agents*, and if the main agent is a *Greedy Agent*, the simulation is a macro simulation of *Greedy Agents* and shows its macroscopic behaviour. Other more complex simulation models could be used with the IDM model, especially for lane changing. Besides, if a performance measure is implemented, it can be plugged to the agent's structure in order to make it take more rational decisions and even learn. As a result, although the model has limits, additional features could be plugged in which is desirable for a simulation framework.

However, the program has limits on flexibility. The number of lanes cannot be changed in this first version: it would require changes in the design of the program. Also positions of cars are given only by their front abscissa x: so accurate perception cannot be performed if lengths of cars vary - which is the case in real life.

4.1.2.2 Computationally possible

Computation time for optimal search will be discussed in Section 4.2. For the online algorithm, time elapsed during one agent loop (cf. Figure 3.9 in Section 3.2.3) has been averaged over all the passages through the loop of the whole complex scenario of Section 4.3.2.4 for each kind of agent (*Greedy* or *BDI*). It has been done five times for each agent, and the average of these running times has been calculated: results are shown on Table 4.4. Tests have been run on Eclipse Juno (4.2.2) with jre 7. One computation at t is done in about 0.1 ms which is satisfying for a real-time application, provided that the sampling time is at least an order of magnitude bigger.

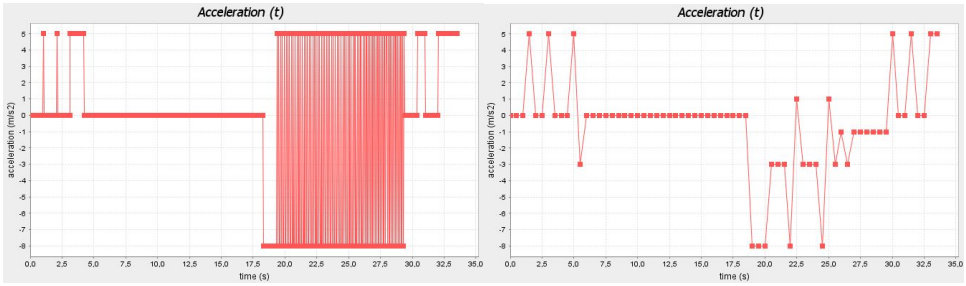
Results						average
Greedy Agent	115.690	86.821	82.651	94.494	100.999	**96.131**
BDI Agent	108.401	104.689	100.322	97.880	98.494	**101.957**

Table 4.4: Running time in micro seconds of an online decision

4.1.2.3 Appropriate sampling time

In the model, the sampling time dt is fixed and cars take decisions every $\delta t = dt$. It simulates a continuous process of decision taking. The sampling time needs to be relatively small since the environment in dynamic: cars evolve fast and continuously. As a result the model has been implemented to support a sampling time dt that can divide reaction time of the drivers, that are typically below 2 seconds. So the times of perception and action match the time steps. A small sampling time enables the agent to take more informed decisions. Moreover, as explained in the section above, the computation time is about 0.1 ms which enables dt to be as small as about 1 ms (one order of magnitude bigger). However, changes in the environment are not that frequent: a car at 40 m/s travels only 2 meters in 0.05 s. Since gap distances are meant to be large (more than 20 m), it is negligible and decisions can be taken at down to 20 Hz: which is the highest frequency at which Stanley was able to compute decisions ([TMD+06]). Nevertheless, the model of this thesis is threshold-based to decide what acceleration to take: at each time step the algorithm computes an acceleration to take from the time t to the time $t + dt$ in order to reach a targeted acceleration (cf. Section 3.2.2). If dt is too small (about 50 ms for example) the algorithm will take decisions for very short terms which can result in saw-teeth decisions, as shown on Figure 4.7). This graphs corresponds to a scenario where the lane of the ego car is blocked and it needs to brake and wait for the lower lane to be free to change lane.

Therefore, a bigger dt will be considered. A car at 40 m/s travels 20 meters in 0.5 s. Gap distances are meant to be even larger than 80 m at that speed; moreover the algorithm takes into account that an emergency braking may occur (cf. B.1.1 related to Gipps' safety distance). Therefore a sampling time of $dt = 0.5$ s is acceptable. As it has shown good behaviour, tests are run with $dt = 0.5$ s.



(a) (b)

Figure 4.7: Acceleration in case of hard braking for $dt = 0.05\ s$ (a) and $dt = 0.5\ s$ (b)

4.1.2.4 Accuracy of cars' positions

The longitudinal position of a car is given by the abscissa x of its front. Although the car has a length, only its front point is considered in the description of the environment. As a result, detection of crash is not adapted to this design and has not been implemented. It is also not adapted to exactly percept 200 meters in front and behind. For example, if the ego car is at $x = 0$ and the car directly in front at $x = 203\ m$ and has a length of 5 m, it is not detected, although the back of the car is within 200 meters. However, it is not crucial in the model to detect cars at 200 or 195 m, it could even vary in a real situation, so this flaw is minor. A more adapted model would be store the whole position of the car (from $x-L$ to x.). However, this simple representation is sufficient for calculating safety distances, as the length of the car in front L_{ft} is considered in Gipps' safety distance, which is desired in this model.

Cars are predicted to be at one given position. However, since the environment is stochastic, it would be more adapted to adopt a stochastic representation of cars' position. But in that case decisions would be probabilistically-based and the agent would need some kind of performance measure to assess what trade-off is best between potential safety and speed. As explained earlier, further studies to elaborate an appropriate performance measure should be conducted to do so.

4.2 Analysis of the optimal search

4.2.1 Purpose of the optimal search

As explained in Section 3.3, optimality of this algorithm is reduced to the relaxed problem where there are only 3 different accelerations considered possible for each adjacent lane at each time step. Moreover, a node is reduced to its path cost for the search: it is considered better or worse than another node only based on this cost. But the optimality criterion has not been properly defined as further studies must be conducted to evaluate an appropriate performance measure. So far the cost is only associated to the time spent, while safety criteria are rather treated as constraints, which is a simplified version. Therefore, the use of this algorithm does not aim at finding the optimal trajectory for the driver, but rather at showing different suboptimal solutions, to be compared with the solution computed online by the agent.

4.2.2 Use of the optimal search

4.2.3 Methodology

Different "optimal" solutions are computed by making the driver's characteristics vary:

- θ: as explained in Section 3.1.2.3, the agent does not consider a trajectory that would induce a headway smaller than $v_i(t)\cdot(3-\theta_j)$ which is equivalent to letting $3-\theta_j$ seconds of separation with the car directly in front - and with the car directly behind in case the ego car changes lane
- μ: the maximum desired speed $v_{md} = v_{max} \cdot (1 - 0.1 * (1 - \mu))$. For $\mu = 1$, the driver accepts a speed up to the maximum speed allowed v_{max}. While μ decreases, the driver accepts lower speeds down to 80% of v_{max} for $\mu = -1$ for this study. μ can actually be decreased down to -9 which is equivalent to an immobile car.

Tests will be run for θ,$\mu = -1$ (case (-1)), θ,$\mu = -0.5$ (case (-0.5)), θ,$\mu = 0$ (case (0)), θ,$\mu = 0.5$ (case (0.5)) and θ,$\mu = 1$ (case (1)) to make the driver accept different ranges of speed and safety distances. As shown on Figure 2.1 of Section 2.1.1.2, it makes the risk threshold of the driver varies.

4.2.3.1 Example of an optimal search

A simulation has been run in order to show the results of this algorithm with the methodology above. Three IDM-cars are on the road. The slowest one is on lane 1, its desired speed is 20% lower than v_{max}. The desired speed of the car in front on lane 2 is 15% lower than v_{max} and the desired speed of the car behind on lane 2 is 10% lower than v_{max}. The ego car starts on lane 1. The results are presented on Figure 4.8. Hybrid A* has been used for these simulations.

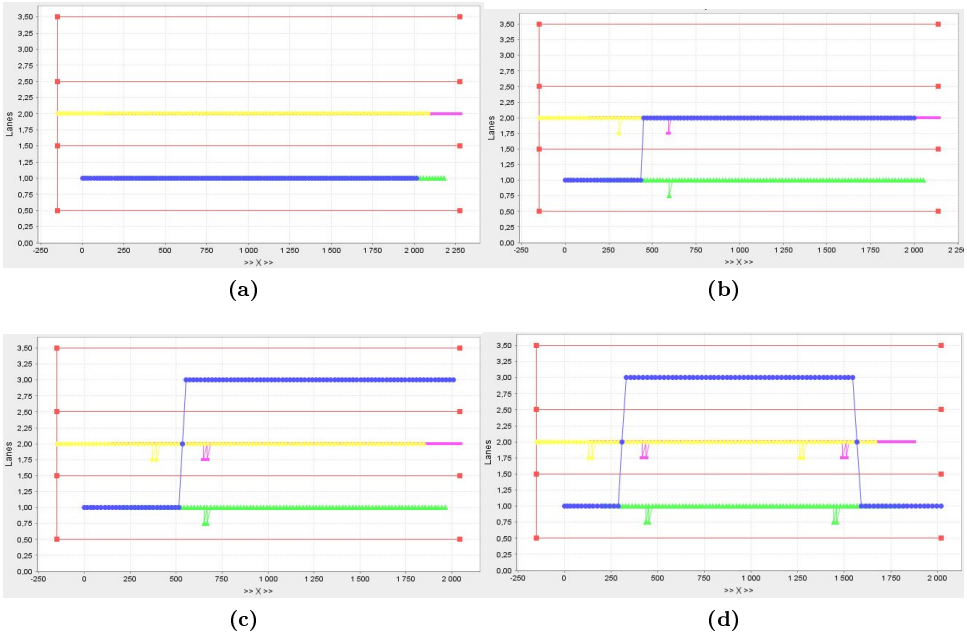


Figure 4.8: Trajectories for case (−1) (a), case (−0.5) (b), case (0) (c) and case (1) (d)

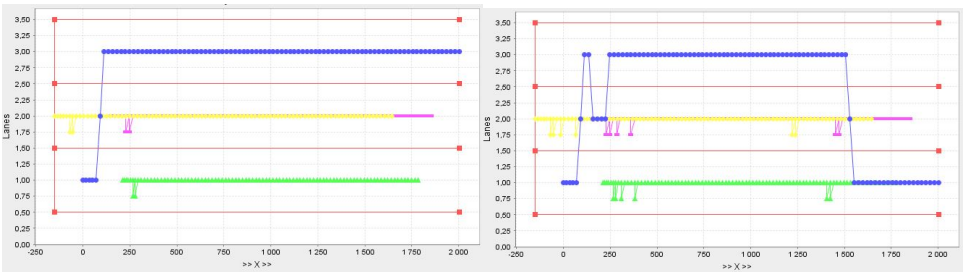
For case (−1), the driver does not want to go faster than the car in front so the agent keeps lane. For case (−0.5), the agent overtakes the car in front at the beginning but does not want to go faster the car in front on lane 2 so it keeps this lane then. However for cases (0), (0.5) and (1), the agents overtakes as the driver wants to go faster than all cars. Cases (0) and (0.5) are very similar thus the trajectory of (0.5) is not displayed. This example shows that diverse trajectories can be computed that satisfy various optimality criteria, which is interesting to compare them and assess which one is actually best. Since there is not accurate performance measure, as it has been explained in previous sections, these trajectories can be compared to check if the gain in time is high enough to

justify taking more risk - but do not provide an accurate benchmark to assess the optimality.

4.2.4 Flaws of the optimal search

4.2.4.1 Unrealistic or undesirable behaviour

Cost-based solutions As explained in Section 3.2.1.1 and through Chapter 3, the manoeuvre parameter γ has not been used for the online search. However, if a penalty is not given for a lane change, the optimal search will consider that a trajectory going straight or changing lane at each time step, at same speeds, are equivalent. As a result, the algorithm may output trajectories with a car changing lane very often. To avoid this problem, a constant lane changing cost $lc = 1$ has been added to the path cost when the car goes to an upper lane, which is equivalent to say that it costs 1 s to go to an upper lane. So the car will not change lane if it does not make it gain 1 s in total. This value is arbitrary and has been set up to avoid a "ping pong" effect only. Figures 4.9 show that phenomenon.



(a) (b)

Figure 4.9: Taking account a lane changing penalty (a) or not (b)

Further studies must be conducted to assess an appropriate lane changing penalty. It should be noticed that there is no cost for getting back on a lower lane, and at constant time cost the algorithm prefers going on a lower lane. As a consequence, the agent does get back on a lower lane when possible, and so still respects the rule *Keep a lower lane when possible* above.

Counterpart of the omniscience As explained in Section 3.3.1.3, the algorithm knows what the other drivers will do. But it can trigger unrealistic

behaviour, as shown on Figure 4.10.

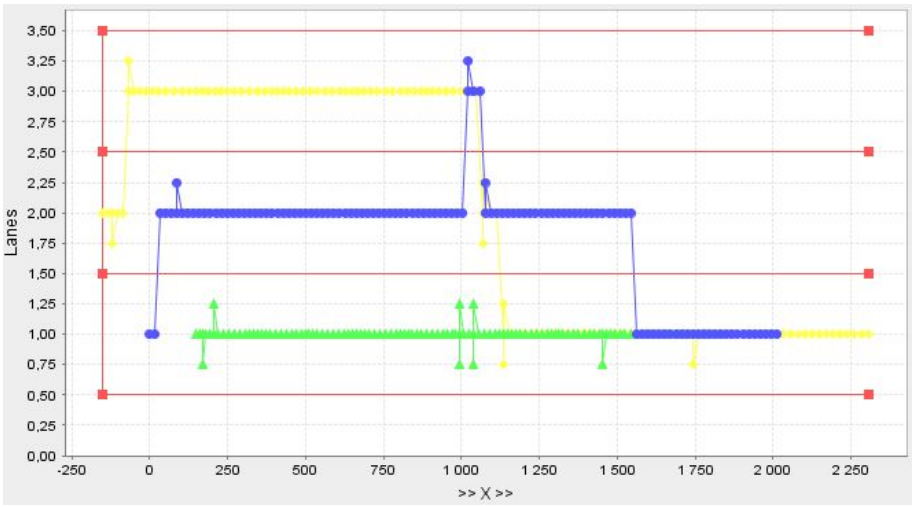


Figure 4.10: Unrealistic behaviour in an optimal search due to the omniscience

At about 1000 m, the algorithm knows that the yellow car will get back. The yellow car is *BDI Agent* and desires to get back once it estimates it is safe. However, its parameter θ is very high: $\theta = 2$ which means that the driver accepts a safety distance that only let 1 s between him and the car behind of in front. But the θ parameter of the blue car is -1, which means that he accepts safety distances that let 3 seconds between him and the other cars. It cannot get back because there is a car on a lower lane. As a result, it overtakes, although at the time step when it takes this decision, the yellow car is on the lane above. It is unrealistic but complies with the optimality criterion since the driver would not accept to keep lane. As a result, one may be careful in the interpretation of results of optimal searches as supposing that everything is known can induce unnatural behaviours.

4.2.4.2 Computation cost

As explained in Section 3.3.1.3, hybrid A* may be used. For the most basic scenario, A* computes solutions but when it gets more complex and computationally expensive, hybrid A* will be used. If not specified, A* is used. As an example for comparisons, A* and hybrid A* have been run on the scenario of Section 4.2.3.1 which was not so complex though. Running times, number of created nodes and checked nodes are displayed on Table 4.5. A created node is a

node that has been expanded during the search. A checked node is a node that has been pulled out the queue. Tests have been run on Eclipse Juno (4.2.2) with jre 7. The symbol *?* is displayed for running times when the algorithm did not find a solution because of an *Out Of Memory* error. Although running time are a lot higher for A* especially for higher values of θ for which more trajectories are possible, the trajectories retrieved are very similar for this scenario as shown on Figure 4.11 and 4.8 (previous section) - it does not lose too much optimality on this test. The high running times, and even the impossibility to retrieve solution when the scenario is slightly more complex (with $\theta = 0.5$ on this very basic scenario for example) highlights the need of relaxing the problem.

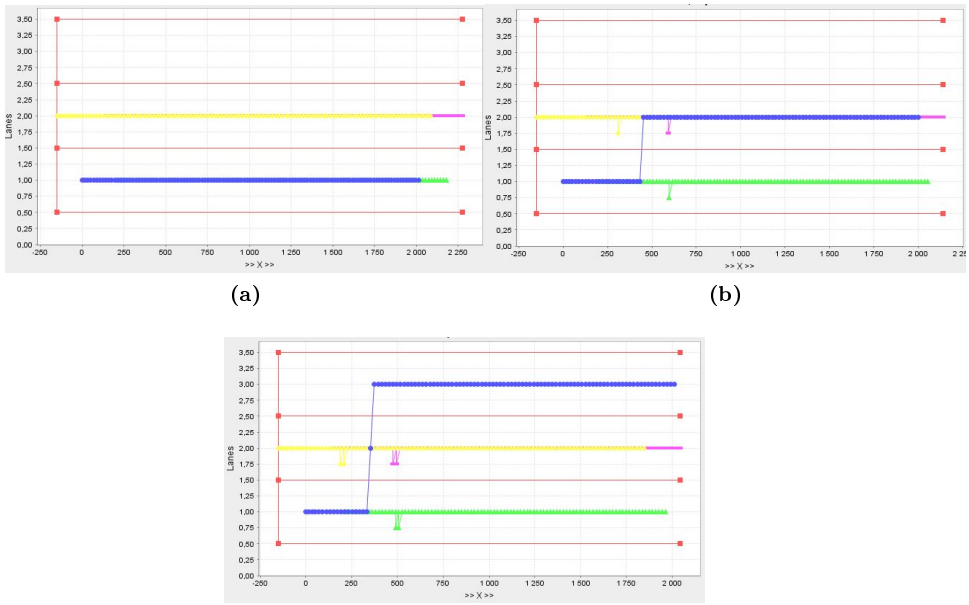


Figure 4.11: Trajectories for case (−1) (a), cases (−0.5) and (0) (b) and case (0) (c)(regular A*)

Results	Regular A*	Hybrid A*
Case -1		
Running time	245 ms	251 ms
#created nodes	364	358
#checked nodes	114	113
Case -0.5		
Running time	5.3 s	474 ms
#created nodes	95796	3262
#checked nodes	29931	835
Case 0		
Running time	66.8s	483 ms
#created nodes	494844	11754
#checked nodes	97373	2158
Case 0.5		
Running time	*?*	345 ms
#created nodes	> 500000	11204
#checked nodes	> 100000	1938
Case 1		
Running time	*?*	220s
#created nodes	> 500000	9683
#checked nodes	> 100000	1623

Table 4.5: Computation cost for Regular and Hybrid A*

4.3 Evaluation and comparison of the agents

4.3.1 Methodology

In this section, the agents *Greedy* and *BDI* will be compared to each other on various scenario. These scenarios have been designed to check if the behaviour of these agents is realistic and "intelligent" on basic situations. Results will also be compared to trajectories computed with optimal searches. Scenarios that will be tested are:

- Accelerate on a free road
- Brake until full stop because of an obstacle in front
- Follow a car
- Overtake a car

- Foresee the need of overtaking: overtake a car early enough before getting stuck by other cars
- React to lane merging
- Travel on a road with many cars (more complex scenario)

In addition to these 7 scenarios, a scenario will be launched with many agents of the same type to assess the validity of the model macroscopically. It will allow lane changes for the other cars - which is not the case with the IDM cars - but also verify if traffic is realistic and acceptable with all cars as such agents on the road.

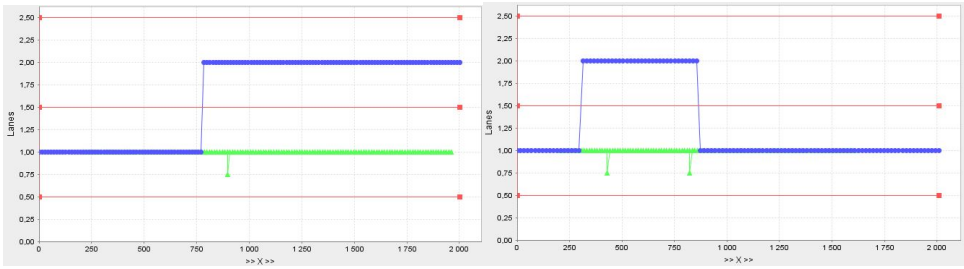
4.3.2 Test scenarios

The first three scenarios have been previously tested in Section 4.1.1.1 and the behaviour is optimal and the same for *Greedy* and *BDI* agents. For the following scenarios, if not specified, the parameters for a car are $m = 1800\ m.s^{-1}$, $L = 4.5\ m$, $g = 5\ m.s^{-1}$ and $b = 8\ m.s^{-2}$ and for a driver $\tau = 1\ s$, $\theta = 0$ and $\mu = 0$. Speed limit is set to 35 $m.s^{-1}$. Results for the other optimal solutions than case 0 (that correspond to the characteristics of the driver for the online agent tests) are almost not presented for basic scenario as these results are trivial; the optimal search will be rather useful for the complex scenario.

4.3.2.1 Overtake a car

At the beginning the agent goes at the maximum desired speed of the driver, which is 10% less than v_{max}. The car is on lane 1. There is another car on lane 1 at 210 m in front, going at 30% slower than v_{max} ($\mu = -2$).

Results for the *Greedy Agent* are shown on Table 4.6, the trajectory and speed on Figure 4.13. Results for the *BDI Agent* are shown on Table 4.7, the trajectory and speed on Figure 4.14. Making (θ, μ) varies just advances or delays the moment of overtaking, to keep distances safe for the corresponding driver, and changes the speed which is maintained, as shown on Figure 4.12. Getting different optimal solutions is not relevant for the purpose of this test. So the results are only given for case 0: these results are actually exactly the same as the *Greedy Agent*.



(a) (b)

Figure 4.12: Optimal trajectories for case (−1) (a) and (1) (b)

Results	
time	63.5 s
#lane changes	2
$sd_{\%}$	100.0 %
TTC_{min}	13.5 s
δT_{min}	3.0 s
h_{min}	94.5 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	35.75 m

Table 4.6: Results for the Greedy Agent for the Overtaking test

Analysis and Comparisons For the *Greedy Agent* the trajectory is safe and the agent goes all long at v_{md}. The agent chooses to overtake at the moment when it should decelerate on the lowest lane to keep the same speed. The *BDI Agent* chooses to overtake as soon as it can perceives the car in front, which makes the trajectory even safer and still the agent goes all long at v_{md}. The optimal solution for case 0 is the same as the *Greedy*'s one as it was programmed to keep a lower lane when possible.

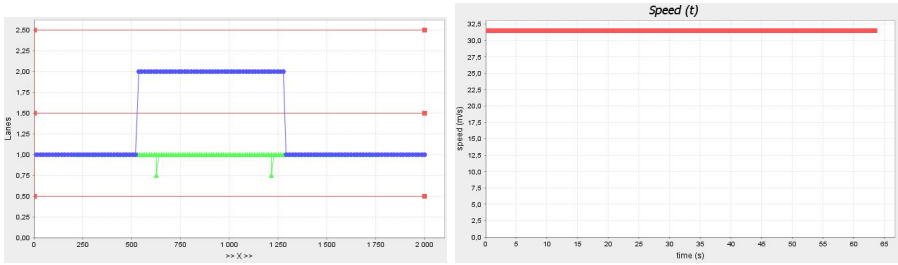


Figure 4.13: Trajectory and Speed for the Overtaking test, Greedy Agent

Results	
time	63.5 s
#lane changes	2
$sd_{\%}$	100.0 %
TTC_{min}	27.5 s
δT_{min}	6.11 s
h_{min}	192.5 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	70.75 m

Table 4.7: Results for the BDI Agent for the Overtaking test

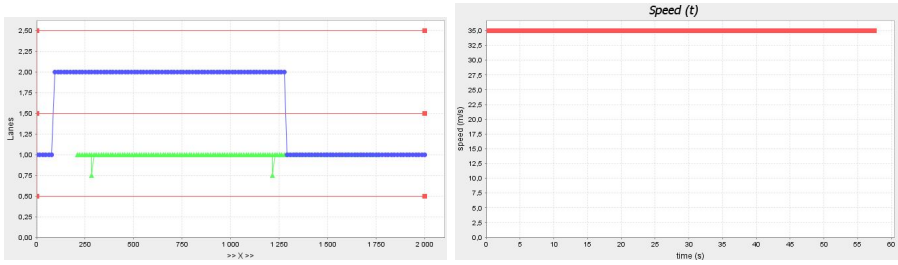


Figure 4.14: Trajectory and Speed for the Overtaking test, BDI Agent

4.3.2.2 Foresee the need of overtaking

At the beginning the agent goes at the maximum desired speed of the driver, which is 10% less than v_{max}. The car is on lane 1. There is another car on lane 1 at 250 m in front, going at 35% slower than v_{max} ($\mu = -2.5$). There is another car on lane 2 at 150 m in front, going at 30% slower than v_{max} ($\mu = -2$). Because of high complexity, hybrid A* is used for this simulation.

Results for the *Greedy Agent* are shown on Table 4.8, the trajectory on Figure 4.15 and the speed and acceleration on Figure 4.16. Results for the *BDI Agent* are shown on Table 4.9 the trajectory on Figure 4.17 and the speed and acceleration on Figure 4.18. Just like the previous scenario, making (θ, μ) varies just advances or delays the moment of overtaking; moreover it is not particularly relevant for this scenario. So the results are only given for case 0 on Table 4.10 and Figures 4.19 and 4.20.

Analysis and Comparisons For the *Greedy Agent* the trajectory is almost always safe: during 1 s only it is not. It happens when the agent chooses to overtake the first vehicle and goes on lane 2 at time t. Then during 1 s (the reaction time of the driver), no decision is taken and instead the agent goes at same speed - because there was no perception on the lane 2 between $t - \tau$ and t. The *BDI Agent* chooses to overtake as soon as it can perceives the car in front, which makes its trajectory safer but especially faster as it does not get blocked by the car of lane 2. The optimal solution for case 0 is quite close to the *BDI*'s one: it overtakes early enough not to get stuck. However it overtakes later as it was programmed to keep a lower lane when possible. Results are very

Results	
time	106.5 s
#lane changes	4
$sd_{\%}$	99.0 %
TTC_{min}	11.07 s
δT_{min}	2.3 s
h_{min}	70.75 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	25.5 m

Table 4.8: Results for the Greedy Agent for the Foresee Overtaking test

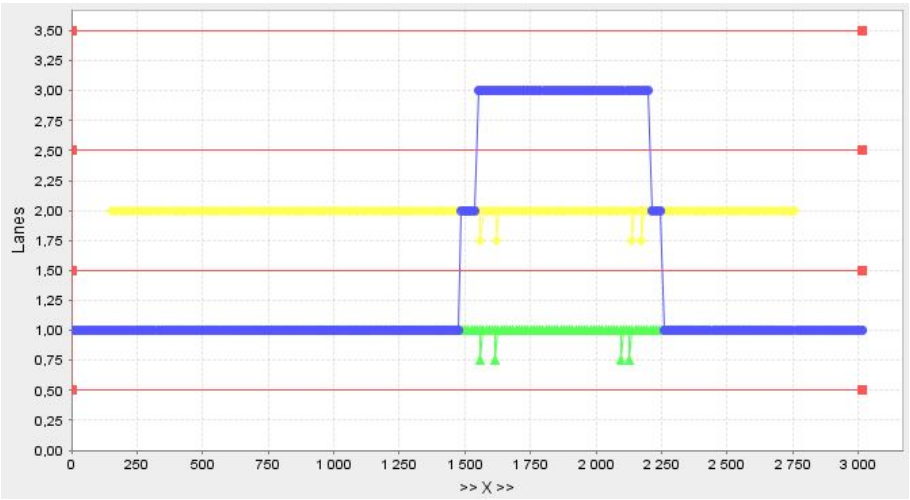
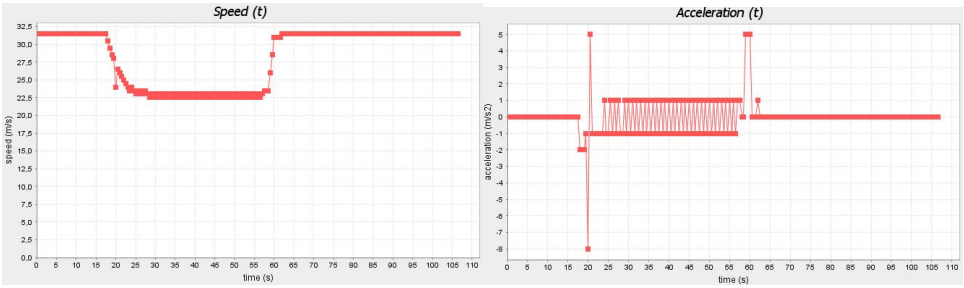


Figure 4.15: Trajectory for the Foresee Overtaking test, Greedy Agent



(a) (b)

Figure 4.16: Speed and Acceleration for Greedy Agent, Foresee Overtaking test

Results	
time	95.5 s
#lane changes	4
$sd_{\%}$	99.0 %
TTC_{min}	12.93 s
δT_{min}	2.87 s
h_{min}	90.5 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	31.75 m

Table 4.9: Results for the BDI Agent for the Foresee Overtaking test

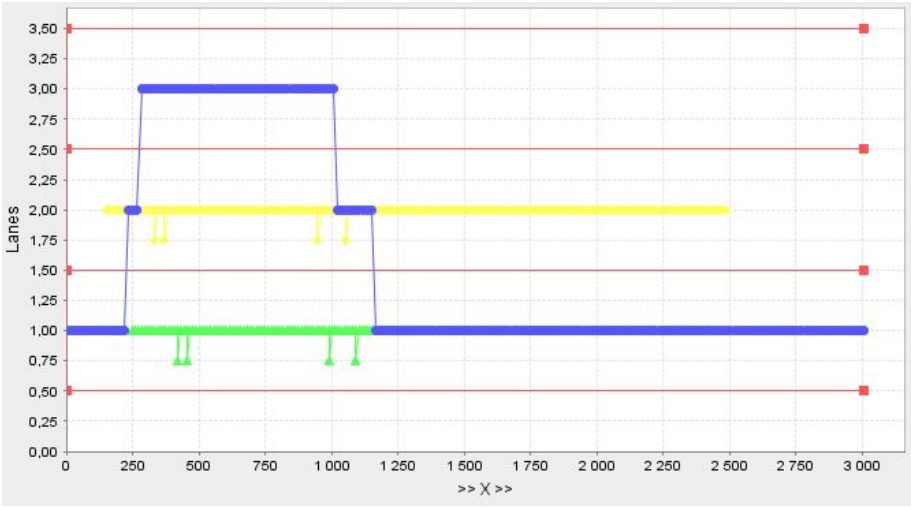
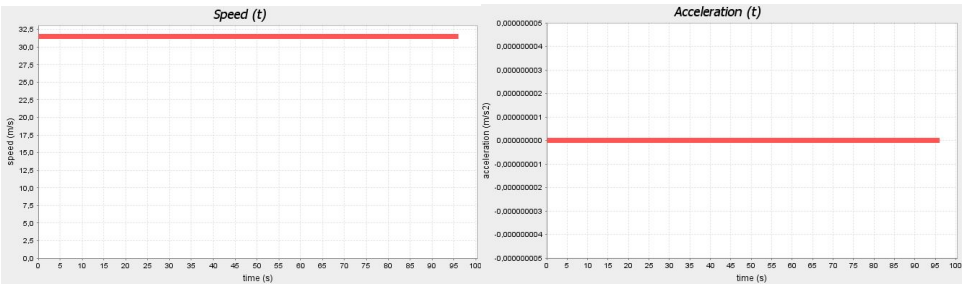


Figure 4.17: Trajectory for the Foresee Overtaking test, BDI Agent



(a) (b)

Figure 4.18: Speed and Acceleration for BDI Agent, Foresee Overtaking test

Results	
time	95.5 s
#lane changes	4
$sd_{\%}$	99.0 %
TTC_{min}	28.12 s
δT_{min}	2.63 s
h_{min}	81.0 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	50.5 m

Table 4.10: Results for the Optimal Solution (0) for the Foresee Overtaking test

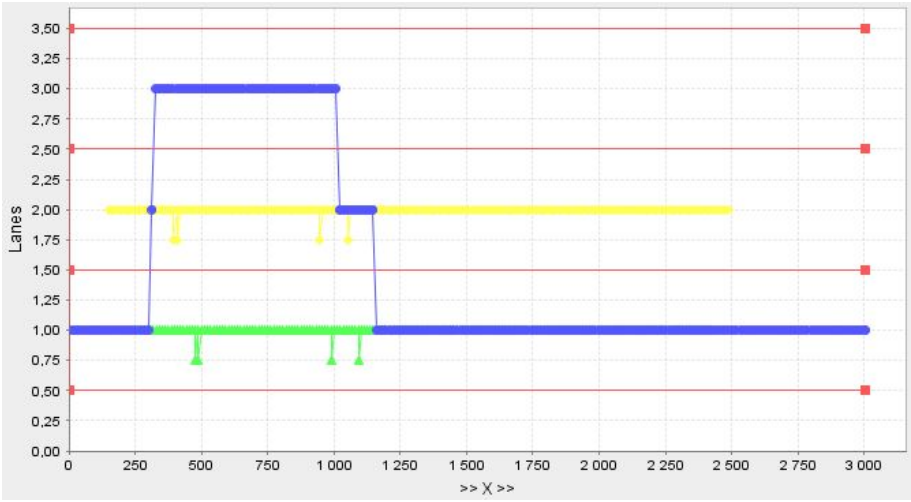
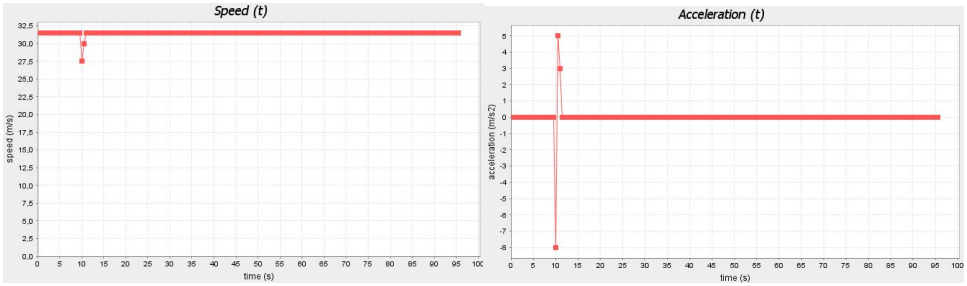


Figure 4.19: Trajectory for the Foresee Overtaking test, Optimal case (0)



(a) **(b)**

Figure 4.20: Speed and Acceleration for the Optimal case (0), Foresee Overtaking test

similar and both trajectories are safe. Also, one should notice that speed is not maintained at the maximum for the optimal search - but still the time to reach 3000 m is 95.5 s like the agents. This is why the TTC_{min} is quite higher for the optimal solution. This is because the optimal search is programmed to keep a lower lane when possible so in that case overtaking was late and there was also some deceleration, nevertheless the loss in time is less than $dt = 0.5$ s. This phenomenon always occurs, so it will not be noticed again in other scenarios' analysis.

4.3.2.3 Lane-merging

At the beginning the agent goes at the maximum desired speed of the driver, which is 10% less than v_{max}. The car is on lane 2. There is another car on lane 2 at 120 m in front, going at 30% slower than v_{max} ($\mu = -2$). There is another car on lane 1 at the same abscissa, going also 30% slower than v_{max} ($\mu = -2$). An immobile car is placed at $x = 1000$ m to simulate a lane merging. These initials conditions make the agent to overtake but they better not do it as lane 3 ends 1 km and there is not enough time to overtake safely. Because of high complexity, hybrid A* is used for this simulation.

Results for the *Greedy Agent* are shown on Table 4.11, the trajectory on Figure 4.21 and the speed and acceleration on Figure 4.22. Results for the *BDI Agent* are shown on Table 4.12 the trajectory on Figure 4.23 and the speed and acceleration on Figure 4.24. For this scenario, making (θ, μ) shows various behaviours; speed and acceleration can be easily inferred from the trajectory graph, therefore only these graphs will be displayed, on Figure 4.25. Results of all the cases are given on Tables 4.13, 4.14, 4.15, 4.16 and 4.17.

Analysis and Comparisons Both agents go on lane 3 and then manage to get back on lane 2 in time. The trajectories are not evaluated as safe because of the obstacle which is considered as a car. Since both agents are programmed to go as fast as possible, they get closer and closer to the obstacle until almost standstill before getting back. Optimal solutions with θ inferior to 0 give trajectories more realistic (better keeping lane 2), safer and that take approximately the same time. The optimal solutions with θ superior to 0 are faster and quite safe. For cases (0.5) and (1), the maximum desired speed is high enough to overtake cars. Case (1) even gets back from the beginning, taking a bit more risk but then having the road free of cars. Case (0) is not realistic as the car overtakes then brakes hard to get back with a low headway behind the car of lane 2 and finally on lane 1. A solution "really optimal" for this theta would

Results	
time	46.5 s
#lane changes	3
$sd_{\%}$	76.0 %
TTC_{min}	2.88 s
δT_{min}	2.06 s
h_{min}	11.75 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	7.25 m

Table 4.11: Results for the Greedy Agent for the Lane Merging test

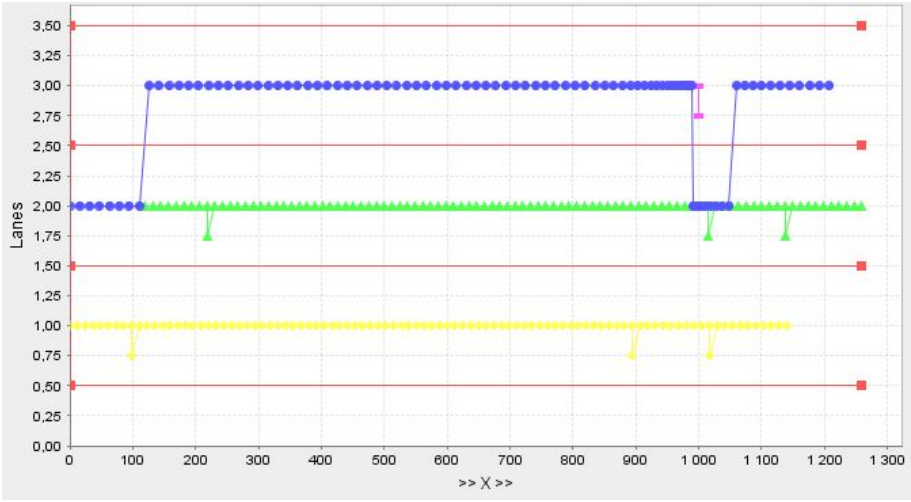
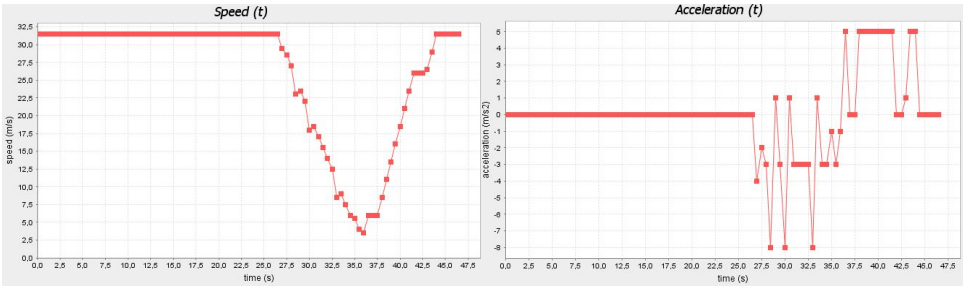


Figure 4.21: Trajectory for the Merging Lane test, Greedy Agent



(a) (b)

Figure 4.22: Speed and Acceleration for Greedy Agent, Lane Merging test

Results	
time	47.0 s
#lane changes	3
$sd_{\%}$	77.0 %
TTC_{min}	2.88 s
δT_{min}	2.06 s
h_{min}	11.75 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	7.25 m

Table 4.12: Results for the BDI Agent for the Lane Merging test

Results	
time	48.5 s
#lane changes	0
$sd_{\%}$	100.0 %
TTC_{min}	32.04 s
δT_{min}	4.09 s
h_{min}	100.25 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	69.13 m

Table 4.13: Results for the Optimal case (−1) for the Lane Merging test

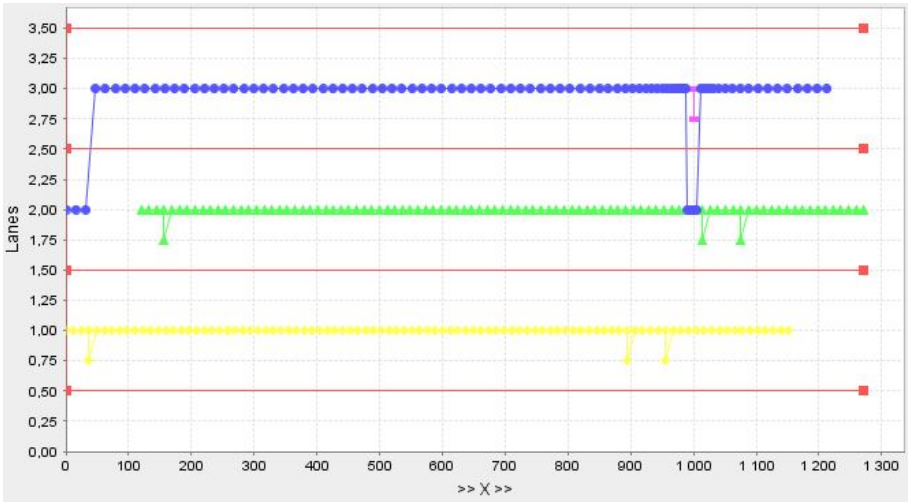
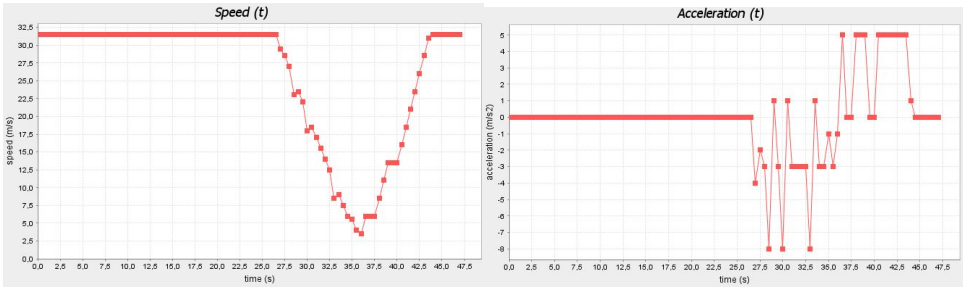


Figure 4.23: Trajectory for the Merging Lane test, BDI Agent



(a) (b)

Figure 4.24: Speed and Acceleration for BDI Agent, Lane Merging test

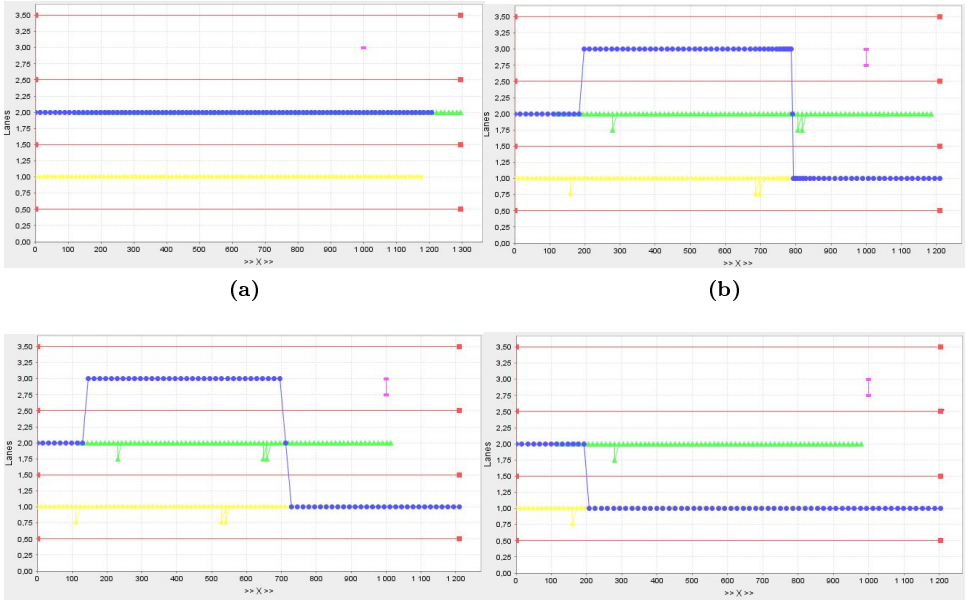


Figure 4.25: Optimal trajectories for cases (−0.5) (a), (0) (b), (0.5) (c) and (1) (d),

Results	
time	48.0 s
#lane changes	0
$sd_{\%}$	100.0 %
TTC_{min}	20.27 s
δT_{min}	3.61 s
h_{min}	88.56 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	55.56 m

Table 4.14: Results for the Optimal case (−0.5) for the Lane Merging test

Results	
time	43.5 s
#lane changes	3
$sd_{\%}$	98.0 %
TTC_{min}	5.77 s
δT_{min}	1.66 s
h_{min}	16.0 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	34.25 m

Table 4.15: Results for the Optimal case (0) for the Lane Merging test

Results	
time	36.5 s
#lane changes	3
$sd_{\%}$	95.0 %
TTC_{min}	11.76 s
δT_{min}	2.66 s
h_{min}	87.94 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	30.31 m

Table 4.16: Results for the Optimal case (0.5) for the Lane Merging test

Results	
time	35.0 s
#lane changes	1
$sd_{\%}$	89.0 %
TTC_{min}	11.7 s
δT_{min}	2.23 s
h_{min}	75.0 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	25.25 m

Table 4.17: Results for the Optimal case (1) for the Lane Merging test

be to keep lane 2 until being able to get back on lane 1. Though, the algorithm chooses this solution because by braking hard on lane 3 before getting back on lane 2, the safety distance to keep with the car on lane 2 is lower than if it was going at a higher speed on lane 2. As a result it can get back a bit earlier on lane 1. It clearly shows the limits of the atomic representation of the state (reducing the state to a cost). However, in this scenario it is interesting to have a large range of solutions provided by optimal searches, and one can choose a trajectory that seemed better. If safety is extremely important and speed not so important, case (0.5) suits well; if speed is important and safety can be just what advocated and no more, cases (0.5) and (1) suit better.

4.3.2.4 Complex scenario

This scenario has been set up to show the behaviour on a longer time. But for visualization purposes, the number of other cars have been limited to 8 and the distance goal is $Xgoal = 5\ km$. There are three cars on lane 1 that go slower than the ego car. There are two cars on lane 2: the one in front goes slower than the ego car but the one behind goes faster. There are three cars on lane 3, all of them go faster than the ego car but the one behind that goes at the same speed.

Results for the *Greedy Agent* are shown on Table 4.18, the trajectory on Figure 4.26 and the speed and acceleration on Figure 4.27. Results for the *BDI Agent* are shown on Table 4.19 the trajectory on Figure 4.28 and the speed and acceleration on Figure 4.30. Optimal results are shown on Tables 4.20 4.21, 4.22, 4.23 and 4.24, and trajectories graphs are presented on Figure 4.29.

Results	
time	169.5 s
#lane changes	12
$sd_{\%}$	99.0 %
TTC_{min}	19.71 s
δT_{min}	2.85 s
h_{min}	74.88 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	29.42 m

Table 4.18: Results for the Greedy Agent for the Complex Scenario

Results	
time	166.5 s
#lane changes	2
$sd_{\%}$	100.0 %
TTC_{min}	13.93 s
δT_{min}	2.94 s
h_{min}	75.25 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	37.0 m

Table 4.19: Results for the BDI for the Complex Scenario

Results	
time	179.0 s
#lane changes	2
$sd_{\%}$	100.0 %
TTC_{min}	42.63 s
δT_{min}	3.88 s
h_{min}	108.46 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	71.73 m

Table 4.20: Results for the Optimal case (-1) for the Complex Scenario

Results	
time	168.5 s
#lane changes	2
$sd_{\%}$	100.0 %
TTC_{min}	21.89 s
δT_{min}	4.02 s
h_{min}	114.94 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	64.06 m

Table 4.21: Results for the Optimal case (-0.5) for the Complex Scenario

Results	
time	160.0 s
#lane changes	4
$sd_{\%}$	100.0 %
TTC_{min}	27.61 s
δT_{min}	2.62 s
h_{min}	80.75 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	56.38 m

Table 4.22: Results for the Optimal case (0) for the Complex Scenario

Results	
time	151.0 s
#lane changes	6
$sd_{\%}$	98.0 %
TTC_{min}	11.65 s
δT_{min}	2.23 s
h_{min}	72.44 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	31.69 m

Table 4.23: Results for the Optimal case (0.5) for the Complex Scenario

Results	
time	143.0 s
#lane changes	8
$sd_{\%}$	98.0 %
TTC_{min}	11.35 s
δT_{min}	1.92 s
h_{min}	66.75 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	17.38 m

Table 4.24: Results for the Optimal case (1) for the Complex Scenario

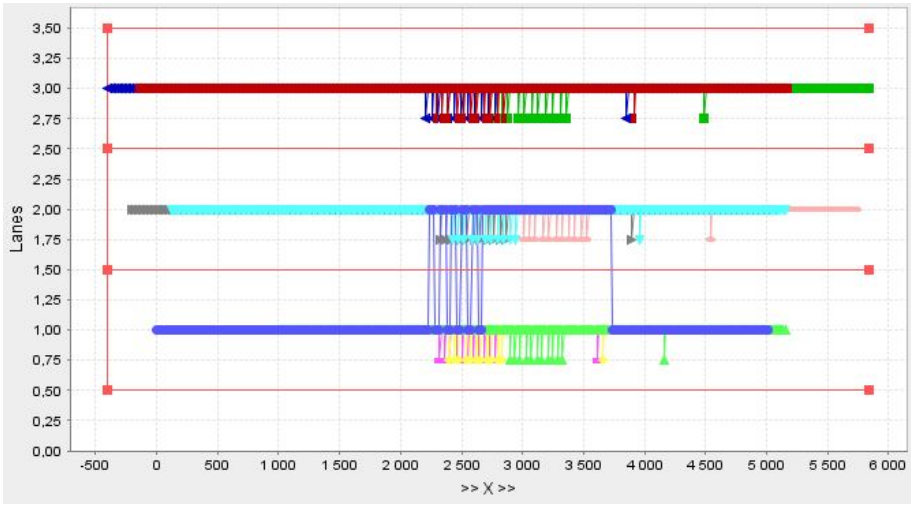
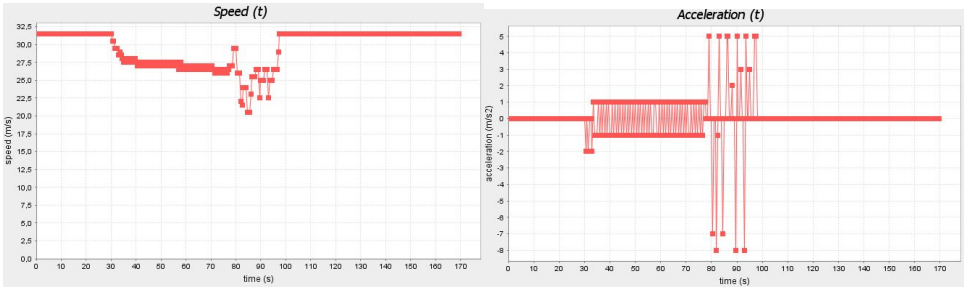


Figure 4.26: Trajectory for the Complex scenario, Greedy Agent



(a) (b)

Figure 4.27: Speed and Acceleration for Greedy Agent, Complex scenario

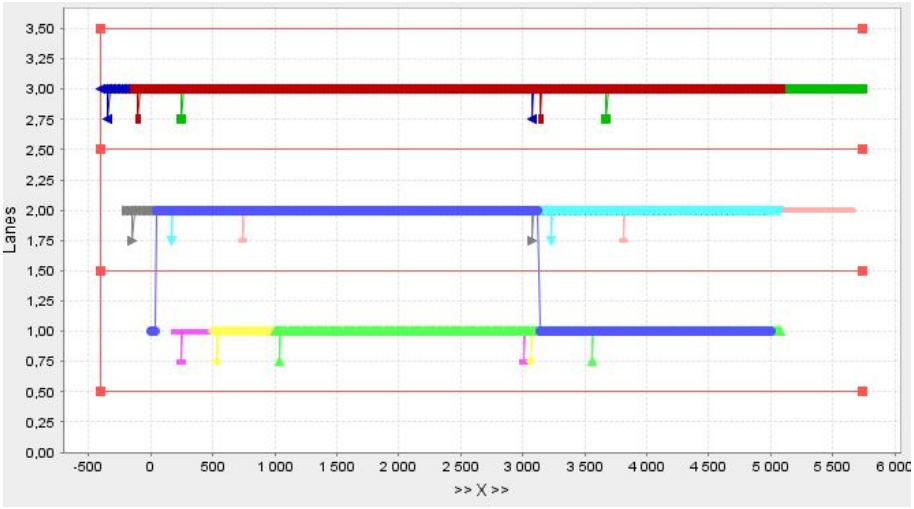


Figure 4.28: Trajectory for the Complex scenario, BDI Agent

Analysis and Comparisons The *Greedy Agent* does not overtake at the beginning and gets stuck. Its trajectory is quite safe but he keeps changing lane, alternating 1 and 2, as the cars in front travel too slow. This effect results from the fact that no penalty charge is applied for the agent. These decisions are not rational and highlight the limits of this agent. Conversely, the *BDI Agent* overtakes at the beginning and gains 4 s of time to travel 5 km as it overtakes in between the cars of lane 2. Its trajectory is even safe as it keeps larger headways with cars in front. The optimal solutions (-1) and (-0.5) travel approximately at the same speed as the *Greedy Agent* although the agents have lower desired speed; moreover they their trajectories are safer as they keep higher headways. They are similar to the *BDI Agent*'s one. The optimal solution (0) performs better than the *BDI Agent* because it does not need a reaction time to perceives that it is safe to go on lane 3 immediately. Optimal solutions (0.5) and (1) go quite faster (10 to 20 seconds faster) as they accept higher speed and lower headways to overtake. Still, they are safe, except case (1) which is a bit more dangerous as it reaches time headways below 2 s. However these two solutions seem to achieve a better trade-off time/safety. Tests have been run with $\mu = 1$ for the agents and results are displayed on Figure 4.31. It shows that even with a higher desired speed, the *Greedy Agent* gets stuck. However, the *BDI Agent* accepts higher speed and therefore is able to overtake more cars and maintain a high speed: it takes 148 s to reach 5 km, and the trajectory is safe as shown on Table 4.25. Which is close to the optimal case (1). The *BDI Agent* performs clearly better on this more complex scenario.

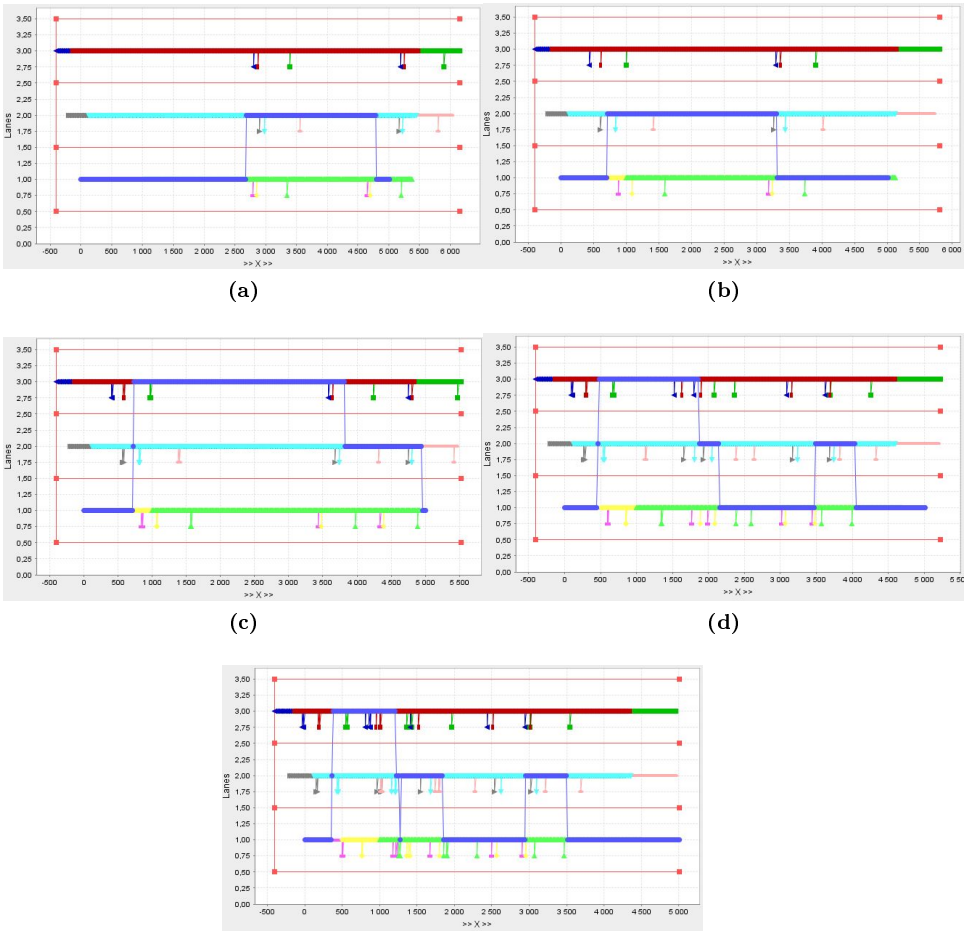
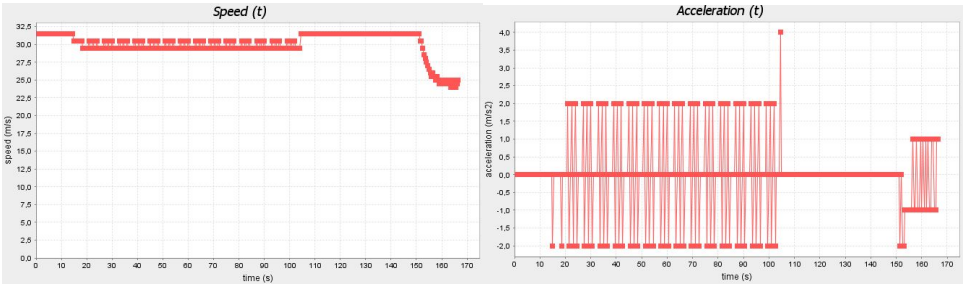
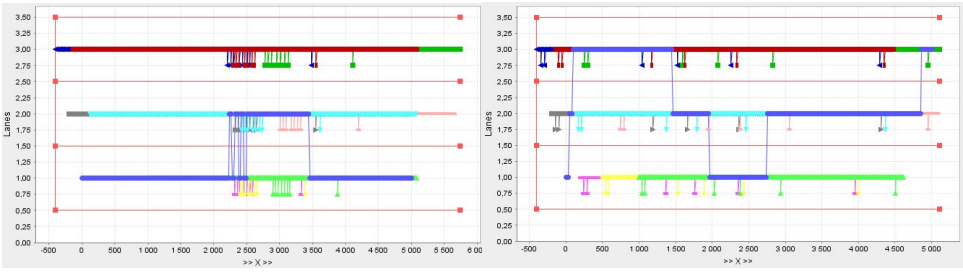


Figure 4.29: Optimal trajectories for cases (−1) (a), (−0.5) (b), (0) (c), (0.5) (d) and (1) (e), Complex scenario



(a) (b)

Figure 4.30: Speed and Acceleration for BDI Agent, Complex scenario



(a) (b)

Figure 4.31: Trajectories for Greedy Agent (a) and BDI Agent (b) with $\mu = 1$, Complex scenario

Results	
time	147.5 s
#lane changes	6
$sd_{\%}$	99.0 %
TTC_{min}	17.75 s
δT_{min}	2.91 s
h_{min}	91.63 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	47.63 m

Table 4.25: Results for the BDI ($\mu = 1$) for the Complex Scenario

One should bear in mind that this scenario has been designed to underline the difference between both agents. If there is not a situation where the agent should foresee that it should overtake, as it is the case at the beginning of this scenario, agents would have similar performances as the *BDI Agent* is quite limited as well.

4.3.3 Macroscopic simulation

The complex scenario above has been simplified for visualization purposes in order to show the macroscopic behaviour of the agents. All cars are equipped with the same agent on these simulations, in order to verify that the overall behaviour is correct. Graphs are shown on Figures 4.32 and 4.33. Optimal solutions for a car are not computed as the aim is rather to check if the overall behaviour seems correct. Tables 4.26 and 4.27 are computed for the blue car.

Analysis and Comparisons The overall behaviour of each agent is correct and trajectories are safe. The macroscopic behaviour shows similar results than the previous tests. On this scenario one should notice though that at the beginning, all cars get back on lower lanes as they are programmed to do so if possible; which is not in accordance with their desire for speed. This shows that their behaviour lacks of rationalism.

Results	
time	83.5 s
#lane changes	1
$sd_{\%}$	100.0 %
TTC_{min}	15.27 s
δT_{min}	2.91 s
h_{min}	85.63 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	41.38 m

Table 4.26: Results for the blue Greedy Agent, Macro scenario

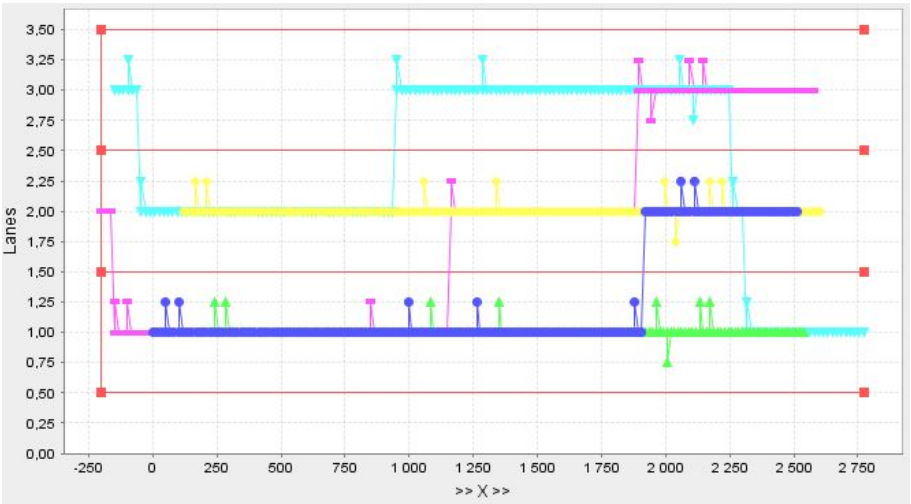


Figure 4.32: Trajectory for the Macro scenario, Greedy Agents

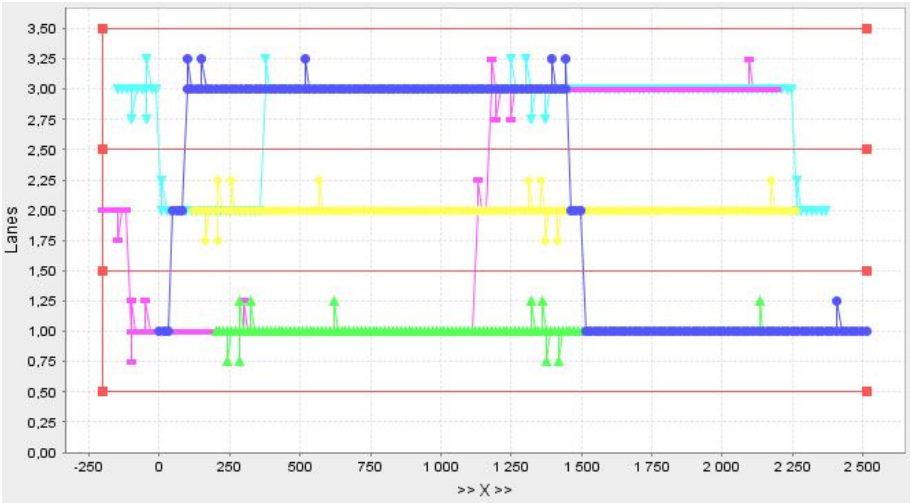


Figure 4.33: Trajectory for the Macro scenario, BDI Agents

Results	
time	72.0 s
#lane changes	4
$sd_{\%}$	100.0 %
TTC_{min}	26.47 s
δT_{min}	3.28 s
h_{min}	112.5 m
EES_{max}	NaN m/s
$p(MAIS_{>2})_{max}$	NaN
d_{min}	58.13 m

Table 4.27: Results for the blue BDI Agent, Macro scenario

Chapter 5

Conclusion

To conclude, this model proves the feasibility of a simulation framework that supports traffic, human and system rules. Its flexibility enables to launch any scenario and retrieves relevant information about the trajectory, considering time and safety indicators so far. Previous works have mainly focused either on the strategic level, such as the global path planner of Junior in the Urban Darpa Challenge, or on a lower tactical level where decisions are taken for shorter terms, such as the collision avoidance program of Stanley in the Desert Darpa Challenge - whereas the model implemented in this thesis targets a higher tactical level and aims at taking medium-term decisions.

In this model agents take safe decisions, pass the basic tests, and contingency plans (maximum deceleration and keep lane in case of there is no solution) do not occur often as shown on speed graphs. However, unrealistic behaviours occur. They lack of rationality as their implemented intelligence is very restricted; especially for the *Greedy Agent* that thinks only one step ahead. Nevertheless their general behaviour is interesting as for basic tests they provide suboptimal solutions that are close to the ones suggested by the optimal search. Besides, macroscopic simulations do not trigger issues, which would enable such a model to be used by several cars that share road structures. This safe behaviour results from the safe distances programmed as constraints in the decision process. On one hand it is desirable that the trajectory is safe but on the other hand agents

do not consider slightly more dangerous choices that could result in a high gain of time - while human drivers would consider such solutions. This flaw could be fixed by the use of a performance measure and stochastic predictions: instead of predicting that the other cars will be at a given position, the agent could consider a probability distribution and take this probability into account in the evaluation of the choice. It is more complex and it would require further studies to elaborate an appropriate performance measure.

Considering the optimal search, it provides a range of different possible trajectories: it does not provide the optimal trajectory for a given driver since no accurate performance measure has been elaborated, but rather suggests what trajectories were potentially good and acceptable. The atomic reduction of a state in the search triggers some unrealistic behaviours that seem inherent to such problems: it could be improved by using a performance measure that suits better to this problem. Also the performance measure should incorporate not only safety and time but also other criteria such as comfort. Also the optimal search could not be tested in a real situation as it needs time backtracking: even provided a performance measure, it seems hard to find an optimal solution in a real situation.

The flexibility of the model enables it to be improved - especially by augmenting the intelligence of the agents. They should be implemented in order to have a higher view of the situations. If the agent is not able to learn, it will repeat again and again the same mistakes and will not be able to succeed in new environments or situations that were not planned. The implementation of a learning structure seems crucial for such a system. But it would require an appropriate performance measure for decisions also, which would require further studies as a decision can have long term consequences. Also, results have been retrieved considering that the other cars follow an Intelligent Driver Model: they are not able to change lane. Further studies should be led to test the model on more realistic models and eventually in a real situation.

Finally, it is a first simple model with limitations but it has shown desirable behaviours for simple tests. Besides the implemented solutions are all computationally possible. This model provides a first simulation framework and algorithms that are able to compute safe trajectories and respect rules. But the decisions taken could be more rational and especially more realistic- avoiding unnecessary lane changes for example. The most important improvement would be to elaborate a relevant performance measure for both decisions and for trajectory evaluation: it would lead to great improvements of the model.

Appendix A

LCM Comparisons

This appendix presents the study unterdaken to select a realistic Longitudinal Control Model. The models which have been tested are the ones of the more simple form as they have mathematical elegance: the equations are explicit about how they try to capture the underlying mechanism of the system. More complex ones such as the rule-based model of Kosonen [Kos99][Ni13b] or the agent-based model of Panwai and Dia [PD05] are more complex and thus their internal mechanism is harder to explicate. Morever they have not been more accredited than the simpler ones. They are not considered in this study. The considered models are the ones from Newell (1961) [New61], Gipps (1981) [Gip81], the Intelligent Driver Model from Treiber (2000) [THH00] and the Field Theory model from Ni (2013) [Ni13a].

A.1 Validation tests

The evaluation of the models is based upon three criteria presented below. These criteria have been suggested by Ni [Ni13c]. It is desirable that a model give realistic results on these basic criteria to be assessed physically sound. For each criterion of each models, graphs are provided to describe the behavior of the car and for each model a table sums up the results on these scenarios. Although

the reaction time tau of the driver may be different from the sampling time dt in real life, Gipps precises that his model works better when they are equal. Therefore tests have been undertaken with $tau = dt = 0.6$ second and so for all the tests.

A.1.1 Acceleration performance

The car starts from standstill with not any obstacle in front and the driver targets a speed of 40 $m.s^{-1}$. The website zeroto60times.com [Zer13] gives empirical data related to this test. Time to go from standstill to 26.8 $m.s^{-1}$ (60 mph) are comprised between 7 and 12 seconds. Time to travel a quarter mile is a second indicator and is comprised between 15 and 19 seconds for these cars.

A.1.2 Deceleration performance

The car goes initially at 32 $m.s^{-1}$, approximately 112 $km.h^{-1}$ and a fixed obstacle is placed 1 km ahead. The vehicle must stop before it. Stopping distance references are given by the government of the UK [oUK13]: for this speed the typical stopping distance is 96 meters. The car is considered to be braking when the speed goes down below 90% of the desired speed. Since the obstacle is far, braking should be smooth and start not too late, not too early neither to be realistic. A distance comprised between 120 (+25% of the stopping distance if braking hard) and 200 meters (+100%) will be considered as realistic. Some of the models show a phenomenon of re-acceleration after a first braking and generally oscillations follow, it is indicated in the summary table. Also in case the car crashes, the speed of the car at the moment of crash is given.

A.1.3 Car-following performance

There is a car in front and the data have been retrieved on the simulator of CARRSQ. The car in front goes in average at approximately 25 $m.s^{-1}$ while the car is willing to go in average at a certain speed v. The simulation has been run for a normal distribution of 10 drivers willing to go at a speed v in average 35 $m.s^{-1}$ with a standard deviation of 5 $m.s^{-1}$. The Time To Collision (TTC) is a meaningful indicator to know if a car is too close: a usual danger threshold is 5 seconds, as taken by Vogel [Vog02]. In the summary table the number of collisions out of the 10 simulations is given.

A.2 Tests on the models

All the models do not manage to stop until standstill, thus an external logic has been implemented to reach a null speed. Also, all the models show a behaviour of an hard acceleration from the beginning. To prevent the car from crashing the other one from the very beginning in the car-following simulation, I have setp up a distance of 100 or 200 meters. The TTC gets low at the very beginning but this problem comes from an unrealistic behavior in that case: other scenarios should be tested to explain this phenomonon, I will not focus on that in this report and rather check that once the car drives, it follows well the one in front by respecting safety distances and not being too far neither. Whenever the maximum acceleration g is a parameter, I use $g = 4m.s^{-2}$. Whenever a maximum deceleration occurs, I use $b = 6m.s^{-2}$.

A.2.1 Newell's model

This model corresponds to the one described in section 2.1.2.1. The parameter λ is set to $0.79s^{-1}$ as Newell empirically advocates [New61]. The initial distance for the car-following has been set to 200 m.

Since the speed is given, the acceleration is unrealistic and the equation does not control the dynamic part which makes the driver goes continuously from a speed v_1 to a speed v_2. It just gives a targeted speed, so the desired speed is directly computed as shown on Figure A.1. For the deceleration, the car brakes very late and the car crashes (Figure A.2) : it is unacceptable. However the car-following is excellent and safety distances are respected (Figure A.3). The deceleration was so bad that other tests have been made changing the parameter λ. It is interesting to notice that making λ vary can fix the problem with the deceleration (Figure A.4 but other problems are triggered: the car-following is not working well as the headway distance is then more than 200 meters in these conditions (Figure A.5 ! Results are sum up in Table A.1. Newell's model gives a good car-following model but fails the acceleration and deceleration tests.

Acceleration	
0-60 mph time	0.6 s
1/4 mile time	10.8 s
Deceleration	
Collision	yes
braking dist	45.0 m
braking time	2.4 s
speed when crash	14.4 m/s
re-acceleration	no
Car-following	
#collisions	0

Table A.1: Results summary for Newell's model

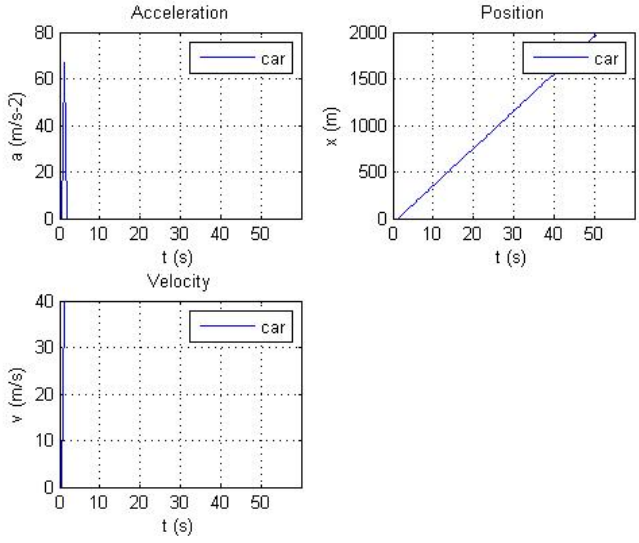


Figure A.1: Acceleration performance for Newell's model

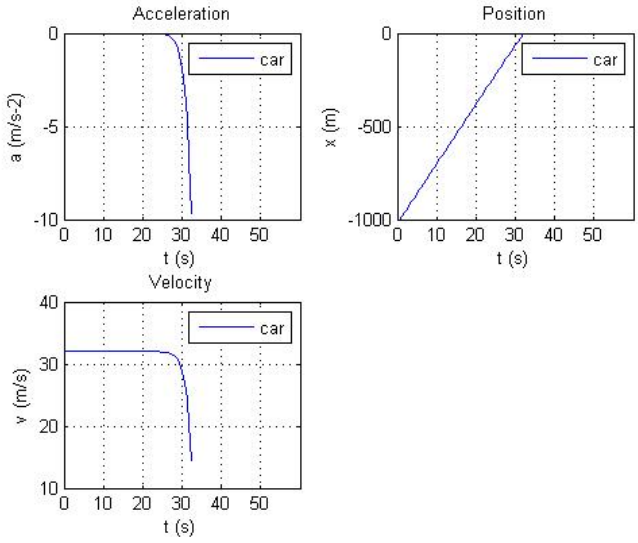


Figure A.2: Deceleration performance for Newell's model

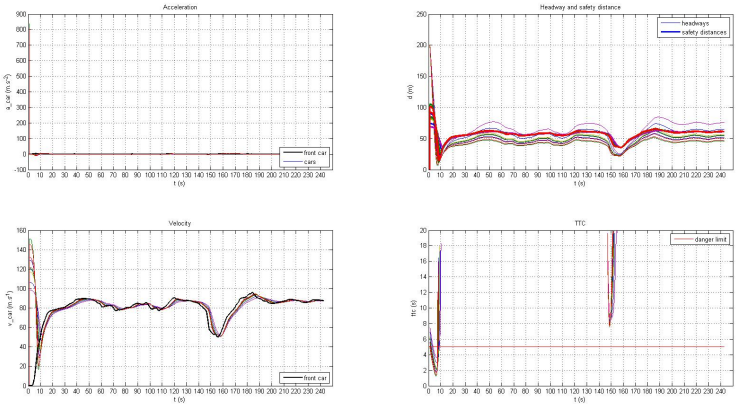


Figure A.3: Car-following performance for Newell's model

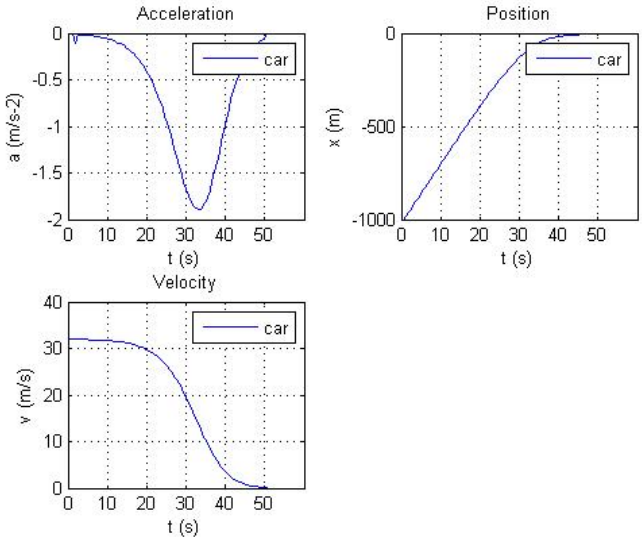


Figure A.4: Deceleration performance for Newell's model ($\lambda = 0.2$)

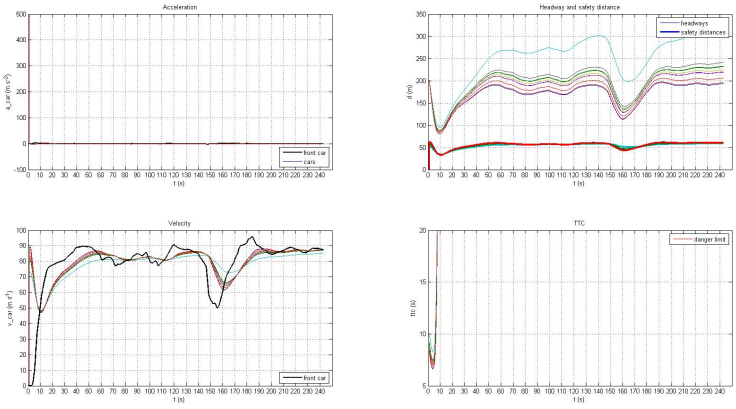


Figure A.5: Car-following performance for Newell's model ($\lambda = 0.2$)

Acceleration	
0-60 mph time	8.4 s
1/4 mile time	17.4 s
Deceleration	
Collision	yes
braking dist	59.2 m
braking time	3.6 s
speed when crash	3.7 m/s
re-acceleration	no
Car-following	
#collisions	0

Table A.2: Results summary for Gipps' model

A.2.2 Gipps' model

This model corresponds to the one described in section 2.1.2.2.

For the acceleration, the car achieves the targeted speed smoothly as shown on Figure A.6 and the time spent on the two tests fit empirical data. The speed given initially is not the one targeted and instead the acceleration is smooth at the beginning: the problem faced by Newell's is overcome. However, the deceleration is late and hard and the car almost manages to stop as the speed is only 3.7 m/s before the crash, as shown on Figure A.7. But this is due to the discretization; test have been made with lower time steps and the car is closer to stop. But at the end of the braking the deceleration is softer, thus with an external logic the car would be able to stop, though the model makes not the car brakes softly and realistically. There is no crash in the car-following but the car keeps a very low distance with the front car, approximately 10 meters as shown on (Figure A.8). This is dangerous and not realistic, at least for this scenario of high speed. Results are sum up in Table A.2. Gipps' model gives a good acceleration model, almost supports the deceleration until standstill but at high speed fails at following a car respecting safety distances.

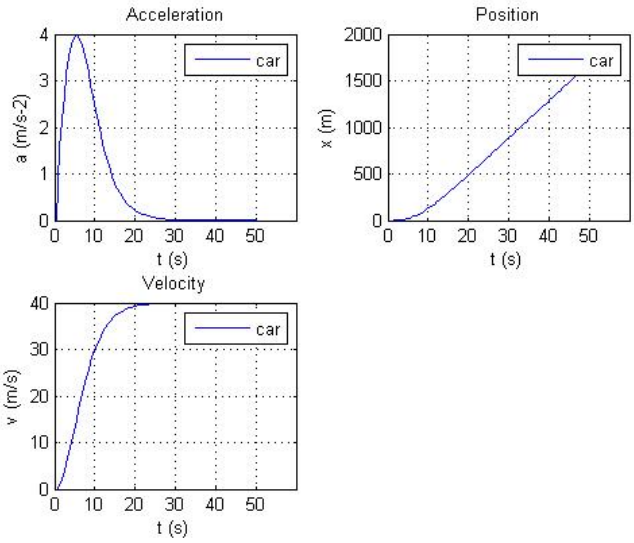


Figure A.6: Acceleration performance for Gipps' model

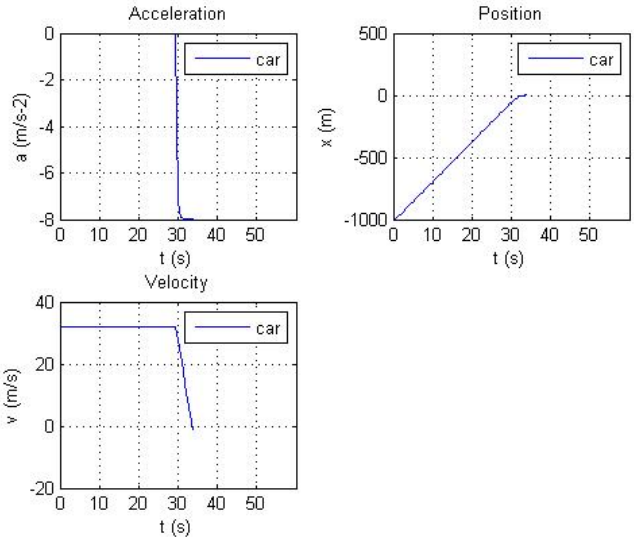


Figure A.7: Deceleration performance for Gipps' model

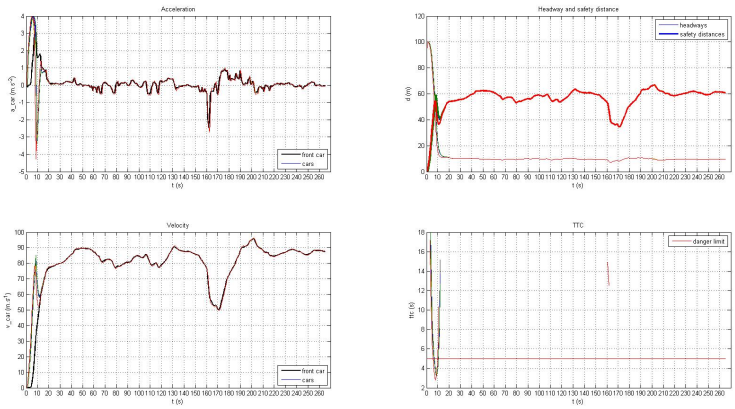


Figure A.8: Car-following performance for Gipps' model (tau=dt)

Acceleration	
0-60 mph time	7.8 s
1/4 mile time	16.8 s
Deceleration	
Collision	no
braking dist	180.1 m
braking time	9.0 s
speed when crash	NaN
re-acceleration	no
Car-following	
#collisions	0

Table A.3: Results summary for the IDM model

A.2.3 Intelligent Driver Model

This model corresponds to the one described in section 2.1.2.3. I use $g = 4$ $m.s^{-2}$ and $\gamma = 4$ as suggested by Treiber [THH00].

The acceleration is very hard at the beginning and then decreases smoothly. The very beginning of the simulation is unrealistic, as shown on Figure A.9. Treiber discusses about these effects and explain that a certain value of γ suits a certain scenario. A given γ will not make everything work perfectly. Nevertheless, the overall speed profile is realistic and the time references match empirical data for the two tests as shown in Table A.3. The braking is slow and smooth and the car manages to stop just before the car in front: it is very realistic, as shown on Figure A.10. The car-following is also excellent and safety distances are respected, as shown on Figure A.11. Results are sum up in Table A.3. Finally the IDM's model gives excellent results and the only flaw for these tests is the unrealistic acceleration from standstill at the very beginning.

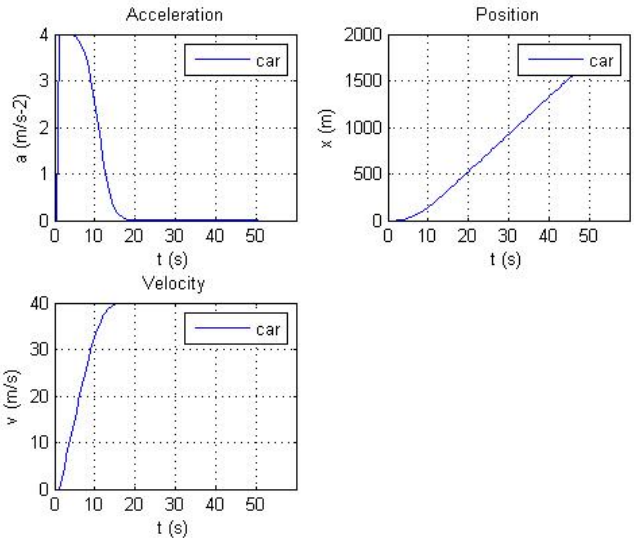


Figure A.9: Acceleration performance for the IDM model

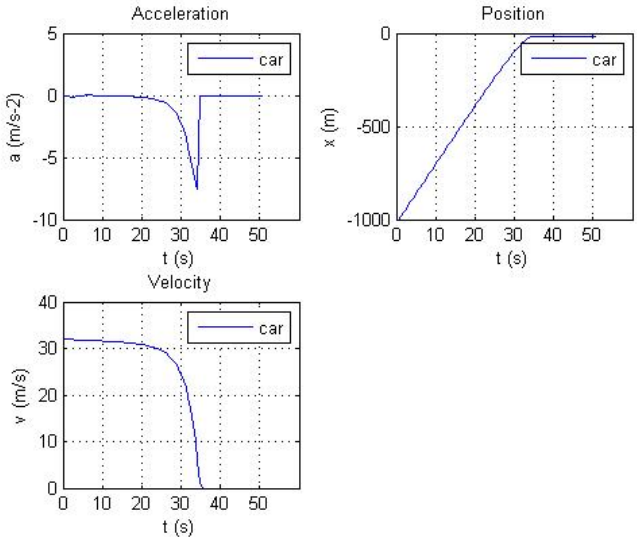


Figure A.10: Deceleration performance for the IDM model

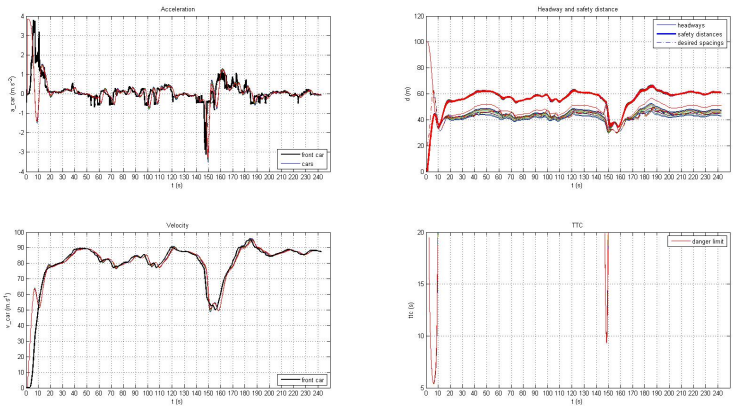


Figure A.11: Car-following performance for the IDM model

Acceleration	
0-60 mph time	10.8 s
1/4 mile time	19.2 s
Deceleration	
Collision	yes
braking dist	267.7 m
braking time	12.6 s
speed when crash	9.2 m/s
re-acceleration	no
Car-following	
#collisions	0

Table A.4: Results summary for the Field Theory model

A.2.4 Field Theory

This model corresponds to the one described in section 2.1.2.4.

Like the other models, the acceleration is very hard at the beginning, but then it decreases softly and the car finally reaches the targeted speed in realistic time as shown on Figure A.12. The braking is early enough and smooth but remains too soft even when the car gets closer hence a crash occurs, as shown on Figure A.13: it is unacceptable. However the car-following part is very realistic and safety distances are respected as Figure A.14 shows. Results are sum up in Table A.4. The Field Theory's model gives excellent results for the car-following part but does not manage to slow down enough to avoid a crash when a fixed obstacle is in front. Also the initial acceleration from standstill is unrealistic.

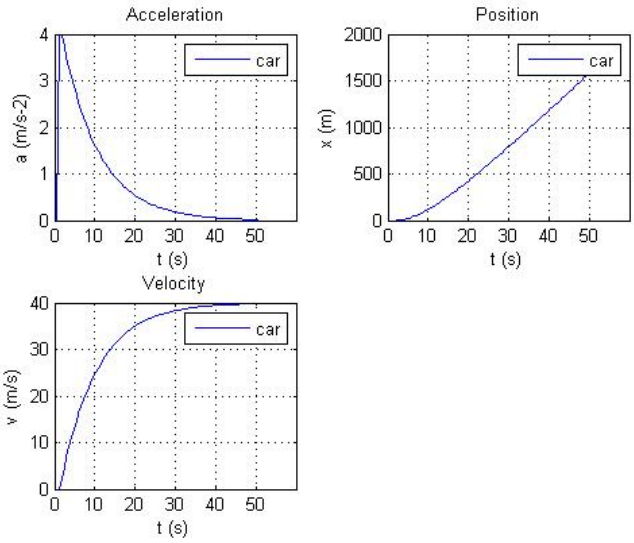


Figure A.12: Acceleration performance for the Field Theory model

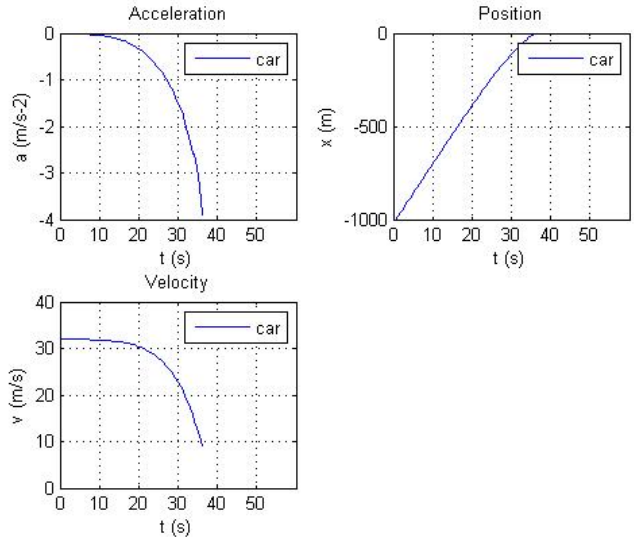


Figure A.13: Deceleration performance for the Field Theory model

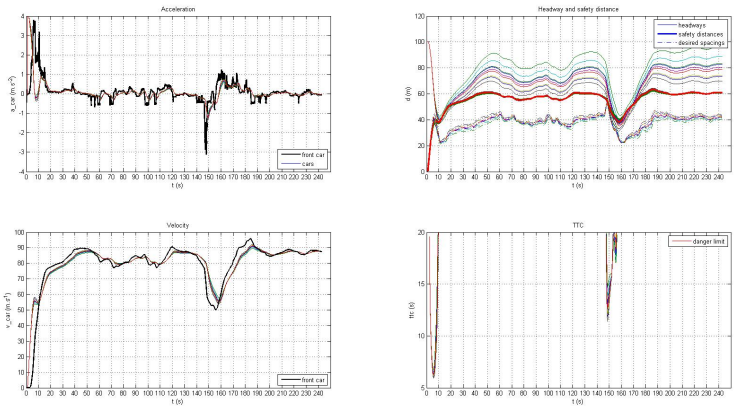


Figure A.14: Car-following performance for the Field Theory model

A.3 Conclusion

The acceleration from standstill is unrealistic for all the models considered in this study. This issue should be fixed to get a more realistic model. External logic should be used to prevent this hard acceleration and makes it softer. All the models give very good results for the car-following test as the cars do not crash and keep reasonable distances. However, only the Intelligent Driver Model passes the deceleration test. As a result the IDM is a very good trade-off between classical models, like Gipps' or Newells's ones, and force-driven models, like Ni's one, and perform well on all the tests. It is clearly the best model for car-following simulation at high speed for these scenarios.

Appendix B

Calculation and Implementation details

B.1 Calculation details

B.1.1 Physically sound safe distance

Gipps' safe distance is given by:

$$d_{i,ft}{}^{*} = v_i(t) \cdot \tau_i + L_{ft} + max(0, (\frac{{v_i}^2(t)}{2b_i} - \frac{{v_j}^2(t)}{2b_{ft}}))$$

It is physically sound because the driver has time to react to an emergency braking. Let us suppose that the car directly in front performs an emergency braking at a constant maximum deceleration rate b_j. The ego car will travel $v_i(t) \cdot \tau_i$ m during the reaction time of the ego driver. Then the ego driver will perform an emergency braking at his car's maximum deceleration rate b_i. Let us calculate the distance travelled by a car j during a constant deceleration at rate b_j. Let us take the moment the car starts braking as the origin of time t_0 = 0. During the braking, the acceleration is given by:

$$a(t) = -b$$

Let us integrate this equation from t_0 to the final time, when the car arrives at standstill t_f. It gives

$$v(t_f) = v(t_0) - b \cdot t_f = 0$$

Which gives the final time $t_f = \frac{v(t_i)}{b}$. Two integrations of the first equations give the distance travelled $x_j(t_f) - x_j(t_0) = v_j(t_0) \cdot t_f - \frac{b \cdot v(t_0)}{2 \cdot b} = \frac{v(t_0)}{2 \cdot b}$. Therefore, the difference of distances travelled by the ego car i and the car directly in front ft is

$$\frac{{v_i}^2(t)}{2b_i} - \frac{{v_j}^2(t)}{2b_{ft}}$$

This term must be added only if superior to 0, which corresponds to a lower distance of braking for the car in front. Also since the point of the car considered is the point most in front of the car, the term L_{ft} must be added to the distance. Finally, this distance is physically sound and given by:

$$d_{i,ft}{}^* = v_i(t) \cdot \tau_i + L_{ft} + max(0, (\frac{{v_i}^2(t)}{2b_i} - \frac{{v_j}^2(t)}{2b_{ft}}))$$

B.1.2 Discretization error

If the sampling time is dt and the discretization step for considered targeted speed is $dv = 1\ m.s^{-1}$, let us calculate the maximum difference of distance travelled during dt between the choice for a targeted speed at $t+dt$ for the maximum speed at v_{opt} and the actual targeted speed v_{tar}. Since the discretization step is dv, the difference between v_{opt} and v_{tar} is upper bounded by $\delta v = \frac{dv}{2}$. Speed at t is $v(t)$. The acceleration to reach v_{opt} at $t + dt$ is:

$$a_{opt} = \frac{v_{opt} - v(t)}{dt}$$

The acceleration to reach v at $t + dt$ is

$$a_{tar} = \frac{v_{tar} - v(t)}{dt}$$

The distance travelled during dt at the acceleration rate a is given by $d = v(t) + \frac{a \cdot dt^2}{2}$. So the difference of distances travelled between an acceleration rate a_{opt} and a_{tar} is:

$$\delta D = |\frac{(a_{opt} - a_{tar}) \cdot dt^2}{2}| = |\frac{(v_{opt} - v_{tar}) \cdot dt}{2}| \leq \frac{\delta v \cdot dt}{2}$$

Therefore, with $dt = 1\ s$ and $dv = 1\ m.s^{-1}$, the maximum error is $0.25\ m$ which is negligible to a typical gap distance of 20 to 50 m.

B.2 Implementation details

B.2.1 Wait for percepts when changing lane

When the agent changes lane, it has not any percept associated to the new lane. It needs to wait τ second before getting a percept. This is implemented by adding the information *delays* at each node state for the optimal search, and keeping this information in the main program of the online algorithm. *delays* is a map that associates a time delay to each agent. At the initial node, or a lane change for agent j, *delays(j)* is set to τ_j. Then at every time step, *delays(j)* is decremented. From a lane change to *delays(j)* reaches 0, the agent keeps the same lane and speed. Then it has percepts and is able to choose an appropriate action. *delays* does not impact on the ego agent behaviour in the optimal search as the best solution should not depend on τ. It is used only for the other agents to simulate their behaviour. For the online algorithm however it is used for every agent.

Bibliography

[Abr03] Jean-Claude Abric. *Pratiques sociales et représentations. Psychologie sociale.* Paris, Presses Universitaires de France, 2003.

[Ass13] Australian Automobile Association. Road safety. *Website*, 2013. "`http://www.aaa.asn.au/issues/road_safety.htm`".

[BA99] Alexander A Borb and Peter Achermann. Sleep homeostasis and models of sleep regulation. *Journal of Biological Rhythms*, 14(6):559–570, 1999.

[BM99] Mark Brackstone and Mike McDonald. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4):181–196, 1999.

[CF97] Bo Cheng and Takehiko Fujioka. A hierarchical driver model. In *Intelligent Transportation System, 1997. ITSC'97., IEEE Conference on*, pages 960–965. IEEE, 1997.

[DBB84] Serge Daan, DG Beersma, and Alexander A Borbély. Timing of human sleep: recovery process gated by a circadian pacemaker. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 246(2):R161–R183, 1984.

[Ful05] Ray Fuller. Towards a general theory of driver behaviour. *Accident Analysis & Prevention*, 37(3):461–472, 2005.

[GHP59] Denos C Gazis, Robert Herman, and Renfrey B Potts. Car-following theory of steady-state traffic flow. *Operations Research*, 7(4):499–505, 1959.

[Gip81] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15:105–111, 1981.

[Gip86] Peter G Gipps. A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414, 1986.

[GLR+11] Daniel Greene, Juan Liu, Jim Reich, Yukio Hirokawa, Akio Shinagawa, Hayuru Ito, and Tatsuo Mikami. An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):942–953, 2011.

[Gov10] Queensland Government. Safe following distances. *Website*, 2010. "`http://www.tmr.qld.gov.au/Safety/Queensland-road-rules/Road-rules-refresher/Safe-following-distances.aspx`".

[GRGN07] Sebastien Glaser, Andry Rakotonirainy, Dominique Gruyer, and Lydie Nouveliere. An integrated driver-vehicle-environment (i-dve) model to assess crash risks. *Australasian Road Safety Research, Policing and Education Conference*, 2007.

[GVGM11] Sebastien Glaser, Benoit Vanholme, Dominique Gruyer, and Said Mammar. Probability and risk based maneuver planning for collision avoidance. *First International Symposium on Future Active Safety Technology toward zero-traffic-accident*, 2011.

[KBL70] DL Kleinman, S Baron, and WH Levison. An optimal control model of human response part i: Theory and validation. *Automatica*, 6(3):357–369, 1970.

[Kos99] Iisakii Kosonen. Urban traffic simulation and control model: Principles and applications. *PhD Thesis*, 1999.

[LS11] Sylvain Lassarre and Farida Saad. An integrated and multidisciplinary approach for studying use and acceptance of new driver support system: The french national project on intelligent speed adaptation (lavia project). In *Reston, VA: ASCEProceedings of the First International Conference on Transportation Information and Safety, June 30. July 2, 2011, Wuhan, China/ d 20110000.* American Society of Civil Engineers, 2011.

[MBB+09] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry

in the urban challenge. In *The DARPA Urban Challenge*, pages 91–123. Springer, 2009.

[MH84] PJ Mills and CA Hobbs. The probability of injury to car occupants in frontal and side impacts. *SAE Technical Paper 841652*, 1984.

[Mic86] John A Michon. *A critical view of driver behavior models: what do we know, what should we do?* Springer, 1986.

[MKMM03] Douglas L. Milliken, Edward M. Kasprak, L. Daniel Metz, and William F. Milliken. *Race Car Vehicle Dynamics: Problems, Answers and Experiments.* SAE International, 2003.

[New61] G. F. Newell. Nonlinear effects in the dynamics of car following. *Operations Research*, 9:209–229, 1961.

[Ni13a] Daiheng Ni. A unified perspective on traffic flow theory, part i: The field theory. *Applied Mathematical Sciences*, 2013.

[Ni13b] Daiheng Ni. A unified perspective on traffic flow theory, part ii: The unified diagram. *Applied Mathematical Sciences*, 2013.

[Ni13c] Daiheng Ni. A unified perspective on traffic flow theory, part iii: Validation and benchmarking. *Applied Mathematical Sciences*, 2013.

[Org10] World Health Organization. Mortality: Road traffic deaths by country. *Website*, 2010. "`http://apps.who.int/gho/data/node.main.A997?lang=en`".

[oUK13] Government of United Kingdom. Typical stopping distances. *Website*, 2013. "`http://www.direct.gov.uk/prod_consum_dg/groups/dg_digitalassets/@dg/@en/@motor/documents/digitalasset/dg_188029.pdf`".

[PD05] Sakda Panwai and Hussein Dia. A reactive agent-based neural network car following model. *IEEE Transactions on intelligent transportation systems*, 6, 2005.

[Pea84] J. Pearl. *Heuristics: Intelligent search strategies for computer problem solving.* Addison-Wesley, Jan 1984.

[PT90] Huei Peng and Masayoshi Tomizuka. Lateral control of front-wheel-steering rubber-tire vehicles. *Research Reports, California Partners for Advanced Transit and Highways (PATH)*, 1990.

[Raj11] Rajesh Rajamani. Lateral vehicle dynamics. In *Vehicle Dynamics And Control*, pages 15–49. Springer, 2011.

[Ran94] Thomas A Ranney. Models of driving behavior: a review of their evolution. *Accident Analysis & Prevention*, 26(6):733–750, 1994.

[RN09] Stuart Jonathan Russell and Peter Norvig. *Artificial intelligence: a modern approach (3rd Edition)*. Prentice hall Englewood Cliffs, 2009.

[Sal06] Dario D Salvucci. Modeling driver behavior in a cognitive architecture. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 48(2):362–380, 2006.

[SCA11] Adnan Shaout, Dominic Colella, and S Awad. Advanced driver assistance systems-past, present and future. In *Computer Engineering Conference (ICENCO), 2011 Seventh International*, pages 72–82. IEEE, 2011.

[SFM85] Ola Svenson, Baruch Fischhoff, and Donald MacGregor. Perceived driving safety and seatbelt usage. *Accident Analysis & Prevention*, 17(2):119–133, 1985.

[Shi78] D. Shinar. The human factor in traffic safety. *Psychology on the Road*, 1978.

[SSW10] Robin Schubert, Karsten Schulze, and Gerd Wanielik. Situation assessment for automatic lane-change maneuvers. *Intelligent Transportation Systems, IEEE Transactions on*, 11(3):607–616, 2010.

[Sum88] Heikki Summala. Risk control is not risk adjustment: The zero-risk theory of driver behaviour and its implications. *Ergonomics*, 31(4):491–506, 1988.

[THH00] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62:1805–1824, 2000.

[TMD+06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[Van12] Benoit Vanholme. Highly automated driving on highways based on legal safety. *PhD Dissertation*, 2012.

[Vog02] Katja Vogel. A comparison of headway and time to collision as safety indicators. *Accident Analysis and Prevention*, 35:427–433, 2002.

[Wil01] G J. S. Wilde. A new psychology of safety and health. In *Target Risk 2*. Pde Pubns, 2001.

[Won01] Jo Yung Wong. *Transient Response Characteristics*. Wiley. com, 2001.

[Zer13] Zeroto60Times.com. Ford 0-60 mph times. *Website*, 2013. "`http://www.zeroto60times.com/Ford-0-60-mph-Times.html`",.

[ZSS85] F Zeidler, H-H Schreier, and R Stadelmann. Accident research and accident reconstruction by the ees-accident reconstruction method. *SAE transactions*, 94:2–399, 1985.