Université Sidi Mohammed Ben Abdellah
Faculté des Sciences Dhar El Mahraz- Fès
Centre d'Etudes Doctorales
"Sciences et Technologies"

Formation Doctorale : Sciences et Technologies de l'Information et de la Communication (STIC)

Discipline : Informatique

Spécialité : Informatique

Laboratoire : Laboratoire d'Informatique, Modélisation et Systèmes (LIMS)

THESE DE DOCTORAT

Présentée par

Afaf Bouhoute

DRIVER BEHAVIOR MODELING : APPLICATIONS FOR DRIVING SAFETY ASSESSMENT AND DRIVERS FINGERPRINTING

Soutenue le 27/10/2018 devant le jury composé de :

Pr. Najiba El Amrani El Idrissi	***Faculté des Sciences et Techniques – Fès***	***Président***
Pr. Akka Zemmari	***Université de Bordeaux – France***	***Rapporteur***
Pr. Hicham Ghennioui	***Faculté des Sciences et Techniques – Fès***	***Rapporteur***
Pr. Mohammed Al Achhab	***Ecole Nationale des Sciences Appliquées – Tétouan***	***Rapporteur***
Pr. Noureddine Rais	***Faculté des Sciences Dhar El Mahraz – Fès***	**Examinateur**
Pr. Saad Bennani	***Ecole Nationale des Sciences Appliquées – Fès***	**Examinateur**
Pr. Lahcen Omari	***Faculté des Sciences Dhar El Mahraz – Fès***	**Directeur de thèse**
Pr. Ismail Berrada	***Faculté des Sciences Dhar El Mahraz – Fès***	**Co-directeur de thèse**

Année universitaire : 2017-2018

Don't ever give up.
Don't ever give in.
Don't ever stop trying.
Don't ever sell out.
And if you find yourself
succumbing to one of the above
for a brief moment, pick
yourself up, brush yourself off,
whisper a prayer, and start
where you left off.
But never, ever, ever give up.

RICHELLE E. GOODRICH

SIDI MOHAMMED BEN ABDELLAH UNIVERSITY

Abstract

Faculty of Science, Fez
Department of computer science

Driver behavior modeling: applications for driving safety assessment and driver fingerprinting

by Afaf Bouhoute

The recent computerization of cars, together with the development of sensor technologies and car communication devices have revolutionized the way researchers and analysts deal with driving behavior. Driving behavior analytics have thus emerged as an important means of improving driving safety and drivers comfort. Depending on the analysis's goals, different mathematical and statistical models have been used and numerous analytics approaches have emerged consequently. Generally, these analytics solutions process data generated by vehicles, solely or combined with road data, and transform it into valuable information to gain a better understanding of drivers behavior. In this PhD thesis, we investigate the application of some well-known approaches in driver behavior modeling and analysis. We aim to provide a holistic framework for driver behavior analysis based on automotive sensors data (such as speed, acceleration, steering angle, etc). Our contributions relate to different steps involved in the data analysis process, mainly those of preprocessing, modeling and analysis. As a very first step, to prepare the driving data for analysis, abstraction using the interval domain is proposed. This abstraction transforms the measurements into a form of intervals, ignoring thus irrelevant details from instantaneous measurements. Then, we propose two graphical models to represent driving behavior, which are probabilistic hybrid automata (PRHIOA) and attributed directed graphs (ADG). In fact, probabilistic graphical models provide a helpful framework for modeling driving behavior: the language of graphs facilitates the representation of the relationships within the driver, vehicle, environment system, while the probability allows the representation of uncertainty. The models are combined with a machine-learning algorithm, based on the learning automata algorithm, to build personalized models of the drivers behavior. Besides their representative power, the models proposed have allowed us to perform profound analyses using formal verification and graph matching techniques. Finally, based on these models, two analyses are proposed: the first analysis uses model checking of the automaton model of the driver to verify the compliance of his driving with the road rules; while the second uses graph matching theory to compute the similarity between the behavior of different drivers. Obtained results reveal the potential of the two approaches in improving driver behavior logs analysis.

Université Sidi Mohammed Ben Abdellah

Résumé

Faculté des Sciences Dhar El Mahraz- Fès
Département Informatique

Modélisation du comportement des conducteurs automobiles: Applications pour évaluation et identification des conducteurs

par Afaf Bouhoute

Les nouvelles technologies de l'information et de la communication et leur contribution à l'informatisation de la voiture ont révolutionné la façon d'étudier le comportement des conducteurs automobiles. L'analyse des données recueillies par les voitures permet d'étudier le comportement des conducteurs dans des situations réelles, d'analyser leurs habitudes ainsi que la performance de leur conduite. Elle constitue ainsi un moyen important pour améliorer la sécurité et le confort des conducteurs. Afin d'analyser le comportement des conducteurs, de nombreuses approches analytiques et méthodologies ont été proposées qui diffèrent en termes des objectifs visés ainsi que des outils utilisés. Dans ce travail de thèse, nous étudions l'application des méthodes formelles reconnues en informatique fondamentale à la modélisation et l'analyse du comportement du conducteur. Notre objectif est de fournir un cadre solide et global pour l'analyse du comportement du conducteur, en se basant sur des données naturalistes de conduite. Les contributions proposées portent sur les différentes étapes du processus d'analyse des données, principalement celles du pré-traitement, de la modélisation et de l'analyse. Dans notre première contribution, nous proposons l'abstraction numérique, sur le domaine des intervalles, comme méthode de réduction des données de conduite. Le but de cette abstraction est de transformer les valeurs collectées en intervalles, ce qui permet d'ignorer les valeurs non pertinentes à l'analyse. La deuxième contribution vise à la création des profils personnalisés des conducteurs. Elle consiste à utiliser (1) les modèles graphiques, notamment les automates probabilistes et les graphes étiquetés comme outils de modélisation du comportement du conducteur, et (2) un algorithme d'apprentissage par renforcement pour la construction des modèles. Le choix des modèles graphiques est justifié d'une part par la puissance représentative des modèles graphiques, et d'autre part, par les analyses approfondies pouvant être effectuées à base de ces modèles. Nous proposons ainsi , dans nos dernières contributions, deux analyses: la première consiste à utiliser la vérification de modèles (model checking) pour une vérification formelle de la conformité du comportement du conducteur aux exigences du code de la route; alors que dans la deuxième nous utilisons l'appariement des graphes (graph matching) pour étudier la similarité des comportements des conducteurs. L'applicabilité des deux analyses est démontrée en utilisant des data sets de conduite publiques. Les résultats obtenus révèlent l'intérêt des deux approches pour l'amélioration des analyses des traces du comportement des conducteurs.

Acknowledgements

"Gratitude is a miracle of its own recognition. It brings out a sense of appreciation and sincerity of a being."

– Auliq-Ice

To all those people who in one way or another contributed in the completion of this thesis: Thank you for being part of my journey!!

Foremost, I would like to express my sincere gratitude to my two supervisors, Pr. *Lahcen Omari* and Pr. *Ismail Berrada.* I am very grateful to Pr. *Lahcen Omari*, without whom this thesis would have only been a dream. I would like to thank him for providing me the opportunity to conduct research under his supervision. My special appreciation and heartily thanks goes to Pr. *Ismail Berrada*, my joint supervisor and long time teacher. This work would not have been possible without his guidance, support and encouragement. His advice on research and on my career development have been priceless. I am also thankful for the excellent example he has provided as a successful and supportive professor.

Besides my advisor, I would like to thank the rest of my thesis committee, Pr. *Najiba El Amrani El Idrissi*, Pr. *Akka Zemmari*, Pr. *Hicham Ghennioui*, Pr. *Mohammed Al Achhab*, Pr. *Noureddine Rais* and Pr. *Saad Bennani* for serving as my committee members, for their encouragement and also for insightful comments that incented me to widen my research from various perspectives.

I take this opportunity to sincerely acknowledge the Centre National Pour La Recherche Scientifique Et Technique (CNRST), for providing excellence research grants which buttressed me to perform my work comfortably.

Last but not the least, I would like to thank my family, the number one reason I am where I am today. Thank you for being there for me during this difficult time. Words cannot express how grateful I am to my parents for all the sacrifices they made on my behalf. For their prayers which I believe have sustained me so far. For the presence of my sisters and my brother who encouraged me and supported me financially and emotionally. I am also indebted to all my friends, colleagues and fellow graduate students whose help and encouragement filled, refueled and re-energize me during my PhD journey.

Contents

List of Figures

List of Tables

Abbreviations

ITS	**I**ntelligent **T**ransportation **S**ystem
NHTSA	U.S **N**ational **H**ighway **T**raffic **S**afety **A**dministration
WHO	**W**orld **H**ealth **O**rganization
DVE	**D**riving, **V**ehicle, **E**nvironment
IV	**I**ntelligent **V**ehicle
ADAS	**A**dvanced **D**riving **A**ssistance **S**ystem
V2X	**V**ehicle-to-everything communication
IVDR	**I**n-**V**ehicle **D**ata **R**ecorder
CAN	**C**ontroller **A**rea **N**etwork
NDD	**N**aturalistic **D**riving **D**ata
GPS	**G**lobal **P**osition **S**ystem
BP	**B**rake **P**edal **P**ressure
GP	**G**as **P**edal **P**ressure
rfd	**r**ight **f**ront **d**istance
lfd	**l**eft **f**ront **d**istance
PRHIOA	**P**robabilistic **R**ectangulat **H**ybrid **I**nput **O**utput **A**utomata
ADG	**A**ttributed **D**irected **G**raph
LA	**L**earning **A**utomata
PCTL	**P**robabilistic **C**omputational **T**ree **L**ogic
SVM	**S**upport **V**ector **M**achines
HMM	**H**idden **M**arkov **M**odel
ANN	**A**rtificial **N**eural **N**etwork
GED	**G**raph **E**dit **D**istance
OBD	**O**n-**B**oard **D**iagnostics
JSON	**J**avaScript **O**bject **N**otation

Chapter 1

General Introduction

"The last thing that we find in making a book is to know what we must put first."

– Blaise Pascal

This introductory chapter aims to give an overview of the present PhD thesis. It starts by giving the general context and motivations of the undertaken research on driver behavior modeling and analysis. Then briefly presents the scope and the principal contributions of this thesis. The chapter also introduces the remaining chapters of this dissertation.

1.1 Background

1.1.1 Road traffic problems and driver behavior

The global number of vehicles around the world is on the rise. As can be seen in Figure 1.1, around 947 million passenger cars and 335 million commercial vehicles were in operation worldwide in 2015. This number is expected to double and reach a 2 billion-unit mark by 2040, according to a report by research house Bernstein [7]. This increase in vehicle ownership is generally driven by the growth in population, the economic prosperity and the powerful desire for a personal means of mobility. The manufacturing of cheaper cars has also contributed to this rise, by opening up new markets, especially in low and middle income countries with high population, e.g. China and India. Despite the multiple benefits of vehicle ownership, especially in providing freedom and convenience of mobility, the increasing rate of vehicles use on roads results in severe consequences to the environment (more vehicles implies more fuel use), and to the society. Some countries with no intention to expand their road network such as Singapore have started taken measures to maintain a zero growth rate for personal cars on its roads until 2020.

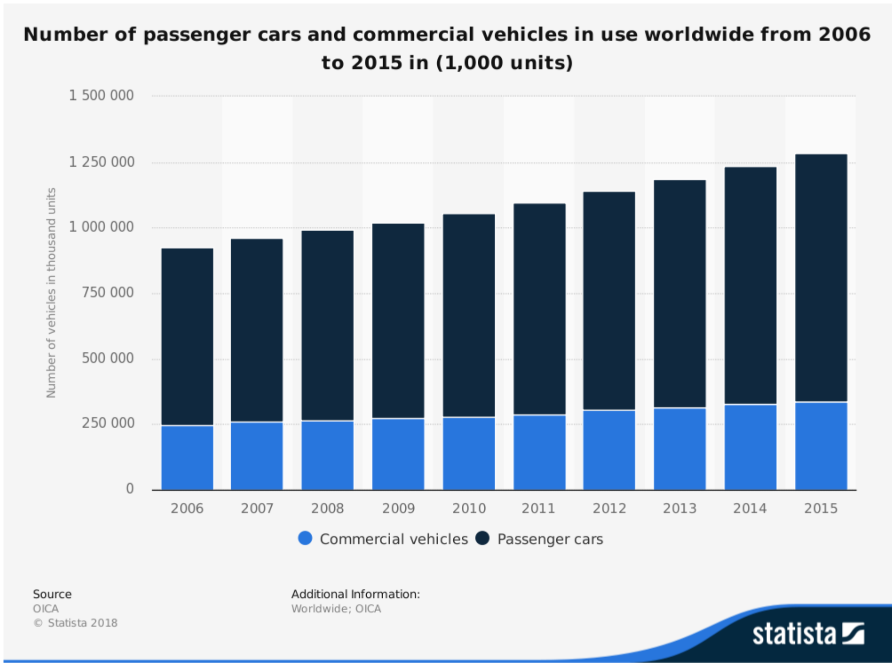


FIGURE 1.1: Number of passenger cars and commercial vehicles in use worldwide from 2006 to 2015 in (1,000 units)[1]

A direct consequence of the increasing vehicle ownership is the rise of road fatalities. Every year, more than 1.25 million people die on roads, and 90% of road traffic fatality are in low-income and middle-income countries, according to the report by the world health organization (WHO) on road safety in 2015. In Figure 1.2, portraying road death rates by countries, we can clearly see the difference in mortality rate, between the WHO African region with the higher rate, and other high-income countries such as United Kingdom and Sweden who are considered as top performers with regard to traffic safety. Predictions reveal that without sustained action, road traffic crashes will rise from the tenth place to become the seventh leading cause of death by 2030 [8].

For all the aforementioned road traffic problems, traffic safety and traffic efficiency have become issues of high priority for cities worldwide. Consequently, more efforts and synergy between different players should be considered to develop a safe and efficient transportation system for all road users.

Traffic safety is influenced by various factors, related to *road users* (i.e. drivers, cyclists, pedestrians, etc. and all people sharing roads), the *vehicle* failures and degradation, and the *road environment* (i.e. road design, weather, etc). Several studies performed on crashes around the world conclude that human error is the main cause of road fatalities.

[1]https://www.statista.com

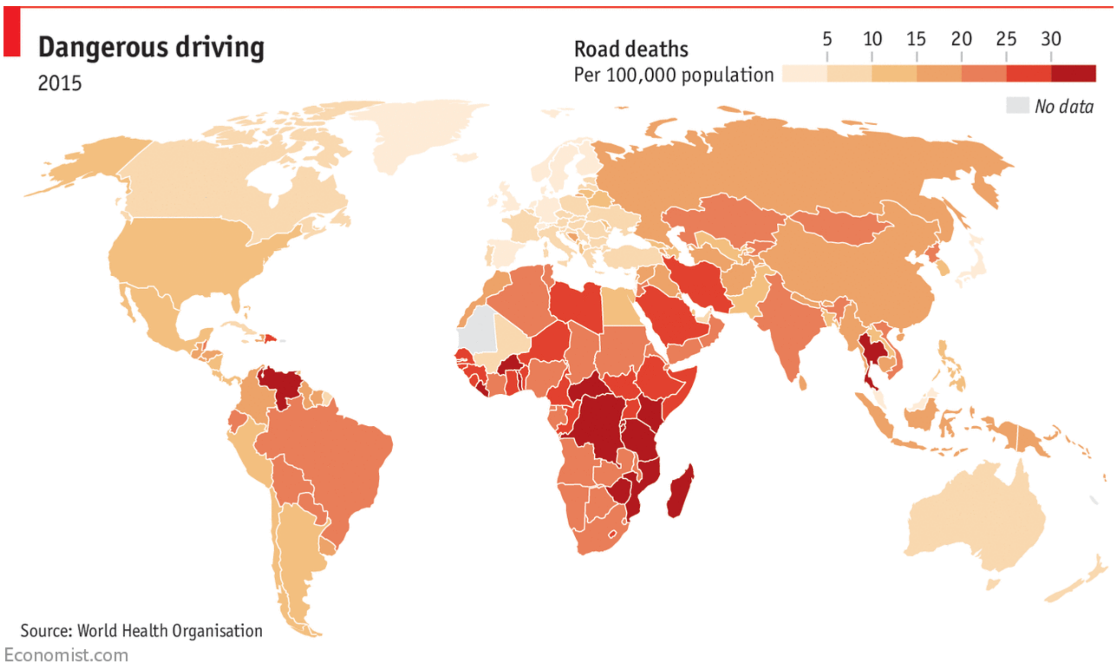


FIGURE 1.2: Road deaths around the world

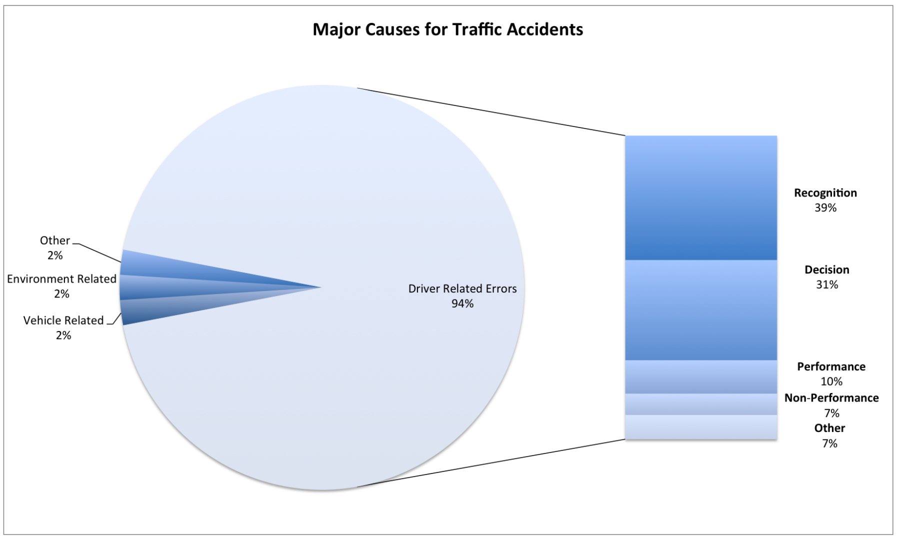


FIGURE 1.3: Major causes for traffic accidents (U.S NHTSA report)

According to a report by the U.S National Highway Traffic Safety Administration, human drivers were a contributing factor of an estimated 94% of crashes occurring between 2005 and 2007; the vehicle and environment contributed only to 2% of crashes each [9]. The distribution of the different driver-related errors, reported in [9], is shown in Figure 1.3. As portrayed in the figure, five categories of errors related to drivers were distinguished. The two most frequent driver-related errors were the errors in *recognition*, e.g. driver's inattention, internal and external distractions, and inadequate surveillance, and in *driver's decision*, e.g. driving too fast for conditions or curve, false assumptions of others' actions, illegal maneuver, misjudgment of gaps or speeds, with percentages

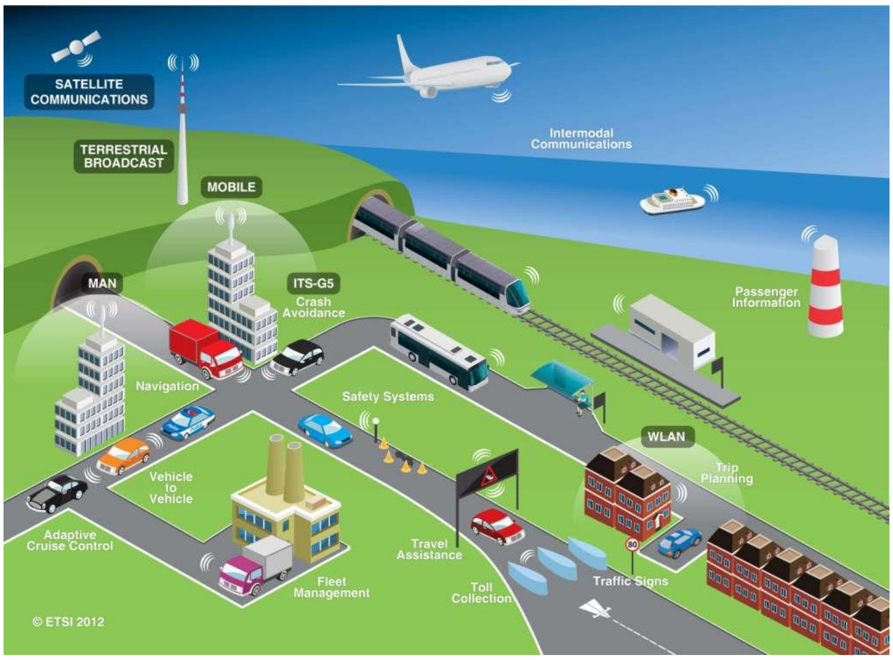


FIGURE 1.4: Intelligent Transportation System (ITS)

of 39% and 31% respectively. Other errors with lower percentages include errors about *performance*, e.g. overcompensation, poor directional control, and *non performance*, e.g. sleep.

In addition to human errors and misjudgment, inappropriate driving attitude and non compliance with traffic rules has a great bearing on road safety. It is therefore necessary to include driver behavior studies in road safety research to allow more thoughtful traffic safety improvement.

1.1.2 Technology at the service of transportation

Following the advances in computing and the emergence of communication technologies in the late 20th century, transportation professionals have started developing technology-based solutions to address road traffic problems. As technology has been boosting innovation in different domains, its application to transportation systems is expected to ensure safer and efficient utilization of vehicles and transportation networks. The term Intelligent Transportation Systems (ITS), originally intelligent vehicle-highway systems, was then invented to define transportation systems where computers, communication and various technologies are applied to roads, vehicles and users(e.g. passengers, pedestrians, etc), to enhance safety and provide efficient mobility and management. Figure 1.4 shows the components of an intelligent transportation system, and how communication technology is used to ensure the connection between them.

FIGURE 1.5: Waymo self driving car. (by EatePurple at English Wikipedia)

FIGURE 1.6: Dashboard of Tesla autopilot model S (from https://www.tesla.com/presskit)

ITS is a multidisciplinary concept that relies on research studies in different areas from electronics and computing to information systems and data analytics, and need efforts from transportation professionals, automotive industry and government decision makers for a successful deployment. The ultimate goal of ITS is a multi-modal transportation system that interfaces different modalities of transportation (i.e. road, rail, maritime, ferry and air transport) to provide efficient mobility services.

Vehicles have always been an integral part of transportation systems. Thus, developing new innovative in-vehicles solutions (e.g. solutions that encourage appropriate driver behavior and secure their compliance with traffic rules) has a significant potential to increase road safety. In the context of ITS, vehicles have developed into intelligent entities with powerful embedded sensors and communication technologies, that make them capable to interact with the other component of the transportation system, including other vehicles, roads, and people. From infotainment to active safety, the market of intelligent vehicles technologies has already been flooded with a broad range of innovative automotive products and services. Car makers and other automotive industry new players have started developing autonomous features for intelligent cars, and initiating the development of self-driving or autonomous cars (Figures 1.5 and 1.6).

1.1.3 Digital technology and the future of the automotive industry

1.1.3.1 Digital disruption

The exponential growth of computing, information and communication technology have opened up huge opportunities for new product/service innovations (Figure 1.7). Digital technologies have been practically changing every aspect of our everyday life, influencing people behaviors and raising their expectations as a result. Companies across industries are already opting for digital transformation to keep up with their costumers' expectations and survive the digitization phenomenon. In order to fully understand the implications of such transformation on industry and society, a definition of the term digitization and its derivatives is essential. The use of the terms digitization, digitalization and digital transformation can lead to confusion, it is therefore important to highlight the difference between the three terms.

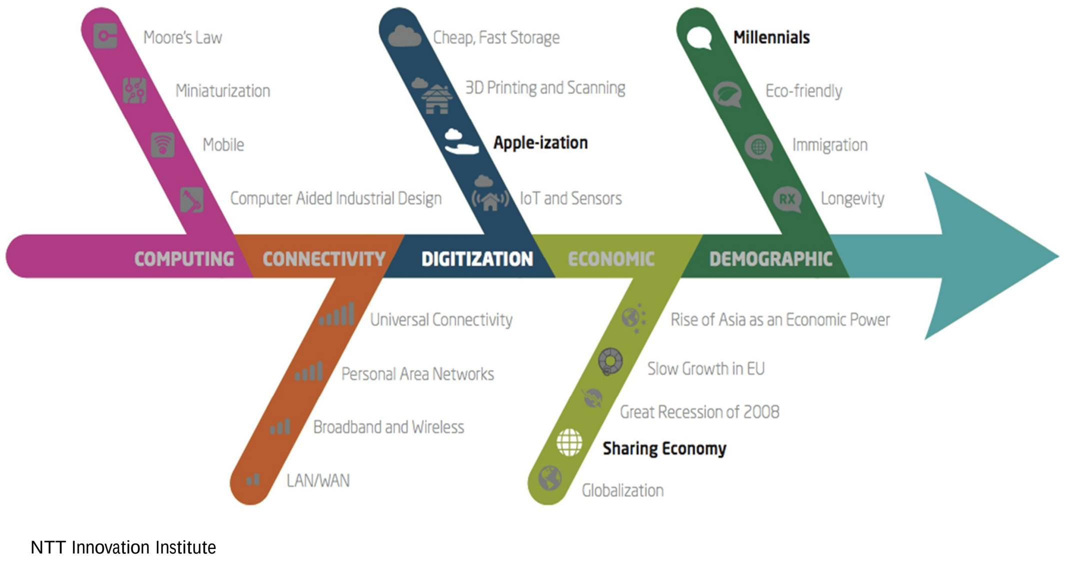


FIGURE 1.7: Trends driving structural change in industries

Digitization and digitalization have been often used interchangeably. Even though they are closely associated there still exists a slight difference between the two terms [10]. In its simplest, digitization refers to the analog-to-digital conversion process, whereby non-digital information, be it signals, images or sound, is converted in some digital format, which can be processed by computer programs. With everything digitalized, companies will dispose of a huge collection of digital data, and by opting for digitalization they can hugely improve their processes and products. On the other hand, digitalization is defined as the use of digital technologies to change a business model and provide new revenue and value-producing opportunities; it is the process of moving to a digital business (*Gartner IT glossary*). Digitalization is converging all technologies, from cloud-based IT, mobile, and the Internet of Things (IoT) to Big Data [11]. Basically, it is the road of moving to digital transformation. Digital transformation goes far beyond digitization and is broader than digitalization; it is an enterprise-wide phenomenon that encompasses the full businesses across all activities and across existing and new ecosystems [12].

Like any other industry, the automotive sector is facing a digital disruption. Digital technologies, such as sensors, navigation systems, communication devices and real time data processors, and their incorporation into cars, have brought brand new innovative services and solutions to the market, driving as a result a digital transformation of the automotive industry. Basically, the digital transformation of "*the automotive industry is the innovative reassembly of customer and company resources, and of products and services, in order to grow value, revenue and efficiency via digital technologies*"[13]. This transformation has also encouraged the emergence of new entrants to the traditionally closed automotive industry. In fact, as the automotive industry is gradually transforming from product-based to software-based industry, traditional peers like auto manufacturers, auto suppliers, retailers, will face new digitally astute entrants like mobility providers, tech providers and specific service providers. In this new ecosystem, many manufacturers are already developing new digital strategies to improve their processes and offerings. In short, digitalization is changing the future of the automotive industry, making car technology more attractive.

1.1.3.2 Key digital trends in the automotive industry

According to analysts, the future roadmap of digitalization in the automotive industry is expected to move rapidly from "digital services" to "car-as-a-service" to "mobility-as-a-service. As portrayed in Figure 1.8, the digitalization of automotive industry revolves around three key themes: connected traveler, autonomous driving, and digital enterprise [13].

Connected traveler. Digitization has allowed the evolution of cars from machines purely mechanical to powerful computers on wheels with advanced connectivity. Generally, car connectivity comprises a set of functions and capabilities that digitally connect automobiles to drivers, services, and other automobiles. Connectivity has recently become an integral part of people' lives, and connected cars provide an extension of this connectivity within the vehicle, by offering services for an optimized vehicle operation and more enjoyable journey. The connected car production is growing rapidly; almost half of all vehicles will have embedded connectivity by 2024. Cars users are now experiencing new connected services -relating to navigation, infotainment, assistance, that will transform their driving experience. The pace with which digital technologies are emerging leads to a change in the behavior of users and their mobility requirement, driving a need for smart mobility services. From carpooling, car sharing to multi-modal mobility, some users today started choosing mobility over ownership; they do no longer

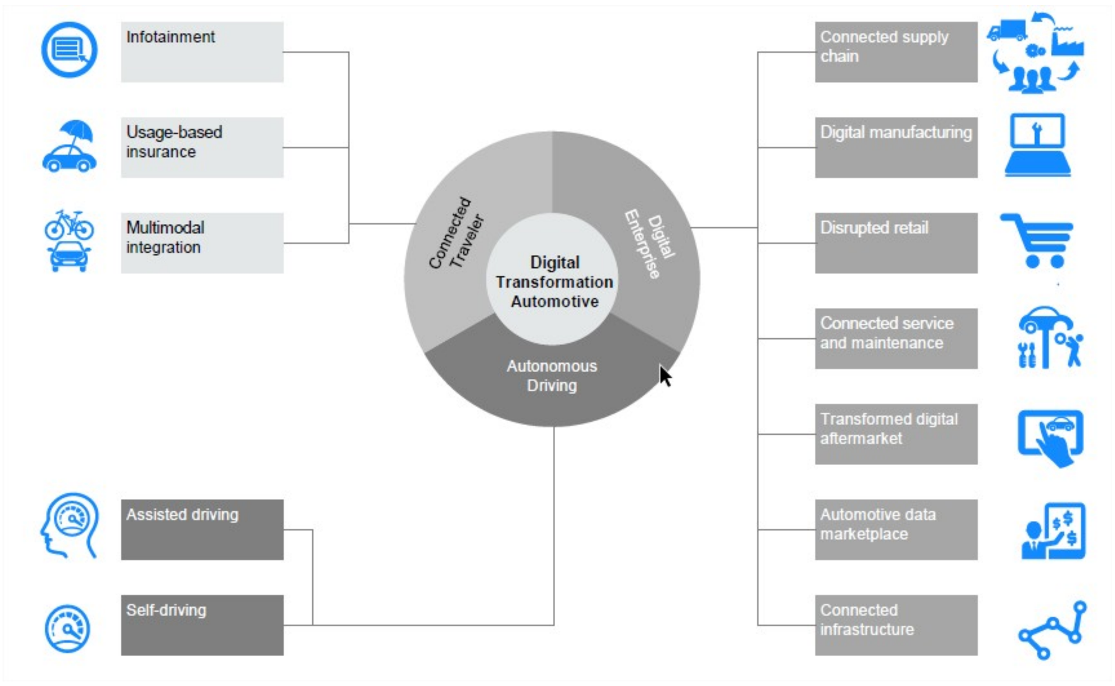
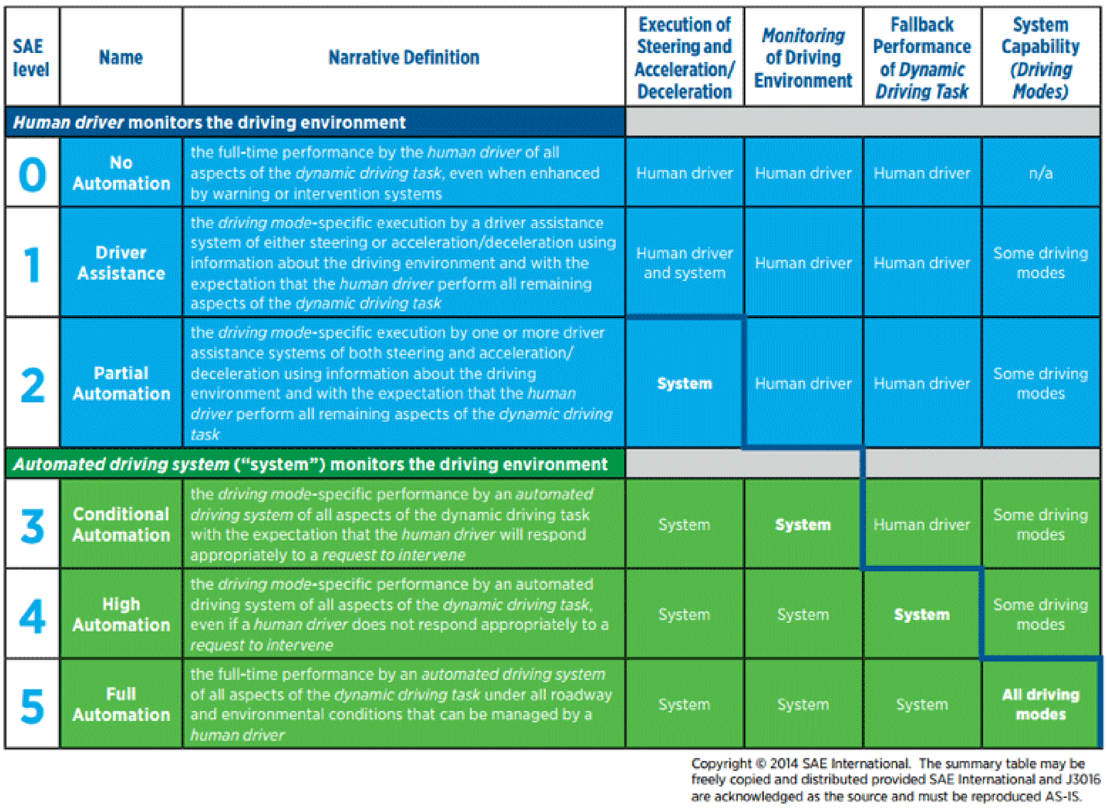


FIGURE 1.8: Key digital trends in automotive industry

consider cars as personal objects but rather a product to be shared in order to reach a better mobility.

Autonomous driving. an autonomous car in its simpler is a vehicle capable to drive itself without a human driver. Such a vehicle uses a variety of sensing technologies (radars, lidars, cameras, GPS, etc.) , V2X connectivity as well as advanced control systems to perform the driving task autonomously. Autonomous cars are characterized by varying degrees of autonomy. NHTSA has identified six levels for autonomous cars ranging from "no automation" to "full automation" (Figure 1.9). Level 0, the lowest level, relies solely on the driver performance to control the car; this later may have warnings or intervention systems but has no vehicle control. The first and second levels, namely assisted and partial automation, rely on the human driver to monitor the driving environment and share the car's control with the driver assistance systems. At the level 1, generally called "hands on", the car is expected to control one driving function, either steering or acceleration/deceleration at the time while the driver performs the remaining aspects of the driving task. Whereas, a second level autonomous car, generally called "hands off" can take the full control of the car (of steering and acceleration/deceleration simultaneously) without human interaction; Yet the driver is still needed to supervise the car operation and take control in case of systems failures. The latter higher levels required less human intervention. Conditional automation or level 3 (also known as "eyes off") allows the driver to turn his attention away from the driving task; the car system will handle all aspects of driving but expects the driver to take the appropriate control



SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
Human driver **monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
Automated driving system **("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

FIGURE 1.9: SAE International's definitions of the six levels of vehicle automation

when requested to intervene. In addition to the degree of autonomy provided at level 3, a high automated (level 4 or also "mind off") car handles all safety critical situations even when the driver fails to take the full control. At the final level of autonomy, the full autonomous car is in complete control of all operations under all roadway and environmental conditions.

Digital enterprise. Digitalization promises substantial change for the economic model of the automotive industry: According to Accenture research, digitization could unlock more that USD$2.3 billion in new value for a manufacturer with USD$55 billion of annual revenue [14]. Digitization of manufacturing will help automakers reduce costs, enhance efficiency and make more innovation. Manufacturers should therefore invest in new technologies to survive in the digital world. Integrating big data simulation and visual modeling can lower development costs and increase productivity, while creating online platforms connecting supply and demand can accelerate operations and improve efficiency of supply chain players. Technology and connected services may open the doors for generating revenue from users connectivity.

1.1.3.3 Issues and challenges of digital transformation

Although it is indisputable that adopting digital technologies has the power to increase the profitability of automobile manufacturers, digitalization creates a brand new challenges. And only by overcoming them, the industry will be able to unlock the digitization's full value potential.

At present, the main important challenge for the automotive industry is how to survive and stay relevant in the digital age. For this reason, factors such as the diversified customers expectations, outdated regulations, the pace of technological innovation should be taken into account when opting for a digital transformation of the industry. Security and data privacy are also one of the most difficult challenges for automotive industry players. Connected car services make the car more vulnerable to hacking and intrusions, affecting thus the privacy and security of consumers. The industry players should therefore consider cyber security solutions to build and guarantee the consumers trust.

1.2 Motivations for driver behavior studies

While the race towards full autonomy has recently accelerated at a fast pace, there are still challenges facing their deployment. The added cost of expensive sensors, security and liability concerns on the one hand, the lack of a regulatory framework and consumer acceptance on the other hand make the widespread adoption of autonomous cars questionable. Estimations reveal that, in UK, a country with a strong interest in self-driving features, only 25% of cars sold in 2035 will be autonomous comprising 15% partially-autonomous cars and 10% fully autonomous cars[15]. This low penetration rate results in a heterogeneity in the users of near future roads (i.e. active drivers, robot drivers, pedestrians, etc.). Before becoming a standard technology, autonomous cars will be sharing roads with human operated vehicles, which makes it practically difficult to remove humans from the equation. Human drivers behavior will undoubtedly remain a key element in road safety, and the development of approaches to improve the safety and comfort of human drivers is still relevant. Modeling and analyzing how drivers perform their daily driving helps to characterize, predict and determine how safe is their behavior, and therefore constitutes an important step in the design of adaptive driving assistance systems.

1.3 Emergence of driving behavior analytics

Quite recently, considerable attention has been paid to driving behavior analytics as a tool to make drivers better and safer. The myriad of sensors integrated into intelligent vehicles offer new wealthy sources of information about the driver, the vehicle and the environment. Such information allows to study the driver behavior in realistic situations, and to better understand the relations between the driver actions, the vehicle performance and the driving environment. The new generations of cars are generating more and more voluminous data. According to IBM [16], car sensors can produce about 1.3 gigabytes of data every hour, and an estimated 312 million gigabytes yearly for 4 hours of daily driving. It is therefore important to emphasize that automotive solutions are facing a big data challenge [17] and more powerful analysis tools are needed.

Driving behavior analytics solutions process vehicle data representing driver behavior and transform it into valuable information, with data visualization functionalities to facilitate the presentation of results. Several analytics solutions are currently offered as standalone products designed for insurers, fleet managers as well as individuals. We cite as examples of existing commercial solutions, *i4drive Driving Analytics*[18], *CLOUD MADE's Car and Driver Analytics*[19] *Zendrive*[20], and *AMODO driving analytics*[21]). In the near future, drivers of cars intelligent versions will benefit from built-in analytics. Data analytics applied to driving behavior will provide comprehensive platforms that process driving data and give more insights on drivers' behaviors.

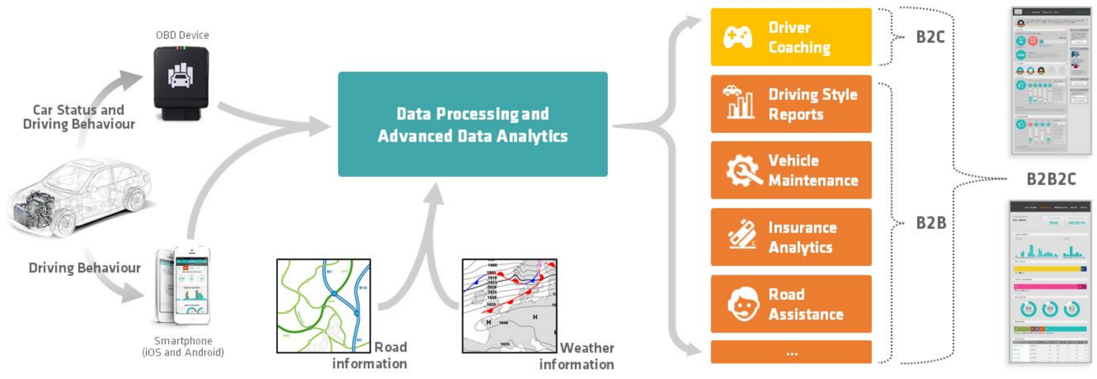


FIGURE 1.10: AMODO Advanced Driving Behavior Analytics

1.4 Scope and contributions of the thesis

This thesis addresses the topic of human driving behavior modeling and its contribution in performing advanced analyses. Though this topic has been widely treated, there still exist some important issues in driving safety that we assume were poorly addressed, as well as diverse mathematical methodologies whose application in driving behavior analysis is to be investigated. The focus, in this thesis, was oriented towards the application of formal approaches from theoretical computer sciences to human driving behavior modeling and analysis.

The main goal is to present a framework for a data-centric modeling (i.e. a modeling using driver behavior data) of driver behavior in a connected environment (the ITS context), which can capture the driver's interactions with its vehicle as well as its driving environment (roads, pedestrians, vehicles, etc). The models established using the proposed framework are to be used for finding characteristics of drivers (i.e. driver profiling) and for developing unique drivers profiles. Throughout this work, we also emphasize the contribution of modeling in enabling more profound analyses of driving behavior, by proposing model-based analysis methods. By using a modeling approach based on graphical models, this thesis brings to light new applications of model checking and graph matching techniques in the context of driver behavior research, which are (1) analysis of drivers compliance with traffic rules and (2) analysis of drivers behavior similarity. To achieve the aforementioned purposes, a complete methodology for driver behavior data analysis, including data abstraction, modeling and analysis is presented.

That being said, the present thesis consists of four contributions:

1. Numerical abstraction as a method for driving data preprocessing: In this first contribution, we deal with the driving data size problem caused by the large number of vehicle sensors and their continuous operation. To reduce the size of driving data, and of the driver behavior model consequently, measurements of selected features are approximated by intervals from the interval abstract domain.

2. Probabilistic Hybrid Automata (PRHIOA) and Attributed Directed Graphs (ADG) models combined with machine learning as tools for driver behavior modeling: In this second contribution, we represent driver behavior as an automaton (resp. graph) model, where states (resp. vertices) represent vehicle dynamic states and transitions (resp. edges) represent driver actions. The learning automata algorithm is applied to learn the transitions probability (resp. edges weights) of the model from driving data, and construct thus the model of driver behavior.

3. Model checking and its application to verification of drivers compliance with safety measures and traffic rules: This contribution introduces model checking as a method for the verification of the driver behavior's safety. Modeling the driver behavior as automaton not only allows a representation at adjustable levels of abstraction but also enables a formal verification of driver behavior safety.

4. Graph matching and its application to drivers behavior similarity analysis: This contribution investigates the possibility of using driving behavior as feature for personal identification. It proposes graph theory as framework for the representation and analysis of drivers similarity. The modeling of driver behavior as graphs has the advantage of providing a better view of the driving behavior, but also allows to perform graph-based analysis, proven useful when applied in many different areas. To analyze the dissimilarity of drivers behaviors, we propose graph matching techniques, namely the graph edit distance, as a comparison technique.

1.5 Thesis outline

The present dissertation is divided into six chapters:

Chapter 2. Driver behavior at the age of intelligent vehicles. This chapter is devoted to presenting the different ingredients needed to start studying driver behavior. First, it starts by presenting the intelligent vehicle technology, its sensory components and the types of enabled advanced driver assistance systems as well as the roles of humans in this new vehicle technology. Then focuses on the human driving behavior, how it can be measured and its different modeling approaches. Finally, I introduce our approach for an intelligent recording of driver behavior.

Chapter 3. Model-based approach for driver behavior profiling. This chapter is dedicated to the methodology proposed for modeling driver behavior. It starts by a literature review of driver profiling and driving data modeling approaches. Then I give an overview of our proposed approach to tackle the uniqueness of drivers' behavior. The approach starts by an abstraction step to reduce the size of generated data and thus the size of the driver model. The different frameworks and tools used to represent and construct the driver models are also presented.

Chapter 4. Analysis of driving behavior safety: How safe is the driver?. This chapter tackles the problem of drivers compliance with safety measures and traffic rules. It presents a formal verification of driver behavior by means of model checking. In this verification approach, desired properties are expressed as logic formulas and a model checking tool is used to verify the driver behavior model. An application of this verification approach on data of one driver is also presented in this chapter.

Chapter 5. Graph-based driving behavior analysis. In this chapter, a methodology to compare drivers similarity using the graph matching method is introduced. At first, the graph matching problem is presented. Then the potential of performing a graph-based analysis of drivers is discussed. The chapter also highlights the utility of graph-based representation in facilitating the visual comparison of drivers.

Chapter 6. Conclusion and Perspectives. This chapter concludes the dissertation. The conclusions drawn from the preformed studies are presented and future perspectives are outlined.

Chapter 2

Driver behavior at the age of intelligent vehicles

"Twenty-five to thirty years ago it was seats, the steering wheel, stereo... There wasn't a lot of value in the interior cabin area. Move forward 30 years... good chance up to half the value of the vehicle is electronic content."

– Michael Robinet, *Managing Director, HIS Automotive*

The previous chapter was an introductory chapter, in which we addressed the digitization phenomenon and its impact on the automotive sector, and how cars digital technologies have been changing drivers behaviors and disrupting the industry. In fact, digitization of cars has led to the emergence of new intelligent vehicle systems, capable of carrying out one or more aspects of the driving task (sensing, perception, etc.), and providing thus more safety and convenience to drivers.

The goal of the present chapter is to cover different aspects related to humans and intelligent vehicle technology. First, we present the key components of intelligent vehicles with a focus on sensory components, which have made the study of driver behavior more realistic and enabled the emergence of advanced solutions for driving behavior analysis. Then, we address different elements needed to better understand and represent driver behavior.

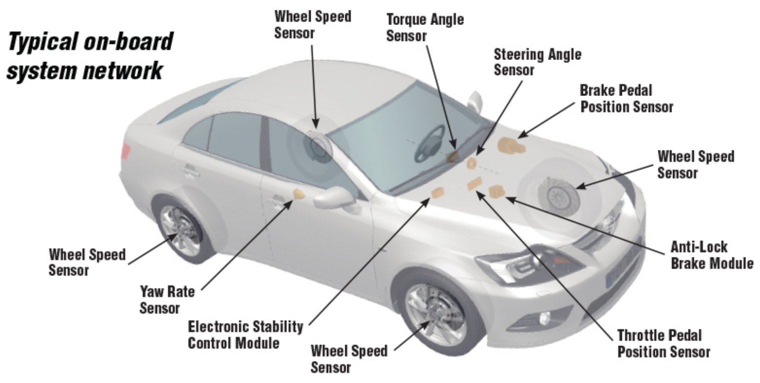


FIGURE 2.1: Components of the car network

2.1 Intelligent vehicles: digitization of the vehicle

2.1.1 Definition and key components of an intelligent vehicle

An intelligent vehicle is defined as a vehicle that performs certain aspects of driving either autonomously or assists the driver to perform his/her driving functions more effectively, all resulting in enhanced safety, efficiency, and environmental impact [22]. Carrying out the driving task requires the car to (1) sense and perceive its environment, (2) analyze and make decisions for driving actions and (3) execute the actions.

Since 1980s, vehicles dispose of a network of sensors that allow for an estimation of their current state. Figure 2.1 shows the different sensors in a typical on-board system network. It is obvious that the more sensors a vehicle have, the more intelligent it will become. The components of an automated/connected vehicle are depicted in Figure 2.2. Vehicles functionalities can thus be enhanced by equipping vehicles with perception sensors, GPS and communication technology that allow them to reliably perceive their surrounding environment. Automotive sensors can be grouped into three different categories:

- **General in-vehicle sensors**, such as accelerometers and wheel speed sensors which are built up together with the vehicle.
- **Perception sensors**, such as radars, lidars and vision systems.
- **Virtual sensors** that are not actual sensors but information sources, such as GPS and wireless communication.

The different sensors are considered as inputs to the vehicle system, which use the sensory information to determine control commands and activate car actuators.

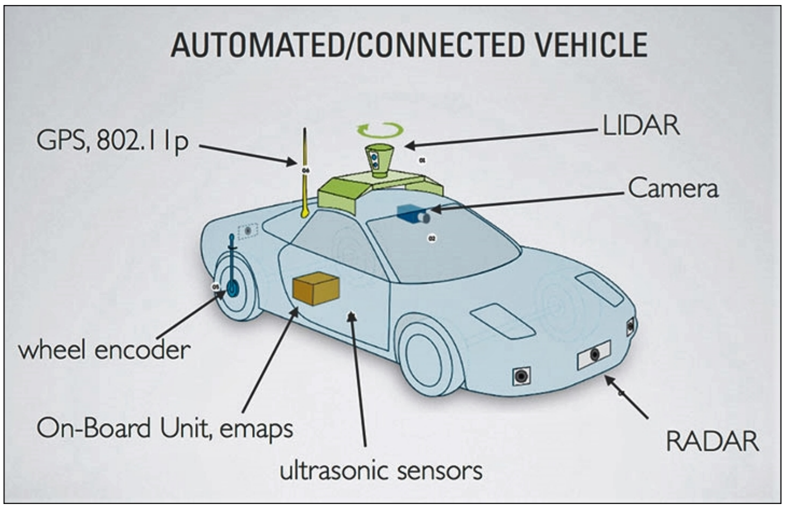


FIGURE 2.2: Components of a self-driving car (from [1])

The vehicle system is actually composed of a number of controllers, known as electronic control units (ECUs), each of which is responsible for a certain functionality in the vehicle, and communicating with each other using a controller area network (CAN) bus [23]. By way of example, door control unit (DCU) which controls and monitors electronic accessories in a vehicle's door, and cruise control which controls the vehicle's speed automatically. Recently, vehicles are equipped with more advanced control technologies that provide more safety and convenience to the driver. These control technologies can range from passive safety like airbags and seat belts lock engagement to active safety of different levels of autonomy.

2.1.2 Advanced driving assistance systems

Advanced driving assistance systems (ADAS) refer to advanced in-vehicle control systems developed to help the driver with the driving task. Their use shows a growing importance as they are expected to help improving road safety, increasing road capacity and providing more comfort for passengers. During the last few years, different kinds of driver assistance systems have been developed such as collision warning, crash avoidance, GPS based navigation system, lane departure warning and parking spot locator, etc. With respect to their level of intervention, the following types of ADAS can be distinguished:

- Driver warning systems: These systems use different kinds of warnings (visual, auditory or tactile) to alert the driver about dangers and/or possible corrective actions. Lane Departure Warning system (LDW) [24] is an example of warning systems that, based on perception of the vehicle lateral movement, warns the driver if he is drifting out of the lane. Authors of [25] present another example of assistance systems, in which vehicular communication technology is used to send warnings not only to the driver, in case of detected abnormal driving behaviors, but also to surrounding vehicles via vehicular communication.

- Perception enhancement systems: Such systems use information from the automotive sensory system to increase the driver perception and his awareness about the driving situation. Because of the complexity of the driving environment and the diversity/heterogeneity of its components (traffic signs, obstacles, environmental conditions, etc.), different perception assistance systems have emerged that differ by their functionalities and the technologies on which they are based (vision/camera systems, Lidar/radar technologies, and vehicular communications, etc.). For instance, vision-based systems using image processing for traffic signs recognition were proposed in [26][27]. Both systems use robust and fast algorithms that have achieved good performance. The algorithm in [27] is however restricted to speed signs recognition. Aside from traffic sign, obstacle (fixed or mobile) detection has been addressed in many studies. In [28], a visual processing system for pedestrian detection is proposed, and its use in assistance applications is discussed. While perception enhancement is still an open research area, automotive industry is recently full of commercialized perception based assistance systems like: blind spot monitoring introduced by Volvo, lane departure warning systems firstly installed on Mercedes commercial trucks, intersection assistance first introduced by Toyota, etc.

- Intervening systems: These systems take over the car's control in crucial situations by performing corrective actions. The lane keeping system in [29] is an example of intervention system that monitors the lateral movement of the car and automatically takes control of its steering to keep it on its lane, when the driver is not responding to the system's warnings. Another example of active safety is the system presented in [30] that provides the driver with brake assistance. The system analyzes the information about the vehicle, environment and driver, identifies the need for braking action, and based on the driver awareness decides if an automatic braking is needed.

- Fully automated systems: These autonomous systems operate without human input, they are considered as enablers of the future driverless cars.

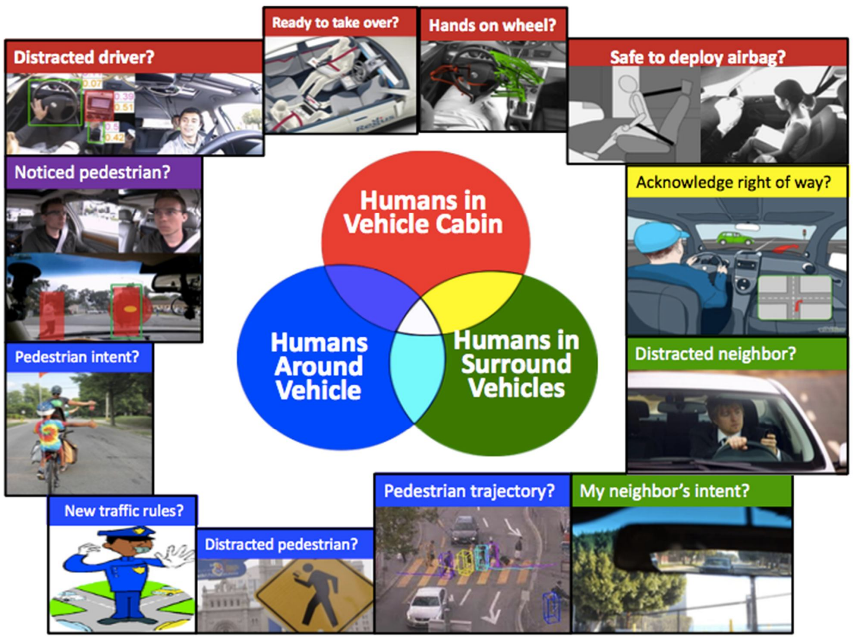


FIGURE 2.3: Human roles in the age of self-driving (from [2])

2.1.3 Human roles in intelligent vehicles

2.1.3.1 Humans and autonomy

The evolution towards autonomous driving has already begun. Semi-autonomous features such as lane keep assistant, adaptive cruise control, etc. are already available in the newer generation of cars. According to automakers and technology developers fully autonomous cars are expected to be on the roads by 2030. However, Lux research analysts claim that the full autonomy remains elusive. In UK for instance, a penetration rate of 25% may be optimistic, according to a report published by KPMG [31], which means that 75% of other cars on the road will be human operated. This heterogeneity in future roads occupants requires intelligent vehicles to take into consideration different aspects of human behavior.

Authors in [2] identify three domains of intelligent vehicle-human interaction: *humans in vehicle cabin*, *humans around the vehicle* and *humans in surrounding vehicle*. Figure 2.3 presented in [2], shows some of the research problematics in relation with the different human roles. In respect with the goals of this thesis, we will focus more on the first domain, that is **humans in vehicle cabin**.

Human drivers inside an intelligent vehicle can have two main roles: system operator, and active driver (i.e. a key player in the driver-vehicle-environment system) [32].

2.1.3.2 Humans inside vehicle cabin

Humans as supervisors. With more driving related tasks automated, the driver is allowed to delegate the authority of driving, under specified driving conditions, to a car's software. The driver is kept engaged and in the loop, to ensure that the car is capable of driving autonomously. He is considered as a supervisor of the automated system and will be the main responsible for all the actions of the car. When the automated driving system is engaged, the human as supervisor would not be in control of the car, which will perform the driving functions depending on the level of automation. For instance, in a vehicle with conditional automation, the driver cede full control of all driving tasks and he is not required to monitor neither the driving environment nor the operation of the car's system. However, a human supervisor should remain alert to system errors related to the dynamic driving and ready to respond appropriately.

Humans as active drivers. Beside his supervisory role of the automated system of an intelligent vehicle, the human has to perform the non-automated driving tasks. In an active driver mode, the human plays a key role in the driver-vehicle-environment triad. He has to assess what is safe, and determine the actions to be executed by the vehicle for its lateral and longitudinal movement. However, because of human errors, vehicles should be equipped with intelligent solutions to assess driving safety and ensure appropriate driving. Hence, the importance of driver studies research in the development of intelligent vehicles and of solutions for ITS.

In the remaining of this dissertation, we study the behavior of humans as active drivers fully engaged in the driving activity. The important contribution of drivers in traffic safety has already being discussed in chapter 1, and we assume that it will remain important during this transitional period toward autonomous driving, especially that autonomous vehicles are designed to enable drivers to transit easily between autonomous and manual modes.

2.2 Driverology: The science of driver behavior

2.2.1 Definition of driving task

Literally speaking, driving a vehicle means moving it from a starting point to destination by relying on available sensory information to perform a set of motoric actions on the vehicle commands and controls. Considering the dynamic nature of the traffic system, driving in traffic is a complex behavioral task, which can be seen as a combination of subtasks associated with specific traffic situations. Generally, the tasks carried out by a driver inside his vehicle fall out in one of the following groups ([33]):

- Tertiary tasks: describe non-driving-related activities that are usually considered as a distraction, like controlling in-vehicle infotainment or assistance systems.
- Secondary tasks: are tasks required to support driving, like turning on the blinkers or activating the wipers...
- Primary tasks: cover all the tasks related to the operation of the car. As depicted in Figure 2.4, primary tasks can be classified in a three-level hierarchy [3]. Navigation, also known as the strategic level, refers the route planning tasks including the adaptation of itineraries to road conditions. Guidance, or the tactical level, refers to the maneuvering tasks like changing lanes, overtaking. And, stabilization, or the operational level, concerns the car control.

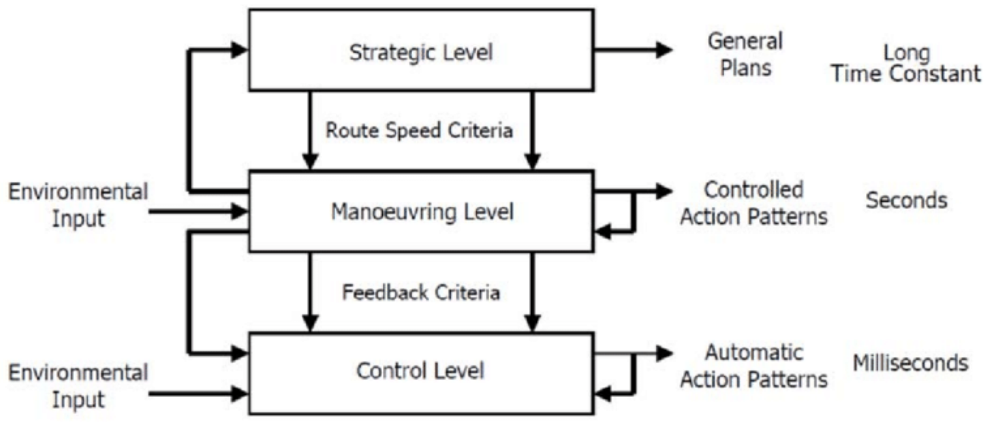


FIGURE 2.4: The hierarchical structure of the driving task (after Michon, 1985 [3]])

2.2.2 The driver-vehicle-environment system

Understanding human driving behavior requires consideration of the traffic system as a whole, emphasizing the relationships between its three interactive parts, the driver, the vehicle and the road environment. These three parts and their interactions are referred to as DVE system (Driver-Vehicle-Environment system). Considering the new components of intelligent vehicles, we present the DVE as a closed loop system which components are presented in Figure 2.5 and detailed bellow.

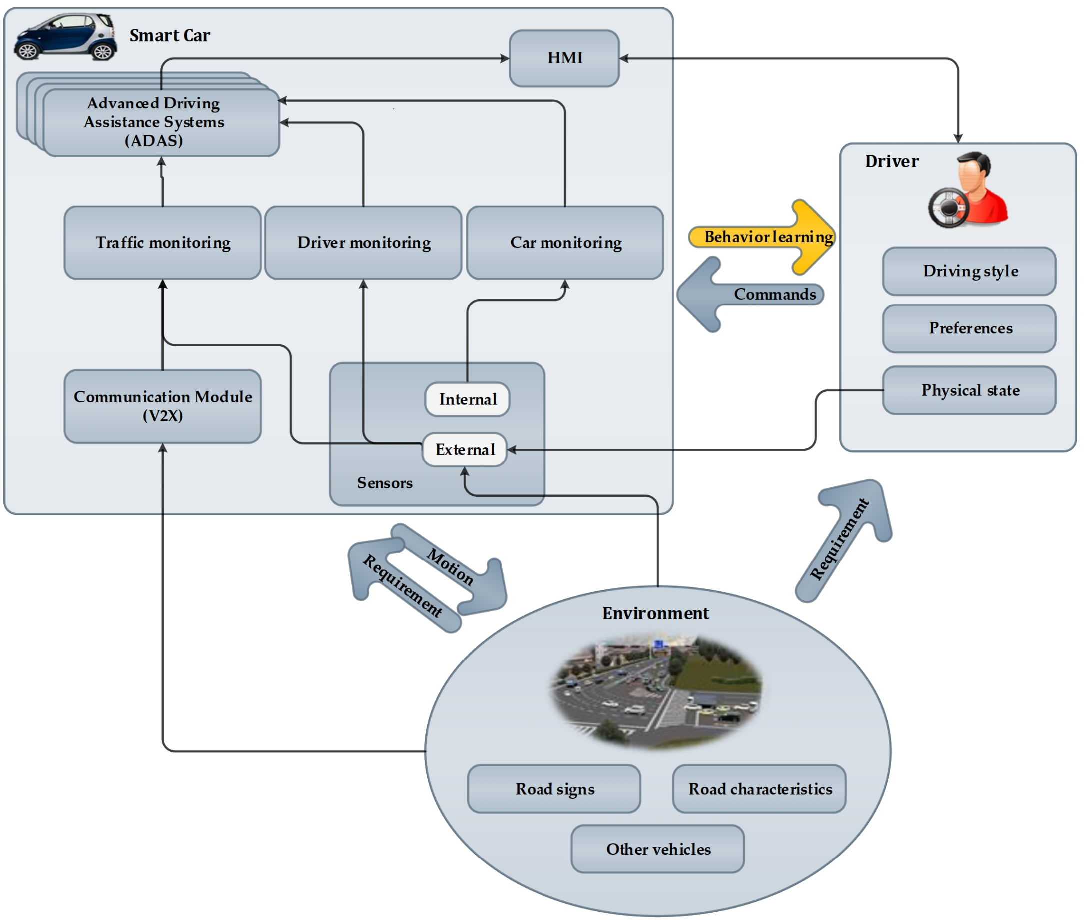


FIGURE 2.5: Interactions within the driver-vehicle-environment (DVE) system

The vehicle. A smart car can be equipped with various subsystems which we categorize into 3 layers:

- A sensing layer that can be composed of (1) car internal sensors such as accelerator and wheel speed sensors, (2) external sensors that can be connected to the car system such as GPS, cameras as well as smartphones sensors, (3) V2X communication module for intra vehicular communication.
- A monitoring layer consists of systems that capture and process signals from the sensing layer. Examples of systems from the monitoring layer are In-Vehicle Data Recorders (IVDR).

- Advanced driving assistance systems (ADAS) complete the higher level, they rely on data from the monitoring layer to provide driver with an advanced and personalized assistance. ADAS use a Human Machine Interface (HMI) to display informative messages and warning to drivers.

The environment. The driving environment consists of a series of traffic situations experienced by the driver in some units of time and space [34]. A traffic situation can be described as a combination of specific parameters that are categorized as:

- Dynamic parameters that refer to parameters about the behavior of other vehicles, pedestrians, etc.
- Static parameters, which are parameters regarding the constructional characteristics of the road (type of the road, curvature, intersection, type of intersection, etc.) and the infrastructural characteristics (regulation type: implicit rules, road signs, etc.).

The driver. Actively engaged in the driving activity, the driver gives commands to the vehicle based on his perception of driving established with the help of in-car perception sensors. The decisions taken by the driver are however influenced by human-related factors such as age group, route preferences, etc. On the other hand the vehicle as a passive component executes the driver decisions and move in the environment accordingly. Decisions of the driver in the next step are based on the feedback from the vehicle with the information from the environment[35].

The complex relations within the DVE system make the design of a framework for driving behavior modeling more challenging. First, it must include various sources of information; vehicle on-board sensors, environmental sensors and body sensors, etc. A second challenge is how to capture and represent the relationships between the driver, the vehicle and the environment. Another challenge is how to process and reason about the available information to get actionable insights.

2.2.3 Measurement of driving behavior

Traffic safety research has been using different tools to observe and measure driving behavior. These tools generally fall in one of the 3 methods: (1) driving simulators, (2) field studies and (3) naturalistic datasets.

2.2.3.1 Driving simulators

Driving simulators have been widely used for entertainment and new drivers training, but also as a tool to monitor driver behavior in traffic safety research and to evaluate new vehicular technologies. A wide range of driving simulators has been made available to researchers, offering different levels of realism, complexity and usage costs [36]. One key aspect of driving simulators is that they allow researchers to study driver behavior in a realistic virtual environment, while providing full experimental control over driving conditions (weather condition, virtual traffic behavior, road layout, distractions, etc.) without compromising safety. In comparison with real vehicles, simulator-based studies have various advantages, which can be summarized in the following points [37]:

- *Controllability, reproducibility, and standardization.* Scenarios created using simulators are highly controllable and easily repeatable, enabling the creation of standardized driving tests and reproducible research results.
- *Ease of data collection.* Simulators allow an easy and accurate measurement of driving data, in contrast to real vehicles for which the accurate recording is more challenging.
- *Possibility of encountering dangerous driving conditions without being physically at risk.* Simulators present a safe way to replicate hazardous driving situations, without exposing drivers to real dangers.
- *Novel opportunity for feedback and instruction.* Using simulator, drivers can experience different types of instructions and feedback ranging from visual, auditory to tactile.

Despite the aforementioned benefits of driving simulators, their use in traffic safety research is still questioned, especially with the lack of studies quantifying the transferability of the simulator results obtained to real world.

2.2.3.2 Field studies

Test tracks are another research methods used for driving behavior studies. Contrary to driving simulators, participants in field tests are asked to drive real vehicles instrumented to measure general driving behavior and are often accompanied by an experimenter to monitor their behavior and provide them with experiment-related instructions. A field study may be performed on a test track closed to public traffic or on a pre-determined route. In comparison with simulator-based studies, a field test offers a higher level of

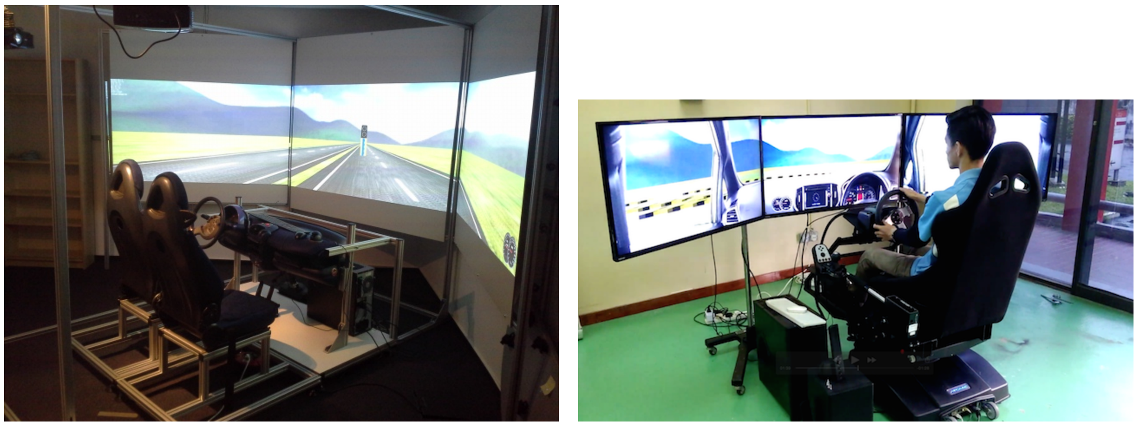


FIGURE 2.6: OpenDS driving simulator. (Image downloaded from https://www.opends.eu)

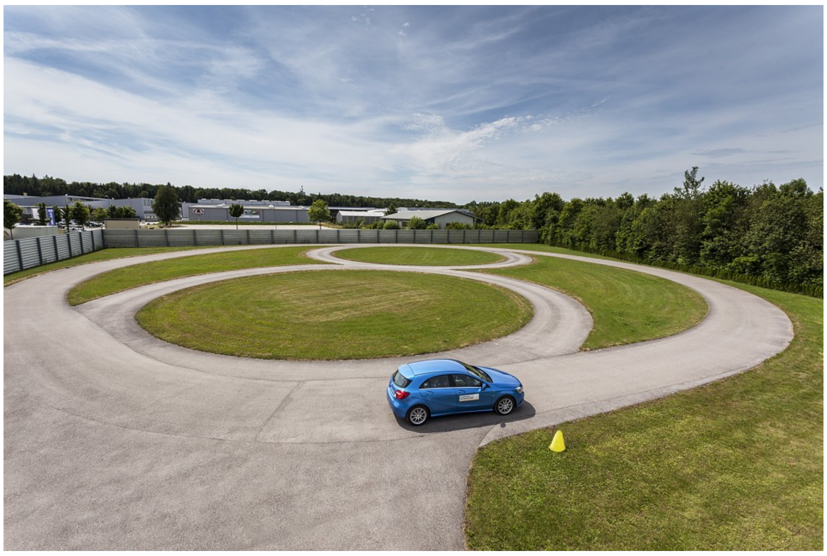


FIGURE 2.7: Test track at AUDIO MOBIL, ©Arno Laminger. Image downloaded from https://hci.sbg.ac.at/outputs/driver-distraction/

fidelity to real road driving, thereby increasing the external validity (generalization of obtained results), but a limited controllability over variables such as weather conditions and level of traffic.

2.2.3.3 Naturalistic datasets

Despite the high realism of recent driving simulators and field tests, the controllability feature makes it unclear whether the drivers will behave the same in their everyday driving. In order to achieve a much greater external validity, the naturalistic approach collects driving data unobtrusively in drivers' vehicles as they drive during their everyday trips.

In a naturalistic study, passenger cars are equipped with devices that continuously record driving data from different sensors. In Vehicle Data Recorders (IVDRs) are one of the tools widely used for on-board data collection. They are devices installed on vehicles that monitor and record continuously the vehicle parameters. Besides measurements, IVDRs may have an analysis module that provides drivers with feedback about their driving. Even though the first commercialized applications of IVDR systems were merely tracking systems designed for fleet management [38], the concept was later extended and other applications have appeared. In the USA for example, the car insurance company "Progressive Corporation" proposes to its customers the use of on-board data recorders to provide payment services in a customized way, based on their recorded driving data [39]. IVDR technology has also being promoted for private vehicle as: car black boxes (also known as video event recorders) installed on the wind-shield, which feature a camera as well as a GPS unit, and record driving performance data such as accelerating, braking and turning. In terms of safety-related traffic enforcement, data recording systems can also be used for recording traffic violation [40][41]. There exist Several other IVDRs which differ in their measurement method and also in the scale of the data collection[42].

For large-scale data collection, a study called "100-Car naturalistic study" [43] was conducted by the Virginia Tech Transportation institute where measurements from 100 equipped cars were collected. The IVDR of the equipped cars continuously records data from the car's CAN-Bus and the different sensors (accelerometer, GPS, video-based lane trackers and radar sensors). The 100-car study was the first study to collect large-scale naturalistic data. However with the era of open science, the shared data may not be realistic in some other countries. This issue has been addressed in [5] where international large-scale data were collected. The study uses 3 equipped cars deployed in 3 different countries (Japan, Turkey and US) that collect data from multiple sensors (brake and gas pedal, steering wheel angle, vehicle speed, GPS, range distance, microphones for speech recording and videos of the inside and outside the car).

2.2.4 Driver behavior modeling

Driver behavior modeling has been considered crucial for understanding behavioral dynamics of drivers and intelligent vehicle systems. The incorporation of such models into driving assistance systems and intelligent vehicles is expected to improve safety and experience of drivers. The different types of need for driver behavior models and their potential applications have led to the appearance of a large number of modeling approaches, where each approach addresses differently particular aspects of driver

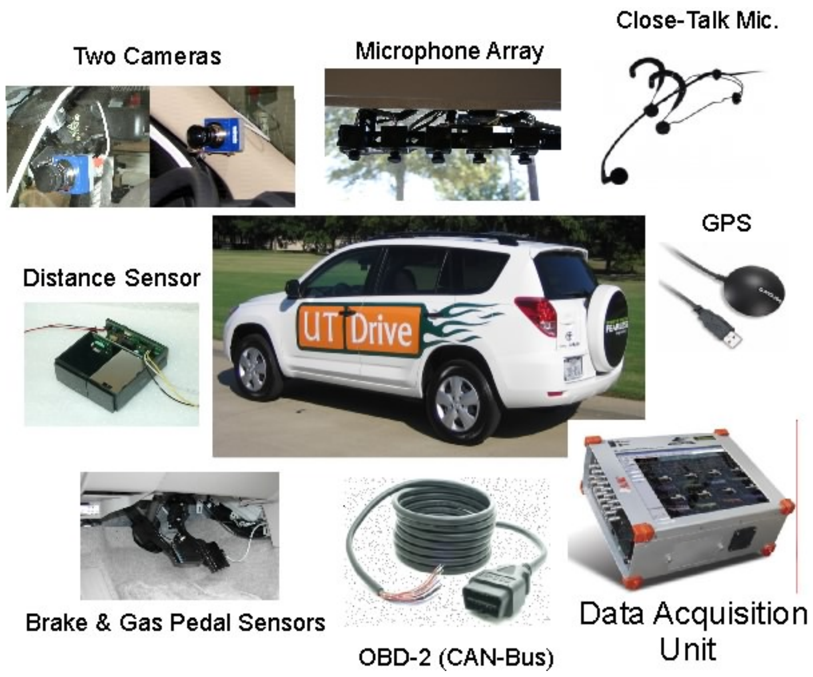


FIGURE 2.8: An instrumented vehicle used to collect naturalistic driving data (UTDrive corpus). Image downloaded from https://www.utdallas.edu/research/utdrive/corpus.html

behavior. Typically, a driver behavior modeling approach uses mathematical and analytical tools to process data from sensors, cameras and the vehicle system and develop models for particular tasks.

Driver behavior models exist in different flavors; they have different purposes, and differ in terms of tools and approaches used to capture the driving behavior. Boyraz, Pinar et al. [44] proposes a classification of driver behavior modeling approaches into four groups:

- *Control theoretic* approach models driver behavior from the control point of view, and focuses generally on specific driving maneuvers such as braking, lane keeping and collision avoidance. In control theoretic approaches, the lateral and longitudinal human driving behavior is modeled with an equation that explicitly represents the physical relationship between input and output variables
- *Human factors* based models consider the physical characteristics of human driver (e.g. visual perception, cognition, mental workload).
- *Stochastic/non-linear* approach uses powerful mathematical tools such as Bayesian and neural networks and Hidden Markov to deal with the uncertain and non-linear nature of driver behavior.
- Hybrid approach combines models from two or more of the aforementioned groups.

Also, AbuAli, Najah et al. have emphasized in their review on driver behavior modeling [45] the reactive/predictive aspect of models; Reactive models analyze driver trips or already performed maneuvers to learn the behavior of the driver, predictive models on the other hand provide a real time identification of the driver behavior.

The driver behavior modeling literature is rich of models for the different levels of the driving task (i.e. strategic, tactical and operational). At the strategic level, route choice behavior has been widely studied, and the use of several mathematical approaches including stochastic learning automata [46], fuzzy logic [47], SVM [48] and neural networks [49] has been investigated. Generally, drivers route choice behavior is modeled as an optimization problem where drivers are assumed to minimize some objective function that includes some travel characteristics[50]. It is also assumed that drivers have a full knowledge of the transportation network when choosing their routes. For instance, authors in [51] assume that drivers attempt to minimize fuel cost. They propose a route search method that suggests drivers personalized routes requiring the least amount of fuel, based on a combination of driving style recognition, and estimation of road traffic and average fuel consumption. An exhaustive study on the human aspect of route choice behavior has been performed by Tawfik, Aly M et al. in [52].

Approaches focusing on the tactical level, deal with modeling complex driving maneuvers. Some studies focuses solely on specific driving situations such as intersections [53][54][55][56], lane changing [57][58], freeway-ramps merging[59][60], or stop signs [61], whereas others address driver maneuvers independently of the driving situation. One of the first attempts to model driving maneuvers was carried out in the early 2000s [62]. Oliver, Nuria et al. in [62] have emphasized the impact of contextual information, such as the driver's gaze, the relative position of the road lanes or the relative position and direction of the surrounding traffic, on the driver's performance. Using real driving data from an instrumented car (car signal data and video signals for contextual information), they build HMMs for seven driving maneuvers: passing, changing lanes, turning, starting and stopping. The experiments have shown that the models are able to recognize and classify the driver maneuver one second before any significant change in the car signals. An approach using naturalistic driving data was presented in [63], where the authors proposed an autoregressive input-output HMM (AIOHMM) to conjointly model contextual information and driving maneuvers. They however focus on the prediction of lane change and turning maneuvers, and use a combination of inputs, including vehicle dynamics, GPS, videos of the inside and outside of the car and street maps.

There exist a number of approaches that deal with driver behavior from the control viewpoint. They aim generally to improve the performance of the vehicle dynamics and to design more human-centered active safety systems for intelligent vehicles. [30] proposes a situation aware predictive braking system that implements predicted behavioral information, vehicle and surround information in the braking system. The proposed framework allows for the assessment of the criticality of the current situation and the need for intervention by an intelligent vehicle safety system. Adaptive Control Cruise (ACC) is among the well known assistance systems, which we can found already implemented in a number of recent cars. ACC automatically adjusts the vehicle speed to maintain a predetermined distance (can be chosen by the driver) from vehicles ahead. An application of driver behavior learning in adaptive control cruise system was presented in [64], the goal was to save on the interactions between drivers and the system by automatically adjusting the gap setting based on driver type and the context of the drive. A recent survey of the current literature on driving behavior modeling from the control point of view has been covered in [65]).

2.3 Smart Driving Behavior Recording System (SDBRS)

The naturalistic driving data collection presents itself as a highly effective information gathering method. The richness of naturalistic driving data is attributable to in vehicle data acquisition systems, which monitor continuously drivers behaviors and collect various form of data. Generally, the data acquisition systems, used in naturalistic driving studies, collect electronic sensors data at short time-step. In addition, contextual details about the driving conditions or the driver physical state are recorded as video data. In fact, the quantity and variety of data recorded increase the complexity of analysis and prompt the need for new methods for processing and analyzing data.

In order to reduce the size of collected data, we propose an architecture of a system for driving data recording and analysis, that keeps an abstracted form of data. The proposed architecture takes into consideration the various sources of driver information as well as the diversity in drivers behaviors. This is by automatically recording the interactions of the driver with the intelligent vehicle system and the environment as a model representing drivers individually.The proposed architecture is represented in Figure 2.9, it mainly consists of two layers. A modeling layer consists of constructing the model of the driver behavior using data from sensors, and an application layer using the constructed model for advanced analysis of the driver behavior.

The main goal of the driving behavior recorder module is to construct a model of an individual driver based on his observed behavior. It consists mainly of a reinforcement learning agent that receives as an input information, the actions performed by the driver and the requirement of the driving environment, and uses it to learn the relationship

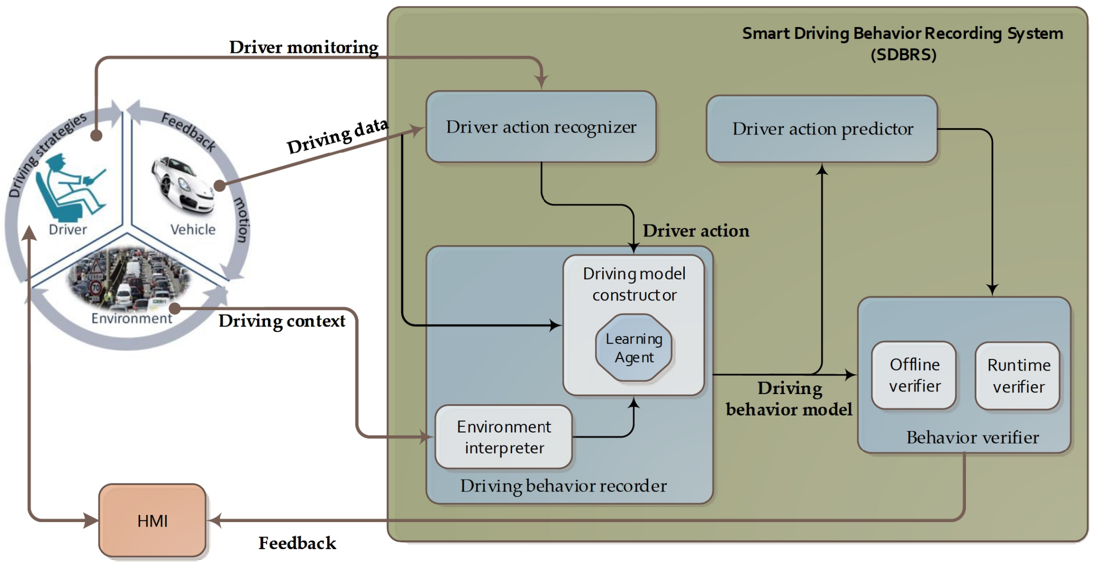


FIGURE 2.9: Architecture proposed for the smart driving behavior recording system

between driver actions and driving contexts. By continuously observing the driver behavior in the different situations encountered, the learning agent will strengthen the action mostly performed by the driver. The approach used for modeling, the parameters and the algorithm used for driving behavior learning are presented later in chapter 3. The driving behavior recording module will thus build a knowledge about the driver reactions in different driving situations, and use this knowledge to reason about similar situations.

Different signals can be considered for driver actions (e.g. stopping, accelerating, etc.) recognition, for instance in-car camera that records driver feet motion, front view camera using lane information and inter vehicle distance or other driving signals such as acceleration and braking information. Actually, the approaches used for the recognition of driving actions are out of the scope of this thesis. As for driving environment, the method used for its representation is detailed in the next paragraph.

2.4 Chapter summary

With vehicle being digitized, car making will be more about digital services that provide users with connectivity, safety and ease of use. Innovative services enabled by digitization has helped and continue to help vehicles become intelligent, and thus safer and more efficient. This chapter was dedicated to address the two important elements of a road transportation system, namely, humans and vehicles. It consists of three main sections.

In the first section, definition of intelligent vehicle is given with a description of its key components, including sensors and driver assistance systems, which have made the car more aware of its state and surrounding. We also discussed briefly the different roles of humans in an intelligent vehicle. In the second section, we approached the topic of human driving behavior modeling. First, by discussing the elements needed to better understand and represent driver behavior. Then, by presenting the methods for its measurement. Finally, by reviewing the existing approaches of driver behavior modeling. The last section addressed the data size problem in naturalistic driving data collection and introduced the need of intelligent methods to process and analyze driving data.

Chapter 3

Model-based approach for driver profiling

"Fortunately, most human behavior is learned observationally through modeling from others."

– Albert Bandura

In the previous chapter, we addressed the main elements related to driver behavior, starting from intelligent technology that have provided the accessibility to driving data, to existing methods of driver behavior modeling and the future of driving behavior analytics. Driver behavior models were proposed for different purposes. The driver behavior modeling approach presented in this chapter aims to build personal models of drivers behavior that can be used as their driving profiles. This chapter presents step by step the methodology to achieve that goal.

3.1 Overview: Driver profiling

A recent trend in driver behavior research is considering driving data analysis for driver profiling. Rather than studying specific maneuvers, driver profiling aims for a general characterization of the driver. This is generally achieved by processing low level driving data using different statistical tools, most of times combined with methods from artificial intelligence and machine learning to analyze, categorize driving styles and subsequently construct profiles of drivers behavior.

Driver profiling has recently become significantly relevant as it enables a number of innovative applications in different driver-related environments, such as insurance and fleet management. As an example, usage-based insurance is a recent concept of automotive insurance that uses profiling to provide personal payment for their customers based on driver habits. Consequently, a number of driving behavior analytics solutions have been made commercially available ([18][19][20][21]) for insurers and fleet managers as well as individuals, providing a sophisticated analysis of drivers habits and a generated score of their driving performance. Profiling drivers can also be used by ADAS to provide a personalized assistance in vehicles with multiple drivers.

Characterization of drivers can be performed based on their driving styles. Driving style is defined as the driver behaviors, belonging to the strategic, tactical or operational level, that have developed into driving habits and which recur reliably within and between trips [66]. At the tactical level for example, driving style can be measured by considering different parameters including the aggressiveness of driving maneuvers, their involvement in accidents, etc. Consequently, approaches studying driver behavior achieve their analysis by setting **recognition, detection** (of drivers, maneuvers, accidents etc.) and/or **classification** (driving styles, etc.) goals, and the models applied in these studies depends primarily on their purposes and applications [67].

Many probabilistic graphical models such as Hidden Markov Models (HMMs) have been used for driver behavior and routes recognition [62][68][69]. Such approaches for driving behavior analysis are known as *model-based methods* [70]; They generally start by establishing a model representing the driver behavior from which they infer driving characteristics. We present as example of model-based methods, the approach presented in [71]. It consists of creating a probabilistic model of the driver, based on the future environment surrounding the car, the state of the driver and the history of steering maneuvers, then applying probabilistic model checking technique to verify the safety and liveness of the driver model.

Other existing approaches, called *direct methods*, extract driving characteristics directly from the data without establishing driver models [70]. Statistical data mining techniques, cluster and principal component analyses mainly, have been proposed in [72] to analyze driving behavior data collected using a GPS-based device. The performed analysis results in the identifying four driving styles, which they label as aggressivity, speed, accelerating, and braking. In [73], Bender et al propose an unsupervised approach to characterize driver behavior as high-level driving actions (e.g. accelerating, braking). To detect these actions, the study uses a bayesian multivariate linear model combined with a sequence segmentation algorithm. A clustering-based approach to analyze car-following behavior is proposed in [74]. The authors define driver behavior as a function that maps traffic states to a driver's actions, they then use segmentation and clustering techniques to decompose the single function describing the driver into several functions which will define the driving pattern. A real-time clustering of driving behaviors using k-means algorithm has been proposed by Sonawane et al. in [75], who also developed an android and web applications implementing the proposed clustering analysis. Other classification methods such as Artificial Neural Networks, Support Vector Machines clustering and fuzzy logic based algorithms have been used in applications dealing with pattern recognition and classification of driving behavior styles and drivers states. ANNs were used in [76] to classify driver performance into four risk levels. An SVM-based algorithm, using driving performance and eye movement features, to detect driver cognitive distraction in real time has been proposed in [77]. SVMs have also been used in [55] to distinguish between compliant and violating driver behaviors at road intersections. Driver scoring based on a fuzzy logic system has been proposed in [78] and [79].

An other approach that emphasizes the heterogeneity in driving behavior is presented in [80], where authors propose Driving Habit Graph (DHG) to represent driver behavior. The DHG models the driving style of one driver in different driving maneuvers, extracted from numerical sensor data. The DHG of a driver, consists of macro nodes, representing the driving maneuvers, linked by arcs indicating the order of their occurrence. The macro nodes are modeled as a Driving Relation Map (DRM), a subgraph built from raw data with nodes reflecting the significant changes in the driving signals.

Different sources of vehicle data have been considered in driving behavior analysis studies, including controller area network (CAN), cameras and embedded sensors. An increasing number of studies have been using smartphones to measure and identify the driver behavior in terms of performed driving maneuvers. In addition to their increasing availability, the explosion in the number and kinds of sensors available inside smartphones provides a no-hardware and low cost alternative to instrumented vehicles, removing thus the burden of device installation and maintenance. In [81], for example, the authors present a system that detect driving maneuvers and road conditions using only the smartphones accelerometers. The system uses a threshold-based detection, by setting appropriate thresholds (for lateral and longitudinal acceleration) for the different studied maneuvers (acceleration, braking, left/right turning, lane change), and thresholds (for vertical acceleration) to detect road bumps. The system also provides drivers with scores that reflect the seriousness of the detected maneuvers and road conditions. An other study using smartphone sensors is presented in [82]. Contrary to [81] which uses fixed thresholds, [82] propose an adaptive method using smartphone sensors and GPS data, for driving maneuver detection (aggressive steering, acceleration and braking maneuvers). The proposed approach build a statistical model of the driver, using a Multivariate Normal (MVN) model that is frequently updated in order to adapt to changing driving conditions. The authors also propose and validate a driver scoring function that takes into account the number, severity and sequence of detected events. Karaduman and Eren use smartphone sensors data in [83] to classify and determine road shapes (straight, left/right curve) and predict driver profiles (safe or aggressive).

Beyond recognition and classification, recent studies in driving behavior analysis are investigating drivers heterogeneity and the possibility of driver fingerprinting [84]. The study performed in [84] states that drivers are distinguishable, even with little data and few sensors availability, and confirming thus the feasibility of driver fingerprinting using CAN bus data. A previous study on driver identification from CAN bus data was presented by Ly et al. in [85]; The data used represent acceleration, braking and turning events. The authors confirms through experiments the potentiality of braking and turning maneuvers in differentiating between drivers. A turning maneuver based fingerprinting of drivers is presented in[86]; the proposed approach uses mobile IMU sensor data taken during vehicle turns, and applies GMM and naive bayes classification. Other relevant works include [87],[88] and [89]. Fingerprinting drivers is very promising research with a plenty of privacy-invasive or anti-theft applications. In [90], Kwak et al. proposed a machine learning based analysis to identify drivers based on CAN data and detect auto-theft.

3.2 Proposed approach for profiling

The way drivers carry out the driving task is unique. While it is logical that static factors such as drivers age, gender and personal preferences are deterministic to distinguish drivers, it has been recently proved that driving signals alone (e.g. velocity, brake pedal pressure, etc.) can be used for drivers identification. Considering this uniqueness of driving behaviors, having individual models for drivers will be more useful to enable adaptive driving assistance. In this context, a driver behavior model is a representation of the driver's driving behavior, which can be defined by the actions performed by the driver to control his vehicle, in different driving situations. For this purpose, we represent driving behavior by a stochastic automaton that will be learned from the human driver through a monitoring of his interactions with the vehicle and the driving environment, as shown in Figure 3.1.

Automaton-based formalisms can be applied at different levels of abstraction. First we must decide what will represent the states of the automaton and what values will they cover. Given that driver behavior can be inferred directly from the vehicle dynamics, we choose automaton states to represent the vehicle dynamic states which will be determined by a set of variables. Since such variables are generally of a continuous nature, we can either opt for a low model with one to one correspondence between variable values and automaton states or for higher level model where the automaton states cover

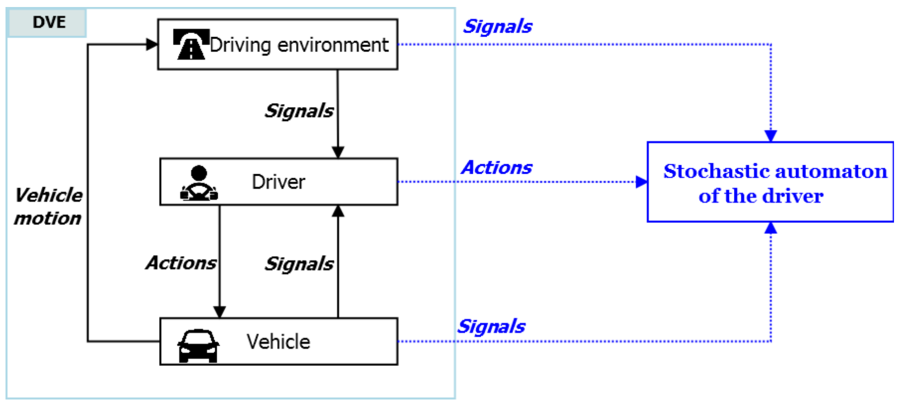


FIGURE 3.1: The learning automaton representing the driving behavior learning problem

a significant range of values. This first step of determining the desired abstraction level is presented in section 3.3.

The problem of learning stochastic automata operating in a random environment is commonly known as *learning automata* (LA). The paradigm of LA has been widely applied in systems with no complete information about the environment in which they operate. In our context, the learning mechanism will attempt to learn from a stochastic driver that constitutes together with the driving environment and the vehicle models, its learning environment Figure 3.1. The basic operation carried out by the learning automaton is the updating of the actions probabilities on the basis of the driver operation. Information about the driving environment, the vehicle dynamics and the driver forms the inputs to the automaton and influences the updating of its states and actions probabilities. The aim of the proposed learning approach is to produce an automaton that models the driver interactions within the DVE system, which can be used by a driver assistance system to predict the driver actions in different situations. In addition, we propose a similar driving behavior learning using graph-based representation, according to which the driver behavior will be represented by a graph instead of automaton. The semantic of the driver behavior automaton, graph and the learning algorithm are detailed later in this chapter.

3.3 Driving Data Abstraction

Data acquisition systems, continuously operating, collect a tremendous amount of heterogeneous data through a multiplicity of in-vehicle sensors. Such sensors allow the measurement of information on variables related to the vehicle, the driver and the surrounding environment. Accelerometers and speedometers, for instance, are used to measure acceleration and speed of the car; radar and lidar sensors for distance to obstacles

Table 3.1: Examples of driving data

Variable	**Description**
Speed	The car speed
Engine speed	The engine's RPM (Revolutions Per Minute)
Acceleration	Acceleration of the car
Steering wheel angle	The position angle of the steering wheel
Heading	The vehicle heading
Fuel level	Level of fuel contained in the fuel tank
Brake/accelerator pedal pressure	The status of the brake/accelerator pedal
Clutch pedal position	Position of the clutch pedal
Transmission gear position	State of the transmission gear
Parking brake	Status of the parking brake

measurement and GPS for location coordinate, etc. We give in Table 3.1 examples of variables related to vehicle operation. The multiplicity of variables which can realistically be collected results in a very large quantity of collected data. This large size of initial data may, in fact, yield much richer information but also makes their analysis impractical or infeasible. For a practical and more effective analysis, driving behavior modeling approaches must use some abstraction techniques that will reduce the size of data and facilitate its manipulation and evaluation.

3.3.1 Numerical abstraction

The concept of abstraction has been used in many different domains from social theory, philosophy to mathematics, and computer science. Generally speaking, abstraction is a generalization process performed by considering only essential characteristics of an object and neglecting the irrelevant details. This concept is usually depicted as shown in Figure 3.2, where a flower (complex geometrical shape) is upper-approximated by two abstract objects. The object in the middle, covering all the flower including the space between the petals, is an abstraction of the object flower. A more abstract object is depicted on the right of the figure, which is an upper-approximation of the flower shape by simple half-planes. Figure 3.3 illustrates examples of numerical values abstraction. The points in the graphs indicate the values taken by a pair of variables (x,y), and represent the concrete domain. Figure 3.3 shows three possible abstractions using three different numerical domains, namely intervals, octagons and polyhedra. The interval domain is the biggest approximation, it is easy to manipulate their constraints (of type $x_i \leq k$) but lacks precision as it includes many extra-solutions comparing to octagon and polyhedra domains. This latter (polyhedra domain) is precise but is costly since it deals with the polyhedral constraints. Thus, the trade-off is to choose an abstract

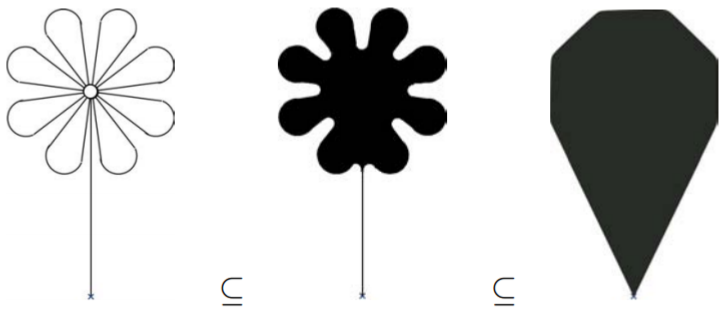


FIGURE 3.2: Abstraction of geometric object (flowers) [4]

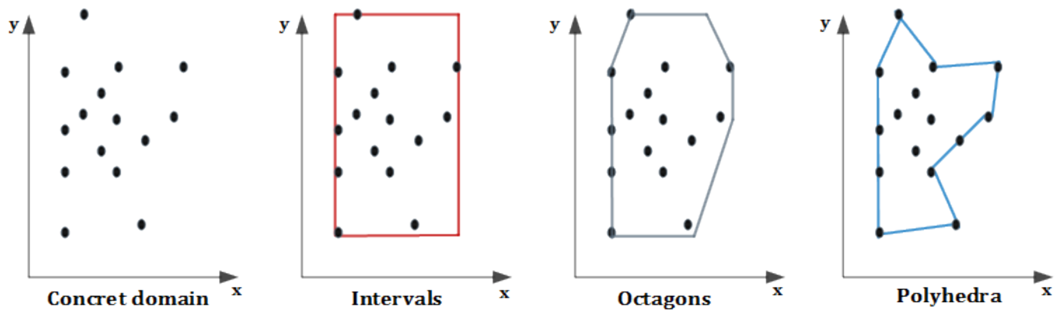


FIGURE 3.3: Numerical abstraction using different domains

domain which is precise enough in the extent that it can express the desired properties with a low-cost implementation.

In theoretical computer science, Patrick and Radhia Cousot have introduced the abstract interpretation theory for abstracting mathematical structures, involved in the formal models of computer systems [91]. The purpose of this theory is to build a sound approximation of the behaviors of complex computer systems, which is used to facilitate the reasoning on and the verification of their behavioral properties (non-termination, correct termination or with errors, etc.). Though it was primarily defined for static program analysis, abstract interpretation has been applied in other problems in computer science (model checking, database queries, malware detection)[92].

According to Patrick and Radhia Cousot, abstraction approximates, a possibly infinite, concrete ordered set $(C, \sqsubseteq_C)$ by an abstract set $(A, \sqsubseteq_A)$. The abstract set represents in fact an over-approximation of the concrete set and should be the tightest one in order to include less extra-elements. To ensure the link between the two sets, the abstraction and concretization functions are defined. It is actually very difficult and computationally expensive to find the exact function that encompasses exact values of the concrete set. The idea of abstraction is to approximate this set even if that might include extra-points. These latter are known as false alarms in code static analysis or extra-solutions in Constraint Problem Solving. The abstraction and concretization functions linking the two domains are defined as:

- Abstraction function:

$$\begin{array}{rccc} \alpha : & C & \rightarrow & A \\ & c & \mapsto & \alpha(c) = a \end{array}$$

- Concretization function:

$$\begin{array}{rccc} \gamma : & A & \rightarrow & C \\ & a & \mapsto & \gamma(a) = c \end{array}$$

In order to have a sound approximation, the abstraction and concretization functions must form a Galois correspondence. α and γ form a Galois connection if and only if, $\forall x \in C \; \forall y \in A : \alpha(x) \sqsubseteq_A y \iff x \sqsubseteq_C \gamma(y)$.

The choice of the suitable abstract domain is based on the structure of the analyzed system and the properties of interest. There exist different abstract domains that can be used to extract the system properties. Herein below, we give examples of commonly used numerical domains:

- **Signs**: If we drop all information about every system variable except whether it is positive, negative or zero, we get the abstract set $A = \{\bot, -, 0, +, \top\}$, where $\bot$ means that the variable has no value, $\top$ means that we do not know the sign of the variable. This abstract domain is non-relational, since it does not take the dependencies between variables into consideration.

- **Intervals**: The lattice of intervals is also a non-relational domain that represents the invariants of the form $x \in [c_1, c_2]$. Although very imprecise, non-relational domains are widespread for their simplicity. However, if the system variables depend on each other and we would like to increase the precision of constraints expressiveness these dependencies are taken into account. Then, we need relational abstract domains.

- **Pentagons**: a weakly relational abstract domain encoding the relations $x < y$.

- **Octagons**: a relational abstraction domain allowing the representation of conjunctions of the inequalities: $\pm\, x \,\pm\, y \,\leq\, k,\; k \in\, \mathbb{R}$.

3.3.2 Abstraction of driving data using interval domain

In this subsection, we propose an adaptation of the interval domain for driving data abstraction. We choose to use the interval domain because it is more adapted to the models that we will use later. Each collected variable is abstracted separately. For the sake of simplicity, we have ignored any correlation between the driving data (such as between acceleration and speed or pedal pressure and acceleration), so we can deal with non-relational domains.

Let $\mathcal{I}_{\bar{\mathbb{R}}} = \{\perp\} \cup \{[a, b[\mid a \in \mathbb{R} \cup \{-\infty\},\ b \in \mathbb{R} \cup \{+\infty\}\ and\ a < b\}$ the set of left-open intervals with bounds in $\mathbb{R}$ and where $\perp$ is the empty interval. The concrete domain of x is defined as the poset $C_x = \langle \bar{\mathbb{R}}, \subseteq \rangle$ where $\subseteq$ is the usual inclusion operator. Whereas, the abstract domain poset $A_x = \langle \mathcal{I}_{\bar{\mathbb{R}}}, \sqsubseteq \rangle$ defines the abstract domain equipped with the ordering relation $\sqsubseteq$. $\sqsubseteq$ is defined by $[a, b[\sqsubseteq [a', b'[\iff a \geq a'\ and\ b \leq b'$, and we define the two operators $\sqcap$ and $\sqcup$ by the following:

$$X \sqcap Y = \perp\ \boldsymbol{if}\ Y = \perp\ \boldsymbol{or}\ X = \perp$$

$$X \sqcap Y = \begin{cases} [max(a, a'), min(b, b')[& \boldsymbol{if}\ X = [a, b[\ \boldsymbol{and}\ Y = [a', b'[\\ & \boldsymbol{and}\ max(a, a') \leq min(b, b') \\ \perp & \boldsymbol{otherwise} \end{cases}$$

$$X \sqcup Y = \begin{cases} [min(a, a'), max(b, b')[& \boldsymbol{if}\ X = [a, b[\ \boldsymbol{and}\ Y = [a', b'[\\ X & \boldsymbol{if}\ Y = \perp \\ Y & \boldsymbol{if}\ X = \perp \end{cases}$$

In this interval abstract domain, a set c of reals is approximated with the interval $[a, b[$ where $a = min(c)$ and $b = max(c)$. The abstraction function α_x is thus defined as:

$$\begin{array}{llll} \alpha_x : & \bar{\mathbb{R}} & \rightarrow & \mathcal{I}_{\bar{\mathbb{R}}} \\ & c & \mapsto & [min(c), max(c)[\end{array} \quad , if\ c \neq \emptyset$$

$$\alpha_x(\emptyset) = \perp$$

The concretization function γ_x is conversely defined as:

$$\gamma_x([a, b[) = \{y \in \mathbb{R} \mid a \leq y < b\} \quad and\ \gamma_x(\perp) = \emptyset.$$

It is easy to see that the functions (α_x, γ_x) form a Galois connection $\langle C_x, \subseteq \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle A_x, \sqsubseteq \rangle$, we can say then that A_x is a sound abstraction of C_x.

The definition of the suitable numerical intervals abstracting driving data depends on the application. Let us take for example the variable "Vehicle speed" denoted as *vel*,

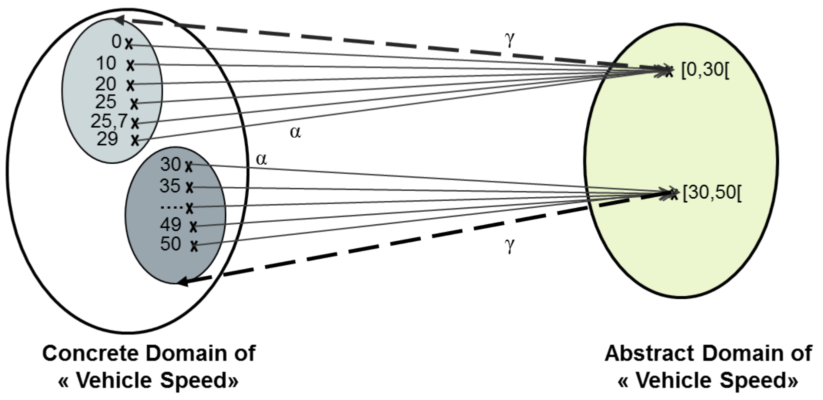

0
10
20
25
25,7
29
30
35
....
49
50
α
α
γ
γ
[0,30[
[30,50[
Concrete Domain of « Vehicle Speed»
Abstract Domain of « Vehicle Speed»

FIGURE 3.4: Abstraction using the interval domain, of the variable "Vehicle speed". (α is the abstraction function, γ is the concretization function)

which values may range from 0 to 240 km/h (the concrete domain); if an application to verify whether the driver respects or not speed limit panels is envisaged, *vel* can be abstracted based on the speed limit panels that exist in the traffic code. The French traffic code defines 6 speed limit panels [93] : 30, 50, 70, 90, 110, 130. *vel* can therefore be abstracted by intervals from the set $A_{vel} = \{[0, 30[, [30, 50[, [50, 70[, [70, 90[, [90, 110[, [110, 130[, [130, 240[, [240, +\infty)\}$. Explicitly, all values of vehicle speed ranging from 0 to 30 will be represented as one abstract value that is the interval [0,30[thus ignoring the speed changes between 0 and 30 and recording only changes between the significant values for the considered application (abstract values). Using functions this can be written as $\forall y \in \mathbb{R}$ such that $0 \le y < 30, \alpha_{vel}(y) = [0, 30[$ while $\gamma_{vel}([0, 30[) = \{y \in \mathbb{R} | 0 \le y < 30\}$. This abstraction is shown graphically in Figure 3.4.

Actually, the example as well as the applications considered here use a static abstraction, which means that the intervals of the abstract domain are set in advance. However, a dynamic approach can be considered where machine learning algorithms are used to retrieve the most suitable intervals from available data. An example of the methodology for dynamic data abstraction using the k-means algorithm is presented in the next paragraph.

3.3.3 Dynamic abstraction of driving data using k-means

k-means is a popular clustering method commonly used to automatically partition a dataset into k groups [94]. Its popularity is presumably due to its speed and its relative simplicity of deployment. Basically, the k-means algorithm takes as input a dataset $X = X_1, X_2, \ldots, X_N$ of N points and an integer k (i.e. the number of clusters to be generated). At the end, the algorithm generates k cluster centroids (i.e. used to define the clusters) and assign each data point to a unique cluster. The clustering is performed

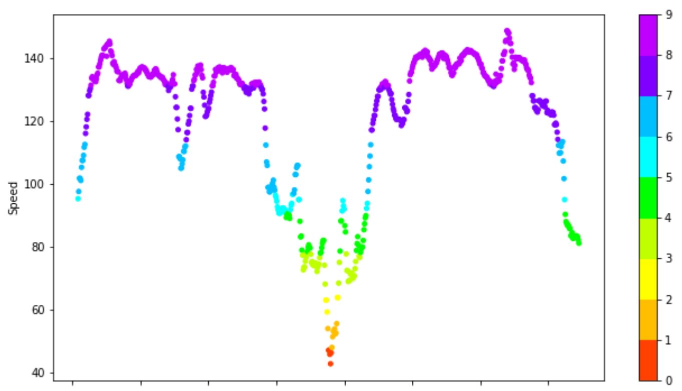


FIGURE 3.5: Dynamic abstraction of speed by means of the k-means algorithm. The number of clusters generated is $k = 9$.

such that the distance between all points and the centroid within a cluster is shorter than any other cluster centroid. This condition is generally expressed by the following expression, where C_i, and μ_i are used to denote respectively a cluster i (among the k clusters) and its centroid:

$$\text{Minimize} \sum_{i=1}^{k} \sum_{x \in C_k} ||x - \mu_i||^2 \text{ with respect to } C_i, \mu_i$$

In our context, each cluster will be represented by an interval defined by the min and max values within the cluster. In this way, the numerical abstraction of a driving variable consists of a segmentation of its values into k disjoint intervals, based on the k-means algorithm. The number of intervals k is estimated from the available data. As we presented in the static approach, each variable from the driving dataset is segmented separately. For more significant reduction of data, principal component analysis (PCA) is carried out to select the most relevant and uncorrelated variables. Figure 3.5 shows an example of segmentation of the variable speed measured for a driver D1 over a trip T, taken from the public dataset UAH-DriveSet[1][95]. The application of k-means results in the generation of 9 clusters of the speed values. In different colors are illustrated the different intervals to be used for speed abstraction. Thus, the variable speed can be abstracted by intervals from the set $A_{speed} = \{[0, 47.1[, [47.1, 56.2[, [56.2, 65.7[, [65.7, 78.0[, [78.0, 90.6[, [90.6, 97.0[, [97.0, 113.7[, [113.7, 131.3[, [131.3, 148.8[, [148.8, 255[, [255, \infty)\}$.

[1]available at http://www.robesafe.uah.es/personal/eduardo.romera/uah-driveset/

An other interesting idea may be to apply the k-means clustering on all driving variables instead of a separate clustering. This is expected to reduce significantly the size of the data.

To summarize, we can say that the dynamic approach for abstraction is data-adapted, which allow to create abstract domains that fit better to each dataset. On the other hand, the static approach is more adapted in specific applications, *i.e. such as analyzing the compliance of the driver with traffic code (as mentioned in the example in the previous paragraph), by fixing the intervals manually it will be easier to verify whether the driver is in authorized/unauthorized state*, and when dealing with different datasets to create a uniform space of the abstract states.

3.4 Proposed modeling formalisms

3.4.1 Probabilistic Rectangular Hybrid Input Output Automata (PRHIOA)

PRHIOA is an adaptation of hybrid IO automata [96] and rectangular automata [97]. It is based on the formalism of rectangular automata for which reachability problem is proved to be decidable[98].

Formally, a PRHIOA is defined as a tuple $(H,U,Y,X,Q,\Theta,inv,E,I,O,D,\mu)$, where:

- H is a set of internal variables, U is a set of input variables and Y is a set of output variables. H, U and Y are disjoint from each other and we write $X \triangleq H \cup U \cup Y$.
- $Q \subseteq val(H)$ is a set of states, where $val(H)$ is the set of valuations of H.
- Θ is a set of initial states.
- $inv : Q \to Rect(H)$ is an invariant function, where $Rect(H)$ is the set of all rectangular predicates over H. A rectangular predicate ϕ over H is a conjunction of rectangular inequalities; it defines the set of vectors $[\![\phi]\!] = \{z \in \mathbb{R}^n \mid \phi\ [H := z]\ \textit{is true}\}$. A rectangular inequality over H is a formula $h_i \sim c$, where $h_i \in H$, c is an integer constant and $\sim$ is one of $<, \leq, >, \geq$. The function inv maps each state to its invariant condition.
- E, I and O are sets of internal, input and output actions, respectively. An internal action of E will be denoted later by "?".
- D is a set of discrete transitions. A discrete transition is labeled with an action, and is defined as a triple (q, o, g, q') where q is a source state, o is an action,

$g \in Rect(H)$ is a guard on the transition, and q' is a target state. To simplify, if the guard is true we will only refer to a transition as a triple (q, o, q').

- μ is a transition probability over output actions O is defined such as: $\sum_{q' \in Q} \sum_{a \in O} \mu(q, a, q') = 1$ for all $q \in Q$ and all $a \in O$.

3.4.2 Attributed Directed Graphs

An *attributed directed graph* G [99][100] is formally defined as a tuple (V, E, μ, δ) where:

- V is a finite set of vertices.
- $E \subseteq V \times V$ a set of directed edges. An edge $e = (s, d)$ is an arrow defined by the pair from the source vertex $s \in V$ to $d \in V$ the destination vertex.
- $\mu : V \rightarrow L_V$ is a vertex labeling function that associates to each vertex from V labels from the set L_V.
- $\delta : E \rightarrow L_E$ is an edge labeling function that associates to each edge from E labels from L_E.

L_V and L_E represent respectively the sets of vertex-labels and edge-labels, where labels are tuples of fixed-size.

3.4.3 Learning Automata (LA)

Learning Automata is one of the most powerful tools in the field of adaptive learning. The term was first introduced in the survey paper by Narendra and Thathachar [101], and was since then used in a wide range of applications including intelligent vehicle control [102] and driver behavior [46]. It is defined as a finite state machine that interacts with a stochastic environment trying to learn the optimal action offered by the environment, via a learning process [103]. The stochastic automaton attempts to solve the learning problem without any a priori information on the optimal action. The learning process consists of learning the optimal action through repeated interactions with the environment. The actions are chosen based on a probability distribution over the action set, which is updated at each instant based on the reinforcement feedback from the environment.

A stochastic learning automata is generally defined by ϕ, α, β, $F(.,.)$, $H(.,.)$, [104] where:

- $\phi = \phi_1, \phi_2, ..., \phi_s$ is a finite set of internal states.
- $\alpha = \alpha_1, \alpha_2, ..., \alpha_r$ is a finite set of output actions.
- $\beta = \beta_1, \beta_2, ..., \beta_m$ is a finite/ infinite set of input actions.
- $F(.,.)$ is a function that maps the current state and input into the next state.
- $H(.,.)$ is a function that maps the current state and input into the current output.

The environment in which the automata operates is modeled as a triple α, c , β , where:

- α represents a finite action/output set.
- β represents an input/response set.
- c is a set of penalty probabilities (an element c_i corresponds to action $\alpha_i \in \alpha$)

The learning process works as follows. At each time step $t = n$, the automaton selects an action $\alpha(n)$ from the set of output actions α (based on the current probability vector). The application of $\alpha(n)$ on the environment generates $\beta(n)$ as the response of the environment. Based on the value of $\beta(n)$), c the set of penalty probabilities is updated, and the next state of the automata $\phi(n + 1)$ is calculated. The procedure is repeated until the optimal action α_m of the environment is found. An optimal action is defined as an action that has the maximum probability of being rewarded.

3.5 Driving environment representation

An important challenge of driving behavior modeling, is how to represent the relationships between driver actions and driving environment. In contrast to traditional approaches that record information about the driving environment as videos, we propose a representation of the driving environment requirement as constraints on the vehicle state, determined using a set of driving variables. Using such a representation (i.e constraints) will facilitate the verification of the driving conformity with the requirement of the driving environment. This representation is implemented as *environment interpreter* module in the driving behavior recording system (Figure 2.9). Though the inputs to the *environment interpreter* module can be any signal from environment; we focus here on traffic regulations because of their relevance in the assessment of driving safety. The following representation is based on the French Highway Code.

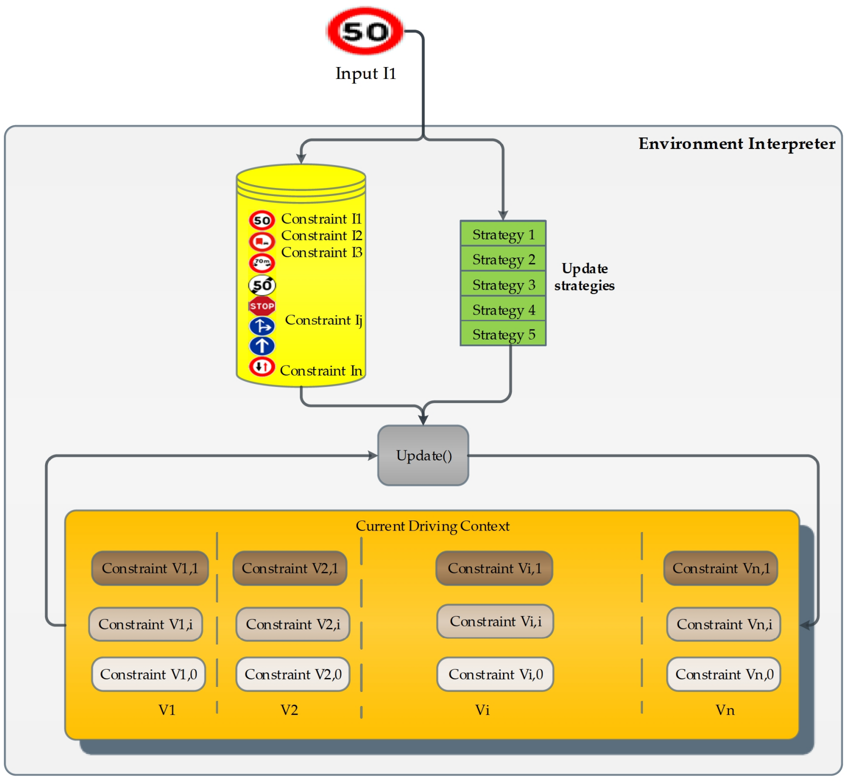


FIGURE 3.6: Architecture proposed for the "Environment interpreter" module

3.5.1 About the French Highway Code

The French Highway Code is characterized by:

- Regulations are either implicit (general requirements) or explicit (road signs, indications...)
- In the absence of explicit rules, implicit rules must be respected.
- Implicit rules depend on the road traffic class. There are two main road classes: in urban area and outside the urban area. Driving outside the urban area may in turn be divided into three more subclasses according to the road type: Express-way, Highway and Free-way.
- French Road signs are classified into four categories:
  - Prescription signs: inform drivers of the specific prohibitions and obligations supplementing the road rules.
  - Danger signs: attract the attention of drivers to places where their must increase their vigilance due to the presence of obstacles or hazards.

  - Indication signs: provide drivers with useful information about lanes use or certain changes.
  - Temporary signs: inform driver about temporary dangers and obstacles.
- French road signs may be combined with tab signs placed below the signs, which clarify or complement their significance. There are different types of tabs signs, where the most relevant to this work are:
  - Distance panels indicate distance between the road sign and its zone of application.
  - Length panels indicate the length of the zone of application of the road sign
- Road signs encountered on a road are no longer applicable after an intersection. The applicable traffic regulations will be brought to the notice of drivers by signs implanted after the intersection.

3.5.2 French traffic signs modeling

The architecture of the environment interpreter is presented in Figure 3.6. This work focuses primarily on some relevant prescription signs. As mentioned earlier, these signs are represented as conditions on the driving variables that were used to define vehicle states. While driving, it is obvious that we cannot have more than one traffic rule that impose two contradictory conditions on the same variable. For example, we cannot encounter, on the same road segment, a sign allowing left turn and a sign that prohibit it. We then perform a classification of road signs according to the variables on which they impose constraints. This classification is presented in Figure 3.7. The end of restrictions signs cancels the conditions already prescribed; those signs are modeled differently, as an update of the driving conditions by removing the indication of the sign.

The traffic signs are modeled by defining a mapping that associates to each of the signs a constraint on a system variable. For example, the sign **AB4** (stop) is associated to the constraint on velocity (V_{max}) indicating that the maximum speed allowed is 0, the sign **B17** (min distance allowed is 70m) is associated to the constraint on the following distance ($D_{min} = 70$) indicating that the minimum following distance that has to be respected 70. Constraints of other inputs are defined in the same way.

The driving context is reconstructed by representing contextual information as a set of stacks, which contain the constraints that have to be respected, where each stack is associated to one variable. For example if a speed limit sign input is received a constraint on the variable velocity V is added to the stack of V constraints. Due to the dynamic nature of driving, strategies for the update of stacks describing the driving context must

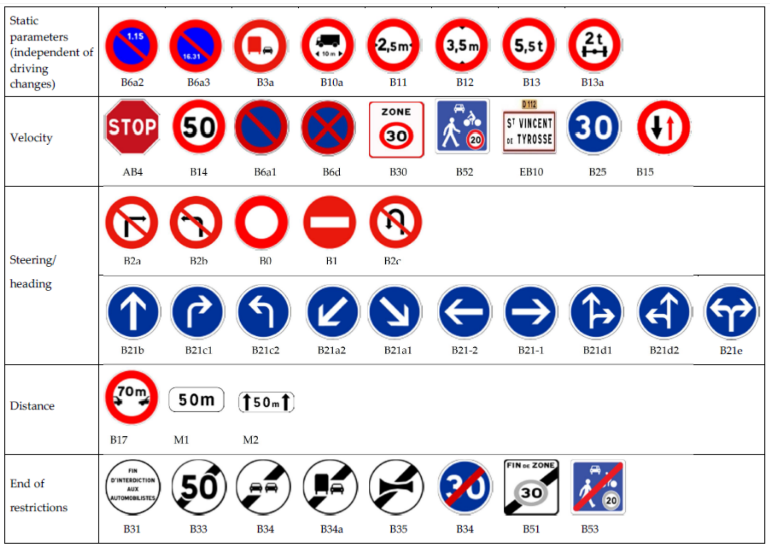

Static parameters (independent of driving changes)
B6a2
B6a3
B3a
B10a
B11
B12
B13
B13a
Velocity
AB4
B14
B6a1
B6d
B30
B52
EB10
B25
B15
Steering/ heading
B2a
B2b
B0
B1
B2c
B21b
B21c1
B21c2
B21a2
B21a1
B21-2
B21-1
B21d1
B21d2
B21e
Distance
B17
M1
M2
End of restrictions
B31
B33
B34
B34a
B35
B34
B51
B53

FIGURE 3.7: Classification of prescription signs according to their semantic

be defined, based the characteristics of the Highway Code. The driving context consists initially of the implicit requirement of the road, and is changing continuously over time while the vehicle is moving. As this context (especially road signs) depends on the effects of the inputs and their period of applicability, a classification of the different inputs according based on these two parameters (sign effect and period of applicability) is proposed. Strategies for stacks management are then defined and associated to each of the defined class. We propose five classes and eventually five strategies, which are described in the following:

- ***Class 1***: for inputs that reset the driving context to its initial state (i.e. intersection, leaving the urban area...)
- ***Class 2***: for inputs that are no longer applicable after a certain period of time (i.e. danger sign...)
- ***Class 3***: for inputs accompanied with one of the two tab signs; distance panel and length panel
- ***Class 4***: for inputs that override the last received input (i.e. speed limit sign...)
- ***Class 5***: for inputs that cancel the last received input (i.e. end of a prescription...)

For better clarification, we use in Fig 3.8 automata representation to describe the update strategies. If a received input I, imposing constraints on a variable A, belongs to:

- ***Class 1*** then the stack of A is initialized

- ***Class 2*** then I is added to the stack, and removed at the expiration of time T computed based on the distance of applicability of I and the velocity.
- ***Class 3*** then I is added to the stack after T time computed based on the indicated distance if accompanied with a distance panel, and/or removed from the stack after T if accompanied with a length panel.
- ***Class 4*** then the last input is removed from the stack before adding I.
- ***Class 5*** then the previous input is removed.

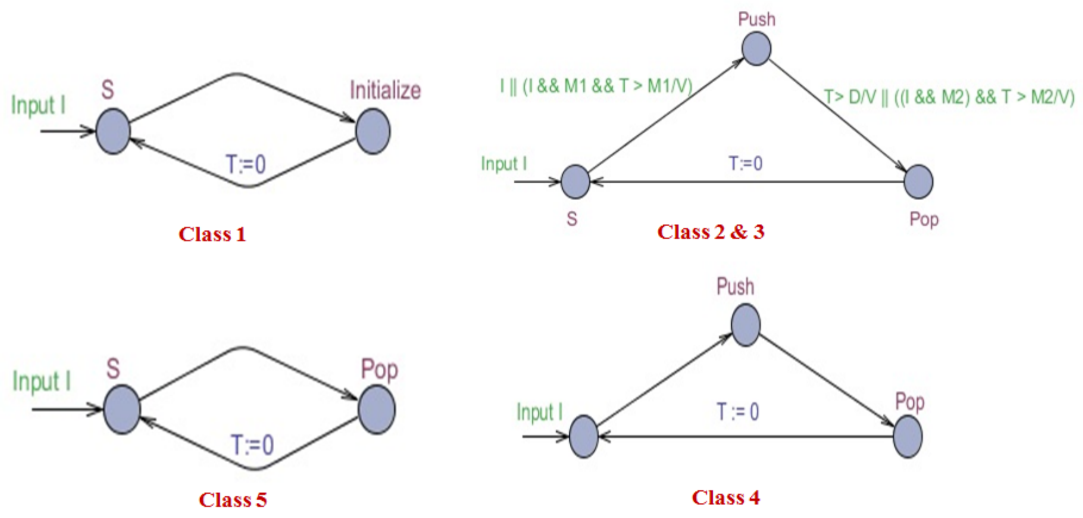


FIGURE 3.8: Update of driving context automaton based on the input class

3.6 Driving behavior modeling

3.6.1 Representation using Rectangular Hybrid Input Output Automata

Driving behavior is represented by the stochastic automaton as follows:

- ***Internal variables***: are driving data related to the state of the vehicle such as *speed*, *acceleration*, *steering angle*... To keep track of the driving environment in each state, a variable referring to the driving context *cxt* is considered. This latter is represented as a set of conditions on the driving data. It consists initially of the implicit requirement of the road (such as the authorized maximal and minimal speed on highways), and is updated continuously due to the dynamic nature of the driving environment.
- ***States***: correspond to the possible valuations of driving data.

- ***Invariants***: A state invariant is defined as a conjunction of conditions over the internal variables. A condition over a variable x takes the form of an interval $[a, b[$ from the abstract domain of x.

- ***Input actions***: represent contextual information about the driving environment. They consist of the real time information about traffic situations received either through in-vehicle sensors or through vehicular communication device. Examples of inputs include prescription road signs (like speed limitation), and warnings of coming vehicles. The inputs are used to construct and update the driving context *cxt*. The update follows a predetermined rules that were designed depending on the nature of the different inputs.

- ***Output actions***: model the response of the driver to the requirement of the driving environment. They represent the driver's actions related to the driving tasks. For instance, accelerating, turning and stopping are modeled as outputs in the automaton model.

- ***Transitions***: we distinguish two kinds of transitions; input-enabled transitions that occur whenever an input is received from the environment and output transitions enabled by the driver operation without an explicit input.

The automaton of the driver behavior is either constructed offline using recorded driving data, or online when sensors measurement are still streaming. As regards the number of the automaton states, it depends on the cardinalities of the variables' abstract domains, elements of which are used for the definition of the invariants. As mentioned earlier in this chapter, the construction is a machine learning algorithm, based on learning automata [105], that learns and updates the probabilities of the transitions between states according to the interactions of the driver with the environment. The pseudo code of the algorithm is depicted in Algorithm.1. The algorithm runs over the available driving data: transitions are added and their probabilities are updated whenever input or output actions occur, using the following reinforcement scheme:

$$P(T) = \begin{cases} 1 & \text{if } r = 0 \\ P(T) + \frac{1-P(T)}{r} & \text{if } r \neq 0 \end{cases} \tag{3.1}$$

$$P(T') = P(T') - \frac{P(T')}{r},\ for\ all\ T' \neq T \tag{3.2}$$

Where $P(T)$ is the probability of the reinforced transition $T \in D$ outgoing from state q, and r is the number of all transitions T' outgoing from q.

Figure 3.9 illustrates an example of driving behavior automaton. The variables velocity, denoted vel, and steering wheel angle, denoted θ, are considered as internal variable

Algorithm 1 The algorithm of the automaton-based model construction

```
Require: $Q$: states , $I$: input action set, $O$: output action set, $q_{initial}$: initial state,
  $Data$: driving data set
Ensure: $D$: transition set, $P$ : transition probabilities
  $q_{current} = q_{previous} = q_{initial}$
  while Data do
    convert $H \in Data$ to its corresponding abstract value $\alpha(H)$
    pick a state $q$ from $Q$ such that $inv(q) = \alpha(H)$
    $q_{current} = q$
    if (input action $I_j \in I$ exists in $Data$) then
      $q_{previous} = q_{current}$
      Update current driving context $cxt$ in $q_{current}$: $q_{current}.cxt = cxt_j$
      if $(q_{previous}, q_{current}, I_j) \notin D$ then
        Add input-enabled transition $T$ to $D$
      end if
      Update transitions probabilities using (3.1) and (3.2)
    end if
    if (output action $O_j \in O$ exists in $Data$) then
      if $(q_{previous}, q_{current}, O_j) \notin D$ then
        Add output transition $T$ to $D$
      end if
      $q_{previous} = q_{current}$
      Update transitions probabilities using (3.1) and (3.2)
    end if
  end while
```

and we add the variable cxt representing the driving context. cxt has been initialized to C_0, the implicit requirement of the driving environment. The arrows of the automaton labeled with numbers represents transitions initiated by driver actions where the numbers represent the probabilities of their occurrence computed during the learning process. Whereas the other arrows represent input-enabled transitions initiated by the environment inputs and which affect the context cxt.

The automaton depicts some usual behaviors of the driver: for instance, we can say that usually in state S_0 the driver either increases the car velocity or both the velocity and the steering angle, each with a probability of 1/2. Also in state S_4 the driver often does not respect the restriction on velocity (a probability of 3/4), the probability he respects this restriction is only 1/4. Another behavior is when restriction on θ (e.g when turning or overtaking is prohibited) ended, the driver often (probability of 7/8) change both the velocity and the steering wheel.

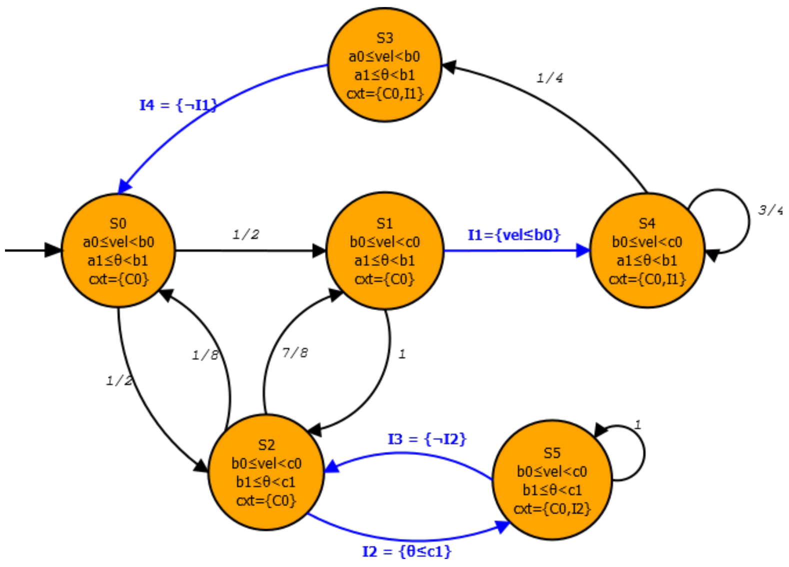


FIGURE 3.9: Example of automata-based representation of driving behavior

3.6.2 Representation using attributed directed graphs

Similarly to the automaton-based modeling approach, a graph of driving behavior consists of:

- *Vertices* : represent the dynamic states of the vehicle.
- *Edges* : represent the transitions between the states of the vehicle.
- *Vertex-labeling function*: Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of driving data, and L_{x_i} be the set of labels of x_i where ϵ_{x_i} is a labeling function for variable x_i defined as:

$$\begin{array}{rccc} \epsilon_{x_i}: & I_{\bar{\mathcal{R}}} & \to & L_{x_i} \\ & [a_i, b_i[& \mapsto & l_{i,x_i} \end{array}$$

  The vertex-labeling function μ is defined as:

$$\begin{array}{rccc} \mu: & V & \to & L_V = L_{x_1} \times \ldots \times L_{x_i} \times \ldots \times L_{x_n} \\ & V_i & \mapsto & l_v \end{array}$$

- *Edge-labeling function*: label each e with a couple $(action, weight)$, where *action* refers to a driver action and $weight \in \mathbb{R}_+^*$ represents the occurrence of *action*. The edge-labeling function is defined as:

$$\begin{array}{rccc} \gamma: & E & \to & L_E \\ & e = (s, d) & \mapsto & l_e = (action, weight) \end{array}$$

The algorithm of the graph construction runs over driving data, its pseudo code is presented in algorithm.2. At each timestamp, the data read is mapped to a label from L_v, edges are added and weights updated. The resulting graph will thus be a personalized representation of the driver behavior.

Algorithm 2 Algorithm of graph construction

```
Require: $V$: A set of labeled vertices, $V_0$: initial state, $Data$: driving data set.
Ensure: $G$: A graph
  $V_i = V_0$, $action =' none'$
  while $Data$ do
    map $H \in Data$ to the corresponding label $lbl = \mu(H)$
    while $lbl = \mu(V_i)$ do
      wait
    end while
    Pick a vertex $V_j$ with $\mu(V_j) = lbl$
    create an edge $e = (V_i, V_j)$
    if driver action $a_i$ known then
      add $a_i$ to the label of $e$: $\delta(e).action = a_i$
    end if
    if $e \in E$ then
      $\delta(e).weight = \delta(e).weight + 1$
    else
      add edge $e$ to $E$
      $\delta(e).weight = 1$
    end if
    $V_i = V_j$
  end while
```

Figure 3.10 depicts an example of driving behavior graph, representing the same driver in Figure 3.9. Velocity (vel) and steering wheel angle θ were considered for the representation of the vehicle states, and their values were used to label the vertices. Considering A_{vel} and A_θ, the abstract domains of vel and θ defined as described in the previous section. Defining labels of vel consists of assigning to each interval $[a_i, b_i[$ in A_{vel} an integer or string label $l_{i,vel}$. $l_{0,vel}$ and $l_{0,\theta}$ for instance refer to the labels assigned to the first interval in A_{vel} and A_θ respectively. For driving context, the label is defined using the notations assigned to the received inputs. $C_0 I_1$ in the label of V_3 means that the driving environment consists of the implicit requirement of the road and the input I_1. In comparison with the automaton in Figure 3.9, the labels of the graph vertices replace the invariants of states, while the edge weights replace the probabilities of transitions. Driver actions, if known, are added to the labels of the edges.

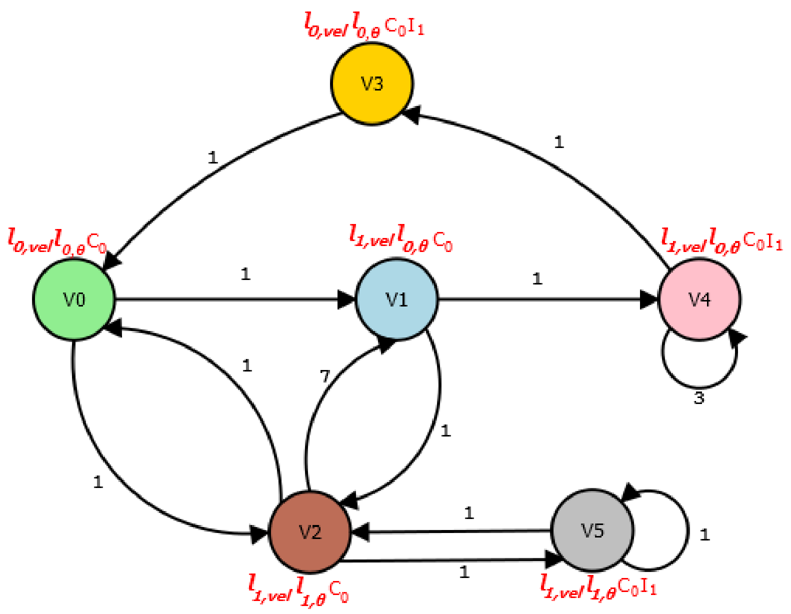

$l_{0,vel} l_{0,\theta} C_0 I_1$
V3
1
1
$l_{0,vel} l_{0,\theta} C_0$
$l_{1,vel} l_{0,\theta} C_0$
$l_{1,vel} l_{0,\theta} C_0 I_1$
V0
1
V1
1
V4
1
7
3
1
1
1
V2
V5
1
1
$l_{1,vel} l_{1,\theta} C_0$
$l_{1,vel} l_{1,\theta} C_0 I_1$

FIGURE 3.10: Example of graph-based representation of driving behavior

3.7 Relationship between driving data abstraction and future application performance

An important issue relevant to performance that should be considered is the relationship between the model size (number of states) and the defined abstraction. This was carried out by conducting an experimentation using sample data from the signal corpus collected by *the behavior signal processing laboratory of Nagoya University* [106]. The corpus uses different kind of sensors to capture real data relevant to the driver, the vehicle and the environment:

- Driver: Driver speech is recorded using microphones, biological signals such as *heart rate*, *skin potential* and *perspiration* are also recorded.
- Vehicle: Recorded driving signals consist of *brake and gas pedal pressure*, *steering wheel angle*, *vehicle speed*, *following distance* and *3D accelerations*. The sampling frequency is 16kHz.
- Environment: Cameras are used to capture the road ahead, GPS positions.

Figure 3.11 depicts the measured values of driving signals of one driver (female) during one trip (7 minutes). Numerical abstraction of driving signals can be intuitively seen as a discretization of the signals. We define 5 different abstractions, varying each time the precision with which we abstract the data. The precision is represented by the length of the intervals used in the abstraction. Table 3.2 presents the intervals considered

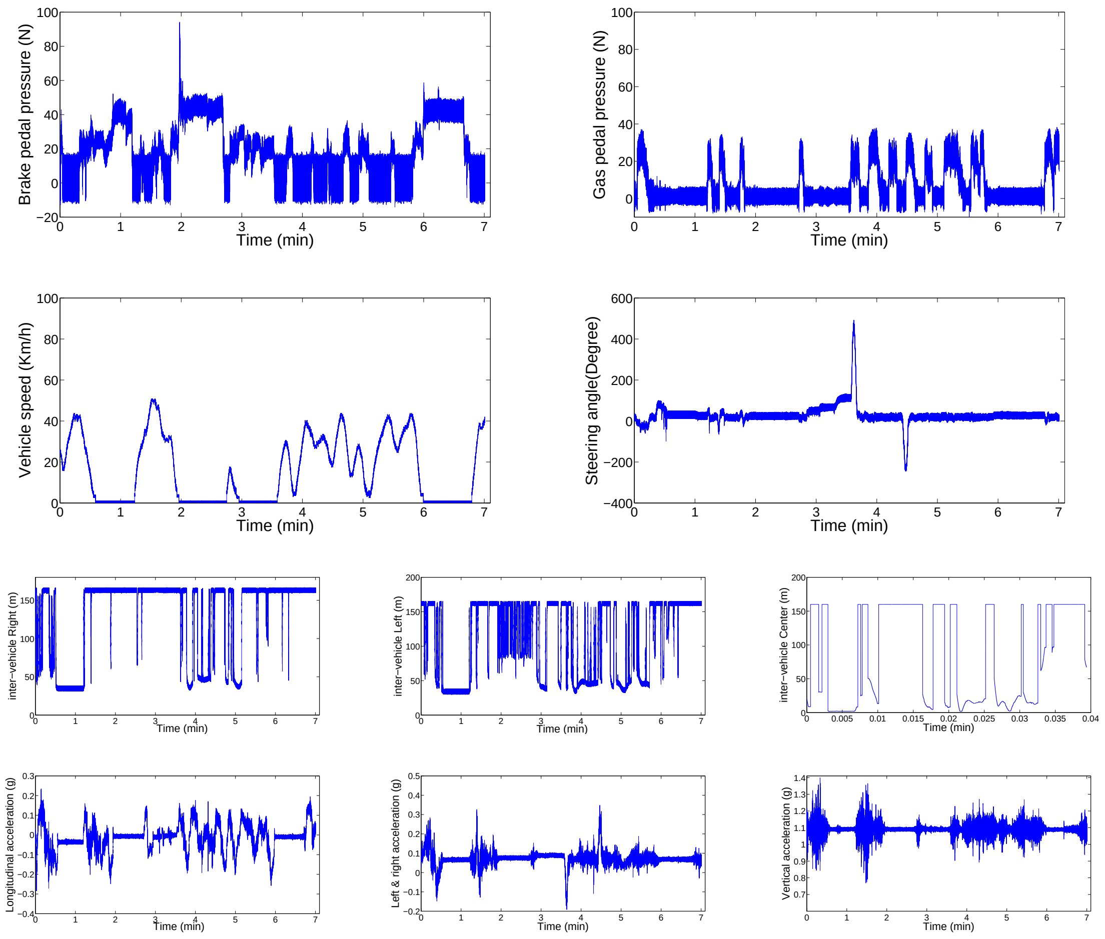


FIGURE 3.11: Distributions of driving signals of one female driver for a 7 minutes drive. The sampling frequency is 16 kHz. The data is provided by CIAIR, Nagoya, Japan (refer to [5] for details)

for the five abstractions defined for the driving signals used in the experiment. The intervals bounds were defined manually by analyzing the different distributions of data. The lengths of intervals vary according to which signal variations the abstraction should capture.

The model state space is generated for each abstraction separately. Figure 3.12 compares the size of the model generated for 1-minute drive (model size), using the different abstractions. It shows also the relation between the model size and the time for its generation (processing time). As expected, an abstraction that is very precise (defined using narrow intervals) increases the size of the model and the processing time. Indeed, abstraction 'A1', the closest abstraction to real measurements has the largest model size (6635520 states) and thus a longer processing time. Whereas the abstraction with lower precision (defined using wider intervals) ('A2'), has smaller model size and shorter processing time. Therefore, in order to make an abstraction better in terms of performance, a compromise between precision and processing time should be found.

Table 3.2: Driving data abstractions

Driving Signals	**Abstractions**				
	Abstraction 1 (A1)	**Abstraction 2 (A2)**	**Abstraction 3 (A3)**	**Abstraction 4 (A4)**	**Abstraction 5 (A5)**
Brake Pedal Pressure (BP)	$\{[-20, 0[,[0, 20[,[20, 40[, [40, 60[,[60, 80[,[80, 100[, [100, \infty[\}$	$\{[-20, 20[,[20, 60[,[60, 100[, [100, \infty[\}$	$\{[-20, 0[,[0, 20[,[20, 40[, [40, 100[,[100, \infty[\}$	$\{[-20, 0[,[0, 40[,[40, 100[, [100, \infty[\}$	$\{[-20, 20[,[20, 50[,[50, 100[, [100, \infty[\}$
Gas Pedal Pressure (GP)	$\{[-10, 0[,[0, 10[,[10, 20[, [20, 30[,[30, 40[,[40, \infty[\}$	$\{[-10, 10[,[10, 30[,[30, 50[, [50, \infty[\}$	$\{[-10, 0[,[0, 10[,[10, 20[, [20, 30[,[30, 40[,[40, \infty[\}$	$\{[-10, 0[,[0, 20[,[20, 40[, [40, \infty[\}$	$\{[-10, 10[,[10, 30[,[30, 40[, [40, \infty[\}$
Vehicle speed (Vel)	$\{[0, 20[,[20, 40[,[40, 60[, [60, \infty[\}$	$\{[0, 30[,[30, 60[,[60, \infty[\}$	$\{[0, 20[,[20, 40[,[40, 60[, [60, \infty[\}$	$\{[0, 30[,[30, 60[,[60, \infty[\}$	$\{[0, 30[,[30, 60[[60, \infty[\}$
Steering wheel angle (θ)	$\{[-300, -100[,[-100, 100[, [100, 300[,[300, 500[,[500, \infty[\}$	$\{[-300, 100[,[100, 500[, [500, \infty[\}$	$\{[-300, -100[,[-100, 100[, [100, 300[,[300, 500[,[500, \infty[\}$	$\{[-300, -100[,[-100, 100[, [100, 500[,[500, \infty[\}$	$\{[-300, -50[,[-50, 50[, [50, 500[,[500, \infty[\}$
Right front distance (d_{rf})	$\{[0, 50[,[50, 100[,[100, 150[, [150, 200[,[200, \infty[\}$	$\{[0, 100[,[100, 200[,[200, \infty[\}$	$\{[0, 50[,[50, 100[,[100, 200[, [200, \infty[\}$	$\{[0, 50[,[50, 170[,[170, \infty[\}$	$\{[20, 40[,[40, 160[,[160, 200[, [200, \infty[\}$
Left front distance(d_{lf})	$\{[0, 50[,[50, 100[,[100, 150[, [150, 200[,[200, \infty[\}$	$\{[0, 100[,[100, 200[,[200, \infty[\}$	$\{[0, 50[,[50, 100[,[100, 200[, [200, \infty[\}$	$\{[0, 50[,[50, 170[,[170, \infty[\}$	$\{[20, 40[,[40, 160[,[160, 200[, [200, \infty[\}$
Center distance (d_c)	$\{[0, 50[,[50, 100[,[100, 150[, [150, 200[,[200, \infty[\}$	$\{[0, 100[,[100, 200[, [200, \infty[\}$	$\{[0, 50[,[50, 100[,[100, 200[, [200, \infty[\}$	$\{[0, 50[,[50, 170[,[170, \infty[\}$	$\{[0, 40[,[40, 160[,[160, 200[, [200, \infty[\}$
Longitudinal acceleration (a_l)	$\{[-0.3, -0.2[,[-0.2, -0.1[, [-0.1, 0[,[0, 0.1[,[0.1, 0.2[, [0.2, 0.3[,[0.3, \infty[\}$	$\{[-0.3, -0.1[,[-0.1, 0.1[, [0.1, 0.3[,[0.3, \infty[\}$	$\{[-0.3, -0.2[,[-0.2, -0.1[, [-0.1, 0[,[0, 0.1[, [0.1, 0.2[,[0.2, 0.3[,[0.3, \infty[\}$	$\{[-0.3, -0.2[,[-0.2, 0[, [0, 0.2[,[0.2, 0.3[,[0.3, \infty[\}$	$\{[-0.3, -0.05[,[-0.05, 0[, [0, 0.2[,[0.2, 0.3[,[0.3, \infty[\}$
Directional acceleration (a_d)	$\{[-0.2, -0.1[,[-0.1, 0[, [0, 0.1[,[0.1, 0.2[,[0.2, 0.3[, [0.3, 0.4[,[0.4, \infty[\}$	$\{[-0.2, 0[, [0, 0.2[,[0.2, 0.4[,[0.4, \infty[\}$	$\{[-0.2, -0.1[,[-0.1, 0[, [0, 0.1[,[0.1, 0.2[,[0.2, 0.3[, [0.3, 0.4[,[0.4, \infty[\}$	$\{[-0.2, 0[,[0, 0.2[,[0.2, 0.4[, [0.4, \infty[\}$	$\{[-0.2, 0[,[0, 0.1[,[0.1, 0.4[, [0.4, \infty[\}$
Vertical acceleration (a_v)	$\{[0.7, 0.9[,[0.9, 1.1[, [1.1, 1.3[,[1.3, 1.5[,[1.5, \infty[\}$	$\{[0.7, 1.1[,[1.1, 1.4[,[1.4, \infty[\}$	$\{[0.7, 1.0[,[1.0, 1.3[,[1.3, 1.5[, [1.5, \infty[\}$	$\{[0.7, 1.0[,[1.0, 1.3[,[1.3, 1.5[, [1.5, \infty[\}$	$\{[0.7, 1.09[,[1.09, 1.13[, [1.13, 1.5[,[1.5, \infty[\}$

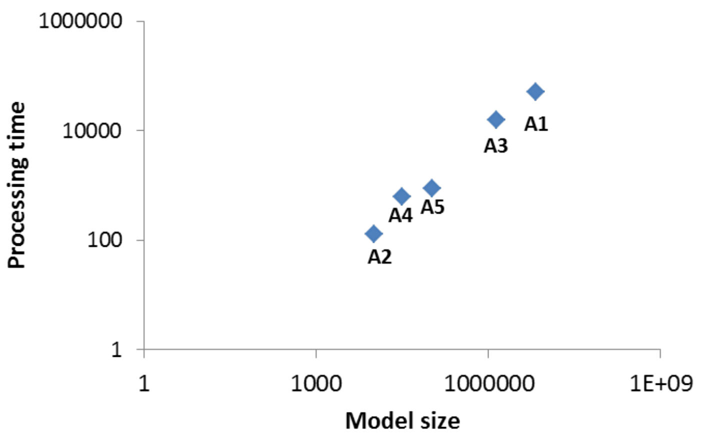


Figure 3.12: Relationships between abstraction and performance

3.8 Chapter summary

In this chapter, we started by giving an overview of the driver profiling trend and its relation with driving behavior analysis. The uniqueness in driver behavior been lately proved, models that display this uniqueness are useful in many applications. Modeling driving behavior is faced with many challenges which have been addressed throughout this chapter. First, the data size problem was addressed by numerical abstraction using intervals. In order to build personalized driver models, we used graphical models with machine learning to update transitions. An approach to model driving environment was also presented.

The complete approach for driver behavior modeling being presented, the next two chapters will be dedicated to two important analyses (1) drivers compliance with traffic rules and (2) drivers behavior similarity.

Chapter 4

Analysis of driving behavior safety: How safe is the driver?

"Get the habit of analysis - analysis will in time enable synthesis to become your habit of mind."

– Frank Lloyd Wright

Driver compliance with road rules, such as speed limits, prohibitions and right of way, is one of the key elements in traffic safety. The assessment of driver behavior conformity to these rules is important in order to distinguish good and bad drivers. To clarify, the behavior of good drivers complies most often with the road rules and requirements in every driving situation, contrast to bad drivers who tends to ignore most of the rules required by the Highway Code. This ability to assess drivers behavior can be of a great relevance either for traffic controllers, insurance companies, driving schools or for drivers themselves. In this chapter, we present an approach that enable automatic checking of whether the driver performs some defined behaviors. For this purpose, we consider Probabilistic Rectangular Hybrid I/O Automaton as driving behavior model and we use model checking to formally verify that it satisfies the properties representing the desired behaviors. The chapter starts by recalling basic concepts of formal verification, model checking and temporal logic. Then presents the methodology of verification of driver behavior using model checking. Finally, results of the verification are discussed.

4.1 Preliminaries

4.1.1 Formal Verification

In the field of computer science, formal methods refer to mathematically rigorous techniques and tools for the specification, development and verification of systems. Adoption of formal methods to problems in various areas (including communication and security protocols, distributed algorithms, source codes and hardware systems), throughout the recent years, has contributed to improving systems reliability and comprehensibility. Using rigorous specifications, formal methods promise a better and more precise reasoning about the behavior of a system. Moreover, they provide mathematical proofs of the correctness of the system behavior. This act of proving or disproving systems correctness is referred to as formal verification. The aim of formal verification is to provide a rigid way of evaluating the conformity of a system behavior to a certain formal specification or property. Typically, a formal verification problem is a problem of proving that a system meets certain specifications. In order to reason about the system mathematically, a model of the this latter, usually some variant of state-transition graph, is built. Specifications are also expressed as logical formulas $\mathcal{F}$. The verification problem is, thus, formalized as a satisfiability problem of the formulas $\mathcal{F}$ in the model $\mathcal{M}$.

There exist many approaches for formal verification which can be broadly classified into two methods: theorem proving and model checking. In theory, model checking consists of a systematic exhaustive exploration of the state space of a system. It is a model-based verification where algorithmic techniques for state space exploration are applied on a model of the system, to prove/disprove that certain correctness criteria holds in the model. The theorem proving approach to verification, on the other hand, require the system under consideration to be modeled in a rich formalism with axioms and inference rules. Correctness proofs are then derived from the axioms using inference rules. Formalizing the two problems: given a model $\mathcal{M}$ of a system $\mathcal{S}$ and a formula ϕ, model checking is the problem of verifying whether ϕ holds in $\mathcal{M}$; theorem proving, on the other hand, is the problem of verifying if ϕ holds in all models of the system $\mathcal{S}$.

4.1.2 Model Checking

Model checking is a verification technique, developed independently by Clarke and Emerson and by Queille and Sifakis in early 1980's, that provides a full automatic verification of finite systems correctness. Model checking is among the most successful approach for verification, its popularity has been boosted by two factors. First, the verification is fast and performed automatically, once the correct model and the required properties are established. Second, it is able to provide a counterexample when a property does not hold in some states; a characteristic that made model checking more useful for fixing systems flaws.

Verification by *model checking* can be defined as the process of proving exhaustively, given a model M ((e.g. Finite State Machine, Deterministic automaton, etc.) of a system whether this model satisfies a given specification (i.e. property) written in a certain logic formalism. Applying model checking involves using a model checking tool, which receives the model and properties as input; if the model checker then outputs a true answer if the model satisfies the specifications or otherwise generate a counterexample. Model checking can also referred to as property checking. There are typically two main classes of logical properties: safety and liveness properties. Safety properties are informally characterized as nothing bad should happen in the system. Typical safety properties include deadlock freedom and mutual exclusion. Liveness properties, on the other hand, assert that something good eventually happens. A typical example for a liveness property is the requirement that certain events occur infinitely often [107]. In the context of communication protocols, for example, ensuring that corrupted messages are never marked as a good one is a safety property, while requiring that a message is eventually transmitted is a liveness property.

Probabilistic model checking supports verification of systems with probabilistic and stochastic characteristics. The probabilistic behavior is generally modeled using random variables and probability distributions, such that transitions does not only reflect their existence but also indicate the likelihood of their occurrence. Specifications in probabilistic model checking are either quantitative properties or qualitative properties. Quantitative properties guarantee that the probability of an event meets given lower or upper bounds (ranging between 0 and 1) while qualitative properties assert that an event holds with probability 0 or 1 [108]. For instance, the requirement that the probability of successful message transmission is greater that 0.99, is a of quantitative property; examples of properties in conventional model checking such as deadlock freedom are considered qualitative properties. Probabilistic model checking is performed through a probabilistic model checker, using a probabilistic logic language to express the specifications.

4.1.3 Properties and temporal logic

One class of formal languages used to express specifications are temporal logics, whose use enable reasoning about system behavior over time. In the context of formal specification, temporal logic is generally used to represent propositions whose truth values

Table 4.1: A list of some temporal modal operators with their meaning

Operator	**Syntax**	**Semantic**
X	$X\ \phi$	*Next*: the property ϕ holds at the next state.
F	$F\ \phi$	*Eventually*: the property ϕ eventually holds (sometime in the future).
G	$G\ \phi$	*Globally*: the property ϕ always holds.
U	$\phi\ U\ \psi$	*Until*: the property ϕ holds until the property ψ holds.
R	$\phi\ R\ \psi$	*Release*: the property ψ is true until the first position where property ϕ is true.
A	$A\ \phi$	*All*: the property ϕ holds on all paths.
E	$E\ \phi$	*Exist*: the property ϕ holds on at least one path.

depend on time. Depending on the underlying nature of time, temporal logic systems can be classified as either linear time logic (LTL) in which the structure of time is linear or branching time logic (CTL) in which time has a treelike nature. In general, a linear temporal logic provides a useful set of temporal operators for expressing events along a single time path. In contrast to branching temporal logic where the temporal operators quantify over possible paths from a given state. The operators used in temporal logic can be distinguished into (i) logical operators of propositional logic (e.g. $\neg : NOT, \wedge : AND, \vee : OR, \rightarrow$: *imply*), and (ii) temporal modal operators, which are presented in Table4.1. LTL formulas are usually built using the temporal operators X, F, G, U. Formulas in CTL also used these temporal operators, but obligatory preceded by the branching operators A and E, also known as path quantifiers. An extension of CTL, called CTL*, allows a free use of temporal operators and path quantifiers is CTL*. That is, in CTL* the condition that path quantifiers have to precede temporal operators is not required. In this sense, CTL* is a superset of LTL and CTL, meaning that any LTL or CTL formula can be expressed in CTL*.

Probabilistic CTL is another extension of CTL proposed for use in systems that exhibit uncertainty. Assume given a denumerable set of *atomic propositions* denoted by $\mathcal{AP}$, which are boolean expression over states. The syntax of PCTL formulas can be defined by the following grammars:

$$\Phi ::= True \mid p \mid \neg\Phi \mid \Phi \wedge \Phi \mid P_{\sim\lambda}[\varphi]$$
$$\phi ::= X\Phi \mid \Phi\, U^{\leq n}\, \Phi$$

Where $p \in \mathcal{AP}$, $\sim \in \{<, >, \leq, \geq\}$ is a comparison operator, $\lambda \in [0,1]$ is a probability threshold and $n \in \mathbf{N} \cup \{\infty\}$.

Formulas ϕ, called *state formulas*, are evaluated over states, and formulas ϕ, called *path formulas*, are evaluated over paths. Path formulas are specified using the two operators,

X (*Next*) and $U^{\leq n}$ (*Bounded Until*). The formula $\phi_1 U^{\leq n} \phi_2$ states that ϕ_1 holds along the path until ϕ_2 holds within n steps. The *unbounded until* operator (U) can be obtained by considering $n = \infty$ ($\Phi \, U \, \Phi = \Phi \, U^{\leq \infty} \, \Phi$).

PCTL properties are usually interpreted over the states of a probabilistic transition system (some variant of discrete Markov chain), which is basically defined as a tuple $(S, s_i, \mathcal{P}, L)$, where S is a finite set of states, $s_i \in S$ is an initial state, $\mathcal{P} : S \times S \to [0, 1]$ is a transition probability function such that $\sum_{s' \in S} \mathcal{P}(s, s') = 1$, for all $s \in S$ and $L : S \to 2^{\mathcal{AP}}$ is a labeling that assigns to each state atomic propositions from a set $\mathcal{AP}$ [109]. Let $\mathcal{M} = (S, s_i, \mathcal{P}, L)$ be a Markov chain. The satisfaction relation $\models$ is defined for state and path formulas separately, and we write $\mathcal{M}, s \models \psi$ (resp. $\mathcal{M}, \pi \models \psi$) to indicate that state s (resp. path π) satisfies ψ where ψ is a state (resp. path) formula. The satisfaction relation for any state s is defined by:

- $\mathcal{M}, s \models True$ for all $s \in S$,
- $\mathcal{M}, s \models a$ iff $a \in L(s)$,
- $\mathcal{M}, s \models \neg\Phi$ iff $\mathcal{M}, s \not\models \Phi$,
- $\mathcal{M}, s \models \Phi_1 \wedge \Phi_2$ iff $\mathcal{M}, s \models \Phi_1$ and $\mathcal{M}, s \models \Phi_2$,
- $\mathcal{M}, s \models P_{\sim\lambda}(\phi)$ iff $Pr(\mathcal{M}, s \models \phi) \sim \lambda$.

For a path $\pi = s_0 s_1 \cdots$ in $\mathcal{M}$, the satisfaction relation is defined as:

- $\mathcal{M}, \pi \models X\Phi$ iff $s_1 \models \Phi$,
- $\mathcal{M}, \pi \models X\Phi \, U^{\leq n} \, \phi$ iff $\exists i \leq n . s_i \models \phi and \forall j < i . s_j \models \Phi$

4.2 Formal verification of human driver behavior

4.2.1 Verification methodology

In this section, we illustrate how probabilistic model checking is used to verify the safety of driver behavior. This methodology of model checking driver behavior is shown Figure 4.1. *Behavior specification* formalizes the desired driving behaviors (i.e. that driver should satisfy for a safe driving) as PCTL formulas. Unlike other qualitative logics (e.g LTL, CTL), PCTL allow us to perform quantitative (as well as qualitative) analysis by expressing properties such as "*can the driver reach a safe (or unsafe) state with a probability less than 30%*". These properties will be fed to the model checker, which

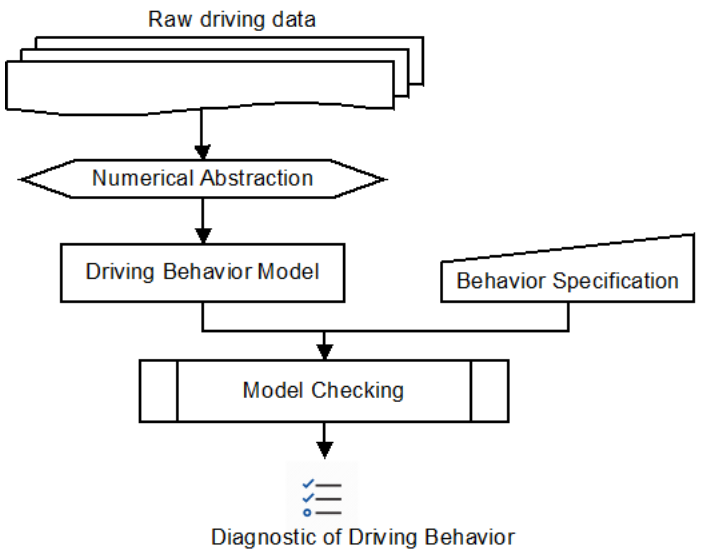


FIGURE 4.1: Driving behavior verification by model checking

in turn starts a model-checking algorithm for verifying the driver behavior model, previously generated from driving data. The verification was carried out using PRISM, an open-source tool that allows building and analyzing several types of probabilistic models: discrete and continuous-time Markov chains, Markov decision processes, probabilistic automata, probabilistic timed automata and extensions of these models with rewards [110].

In order to validate the driver behavior using PRISM, the driver behavior model must be translated to PRISM language, a textual language, based on guarded command notation. The following description of the PRISM language is taken from the manual section on the software website[1]. The fundamental components of the PRISM language are *modules* and *variables*. A PRISM model is composed of a number of modules, representing the components of the system been modeled, which can interact with each other. Each module contains a set of local variables that determine its local state, and a set of guarded commands that describe its behavior. The combination of modules local states determines the global state of the whole model. The commands describing modules behavior are of the form

$$[\,]\ guard \rightarrow prob_1 : update_1 + \ldots + prob_n : update_n$$

The *guard*, is a predicate over the variables of the model, and is associated with each command to determine when the command is applicable. $update_i$ and $prob_i$ indicates that a transition updating the values of the module's variables can occur with a probability $prob_i$ if the *guard* is true. For a detailed description of the structure of a PRISM

[1]http://www.prismmodelchecker.org/

code, refer to the PRISM manual.

4.2.2 Translation of driver behavior models into PRISM

As mentioned before, in order to perform model checking analysis on a driver behavior model using PRISM, we must translate it into the PRISM language. In the last chapter, we presented a probabilistic model of driver behavior where the states are defined by predicates over driving variables, and transitions are enriched with probabilities reflecting the frequency of their occurrence. This is mapped into a continuous-time Markov chain (CTMC) model in PRISM using a module with *local variables* representing driving variables and *commands* describing transitions. In order to translate the states predicates into PRISM language, we define for each driving variable V_i two local variables $V_{i_{min}}$ and $V_{i_{max}}$ representing the minimum and maximum values of intervals (the predicates). For instance, the variables vel_{min} and vel_{max} in the PRISM code, are used to represent the intervals taken by the driving variable vel. Consequently, when expressing properties vel_{max}, resp. vel_{min}, will be used verify whether velocity is lower, resp. greater, than a certain value. In PRISM, transitions of a CTMC model are labeled with rates (positive integers) instead of probabilities. To be conform with this modeling, probabilities from the driving behavior model are mapped to positive integers representing the rates in PRISM commands.

4.2.3 Some specifications of driver behavior

With the availability of numerous driving variables, different behaviors can be expressed as logical properties. With no consideration of the driving environment in this analysis (video processing techniques are needed to extract road information from the dataset used in this analysis), only verification of some driving habits, which may reflect the driver performance, such as **tailgating**, **turning behavior**, will be considered regardless of traffic signs.

For behavior as properties, we distinguish two kinds of properties, *parameterized* properties for which the values used for verification are supplied at verification time and *non-parameterized* properties in which the values are fixed by road regulations. Table 4.2 presents a list of properties that we will use in this analysis expressed in PCTL. The two first properties P_1 and P_2 verifying the tailgating behavior are *non-parameterized* properties, because the safe following distance is regulated by the traffic law. The other properties in Table 4.2 are *parameterized*. The following driving behavior were considered in this analysis.

- ***Tailgating.*** In order to avoid eventual collisions, a driver should keep a safe distance behind its preceding vehicle. For instance, the French Highway Code define the safe following distance (in meter) as $\frac{5}{9} \times vel$, with vel expressed in km/h. This is evaluated by using the properties P_1 and P_2. P_2 will be true if the safe following distance is always respected by the driver, whereas P_1 computes the probability of compliance with the safe following distance rule.

- ***Braking/accelerating behavior.*** One dangerous behavior that can damage the car mechanical system is pressing the accelerator and brake pedal at the same time. The driver should not apply pressure on both pedals at the same time. This behavior is expressed by P_6, which will be true if in the model there exists no state with BP and GP both greater than 10. Also, it can be useful to verify the response of the car to the driver action. For instance, we can verify that the car acceleration is positive when the driver presses the gas pedal. This is expressed by The property P_5, which will be true if the model does not contain a state where the longitudinal acceleration a_l is greater than a value a and the gas pedal pressure GP is negative.

- ***Behavior when turning.*** A cautious driver should keep his foot off of the gas pedal while turning to avoid a skid. This behavior is expressed by P_8 and P_9. It is also important to avoid crash into obstacles in both sides of the car especially when turning. We propose P_3 and P_4 to express that the driver should keep a safe right front/left front distance from right/left obstacles, which can either be another vehicle or the road side.

4.3 Results and discussion

In order to illustrate the applicability of the model checking analysis of driver behavior, we use the sample data from the signal corpus collected by *the behavior signal processing laboratory of Nagoya University* [106], presented in the previous chapter. To assess whether the analysis results are impacted with the abstraction used, we perform model checking on the driver models generated by the 4 the abstractions presented in Table 3.2, and for different driving times, mainly 1, 3 and 7 minutes drive.

Table 4.2 shows the logical properties corresponding to the behaviors presented in the previous section. The model checking was carried out by the version 4.3.1 of PRISM. Table 4.3 presents verification results for the models generated for 1, 3 and 7 minutes drive. To check the veracity of the results, these latter were compared to the variables frequencies (Figure 3.11). Figure 4.2 illustrates the safe following distance calculated

TABLE 4.2: A list of logical properties used in verification

		Property ϕ	Properties parameters
Non parameterized	$\mathbf{P_1}$:	$P=?[F^{<100} d_{c_{min}} > 5/9 \times vel_{max}]$	
	$\mathbf{P_2}$:	$A[G\ d_{c_{min}} > 5/9 \times vel_{max}]$	
Parameterized	$\mathbf{P_3}$:	$A[G\ d_{lf_{min}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r \ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0 \ \&\ \theta_{min} \geq \theta_l))]$	d_{safe}: the distance considered as safe.
	$\mathbf{P_4}$:	$A[G\ d_{rf_{max}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r \ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0 \ \&\ \theta_{min} \geq \theta_l))]$	θ_r: the right turning angle.
	$\mathbf{P_5}$:	$A[\ G(a_{l_{min}} \geq a \rightarrow GP_{min} \geq 0)]$	θ_l: the left turning angle.
	$\mathbf{P_6}$:	$A[G\ !(GP_{min} > p \ \&\ BP_{min} > p)]$	a: the value of acceleration to not be exceeded
	$\mathbf{P_8}$:	$A[\ G(a_{l_{min}} \geq a \rightarrow (\theta_{max} < \theta_r \ \&\ \theta_{min} > \theta_l))]$	p: the tolerable value of pressure
	$\mathbf{P_9}$:	$P=\ ?[a_{l_{min}} > 0 \ \&\ (\theta_{min} > \theta_r \mid \theta_{max} < \theta_l)]]$	

TABLE 4.3: Verification results

Abs	**A2**			**A3**			**A4**			**A5**			**Parameters**
T	**1**	**3**	**7**	**1**	**3**	**7**	**1**	**3**	**7**	**1**	**3**	**7**	
P_1	0.999949	0.999949	0.999949	0.999999	0.999999	0.999999	0.999652	0.999652	0.999652	0.999793	0.999793	0.999793	
P_2	False	False	False	False	False	False	False	False	False	False	False	False	
P_3	False	False	False	False	False	False	False	False	False	False	False	False	$d_{safe}=30$, $\theta_r=30$, $\theta_l=-30$
P_4	True	True	True	True	True	True	True	True	True	True	True	True	
P_5	False	False	False	True	False	False	True	True	True	True	True	True	$a=1$
P_6	True	True	True	True	True	True	True	True	True	True	True	True	$p=10$
P_8	True	True	True	False	False	False	False	False	False	True	True	True	$a=2$, $\theta_r=60$, $\theta_l=-60$
P_9	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Abs: Abstraction

T: Time of recording in minutes

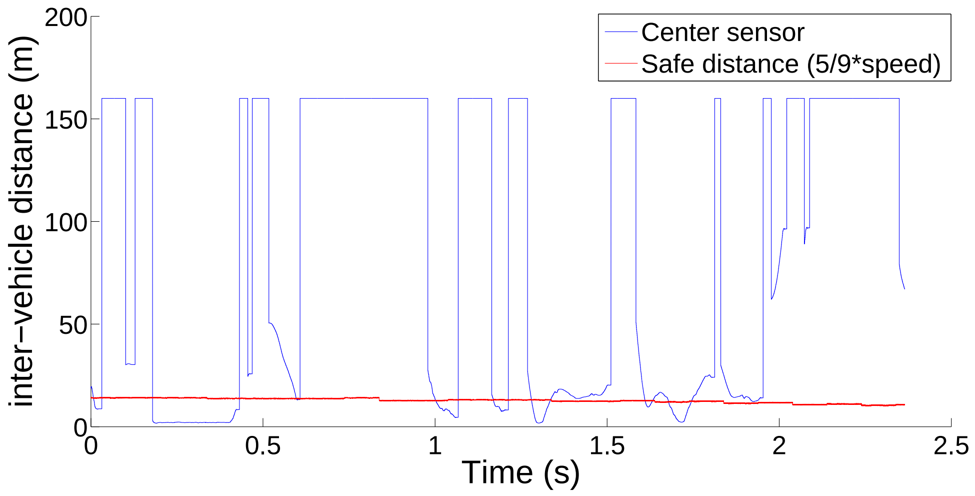


FIGURE 4.2: Following distance reported by the sensor versus safe distance deduced from the speed values

from speed values (the red graph) compared with the distance values from the sensor. We notice from the figure that in some points the safe distance is not respected, i.e where the distance is lower than 20m. In fact, checking this behavior by P_2 gives "false" as a result meaning that the subject driver has broken this rule at least once. P_1 gives us more insights because it quantifies the verified behavior. The results show that for 99% of times the driver has kept a safe following distance. In the frequency graphs displayed in Figure 3.11, we can see that in some points of time where left front distance was ≤ 30, it happens that the steering wheel angle values exceed 30 or are below -30. This was captured by property P_3, proving that the driver does not always keep safe distance from both sides of the car when performing turning maneuvers. P_4 aimed to verify the same behavior considering the right front distance, however, to illustrate the impact of using the min and max variable on the results, we use rfd_{max} in the property expression. As predicted, the checking gives wrong results because by using the upper value, the distance values lower than rfd_{max} are overestimated.

When we compare the verifications for the 4 abstractions, we notice contradictory results for P_5 and P_8. This is actually due to the values of parameters used for verification. For instance, the fact that only abstractions A_2 and A_3 have succeed in capturing a state with longitudinal acceleration greater than $a = 0.1$ and a positive gas pedal pressure is because of the value of $a = 0.1$ is an interval bound which means that the value of interest (0.1) has not been overestimated by the model. We notice also that, in A_3, P_5 becomes false when the recording time increases. However, it is logical because as we record more data, new model states are traversed, and hence states where a property is not satisfied can be found.

In summary, we can say that model checking driving behavior offers a formal approach for an easy traffic safety verification. The results obtained so far have reflected the real behavior of the driver, which was initially represented by plots of driving signals in Figure 3.11 . Moreover, the use of PCTL logic allowed us to measure the probability of occurrence of verified behaviors. The comparison of results reveals that the verification accuracy depends primarily on the abstraction used and on properties parameterization consequently. Therefore, in order to get correct verification results, the abstraction has to be defined considering the parameters of properties.

4.4 Chapter summary

In this chapter, we have presented a formal verification of driver behavior by an automated model checking tool. In this approach of verification, desired driving behaviors were expressed in a temporal logic and used to verify the driver behavior, modeled as a Probabilistic Rectangular Hybrid I/O Automaton. The usability of the method was demonstrated through a verification of vehicle traces belonging to one driver. The impact of abstraction on verification results has also been investigated.

Chapter 5

Graph-based driving behavior analysis

"The cars we drive say a lot about us."

– Alexandra Paul

Driving behaviors in different road conditions vary from one driver to another. For instance, drivers differ in the way they maintain certain speed, how they press the pedals, and also in the distance they keep from other vehicles [111]. Either for personal preferences, habits or vehicle characteristics, drivers tend to have each a unique driving behavior. An analysis that focus on and quantify this heterogeneity is of great applicability for driver identification. In this chapter, the aim is to answer the question, can driving behavior be used as a feature for personal identification?. To answer this question, we propose the application of graph theory to represent and analyze driving behavior; since graphs will provide a better view of the driving behavior, and will give us the ability to perform graph-based analysis, which was proven useful when applied in many different areas. In order to investigate the dissimilarity of drivers behaviors, graph matching techniques, namely the graph edit distance, are used as a comparison technique.

5.0.1 The Graph matching problem

Theoretical concepts from graph theory have been highly applied to study various applications. The graphs' ability to represent complex entities and their relational properties makes of them a powerful tool used in nearly every branch of science [112], e.g. software engineering, data mining and networking. Moreover, the genericity of graphs, i.e their

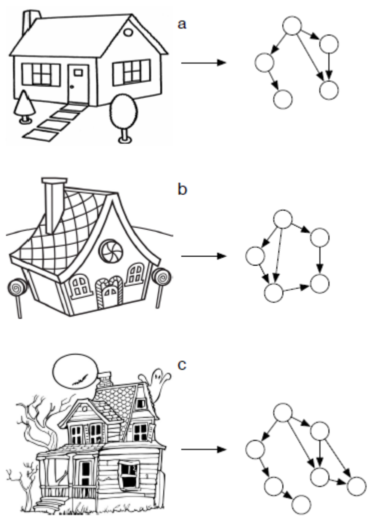


FIGURE 5.1: Examples of graph based representation([6]). The objects "houses" are represented by graphs, with the nodes representing the constituent parts of houses, i.e wall, door, window, roof, and the edges representing the connections between these parts

invariance to transformations such as rotation and translation, allows them to be well-suited for applications such as structural pattern recognition. In this domain, objects are often represented by graphs, and the recognition is formulated into a *graph matching problem*. For instance, objects "houses" are represented, in figure 5.1 as graphs; the constituent parts of a house such as walls, roofs and windows are represented as nodes with the connected parts linked by edges. In the figure, (a) represents a model of the house, (b) and (c) represent real world objects and classifying them as houses goes back to comparing the graphs with the model graph (a).

In the field of graph theory, graph matching is the problem of finding a similarity between graphs; it can be formulated as finding a mapping $s : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{R}$, given two G and G' from the space of graphs $\mathcal{G}$, such that $s(G, G')$ measures the similarity (or dissimilarity) of G and G' [113]. Because of its combinatorial nature, graph matching is considered as one of the most complex issues. Its complexity class still remains unsettled, and depends on the matching approach. The correspondence between two graphs is based on the three structural characteristics of the graph [114]:

- The labels on the vertices should be the same in the two graphs.
- Edges existing between vertices in one graph should match the edges existing between their corresponding vertices in the second graph.
- The labels on the edges of the two graphs should much each other.

The use of graphs based techniques for pattern representation and classification in pattern recognition dates back to the late seventies. Since then, research efforts have been made to solve the matching problem, and to find efficient similarity measures. Two approaches have been adopted for defining similarity measures; they are either computed directly in the graphs domain, or recently through a representation of them in a suitable space [100]. The objective behind such representation is to bring into use vector-based techniques for classification to graphs. *Graph Kernels*, for instance, allow the application of kernel machines, e.g SVM, directly on graphs by extending the inner product defined on vectors to the domain of graphs. Let $\mathcal{G}$ be the space of graphs, a graph kernel is a kernel function (i.e symmetric and positive semi-definite) $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{R}$ that maps a pair of graphs onto a real number. In that sense, a graph kernel can be intuitively considered as a similarity measure, however its formal properties make the whole family of kernel methods applicable to graphs. The *Graph embedding* approach opts for another representation by mapping the graphs onto points in a vector space, in such a way that the distance between the points reflect the similarity between the graphs. In the traditional graph matching problem, a similarity measure is generally obtained by comparing both the semantic and structural information of the graphs. The methods for graph matching can be roughly divided into two broad classes: exact matching methods and inexact matching methods. The exact matching approach aims at finding a strict correspondence between the two graphs being matched, at least among their subparts, while in the inexact matching approach a matching can occur even if the two graphs being compared are structurally different to some extent [115]. The two classes are explained in details in the following. Figure 5.2 presents some of the most popular graph matching methods.

5.0.2 Exact graph matching

Exact matching relies on a boolean evaluation of the (dis)similarity of two graphs, and determining whether they are identical in terms of topology and labeling. The main characteristic of exact graph matching is that the mapping between the vertices of the two graphs must be *edge-preserving*, in the sense that if an edge exists between two vertices in one graph, then that edge must also exist between the vertices they are mapped to in the second graph. Formally, the *edge preserving* characteristic can be defined as follows : Given two graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where V_i and E_i are the sets of vertices and edges respectively ($i \in 1, 2$), and a mapping $\varphi : V_1 \rightarrow V_2$. The mapping φ is said to be *edge preserving* iff:

$$\forall u, v \in V_1, (u, v) \in E_1, \exists(\varphi(u), \varphi(v)) \in E_2$$

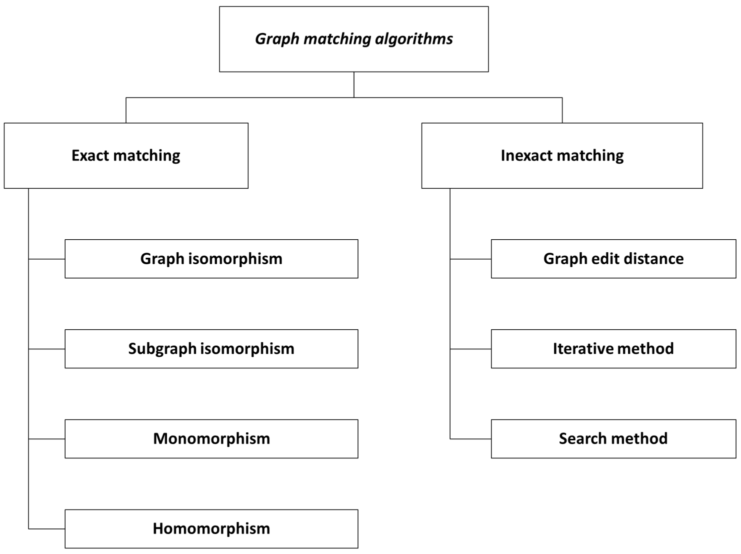


FIGURE 5.2: Classification of graph matching approaches

There exist different types of exact matching, e.g *Graph isomorphism*, *subgraph isomorphism*, *monomorphism*, *homomorphism.*

Graph isomorphism, the stricter type, is when a bijective mapping exists between the vertices of two graphs that preserves the edge structure and all labels. Thus, isomorphic graphs are similar/identical in both topology and labels. This problem is known to be in the complexity class NP, however it is shown to be solved efficiently is some special cases such as for planar graphs. More formally, a graph isomorphism between two labeled graphs $G_1 = (V_1, E_1, \mu_1, \delta_1)$ and $G_2 = (V_2, E_2, \mu_2, \delta_2)$ (where μ_i and δ_i are respectively vertex labeling and edge labeling functions) is bijective function $f : V_1 \to V_2$ satisfying [100]:

- $\forall v \in V_1, \mu_1(v) = \mu_2(f(v))$
- $\forall e_1 = (u, v) \in E_1, \exists e_2 = (f(u), f(v)) \in E_2$ such that $\delta_1(e_1) = \delta_2(e_2)$
- $\forall e_2 = (u, v) \in E_2, \exists e_1 = (f^{-1}(u), f^{-1}(v)) \in E_1$ such that $\delta_1(e_1) = \delta_2(e_2)$

Subgraph isomorphism, is a weaker form of matching requiring an isomorphism between one graph and a subgraph of the other graph. This type can be useful, in the pattern recognition field, for identifying corresponding substructures in two graphs, e.g identifying objects in larger scenes. Formally, an injective mapping $f : V_1 \to V_2$ from $G_1 = (V_1, E_1)$ to $G_2 = (V_2, E_2)$ is called a subgraph isomorphism, if there exists a subgraph $G \subseteq G_2$ such that $f(\cdot)$ is a graph isomorphism between G_1 and G [100].

Other forms of matching drop conditions about the bijective mapping and the matching in the number of vertices, such as *monomorphism* that requires an injective mapping only, and *homomorphism* where many-to-one correspondence between the two graphs is considered. Another interesting type of exact graph matching is *maximum common subgraph* which goal is to find the largest subgraph for which a mapping between a subgraph of the the first graph and an isomorphic subgraph of the second one.

5.0.3 Inexact graph matching

In fact, the strict constraints imposed by graph/subgraph isomorphism (i.e. similarity in terms of topology and labeling) are too rigid for comparing two graphs, making exact matching inapplicable in some circumstances. In many applications, especially in pattern recognition, the intrinsic variability of patterns and the presence noise raise the possibility of having graphs that represent patterns from the same class, but are not completely identical in their structure. Hence, the need to integrate some degree of error tolerance into the graph matching process. In addition, because of the high computational complexity of exact matching algorithms, providing an approximate solution in a reasonable time may be a wiser alternative. Inexact graph matching was then proposed as a more practical alternative that overcomes these drawbacks of exact matching.

In inexact matching algorithms, the edge-preservation constraint used in exact matching is generally not required. Instead, edges not satisfying this constraint are penalized with a cost that takes into account other differences. That is, inexact matching aims to find a mapping from one graph to another that minimizes the overall matching cost. There exist several approaches to inexact graph matching that propose different definitions of the cost function. One common technique is to use the matching cost as a measure of the graphs dissimilarity, such as *graph edit distance*.

5.0.4 Graph edit distance

Graph Edit Distance method (GED) belongs to the class of inexact graph matching, and is known for its flexibility in comparison with other methods as it can be applied to a wide class of graphs [100]. The idea of GED is to measure the dissimilarity between two graphs G_1 and G_2 by the cost of *graph edit operations*, i.e *Insertions*, *Deletions* and *Substitutions* of vertices and edges, needed to transform G_1 into G_2. Let the graphs $G_1 = (V_1, E_1, \mu_1, \delta_1)$ and $G_2 = (V_2, E_2, \mu_2, \delta_2)$ be labeled graphs and ϵ a dummy vertex or edge used to represent insertions and deletions, edit operations are defined as one of the following operations applied on the vertices or edges of G_1 [6]:

- *Insertions* ($\epsilon \to v_2$ for vertex insertion, $\epsilon \to e_2$ for edge insertion). insert a vertex or edge from sets V_2 or E_2. A label is also associated to the inserted element.
- *Deletions* ($v_1 \to \epsilon$ for vertex deletion, $e_1 \to \epsilon$ for edge deletion). remove a vertex or edge from sets V_1 or E_1.
- *Substitutions* ($v_1 \to v_2$, $e_1 \to e_2$). replace a vertex v_1 or edge e_1 with a vertex v_2 or edge e_2 from sets V_2 or E_2.

A sequence of operations $(o_1, \cdots, o_k)$ applied to transform G_1 into G_2 is referred to as an *edit path* $\pi = (o_1, \cdots, o_k)$ between G_1 and G_2. Figure 5.3 (taken from [6]) shows an example of a possible edit path between two graphs. It must be emphasized that insertions and deletions in a edit path have to satisfy the following two constraints:

- Deleting a vertex implies deleting all its incident edges,
- An edge can be inserted only between two existing or already inserted nodes.

As many edit paths could exist, a non negative edit cost $c(o_i)$ is associated to each graph edit operation o_i. Intuitively, a low cost edit path means that the two graphs are similar. To avoid unnecessary edit operations, the edit cost function must satisfy the following inequalities:
$c(\epsilon \to v_2) \leq c(\epsilon \to v) + c(v \to v_2)$
$c(\epsilon \to e_2) \leq c(\epsilon \to e) + c(e \to e_2)$
$c(v_1 \to \epsilon) \leq c(v_1 \to v) + c(v \to \epsilon)$
$c(e_1 \to \epsilon) \leq c(e_1 \to e) + c(e \to \epsilon)$
$c(v_1 \to v_2) \leq c(v_1 \to v) + c(v \to v_2)$
$c(e_1 \to e_2) \leq c(e_1 \to e) + c(e \to e_2)$

Through this concept of edit cost functions, graph edit distance can be tailored to meet the requirements of various applications. When GED is computed between two labeled graphs, edit costs are defined as functions of labels [116]. In this case, the costs of substitutions are defined as a function of the labels of the substituted elements, whereas insertions and deletions are penalized with values linked to the labels of the inserted/deleted element. It is also possible to parametrize graph edit distance by integrating domain specific knowledge when defining edit cost functions. Given the costs of the edit operations, the cost $\gamma(\pi)$ of an edit path $\pi \in \Pi(G_1, G_2)$, where $\Pi(G_1, G_2)$ is the set of all edit paths between G_1 and G_2, is defined as the sum of the costs of the edit operations

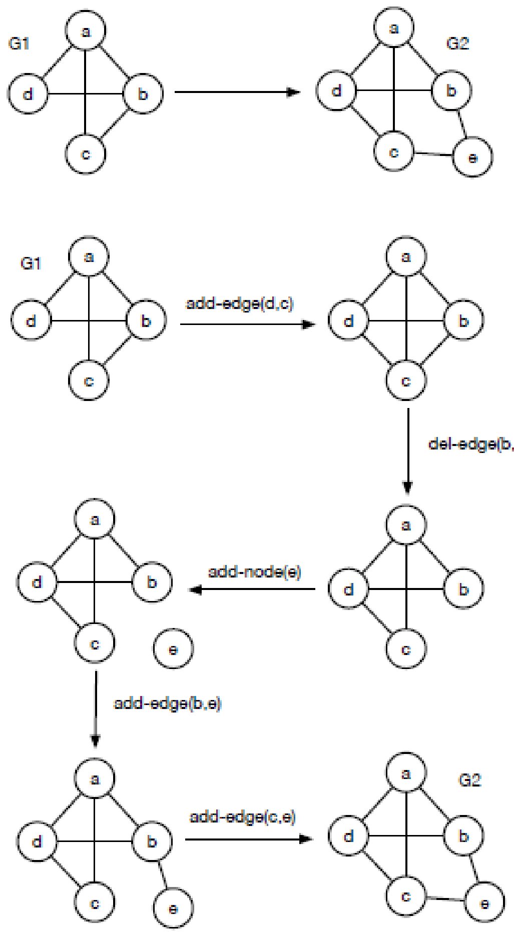


FIGURE 5.3: A possible edit path between the graphs $G1$ and $G2$

of the path π.

$$\gamma(\pi) = \sum_{o_i \in \pi} c(o_i)$$

Considering all the edit paths between $G_1 = (V_1, E_1, \mu_1, \delta_1)$ and $G_2 = (V_2, E_2, \mu_2, \delta_2)$, the edit distance is defined by the minimum edit path cost between the two graphs. The graph edit distance $d(\cdot, \cdot)$ is therefore formally defined as a function:

$$\begin{array}{llll} d: & \mathcal{G} \times \mathcal{G} & \rightarrow & \mathcal{R}^+ \\ & (G_1, G_2) & \mapsto & GED(G_1, G_2) = \min\limits_{\pi \in \Pi(G_1, G_2)} \gamma(\pi) = \min\limits_{(o_1, \ldots, o_k) \in \Pi(G_1, G_2)} \sum_{i=1}^{k} c(o_i) \end{array}$$

The problem of GED computation is in general NP hard. There exist several algorithms for solving the GED computation problem, which can be distinguished into exact methods and approximate methods. Exact methods are reported to compute GED only for small graphs, while for larger graphs approximations by means of lower and upper bound computation are often used [116]. A wide number of exact algorithms are based on the A^* algorithm; they consider the set of all possible edit paths between two graphs as an ordered tree. In this tree, each node corresponds to a partial edition; a leaf node corresponds to a complete edit path between two graphs. The problem of GED computation is then transformed into a tree search algorithm, evaluating the different edit paths to find the path with the minimum edit cost. Other exact algorithms formulate the graph edit distance into a binary linear program, i.e a linear program where all variables must take values from the set $0, 1$, modeling the graphs by means of adjacency matrix and the edit distance as the permutation matrix that minimizes the cost of graph transformation [116]. Unfortunately, exact computation of GED is solved in exponential time, i.e between two graphs G_1 and G_2 with m and n vertices respectively, there exist $\mathcal{O}(mn)$ edit path to explore, restricting its applicability to graphs of small size. The goal of approximate methods is to make GED computation faster, i.e in polynomial time. For instance, the A^*- *beam search* algorithm [117], a variant of A^*, proposes an approximation based on the idea of beam search. In order to reduce the complexity of A^*, A^*-*beam search* does not explore the full search space, but expands only nodes that belong to the most promising partial matches. Another approximate approach solves the GED problem by means of *bipartite graph matching* [118]. This heuristic relies on the idea that a mapping between nodes with similar neighborhoods should induce a low cost edit path associated [119]. In *bipartite graph matching*, the GED problem is reduced to a bipartite assignment problem to find the optimal mapping between the two relevant graphs.

5.1 The graph matching toolkit

In the analyses presented in this chapter, this graph comparison task is performed using the graph matching tool developed by Riesen et all [120]; it provides an algorithmic framework for fast suboptimal graph edit distance computation implementing a number of approximations previously proposed by the authors. The software features and the implemented approximation methods are extensively described in [120]. The graph matching tool enables the parametrization of graph edit distance to adapt to different problems by providing the following:

- Five algorithms to compute the graph edit distance, namely, *A**, *Beam Search*, *Munkres' Algorithm*, *Hungarian Algorithm* and the algorithm of *Volgenant and Jonker*.

- Four distance functions, to parametrize to edit costs, the first two distances are applicable to numerical attributes only whereas the last distance is applicable to string attributes.

  1. *absolute value difference*: $d(N_1, N_2) = |N_1 - N_2|)$, where N_1 and N_2 are numerical values.
  2. *squared difference*: $d(N_1, N_2) = (N_1 - N_2)^2)$, where N_1 and N_2 are numerical values.
  3. *discrete metric*: $d(A_1, A_2) = \begin{cases} \mu, & if A_1 = A_2 \\ \nu, & else \end{cases}$, where μ, v are non-negative real values ($\mu, v \in \mathbb{R}^+$) defined by the user.
  4. *Levenshtein distance*: also known as string edit distance (sed), defined as $d(S_1, S_2)$ = minimal number of single-character edit operations (deletions, insertions, substitutions) required to change string S_1 into string S_2.

- Four similarity kernels enabling the transformation of graph edit distance $d(G_1, G_2)$ to a similarity measure $k_i(G_1, G_2)$.

  - $k_1 = -d(G_1, G_2)^2$
  - $k_2 = -d(G_1, G_2)$
  - $k_3 = tanh(-d(G_1, G_2))$
  - $k_4 = exp(-d(G_1, G_2))$

The software of Riesen et al. supports the matching of labeled and unlabeled graphs; unlabeled graphs are implemented by assigning a same symbolic label to all nodes and edges. The labels for edges and nodes can be defined as integers in $\mathcal{R}$, strings over an alphabet, or as an arbitrary combination of different labels. The graph edit matching is parametrized by defined costs for nodes/edges insertions, deletions, and substitution. For the sake of symmetry, the insertion and deletion costs are assumed to be identical. For the substitution operation, the cost is measured by means of one of the four distances mentioned above defined on each of the attributes. The software enables a single node to be labeled with five up to five attributes defined by the user. The software also gives the user the ability to scale the relative importance of an attribute distance value by using a weighting parameter $\sigma_i \in]0, 1]$. Given two nodes, resp. edges, u and v labeled with k attributes A_i, $i \in 1 \leq i \leq k$, and p, a parameter indicating that the p-th root is extracted from the combined cost; the cost of node, resp. edge substitution is obtained by building the sum or the product of individual attributes costs.

$$\left(\sum_{i=1}^{k} \sigma_i.d_i(u.A_i, v.A_i)\right)^{1/p} \text{ or } \left(\prod_{i=1}^{k} \sigma_i.d_i(u.A_i, v.A_i)\right)^{1/p}$$

In order to control the importance of nodes costs and edges costs, a weighting parameter $\alpha \in [0, 1]$ is defined. The node operation costs are then multiplied by α, whereas edge operations are multiplied by $(1 - \alpha)$.

5.2 Graph matching for driving behavior similarity analysis

5.2.1 Data sets

5.2.1.1 OpenXC trace files

As mentioned in the earlier chapters, several researches have focused on the collection and measurement of vehicle data. Ford Bug Labs, for instance, have developed the *OpenXC platform*, an open source hardware and software API for accessing vehicle data, and made it available for research and automobile applications development. The *OpenXC* vehicle interface, i.e the *OpenXC* hardware, is based on a microcontroller with two external connections, one to the CAN bus via the OBD-II port, and one to the host device via USB or serial. The hardware listens to CAN messages (or a filtered subset of them), performs required unit conversion or factoring and outputs a generic version to a USB, Bluetooth or network interface [121][122]. Figure 5.4, taken from *OpenXC* website illustrate the use diagram of the *OpenXC* platform.

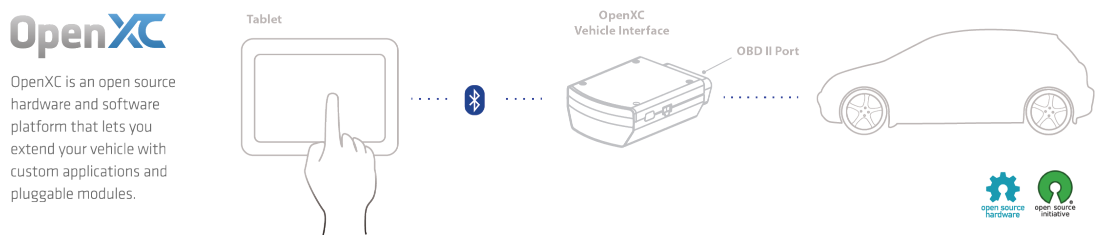

OpenXC
OpenXC is an open source hardware and software platform that lets you extend your vehicle with custom applications and pluggable modules.
Tablet
OpenXC Vehicle Interface
OBD II Port
open source hardware
open source initiative

FIGURE 5.4: *OpenXC*'s use diagram

When the *OpenXC* vehicle interface is plugged into a car, and from an android device, it is possible to retrieve real time vehicle data such as steering wheel angle, vehicle speed, etc. The data is generally formatted in JSON format, and is accessible from android application using the *OpenXC* library. The list of signals officially supported by the Android library is presented in table 5.1. More details about the *OpenXC* library can be found on the platform website.

The *OpenXC* platform makes also available a number of vehicle trace files of anonymous drivers, collected in different scenarios, and for different behaviors. These vehicle trace

TABLE 5.1: *OpenXC* supported driving signals

Signal	**Range**	**Frequency**
steering_wheel_angle	-600 to +600 degrees	max 10Hz
torque_at_transmission	-500 to 1500 Nm	max 10Hz
engine_speed	0 to 16382 RPM	max 10Hz
vehicle_speed	0 to 655 km/h	max 10Hz
accelerator_pedal_position	0 to 100%	max 10Hz
parking_brake_status	boolean (true when brake engaged)	1Hz and immediately on change
brake_pedal_status	Boolean (true when pedal pressed)	1Hz and immediately on change
transmission_gear_position	first, second, third, fourth, fifth, sixth, seventh, eighth, reverse, neutral	1Hz
odometer	0 to 16777214.000 km, with about .2m resolution	max 10Hz
ignition_status	off, accessory, run, start	1Hz and immediately on change
fuel_level	0 - 100%	max 2Hz
fuel_consumed_since_restart	0 - 4294967295.0 L	max 10Hz
door_status	driver, passenger, rear_left, rear_right	1Hz and immediately on change
headlamp_status	Boolean (true is on)	1Hz and immediately on change
high_beam_status	Boolean (true is on)	1Hz and immediately on change
windshield_wiper_status	Boolean (true is on)	1Hz and immediately on change
latitude	-89.0 to 89.0 degrees with standard GPS accuracy	max 1Hz
longitude	-179.0 to 179.0 degrees with standard GPS accuracy	max 1Hz
button_event	left, right, up, down, OK	Sent only if value changes

files can be downloaded from *OpenXC* website[123]. The files with the names shown in Table 5.2 were used in the present analysis.

TABLE 5.2: *OpenXC* Vehicle trace files used in the analysis

Vehicle Trace Files for New York City, USA	
Symbol	File name
DC	*Downtown, Crosstown*
DE	*East Downtown*
DW	*West Downtown*
DW2	*West Downtown 2*
UC	*Uptown Crosstown*
UC2	*Uptown Crosstown 2*
UW	*West Uptown*
UW2	*West Uptown 2*

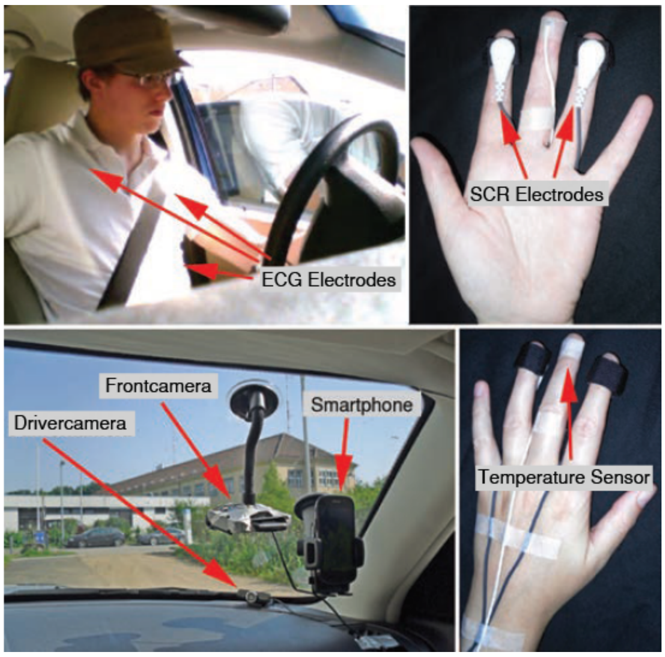


FIGURE 5.5: Sensors used for data collection in hcilab driving dataset

5.2.1.2 hcilab driving data set

For a second experiment, we use the driving data set collected by the human-computer interaction lab (hciLab) as part of their research study to analyze drivers workload in different road types. The sensors used for data collection are shown in Figure 5.5. The dataset contains anonymous information about GPS, brightness, acceleration, physiological data of 10 drivers, and is downloaded as an archive of comma separated files[1] where each file contains the recordings of one driver. Each driver drove for about 30 minutes, in five different road types: 30 km/h zone, 50 km/h zone, highway, freeway, and tunnel. The different signals recorded during the study are presented in table 5.3. The videos of drivers and driving scenarios recorded by the *Driver camera* and the *Front camera* were excluded from the public dataset for privacy reason. More detailed information about the data set and the performed study can be found in [124].

5.2.2 Data sets processing

The *OpenXC* trace files are plain text files that contains OpenXC JSON messages with an additional timestamp field, separated by newlines. JSON is a language independent, human-readable data format; it was used in *OpenXC* because of its flexibility and ease of use. The following lines show the JSON syntax of *OpenXC* messages:

```
{"name": "accelerator_pedal_position", "value": 0, "timestamp": 1364323939.012000}
{"name": "engine_speed", "value": 772, "timestamp": 1364323939.027000}
```

[1]https://www.hcilab.org/research/hcilab-driving-dataset/

Table 5.3: Data collected in the hcilab study

Data		Frequency	Sensor
GPS information	Latitude (m)	1HZ	Mobile phone
	Longitude (m)		
	Speed (m/s)		
	Bearing (degree)		
Brightness and Acceleration	Brightness	Between 8HZ and 12 HZ	
	Acceleration X		
	Acceleration Y		
	Acceleration Z		
Physiological data	Heart rate	128HZ	Electrocardiogram
	Skin conductance		Skin conductance sensor
	Body temperature		Temperature sensor

```
{"name": "vehicle_speed", "value": 0, "timestamp": 1364323939.029000}
{"name": "accelerator_pedal_position", "value": 0, "timestamp": 1364323939.035000}
```

The timestamp is in UNIX time, i.e. the number of seconds since the UNIX epoch, 00:00:00 UTC, 1/1/1970.

As shown in Table 5.1, *OpenXC* supports a set of car signals, however, in this study we restrict ourselves to those listed in Table 5.4. Table 5.4 shows for each variable the range of its values, the frequency of measurement, as well as the abstract values from the interval domain that we have defined; we do not define any abstraction for the variable *transmission_gear_position* because it is discrete.

Table 5.4: *OpenXC* driving signals used in the analysis

Signal	**Range**	**Abstract Domain Used**
engine_speed	0 to 16382 RPM	{[0,500[, [500,1000[, [1000,1500[, [1500,2000[, [2000,2500[, [2500,3000[, [3000,4000[, [4000,16382[, [16382,∞)}
fuel_level	0-100%	{[0,20[, [20,40[, [40,60[, [60,80[, [80,100[, [100,∞)}
torque_at_transmission	-500 to 1500 Nm	{[-500,-100[, [-100,0[, [0,50[, [50,100[, [100,150[, [150,200[, [200,500[, [500,1500[, [1500,∞[}
transmission_gear_position	{ first, second, third, fourth, fifth, sixth, seventh, eighth, reverse, neutral}	N/A*
vehicle_speed	0 to 655 km/h	{[0,10[, [10,20[, [20,30[, [30,40[, [40,50[, [50,60[, [60,655[,[655,∞[}

* N/A: Non abstraction applicable.

For the hciLab driving data set, we use the signals presented in table 5.5. The intervals used for abstraction have been deduced from the data set using k-means, as presented in the data abstraction section of chapter 3.

Now that abstractions have been defined, labels for the graph vertices have to be designed. Table 5.6 shows the labels we have used for OpenXC data set. For all variable,

Table 5.5: hcilab driving signals used in the analysis

Signal	**Abstract Domain Used**
SPEED_GPS	{ [0, 1.07155[, [1.07155, 3.64651[, [3.64651, 6.92086[, [6.92086, 10.11305[, [10.11305, 13.08576[, [13.08576, 16.29244[, [16.29244, 19.33357[, [19.33357, 21.43441[, [21.43441, 24.52137[, [24.52137, 26.77736[, [26.77736, 28.96064[, [28.96064, 32.94277[, [32.94277, 37.47226[, [37.47226, 80] }
ACCELX	{ [-180, -7.50822[, [-7.50822, -6.30154[, [-6.30154, -5.51624[, [-5.51624, -4.59687[, [-4.59687, -4.07972[, [-4.07972, -3.21781[, [-3.21781, -2.37505[, [-2.37505, -1.58975[, [-1.58975, -0.93853[, [-0.93853, -0.55545[, [-0.55545, 0.01915[, [0.01915, 0.38307[, [0.38307, 0.74699[, [0.74699, 1.18752[, [1.18752, 1.83875[, [1.83875, 2.79643[, [2.79643, 4.09887[, [4.09887, 4.84586[, [4.84586, 5.55455[, [5.55455, 6.24408[, [6.24408, 8.02536[, [8.02536, 180]}
ACCELY	{[-180, -0.42138[, [-0.42138, 1.95367[, [1.95367, 3.83072[, [3.83072, 4.97994[, [4.97994, 5.66947[, [5.66947, 6.22492[, [6.22492, 8.00621[, [8.00621, 8.54251[, [8.54251, 9.30866[, [9.30866, 9.71088[, [9.71088, 10.05565[, [10.05565, 10.41957[, [10.41957, 10.91756[, [10.91756, 11.99016[, [11.99016, 12.85207[, [12.85207, 14.11621[, [14.11621, 15.07389[, [15.07389, 15.8975[, [15.8975, 16.81687[, [16.81687, 180]}
ACCELZ	{ [-180, -8.48505[, [-8.48505, -6.95276[, [-6.95276, -5.42047[, [-5.42047, -4.53941[, [-4.53941, -4.00311[, [-4.00311, -2.52828[, [-2.52828, -2.01113[, [-2.01113, -1.16837[, [-1.16837, -0.68953[, [-0.68953, -0.2873[, [-0.2873, 0.09577[, [0.09577, 0.47884[, [0.47884, 0.88107[, [0.88107, 1.37906[, [1.37906, 2.04944[, [2.04944, 3.12204[, [3.12204, 4.29041[, [4.29041, 5.51624[, [5.51624, 6.53138[, [6.53138, 8.40844[, [8.40844, 180] }

Table 5.6: OpenXC: Matching the vertices labels to abstract values

label	**fuel_level**	**engine_speed**	**torque_at_transmission**	**vehicle_speed**	**transmission_gear_position**
-1	N/A**	N/A	”reverse”	N/A	N/A
0	NO_DATA*	NO_DATA	NO_DATA	NO_DATA	”neutral”
1	[0, 20[	[0, 500[	[-500, -100[	[0, 10[	”first”
2	[20, 40[	[500, 1000[	[-100, 0[	[10, 20[	”second”
3	[40, 60[	[1000, 1500[	[0, 50[	[20, 30[	”third”
4	[60, 80[	[1500, 2000[	[50, 100[	[30, 40[	”fourth”
5	[80, 100[	[2000, 2500[	[100, 150[	[40, 50[	”fifth”
6	[100,∞[	[2500, 3000[	[150, 200[	[50, 60[	”sixth”
7	N/A	[3000, 4000[	[200, 500[	[60, 655[	”seventh”
8	N/A	[4000, 16382[	[500, 1500[	[655,∞[	N/A
9	N/A	[16382,∞[	[1500,∞[	N/A	N/A

*NO_DATA : No data received at that time

**N/A : No corresponding valued

each abstract value (interval) is mapped to a number from the set $\{-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ as seen in Table 5.6; and we construct a vertex label as a combination of the variables labels (in the same order presented in Table 5.6). For instance, a vertex labeled 02000 means that the value of *engine_speed* is in *[500, 1000[* and the *transmission_gear_position* is in ”*neutral*” position.

A Python program implementing the algorithm of driver behavior graph construction (Algorithm 2) was developed. The program processes the trace files and outputs graphs of the drivers behavior which will be used as drivers profiles. Considering the anonymity of the OpenXC data set, the trace files are supposed to belong to different drivers.

Consequently, one driver graph is constructed per each vehicle trace file. Similarly, processing the hciLab driving data set generates 10 driver graphs for the 10 drivers. The processing of the OpenXC data set results in 8 driver graphs with 618 nodes and 2356 edges. For the hciLab data set, because of hardware limitation we stop the graphs construction at 1544 nodes. In order to evaluate the potential of graph matching for driver recognition, we also generate graphs for sub parts of the traces, which we will call subgraphs in the remaining of this chapter. Table 5.7 displays the size of the generated graphs. A driver graph refers to the graph resulting from processing the entire trace file of one driver. Subgraph 1 and subgraph 2 are graphs resulting from processing sub parts of drivers data.

TABLE 5.7: Sizes of the generated graphs

Data set		Driver graph size	Subgraph 1	Subgraph 2
OpenXC	nodes	618	326	484
	edges	2356	865	1852
hciLab	nodes	1544	488	999
	edges	3090	1247	2285

Now, in order to compare the drivers behaviors, we analyze the similarity between the graphs generated, by means of GED. We also compute the edit distance between subgraphs and driver graphs to investigate the applicability of graph matching to driver pattern recognition. The computation of the GED between graphs has been carried out using the graph matching toolkit of Riesen et all, presented in the previous section. The software parameters used for matching are presented in figures 5.6 and 5.7.

```
#Graph edit distance procedure: Hungarian
#Edge mode: directed
#Cost for node deletion/insertion: 1.0
#Cost for edge deletion/insertion: 1.0
#Alpha weighting factor between node and edge costs: 0.5
#Node attribute 0: engine_speed; Cost function: discrete; mu=0 nu=0.2; Soft factor: 1.0
#Node attribute 1: fuel_level; Cost function: discrete; mu=0 nu=0.2; Soft factor: 1.0
#Node attribute 2: torque_at_transmission; Cost function: discrete; mu=0 nu=0.2; Soft factor: 1.0
#Node attribute 3:transmission_gear_position; Cost function:discrete; mu=0 nu=0.2;Soft factor:1.0
#Node attribute 4: vehicle_speed; Cost function: discrete; mu=0 nu=0.2; Soft factor: 1.0
#Edge Attribute 0: weight; Cost Function: absolute; Soft Factor: 1.0
#Individual node costs are added
#Individual edge costs are added
#(Combined node cost)^(1/1.0)
#(Combined edge cost)^(1/1.0)
```

FIGURE 5.6: Matching parameters for OpenXC data set

```
#Graph edit distance procedure: Hungarian
#Edge mode: directed
#Cost for node deletion/insertion: 1.0
#Cost for edge deletion/insertion: 1.0
#Alpha weighting factor between node and edge costs: 0.5
#Node attribute 0: Speed_GPS; Cost function: discrete; mu=0.0 nu=0.25; Soft factor: 1.0
#Node attribute 1: AccelX; Cost function: discrete; mu=0.0 nu=0.25; Soft factor: 1.0
#Node attribute 2: AccelY; Cost function: discrete; mu=0.0 nu=0.25; Soft factor: 1.0
#Node attribute 3: AccelZ; Cost function: discrete; mu=0.0 nu=0.25; Soft factor: 1.0
#Edge Attribute 0: weight; Cost Function: absolute; Soft Factor: 1.0
#Individual node costs are added
#Individual edge costs are added
#(Combined node cost)^(1/1.0)
#(Combined edge cost)^(1/1.0)
```

Figure 5.7: Matching parameters for hciLab data set

5.2.3 Results and discussion

This section summarizes the analysis performed on the both data set; it illustrates and discusses some experimental results. Table 5.8 shows the results obtained for the OpenXC data set. The first table presents the inter distance between the complete graphs of the data set. The high values of the distances show the high dissimilarity within drivers behaviors and support thus the potential of graph matching for driver identification. We can also notice that the graph for DC has the higher GED value, which may be due to the difference of the driving environment. Also, the GED value tends to decrease for vehicle traces from the same driving environment, such as for DW vs DW2 and UW vs UW2. The last two tables presented in table 5.8 show the results of matching the subgraphs, constructed from sub parts of the traces, to the drivers graphs. It can be seen from the results that edit distance values between subgraphs considered in this analysis and their actual driver graphs are small, which mean that subgraph 1 and subgraph 2 have been successfully recognized. Moreover, by comparing the distances in these two tables we can clearly see that these distances are smaller for subgraph 2 which corresponds to a bigger part from the behavior logs, while the distances from other trace files have greater values. This result is quite logical because the more information we have about a driver behavior the bigger is the probability of his identification.

Similar results have been found in the analysis of HciLab data set, which are presented in table 5.9. As for the previous data set, we obtain high edit distance values as shown in the first table of table 5.9. It is worth mentioning that the drivers, who participated in the data collection, have droved in the same environment. This confirms the high dissimilarity and the heterogeneity in drivers behaviors even under the same driving conditions. For the last two tables of table 5.9 showing the matching results of subgraph 1 and subgraph 2, we notice that, similarly to the openxc data set, the diagonal values are smaller for subgraph 2, however, we do not see a big difference in other distance

TABLE 5.8: Similarity distances of OpenXC driver behavior graphs

	Data graph							
Model graph	DC	DE	DW	DW2	UC	UC2	UW	UW2
DC	0.00000	2056.50257	2073.95075	2093.10844	2059.03047	2113.25615	2114.93765	2091.94170
DE	2054.00257	0.00000	1957.75123	1997.88723	1913.30240	1982.78196	2044.35497	2019.79011
DW	2057.45075	1960.75123	0.00000	1943.03156	1945.19991	1966.45163	2001.86077	1992.70756
DW2	2094.10844	1995.88723	1951.53156	0.00000	1990.67642	1904.84429	1974.17923	1988.26766
UC	2056.53047	1906.30240	1952.19991	1991.17642	0.00000	1958.65330	2014.52274	2043.65429
UC2	2115.75615	1979.78196	1956.45163	1908.84429	1956.65330	0.00000	1982.60797	1987.54391
UW	2113.93765	2044.35497	2000.86077	1965.17923	2016.52274	1981.60797	0.00000	1848.89301
UW2	2092.94170	2022.29011	1996.20756	1986.26766	2043.65429	1995.54391	1846.89301	0.00000

	Data graph							
Model graph	DC-2	DE-2	DW-2	DW2-2	UC-2	UC2-2	UW-2	UW2-2
DC	754.96510	1935.70923	1982.41664	1955.68725	1962,.7052	1956.69554	1962.65151	1947.01850
DE	1911.32963	319.00000	1924.39812	1908.50837	1907.93881	1917.20529	1951.87485	1951.14876
DW	1971.69568	1895.55784	319.00000	1920.98218	1910.75277	1910.33333	1951.67162	1940.71512
DW2	1947.12534	1921.20609	1878.74870	319.00000	1916.23672	1903.47287	1916.93548	1927.01432
UC	1957.16451	1874.00143	1946.14779	1931.75607	319.00000	1921.80100	1964.43565	1948.77117
UC2	1945.61971	1876.60017	1909.64123	1876.08197	1920.78026	319.00000	1919.93998	193.,05971
UW	1969.46870	1938.19506	1887.72817	1938.52178	1925,.2737	1919.33962	319.00000	1906.90397
UW2	1948.99027	1939.33800	1943.62392	1903.53000	1931.70679	1942.36092	1890.36821	319.00000

Data graphs generated for sub part 2

	Data graph							
Model graph	DC-1	DE-1	DW-1	DW2-1	UC-1	UC2-2	UW-1	UW2-1
DC	1591.53882	1640.43967	1640.18677	1637.12164	1619.11469	1645.83702	1632.18826	1636.43894
DE	1623.81021	1459.82527	1637.34044	1637.06409	1607.52039	1637.84527	1644.11703	1641.72188
DW	1630.31406	1635.05615	941.33640	1599.64984	1619.89528	1612.20249	1630.78967	1624.19915
DW2	1630.67968	1625.27924	1608.49581	1334.16718	1624.20151	1617.24393	1617.12146	1619.83041
UC	1639.95031	1623.50301	1645.96672	1641.73681	1423.55414	1659.25441	1636.67786	1645.07085
UC2	1630.29116	1654.13184	1624.13223	1620.62960	1623.33770	1418.99023	1617.70019	1620.76252
UW	1626.17448	1630.44310	1621.41008	1632.72611	1637.43513	1631.73166	1586.03848	1631.16784
UW2	1630.49693	1643.19935	1631.95487	1609.40638	1636.77580	1625.31303	1628.78667	1547.33329

Data graphs generated for sub part 1

values between the two subgraphs. This means that even with the small part of the behavior logs, the driver can be distinguished, as the distance values remain close to its model graph, and very far from other drivers model graphs.

As a conclusion, we can say that graph matching provides a rigid framework for driver behavior analysis. The goal of the experiments performed so far is to emphasize the potential application of the graph-based analysis for comparing drivers behaviors and for driver identification. The results discussed above also show the applicability of this analysis method for investigating the occurrence of given driving patterns among multiple drivers. However, because of the lack of richer information about drivers in public data sets, we decided to not investigate the performance of the analysis. To do this, multiple trips of each driver must be available where a list of trips will be left for validation.

Table 5.9: Results for dissimilarity measurement of driving behavior graphs

	Data graph									
Model graph	Driver 1	Driver 2	Driver 3	Driver 4	Driver 5	Driver 6	Driver 7	Driver 8	Driver 9	Driver 10
Driver 1	0.00000	2667.9073	2616.13573	2669.14375	2653.68632	2637.25759	2661.75518	2710.32218	2676.1233	2637.3065
Driver 2	2686.49064	0.00000	2694.25277	2686.16298	2683.98214	2677.66216	2760.20485	2659.14922	2704.80207	2714.45177
Driver 3	2614.46906	2688.91944	0.00000	2718.34636	2687.89817	2576.34997	2685.76627	2624.34392	2692.77529	2690.7667
Driver 4	2678.64375	2690.66298	2713.34636	0.00000	2640.82925	2598.30617	2698.75724	2628.72022	2684.88464	2703.04625
Driver 5	2648.68632	2702.48214	2712.89817	2652.32925	0.00000	2653.33582	2637.85617	2672.41027	2667.53709	2560.87778
Driver 6	2617.75759	2659.16216	2547.34997	2618.55617	2634.83582	0.00000	2586.90357	2656.67801	2666.49394	2669.88293
Driver 7	2646.75518	2755.20485	2683.93294	2698.75724	2638.6895	2583.7369	0.00000	2682.26608	2723.00671	2672.3643
Driver 8	2702.98885	2658.64922	2621.34392	2619.72022	2652.91027	2628.84468	2659.43275	0.00000	2722.19666	2604.90962
Driver 9	2667.6233	2699.96874	2671.60862	2683.38464	2686.53709	2647.49394	2726.00671	2712.69666	0.00000	2719.16406
Driver 10	2624.8065	2725.45177	2683.10003	2697.04625	2590.62778	2676.38293	2668.95716	2620.40962	2721.91406	0.000000

	Data graph									
Model graph	Driver 1-2	Driver 2-2	Driver 3-2	Driver 4-2	Driver 5-2	Driver 6-2	Driver 7-2	Driver 8-2	Driver 9-2	Driver 10-2
Driver 1	713.22500	2605.45131	2603.38918	2602.50371	2617.54314	2580.84493	2580.42148	2583.55197	2595.53163	2572.20725
Driver 2	2593.57754	677.43333	2605.46276	2631.66398	2641.74516	2624.37151	2649.65437	2603.58968	2640.94722	2614.85026
Driver 3	2578.4289	2608.02063	714.75000	2607.80187	2597.57674	2572.54419	2584.27630	2608.67788	2573.69969	2587.17741
Driver 4	2637.88701	2626.07691	2600.21215	675.00000	2626.50458	2597.95216	2593.39431	2617.03762	2602.57058	2594.36093
Driver 5	2640.10147	2644.62756	2624.79041	2588.16854	675.50000	2605.69495	2581.58573	2582.03631	2606.38727	2592.05518
Driver 6	2614.77235	2617.23776	2545.22096	2594.63236	2613.45504	750.23493	2579.15088	2575.21094	2570.91860	2604.10185
Driver 7	2623.67602	2650.56190	2605.96448	2623.11524	2612.35935	2600.50066	702.55000	2632.59960	2644.20713	2623.32978
Driver 8	2609.98305	2639.16561	2541.25596	2606.23308	2600.69879	2576.31503	2591.33763	781.79167	2623.59755	2590.62971
Driver 9	2601.39237	2614.41667	2608.27233	2575.73592	2623.17206	2615.62885	2634.19960	2620.89214	687.43810	2619.26264
Driver 10	2596.41322	2632.71364	2623.99021	2560.73535	2599.19492	2583.31645	2620.50965	2588.41342	2633.36127	675.50000

	Data graph									
Model graph	Driver 1-1	Driver 2-1	Driver 3-1	Driver 4-1	Driver 5-1	Driver 6-1	Driver 7-1	Driver 8-1	Driver 9-1	Driver 10-1
Driver 1	1450.33333	2447.95915	2462.14831	2466.78137	2441.30388	2378.96910	2454.09820	2457.45627	2446.36975	2412.81182
Driver 2	2432.84421	1502.00000	2449.71237	2435.97083	2439.62571	2421.91012	2467.74327	2470.52302	2461.80000	2455.81560
Driver 3	2398.04457	2447.86040	1480.12341	2436.20827	2408.04582	2395.61786	2435.98883	2418.00964	2456.66397	2421.91792
Driver 4	2447.74937	2466.12500	2418.71296	1449.96667	2429.02652	2431.58474	2458.30823	2406.87102	2431.66508	2400.98891
Driver 5	2412.69692	2445.55842	2443.08242	2424.73203	1454.08681	2406.92939	2461.30566	2432.35672	2457.14353	2443.34068
Driver 6	2425.90615	2450.42514	2405.94466	2413.72572	2409.11848	1472.59087	2462.62734	2444.05536	2429.90672	2448.19715
Driver 7	2467.12207	2465.40043	2435.28348	2447.41462	2447.36751	2455.91987	1470.54064	2453.09295	2449.76267	2441.14434
Driver 8	2436.39287	2437.65833	2436.41550	2415.76441	2415.23323	2396.93515	2457.08281	1496.66667	2442.19548	2402.07264
Driver 9	2437.46142	2452.00000	2417.43595	2399.11964	2395.49706	2398.13854	2462.82698	2433.32821	1471.97500	2444.32468
Driver 10	2443.82327	2446.45741	2430.06606	2406.64242	2421.61128	2432.52104	2449.72782	2440.33655	2447.73688	1451.50000

5.3 vCar: the plateform for driving data visualization and analysis

Thanks to on-going digitization, cars are increasingly becoming data centers on wheels. The constantly growing number of sensors inside cars and the technology evolution towards higher resolutions of data types are challenging the scientific community to develop efficient algorithms and strategies for vehicle data management and analysis that ensure safe and comfortable driving. The development of such solid algorithms passes through

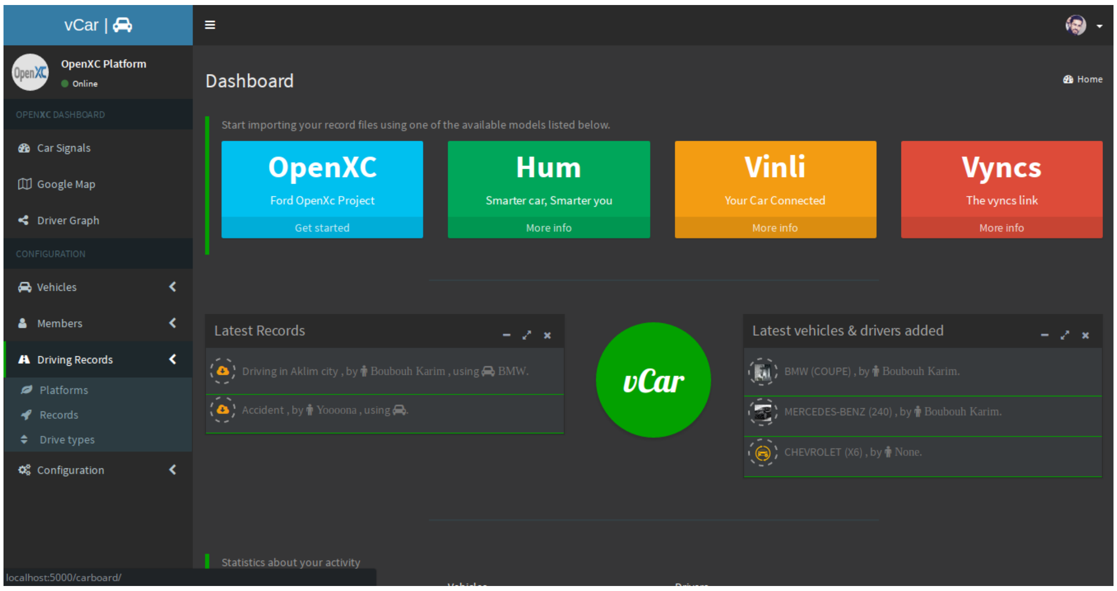


FIGURE 5.8: Home page of the prototype developed "vCar"

several steps, from data acquisition and storage to analysis and visualization, and requires many experiments to prove their applicability and efficiency. In order to simplify this tedious process, we started developing vCar[2], a platform that encompasses all these steps in one environment. The platform can be used as data storage, visualization and analysis tool, and most importantly as a platform to develop and test algorithms. It will offer researchers and developers a solid background to work in the same environment where data is globally available for the whole process of development and testing, ans will also be a nice tool for drivers to have feedback about their driving.

The vCar Platform is a modular web application developed using Python alongside with many technologies, the core provides basic modules such as importer, explorer, editor, etc., and it can be extended easily with a flexible plugin system. Developed algorithms can also be attached to the platform as plugins that can be reused or coupled with other applications to empower some functionalities of the car.

Currently the platform is still in an alpha stage but there is already a number of extensions developed including:

- Classification and scoring of driver behaviors
- Driver Distraction Detector.
- Driver Maneuver detection and anticipation.
- Traffic panels detection and informational panels translation.

[2]http://vcar.ai/

- Graph-based visualization of driver behavior

The platform is packed with a set of reusable modules developed to handle common tasks in the process of developing algorithms, and they are structured in such a way to be easily extended or replaced by other modules.

5.4 Graphs and driving Behavior Visualization

With the huge amount of data generated by car sensors, data visualization has become an important means for data interpretation. It consists of using visual graphics to visually communicate information, aiming for a better understanding and analysis of complex data. While frequency distribution graphs are used to analyze driving behavior patterns over time, the graph representation, as proposed in section 3.2, allows an abstract representation and makes the visual comparison easier.

We present in Figure 5.9 two driving behavior graphs created from the traces *"Downtown,Crosstown"* and *"East Downtown"* respectively. For simplicity, we constraint ourselves to graphs of 30 nodes. Just from comparing the two graphs visually, we can look for similarities in the driving behavior of the two drivers. Examples of the information we can get from the comparison are:

- There are some common states between the graphs of the two drivers: The nodes labeled 02310, 52310, 52311, 53311, 53411, 54411, 54421, 54333, 53333, 53233, 53223, 52223, 52323. This means that the drivers have some similarities in the visited states

- The behavior of the driver in Figure 5.9a is characterized by an increasing in the value of the *engine_speed* whereas an increasing in the value of *torque_at_transmission* is observed in the behavior of the driver in Figure 5.9b, until they both reach the node 02310.

- As regards the states in common, differences in the weights of the self loop edges is observed. This explained by the fact that the driver spends more time in the nodes with high weighted loop. For instance, the driver in Figure 5.9b spends more time in the nodes 52311 53411 5411 54421; while the one in Figure 5.9a tends to spend more time in the nodes labeled 52310, 53311, 54333, 53333.

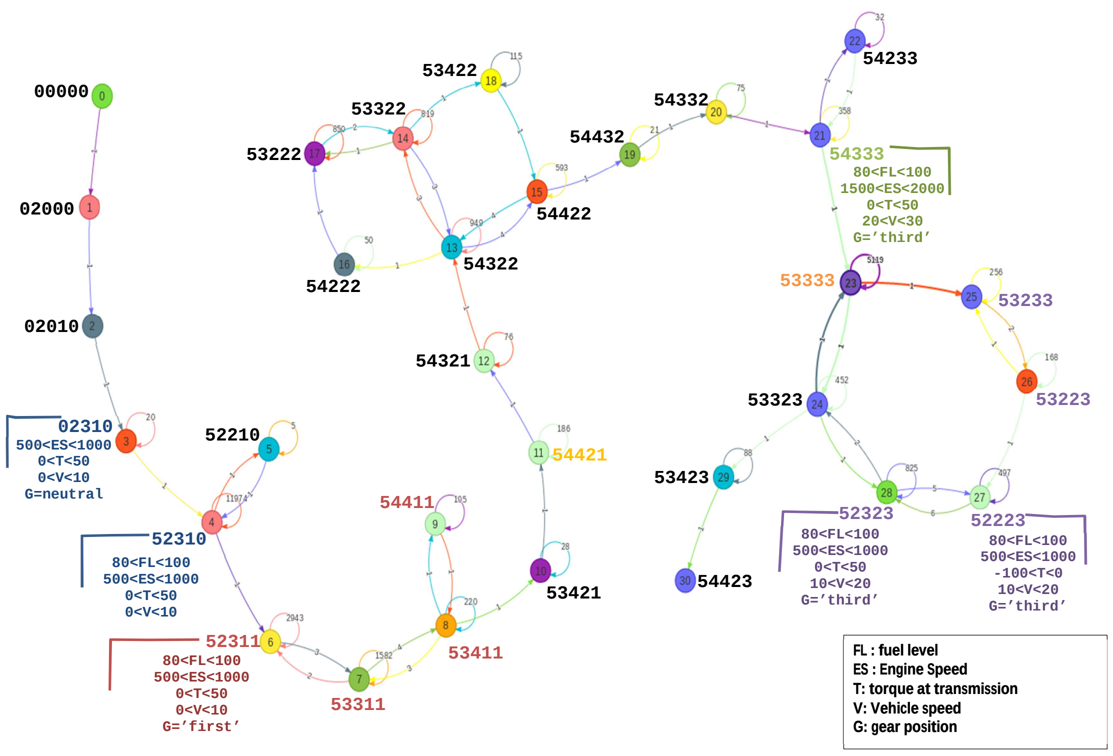
- Both drivers seem to have similar behavior in the node labeled 02310, which is always followed by node 52310; and the node 52311 followed by node 53311.

- While the driver in Figure 5.9b tends to leave node 52310 to only reach node 52311, the driver in Figure 5.9a sometimes passes by node 52210. This means that when in node 52310, the value of *torque_at_transmission* for the vehicle of the latter driver may reach a negative value before returning to its positive value and perform shifting.

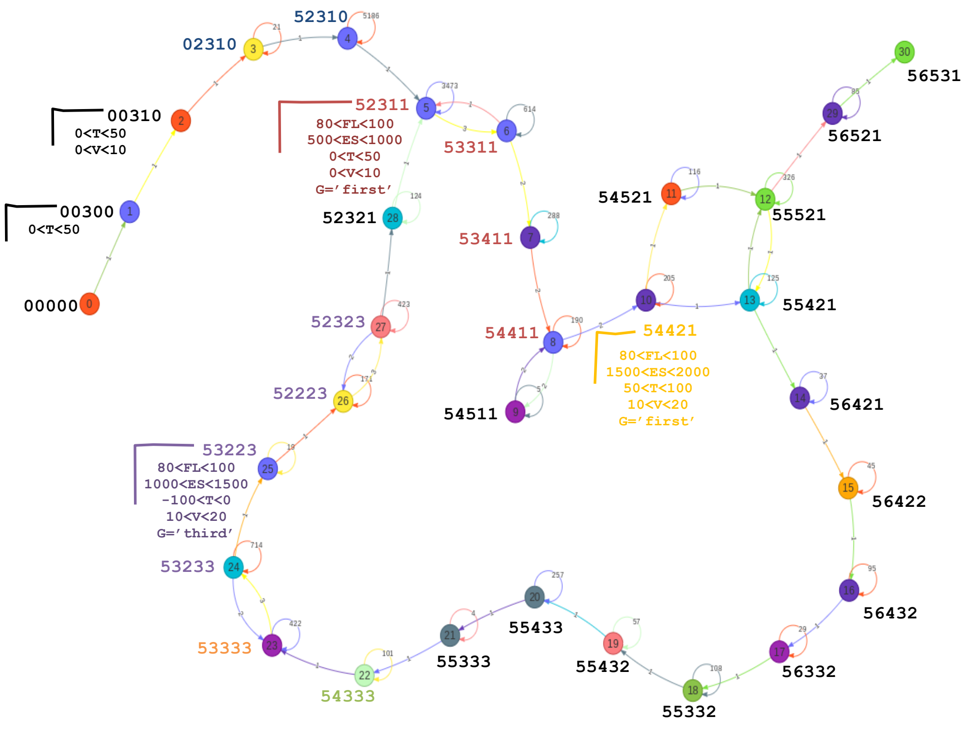
In a similar way, a more profound analysis can be conducted by analyzing the labels of the two graphs vertices.

5.5 Chapter summary

In this chapter, we introduced a new approach for analyzing drivers similarity using graph matching technique. In this approach of analysis, drivers graphs are used as a model representing drivers behaviors, and graph edit distance is used as a metric to study the similarity between the drivers. The chapter can be divided into two main parts. We dedicate the first part to present the graph matching problem; we recalled its basic concept and presented the graph edit distance and its computation algorithms. We also presented the software used for graph edit distance computation. The second part is dedicated to the performed experimentations; we apply graph representation and graph matching based analysis on two public data sets, namely the OpenXC data set and hciLab data set. In each data set, graphs representing individual drivers are generated from data and drivers similarity analysis has been performed based on the graph edit distances. Obtained results show the applicability of graph-based analysis in driver behavior studies and the potential of using graph matching technique for driver identification. The chapter also straightened out the importance of graph-based driver behavior representation in facilitating data visualization.



(A) Driving behavior graph for the Downtown-Crosstown scenario.



(B) Driving behavior graph for the East Downtown scenario.

FIGURE 5.9: Example of driving behavior graphs visualization.

Chapter 6

Conclusion and Perspectives

"I think if you do something and it turns out pretty good, then you should go do something else wonderful, not dwell on it for too long. Just figure out what's next."

– Steve Jobs, *NBC Nightly News, May 2006*

There is a growing trend towards using technology to improve driver behavior and traffic safety. The recent computerization of cars, together with the development of sensor technologies and car communication devices have revolutionize the way researchers and analysts deal with driving behavior. Driver behavior being a main cause of road injuries, more attention is given to solutions and approaches that offer more advanced analysis of driving behavior. Driving behavior analytics have thus emerged as an important means of improving driving safety and drivers comfort; they process data generated by vehicles, solely or combined with road data, and transform it to valuable information to gain a better understanding of drivers behavior. Depending on the analysis's goals, different mathematical and statistical models have been used and numerous analytics approaches have emerged consequently.

In this PhD thesis work, we investigated the application of some well-known approaches to driver behavior modeling and analysis. The developed methodology uses models of driver behavior, established from driving data, to analyze driver behavior and create drivers profiles. In general, a data analysis process involves three common tasks: ***acquisition*** task, ***preprocessing***, and ***modeling and analysis***. With respect to this process, the contributions proposed relate to the last two tasks.

Based on the overview provided in chapter 2, we presented recent generations of vehicles as intelligent entities with numerous electronic components that offer a wealth of information about driver behavior. Since it depends strongly on personal and situational factors, studying driver behavior requires consideration of the three interactive parts of the traffic system, the driver, vehicle and road environment. We then concluded that a driver behavior modeling framework should capture the causal relationships between the driver actions and the driving situations determined by the states of the vehicle and environment. Moreover, methods and techniques to deal with the quantity and heterogeneity of vehicle generated data should be implemented. We have also pointed out that driver behavior modeling has been addressed from many perspectives, however, in this work we aim for a general characterization of driver behavior, i.e driver profiling by proposing an approach to infer individual models of drivers from driving behavior logs.

In chapter 3 we have presented the main contribution which consists of a complete methodology for driver modeling and profiling. We first provided an overview about the different approaches considered for driver profiling. Generally, this aim is achieved by processing low driving data by means of statistical tools, which are most of times combined with methods from artificial intelligence and machine learning. We have distinguished between two approaches of driver profiling, direct methods that extract driver characteristics directly from data and model-based methods that infer driver characteristic from a model of driver behavior established from data. Among the model-based methods, there is probabilistic graphical models well known for their merging of graph theory and probability theory. They provide a helpful framework for modeling driving behavior: the language of graphs facilitates the representation of the relationships within the driver, vehicle, environment system, while the probability allows the representation of uncertainty. In our model-based approach for driver profiling, we proposed two modeling formalisms, PRHIOA and ADG, which besides their representative power have allowed us to perform profound analyses using formal verification and graph matching techniques. In order to construct the models of driver behavior, we used the learning automata algorithm, one of the most powerful approaches in the field of reinforcement learning, to learn the transitions probability of the model based on observations of the driver behavior. To enrich the model with contextual information about the driving environment, a study of the French traffic code have been conducted to retrieve the different rules regulating driving behavior on roads. These rules was represented by constraints on the variables of the driver behavior model, and added to the model states as *driving context*. The chapter has also addressed the data size problem. *Abstraction using numerical intervals domain* has been presented as a driving data abstraction framework, and distinguished between two approaches, a static abstraction where the abstract domain is defined manually and a dynamic abstraction where machine learning is used to retrieve the abstract domain from data. To summarize, the proposed approach implements a set of formal methods for data abstraction and modeling; it can create models of drivers behavior from vehicle sensors data and is able, thanks to the proposed representation of the driving environment, to capture driver behavior with respect to traffic rules. Based on this modeling approach, two analyses were proposed.

The first analysis, presented in chapter 3, addressed the problem of drivers compliance with safety measures and traffic rules. Our contribution consists of using formal verification, by model checking, of driver behavior. This was motivated by the broad application of model checking for analyzing behaviors of systems in various domains, the proven rigor and accuracy of its analysis, and our intention in investigating the applicability of this technique in human driving behavior verification. Using a model checking tool, to which the driver behavior automaton model and desired behavior expressed as logic formulas were fed, the analysis provided an automatic verification of driving behavior. The proposed analysis methodology has been used to analyze some driving habits, mainly tailgating, turning and braking/acceleration behavior. The obtained results have successfully reflected the real behavior of the driver, displayed in driving signals plots. In addition, we studied the impact of abstraction on the analysis results, and concluded the verification accuracy depends primarily on the abstract domain used in the abstraction step and on properties parametrization consequently.

Chapter 4 was devoted to the second analysis, our last contribution, which consists of graph matching as a tool for analyzing drivers similarity. Graph matching being successfully applied for pattern recognition, the objective was to investigate their applicability in driving behavior studies for driving pattern analysis. We applied graph edit distance to compute the dissimilarity between graphs of drivers from two data sets. The results obtained so far confirmed the high dissimilarity between drivers behavior and showed the potential of the proposed approach in driver identification. We concluded that the advantage of graph matching is to provide a rigid framework for analyzing drivers similarity, however, richer data sets are needed to explore other potential conclusions.

The contributions proposed in this work open new perspectives to be explored. First, we intend to make the implemented methods accessible to researchers and academia through the vcar platform. We also intend to investigate their application on more voluminous data of heterogeneous drivers with multiple trips. The driver behavior research was recently marked by the appearance of large databases of open driving data, with recorded videos of the driving environment. An important extension of our work is to explore video processing and automatic annotation techniques to incorporate video data in our modeling framework. We are also motived to develop a commercial application of our contributions. For this we have explored different application areas that we presented in the remaining paragraphs.

One of the important applications that we have considered is that of a monitoring system for driver's education purposes. Monitoring systems have been recently an important trend for measuring driving styles. Monitoring systems have been mostly used in fleet management[38], and they have shown promising results confirming the contributing role of monitoring in the improvement of a range of safety-related behaviors. Therefore for road safety purposes, monitoring systems must allow the assessment of the safety of the driver behavior in different traffic situations. Generally, the existing monitoring systems focus on the record of time-step driving data (acceleration, speed, etc.) while the contextual details about the driving conditions are either not recorded or provided as videos of the external environment. Moreover, the large size of the generated files by these systems makes their analysis more complicated and increases their processing time. Comparing to the existing monitoring systems, driving data together with the contextual constraints of the environment are recorded as one mathematical model, and tools to analyze the convenience of recorded data to driving conditions are provided. In addition, smaller log files are generated, as only abstracted states of driver-vehicle with the emphasis on the input/output nature of the driving behavior are kept instead of storing the time step data of driving. As driving safety and performance has to be evaluated according to the contextual driving environment, focus has been put on modeling and recording of the driving behavior observed with the contextual information expressed as context constraints, which makes the analysis and verification process less complicated. Therefore, applications can be found in different other sectors with driving safety and driving analysis concerns can be found.

It can be used for example by insurance companies to reward/punish their customers according to the performance of their driving. Insurance companies are recently using driving monitoring to provide personal payment for car insurance based on driver habits, and to encourage their customer to improve their driving performance. The systems that are used until now track the driver's habits (braking and other data) to figure out if he will receive a good driver discount or not. The evaluation of driving habits is made primarily on data from the car engine computer. Yet the system that we proposed here combines information from the vehicle with the information from environment to evaluate how the driver behaves in different driving situations. Information about environment consists of road types, traffic regulation, behavior of other vehicles, obstacles, etc. Insurers can then evaluate the driver habits in respect to the perceived environment instead of recognizing it from driving data. Examples of behavior insurers can evaluate are: the velocity of the car taking into account the road type or curvature and an overtaking maneuver considering the other vehicles on the road.

As a part of our approach focus on modeling traffic rules, traffic police can use it for the detection and generation of infringement notices. Recently, multiple intelligent solutions have been proposed to support traffic enforcement such as speed cameras that capture speeding vehicles, in-vehicle data recorders that record driving data to be used in a crash analysis,... These solutions contribute to the improvement of road safety without requiring an increase in human police resources. Our approach of modeling offer a way to detect road rules infringement without the need of more intelligent road infrastructure as every vehicle will be able to auto-detect and record any disobedience to road rules and send data to police center for evaluation. The formal verification proposed will facilitate the evaluation of recorded data.

Another important application is to use the driver behavior model as a basis of a personalized driving assistance. The automotive technologies available in recent cars can be a source of a lot of information that can be used to learn driver preferences either in terms of driving, navigational or infotainment tasks. This information gathered from the in-vehicle technologies can be used to enrich the personal behavior model presented in this work. Because this probabilistic model can predict the driver behavior in the different situations that were experienced previously by the driver, we can anticipate and avoid risky behaviors and assistance functions can be implemented by the design of the controller unit to take control of the physical plant if a risky behavior is anticipated.

List of Publications

International Journals (Peer reviewed)

1. Afaf Bouhoute, Rachid Oucheick, Karim Boubouh and Ismail Berrada (2018). "Advanced Driving Behavior Analytics for an Improved Safety Assessment and Driver Fingerprinting". IEEE transactions on intelligent transportation systems.
2. Salah Eddine Ramah, Afaf Bouhoute, Karim Boubouh and Ismail Berrada (2017). "One Step Further Towards Real-Time Driving Maneuver Recognition Using Phone Sensors". IEEE transactions on intelligent transportation systems. (Under review)
3. Abdellah EL Mekki, Afaf Bouhoute and Ismail Berrada (2018). "Improving Driver Identification for the Next Generation of In-vehicle Software Systems". Expert Systems with Applications. (Submitted)

Books and Book chapters

1. Afaf Bouhoute, Ismail Berrada and Mohammed El Kamili. "A Formal Driving Behavior Model for Intelligent Transportation Systems". In Networked Systems (pp. 298-312). DOI:10.1007/978-3-319-09581-3_21. Springer International Publishing. 2014.
2. Afaf Bouhoute, Rachid Oucheikh and Ismail Berrada. "Context-Aware Driving Assistance: An Approach for Monitoring-Based Modeling and Self-learning Cars". In Advances in Ubiquitous Networking 2 (pp. 587-597). DOI:10.1007/978-981-10-1627-1_46. Springer Singapore. 2017

International Conferences (Peer reviewed)

1. Afaf Bouhoute, Ismail Berrada and Mohammed El Kamili . "A formal model of human driving behavior in vehicular networks". In Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International (pp. 231-236). DOI: 10.1109/IWCMC.2014.6906362 IEEE. 2014

2. Afaf Bouhoute, Rachid Oucheikh and Ismail Berrada. "Monitoring-based driving behavior modeling for a self-learning car". In 22nd Intelligent Transportation System (ITS) world congress 2015, from 5-9 October, Bordeaux, France. 2015

3. Afaf Bouhoute, Rachid Oucheikh, Yassine Zahraoui and Ismail Berrada. "A holistic approach for modeling and verification of human driver behavior". In Wireless Networks and Mobile Communications (WINCOM), 2015 International Conference on (pp. 1-7). DOI: 10.1109/WINCOM.2015.7381323. IEEE. 2015

4. Kawtar Sefrioui Boujemaa, Ismail Berrada, Afaf Bouhoute and Karim Boubouh, "Traffic sign recognition using convolutional neural networks". 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM), Rabat, 2017, pp. 1-6. DOI: 10.1109/WINCOM.2017.8238205. 2017

5. Abdelhak Zouzou, Afaf Bouhoute, Karim Boubouh, Mohammed El Kamili and Ismail Berrada. "Predicting lane change maneuvers using inverse reinforcement learning". 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM), Rabat, 2017, pp. 1-7. DOI: 10.1109/WINCOM.2017.8238204. 2017

National Conferences

1. Afaf Bouhoute and Ismail Berrada (2013). A New Approach for Multisource Multicast Routing in VANETs. Deuxième Journées Doctorales en Systèmes d'Information, Réseaux et Télécoms (JDSIRT 2013).

2. Afaf Bouhoute, Imane Daha, Ismail Berrada and Lahcen Omari (2013). Driver Behavior Modeling Using I/O Automata. In the First International Workshop on Wireless Networks and Mobile COMmunications (WINCOM'13).

3. Afaf Bouhoute, Rachid Oucheikh, Ismail Berrada, and Lahcen Omari (2014). A New Formal Approach to model Human Driving Behavior in Vehicular Networks. In the first International Workshop on WIreless Technologies, embedded and intelligent Systems (WITS'14).

4. Imane Rahmouni, Ahmed El Ouadrhiri, Afaf Bouhoute, Ismail Berrada and Mohammed El-kamili (2014). Energy and Buffer Management in Delay Tolerant Network. In the 2nd International Workshop On RFID And Adaptive Wireless Sensor Networks (RAWSN'14).

Bibliography

[1] Lentin Joseph. *ROS Robotics Projects*. Packt Publishing, March 2017. ISBN 978-1-78355-471-3.

[2] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Looking at humans in the age of self-driving and highly automated vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):90–104, 2016.

[3] John A Michon. A critical view of driver behavior models: What do we know, what should we do. *Human behavior and traffic safety*, pages 485–520, 1985.

[4] Patrick Cousot. Mathematical foundations: (6) abstraction - part i. University Lecture, 2005. URL http://web.mit.edu/16.399/www. Accessed: 2016-05-04.

[5] Kazuya Takeda, John HL Hansen, Pinar Boyraz, Lucas Malta, Chiyomi Miyajima, and Hüseyin Abut. International large-scale vehicle corpora for research on driver behavior on the road. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1609–1623, 2011.

[6] Vincenzo Carletti. *Exact and Inexact Methods for Graph Similarity in Structural Pattern Recognition PhD thesis of Vincenzo Carletti*. PhD thesis, Université de Caen; Universita degli studi di Salerno, 2016.

[7] The number of cars worldwide is set to double by 2040. URL https://www.weforum.org/agenda/2016/04/the-number-of-cars-worldwide-is-set-to-double-by-2040. Accessed: 2017-09-11.

[8] Road traffic injuries. URL http://www.who.int/mediacentre/factsheets/fs358/en/. Accessed: 2018-01-29.

[9] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, 2015.

[10] J Scott Brennen and Daniel Kreiss. Digitalization. *The International Encyclopedia of Communication Theory and Philosophy*, 2016.

[11] The digital revolution is creating new opportunities for leadership. URL https://www.weforum.org/agenda/2016/10/the-digital-revolution-is-creating-new-opportunities-for-leadership. Accessed: 2017-09-11.

[12] Digitization, digitalization and digital transformation: the differences. URL https://www.i-scoop.eu/digitization-digitalization-digital-transformation-disruption/. Accessed: 2017-09-11.

[13] Bruce Weindelt. Digital transformation of industries: Automotive industry. In *World Economic Forum in collaboration with Accenture*, page 4, 2016.

[14] Christine Knackfuß Andreas Gissler, Clemens Oertel and Franziska Kupferschmidt. Driving digitization in the auto industry. URL https://www.accenture.com/t00010101T000000__w__/es-es/_acnmedia/PDF-10/Accenture-Strategy-Driving-Digitization-Auto-Industry-1.pdf.

[15] Autonomous cars: Everything you need to know about market and research trends. URL https://www.greyb.com/autonomous-cars/. Accessed: 2017-10-10.

[16] Kimberly Madia. Big data on wheels. URL http://www.ibmbigdatahub.com/blog/big-data-wheels. Accessed: 2016-04-20.

[17] Evangelos Simoudis. The automotive industry's big data challenge. URL https://www.enterpriseirregulars.com/104636/. Accessed: 2016-04-24.

[18] i4drive: Advanced driving experience. URL https://www.i4drive.com/driving-analytics. Accessed: 2016-11-16.

[19] Cloud made: Car & driver analytics. URL http://cloudmade.com/solutions/car-driver-analytics. Accessed: 2016-11-25.

[20] Zendrive. URL https://www.zendrive.com. Accessed: 2016-09-18.

[21] Amodo: Advanced driving behavior analytics. URL https://www.amodo.eu. Accessed: 2016-11-22.

[22] Azim Eskandarian. *Handbook of intelligent vehicles*. Springer London, 2012.

[23] Dennis K Nilsson, Ulf E Larson, and Erland Jonsson. Creating a secure infrastructure for wireless diagnostics and software updates in vehicles. In *International Conference on Computer Safety, Reliability, and Security*, pages 207–220. Springer, 2008.

[24] Claudio Rosito Jung and Christian Roberto Kelber. A lane departure warning system based on a linear-parabolic lane model. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 891–895. IEEE, 2004.

[25] Saif Al-Sultan, Ali H Al-Bayatti, and Hussein Zedan. Context-aware driver behavior detection system in intelligent transportation systems. *Vehicular Technology, IEEE Transactions on*, 62(9):4264–4275, 2013.

[26] Monika, Dev Dutt Yadav, N Avinash, Ho Gi Jung, and Hyuckmin Na. Real time traffic sign recognition system as speed regulator in iav. In *IICAI*, pages 1936–1951, 2009.

[27] Bjorn Johansson. Road sign recognition from a moving vehicle. Technical report, 2002.

[28] Amnon Shashua, Yoram Gdalyahu, and Gaby Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 1–6. IEEE, 2004.

[29] N Minoiu Enache, Mariana Netto, Said Mammar, and Benoît Lusetti. Driver steering assistance for lane departure avoidance. *Control engineering practice*, 17 (6):642–651, 2009.

[30] Joel C McCall and Mohan M Trivedi. Driver behavior and situation aware brake assistance for intelligent vehicles. *Proceedings of the IEEE*, 95(2):374–387, 2007.

[31] MIKE Hawes. Connected and autonomous vehicles-the uk economic opportunity. *En ligne http://www. smmt. co. uk/wp-content/uploads/sites/2/CRT036586F-Connected-and-Autonomous-Vehicles—-The-UK-Economic-Opportu*, 1, 2015.

[32] SS Sunder. Foundations for innovation in cyber-physical systems. In *proceedings of the NIST CPS Workshop, Chicago, IL, USA*, volume 13, 2012.

[33] Heiner Bubb. Ergonomie. *3. Aufl., Kap. 5.3 Systemergonomische Gestaltung, S. 390–420). München: Carl Hanser Verlag*, 1993.

[34] Marina Plavšic. Analysis and modeling of driver behavior for assistance systems at road intersections. *Dr.-Ing. dissertation, Faculty Mech. Eng., TU München, München, Germany*, 2010.

[35] Daiheng Ni. Traffic flow theory: Characteristics, experimental methods, and numerical techniques. 2015.

[36] Brian Philips and Tom Morton. Making driving simulators more useful for behavioral research-simulator characteristics comparison and model-based transformation: Summary report. Technical report, 2015.

[37] J De Winter, PM Van Leeuwen, and Riender Happee. Advantages and disadvantages of driving simulators: a discussion. In *Proceedings of Measuring Behavior*, pages 47–50, 2012.

[38] AA Marzooqi. Road safety system for monitoring fleet drivers. In *Safer driving, reducing risks, crashes and casualties. Proceedings of the 68th road safety congress held blackpool, 3-5 march 2003*, 2003.

[39] Use less, pay less: A simple concept that reduces the cost of car insurance now available to michigan and oregon drivers. URL `https://www.progressive.com/newsroom/article/2007/january/tripsense-mich-ore/`. Accessed: 2016-09-11.

[40] Ahmed M Elmahalawy. A car monitoring system for self recording traffic violations. *world*, 4:5, 2014.

[41] Omid Nejati. Smart recording of traffic violations via m-rfid. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1–4. IEEE, 2011.

[42] Tomer Toledo and Tsippy Lotan. In-vehicle data recorder for evaluation of driving behavior and safety. *Transportation Research Record: Journal of the Transportation Research Board*, (1953):112–119, 2006.

[43] Thomas A Dingus, SG Klauer, VL Neale, A Petersen, SE Lee, JD Sudweeks, MA Perez, J Hankey, DJ Ramsey, S Gupta, et al. The 100-car naturalistic driving study, phase ii-results of the 100-car field experiment. Technical report, 2006.

[44] Pinar Boyraz, Amardeep Sathyanarayana, JL Hansen, and E Jonsson. Driver behavior modeling using hybrid dynamic systems for 'driver-aware'active vehicle safety. *Proc. Enhanced Saf. Veh*, pages 1–8, 2009.

[45] Najah AbuAli and Hatem Abou-zeid. Driver behavior modeling: Developments and future directions. *International Journal of Vehicular Technology*, 2016, 2016.

[46] Kaan Ozbay, Aleek Datta, and Pushkin Kachroo. Modeling route choice behavior with stochastic learning automata. *Transportation Research Record: Journal of the Transportation Research Board*, (1752):38–46, 2001.

[47] Yetis Sazi Murat and Nurcan Uludag. Route choice modelling in urban transportation networks using fuzzy logic and logistic regression methods. 2008.

[48] Bingrong Sun and Byungkyu Brian Park. Route choice modeling with support vector machine. *Transportation Research Procedia*, 25:1811–1819, 2017.

[49] SooBeom Lee, YoungChan Kim, Moon Namgung, and JangWook Kim. Development of route choice behavior model using linkage of neural network and genetic algorithm with trip information. *KSCE Journal of Civil Engineering*, 9(4):321–327, 2005.

[50] Aly M Tawfik, Hesham A Rakha, and Shadeequa D Miller. Driver route choice behavior: Experiences, perceptions, and choices. In *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pages 1195–1200. IEEE, 2010.

[51] Yuanyuan Bao and Wai Chen. A personalized route search method based on joint driving and vehicular behavior recognition. In *Wireless Symposium (IWS), 2016 IEEE MTT-S International*, pages 1–6. IEEE, 2016.

[52] Aly M Tawfik, Hesham A Rakha, et al. Human aspects of route choice behavior: Incorporating perceptions, learning trends, latent classes, and personality traits in the modeling of driver heterogeneity in route choice behavior. 2012.

[53] Toru Kumagai, Yasuo Sakaguchi, Masayuki Okuwa, and Motoyuki Akamatsu. Prediction of driving behavior through probabilistic inference. In *Proc. 8th Intl. Conf. Engineering Applications of Neural Networks*, pages 117–123, 2003.

[54] Isam A Kaysi and Ali S Abbany. Modeling aggressive driver behavior at unsignalized intersections. *Accident Analysis & Prevention*, 39(4):671–678, 2007.

[55] Georges S Aoude, Vishnu R Desaraju, Lauren H Stephens, and Jonathan P How. Behavior classification algorithms at intersections and validation using naturalistic data. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 601–606. IEEE, 2011.

[56] Stéphanie Lefèvre, Javier Ibañez-Guzmán, and Christian Laugier. Context-based estimation of driver intent at road intersections. In *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2011 IEEE Symposium on*, pages 67–72. IEEE, 2011.

[57] Nobuyuki Kuge, Tomohiro Yamamura, Osamu Shimoyama, and Andrew Liu. A driver behavior recognition method based on a driver model framework. Technical report, SAE Technical Paper, 2000.

[58] Vadim A Butakov and Petros Ioannou. Personalized driver/vehicle lane change models for adas. *IEEE Transactions on Vehicular Technology*, 64(10):4422–4431, 2015.

[59] Lina Lwambagaza. Modeling older driver behavior on freeway merging ramps. 2016.

[60] Alexandra Kondyli and Lily Elefteriadou. Driver behavior at freeway-ramp merging areas based on instrumented vehicle observations. *Transportation Letters*, 4 (3):129–142, 2012.

[61] Yasuo Sakaguchi, Masayuki Okuwa, Ken'ichiro Takiguchi, and Motoyuki Akamatsu. Measuring and modeling of driver for detecting unusual behavior for driving assistance. In *Proceedings of 18 th International Conference on Enhanced Safety of Vehicles*, 2003.

[62] Nuria Oliver and Alexander P Pentland. Driver behavior recognition and prediction in a smartcar. In *AeroSense 2000*, pages 280–290. International Society for Optics and Photonics, 2000.

[63] Ashesh Jain, Hema S Koppula, Bharad Raghavan, Shane Soh, and Ashutosh Saxena. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3182–3190, 2015.

[64] Avi Rosenfeld, Zevi Bareket, Claudia V Goldman, Sarit Kraus, David J LeBlanc, and Omer Tsimhoni. Learning driver's behavior to improve the acceptance of adaptive cruise control. In *IAAI*, 2012.

[65] Wenshuo Wang, Junqiang Xi, and Huiyan Chen. Modeling and recognizing driver behavior based on driving data: a survey. *Mathematical Problems in Engineering*, 2014, 2014.

[66] Fridulv Sagberg, Selpi, Giulio Francesco Bianchi Piccinini, and Johan Engström. A review of research on driving styles and road safety. *Human factors*, 57(7): 1248–1275, 2015.

[67] Gys Albertus Marthinus Meiring and Hermanus Carel Myburgh. A review of intelligent driving style analysis systems and related artificial intelligence algorithms. *Sensors*, 15(12):30653–30682, 2015.

[68] Amardeep Sathyanarayana, Pinar Boyraz, and John HL Hansen. Driver behavior analysis and route recognition by hidden markov models. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 276–281. IEEE, 2008.

[69] N Dapzol. Driver?s behavior modeling using the hidden markov model formalism. young researcher seminar of the european conference of transport research institute. 2005.

[70] Wenshuo Wang and Junqiang Xi. A rapid pattern-recognition method for driving styles using clustering-based support vector machines. In *American Control Conference (ACC), 2016*, pages 5270–5275. IEEE, 2016.

[71] Dorsa Sadigh, Katherine Driggs-Campbell, Alberto Puggelli, Wenchao Li, Victor Shia, Ruzena Bajcsy, Alberto L Sangiovanni-Vincentelli, S Shankar Sastry, and Sanjit A Seshia. Data-driven probabilistic modeling and verification of human driver behavior. *Formal Verification and Modeling in Human-Machine Systems*, 2014.

[72] Zoran Constantinescu, Cristian Marinoiu, and Monica Vladoiu. Driving style analysis using data mining techniques. *International Journal of Computers Communications & Control*, 5(5):654–663, 2010.

[73] Asher Bender, Gabriel Agamennoni, James R Ward, Stewart Worrall, and Eduardo M Nebot. An unsupervised approach for inferring driver behavior from naturalistic driving data. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3325–3336, 2015.

[74] Bryan Higgs and Montasir Abbas. Segmentation and clustering of car-following behavior: Recognition of driving patterns. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):81–90, 2015.

[75] Akshay Sonawane, Saurabh Dhabe, Utkarsh Nadgouda, Vipul Jadhav, Gopika V Mane, and UG Scholar. A systematic approach for real-time clustering and analysis of drivers' driving behavior using obd-ii. *International Journal of Engineering Science*, 5047, 2016.

[76] Ishika Zonina Towfic. A method for classifying driver performance. 2014.

[77] Yuan Liao, Shengbo Eben Li, Guofa Li, Wenjun Wang, Bo Cheng, and Fang Chen. Detection of driver cognitive distraction: An svm based real-time algorithm and its comparison study in typical driving scenarios. In *Intelligent Vehicles Symposium (IV), 2016 IEEE*, pages 394–399. IEEE, 2016.

[78] German Castignani, Raphaël Frank, and Thomas Engel. An evaluation study of driver profiling fuzzy algorithms using smartphones. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–6. IEEE, 2013.

[79] Ayse Cisel Aras and Ismail Gocer. Driver rating based on interval type-2 fuzzy logic system. *IFAC-PapersOnLine*, 49(11):95–100, 2016.

[80] Sei-Wang Chen, Chiung-Yao Fang, and Chih-Ting Tien. Driving behaviour modelling system based on graph construction. *Transportation research part C: emerging technologies*, 26:314–330, 2013.

[81] Yantao Li, Fengtao Xue, Lei Feng, and Zehui Qu. A driving behavior detection system based on a smartphone's built-in sensor. *International Journal of Communication Systems*, 30(8), 2017.

[82] German Castignani, Thierry Derrmann, Raphaël Frank, and Thomas Engel. Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study. *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[83] Mucahit Karaduman and Haluk Eren. Classification of road curves and corresponding driving profile via smartphone trip data. In *Artificial Intelligence and Data Processing Symposium (IDAP), 2017 International*, pages 1–7. IEEE, 2017.

[84] Miro Enev, Alex Takakuwa, Karl Koscher, and Tadayoshi Kohno. Automobile driver fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016(1): 34–50, 2016.

[85] Minh Van Ly, Sujitha Martin, and Mohan M Trivedi. Driver classification and driving style recognition using inertial sensors. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1040–1045. IEEE, 2013.

[86] Dongyao Chen, Kyong-Tak Cho, and Kang G Shin. Mobile imus reveal driver's identity from vehicle turns. *arXiv preprint arXiv:1710.04578*, 2017.

[87] Abdul Wahab, Tan Chin Keong, Hüseyin Abut, and Kazuya Takeda. Driver recognition system using fnn and statistical methods. In *Advances for In-Vehicle and Mobile Systems*, pages 11–23. Springer, 2007.

[88] Chiyomi Miyajima, Yoshihiro Nishiwaki, Koji Ozawa, Toshihiro Wakita, Katsunobu Itou, Kazuya Takeda, and Fumitada Itakura. Driver modeling based on driving behavior and its evaluation in driver identification. *Proceedings of the IEEE*, 95(2):427–437, 2007.

[89] Angela Burton, Tapan Parikh, Shannon Mascarenhas, Jue Zhang, Jonathan Voris, N Sertac Artan, and Wenjia Li. Driver identification and authentication with active behavior modeling. In *Network and Service Management (CNSM), 2016 12th International Conference on*, pages 388–393. IEEE, 2016.

[90] Byung Il Kwak, JiYoung Woo, and Huy Kang Kim. Know your master: Driver profiling-based anti-theft method. In *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, pages 211–218. IEEE, 2016.

[91] Patrick Cousot and Radhia Cousot. Basic concepts of abstract interpretation. In *Building the Information Society*, pages 359–366. Springer, 2004.

[92] Patrick Cousot and Radhia Cousot. Abstract interpretation: past, present and future. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 2. ACM, 2014.

[93] Wikipédia. Vitesse maximale autorisée sur route en france — wikipédia, l'encyclopédie libre, 2016. URL http://fr.wikipedia.org/w/index.php?title=Vitesse_maximale_autoris%C3%A9e_sur_route_en_France&oldid=131825962. En ligne; Page disponible le 4-décembre-2016.

[94] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[95] Eduardo Romera, Luis M Bergasa, and Roberto Arroyo. Need data for driver behaviour analysis? presenting the public uah-driveset. In *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pages 387–392. IEEE, 2016.

[96] Nancy Lynch, Roberto Segala, and Frits Vaandrager. Hybrid i/o automata. *Information and Computation*, 185(1):105–157, 2003.

[97] Thomas A Henzinger and Peter W Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1):369–392, 1999.

[98] A. Puri and P. Varaiya. Decidable hybrid systems. *Mathematical and Computer Modelling*, 23(11):191 – 202, 1996. ISSN 0895-7177. doi: http://dx.doi.org/10.1016/0895-7177(96)00072-6. URL http://www.sciencedirect.com/science/article/pii/0895717796000726.

[99] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13(1):113–129, 2010.

[100] Lorenzo Livi and Antonello Rizzi. The graph matching problem. *Pattern Analysis and Applications*, 16(3):253–283, 2013.

[101] Kumpati S Narendra and Mandayam AL Thathachar. Learning automata-a survey. *IEEE Transactions on systems, man, and cybernetics*, (4):323–334, 1974.

[102] Cem Unsal, Pushkin Kachroo, and John S Bay. Multiple stochastic learning automata for vehicle path control in an automated highway system. *IEEE Transactions on Systems, Man, and Cybernetics-part A: systems and humans*, 29(1): 120–128, 1999.

[103] Mohammad S Obaidat, Georgios I Papadimitriou, and Andreas S Pomportsis. Efficient fast learning automata. *Information Sciences*, 157:121–133, 2003.

[104] Cem Unsal. *Intelligent navigation of autonomous vehicles in an automated highway system: Learning methods and interacting vehicles approach.* PhD thesis, Virginia Tech, 1998.

[105] Kumpati S Narendra and Mandayam AL Thathachar. *Learning automata: an introduction.* Courier Dover Publications, 2012.

[106] Cristina Olaverri. Behavior signal processing laboratory [its research lab]. *Intelligent Transportation Systems Magazine, IEEE*, 8(1):72–76, 2016.

[107] Christel Baier, Joost-Pieter Katoen, and Kim Guldstrand Larsen. *Principles of model checking.* MIT press, 2008.

[108] Christel Baier. On algorithmic verification methods for probabilistic systems. *Universität Mannheim*, 1998.

[109] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal aspects of computing*, 6(5):512–535, 1994.

[110] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[111] Kei Igarashi, Kazuya Takeda, Fumitada Itakura, and Hüseyin Abut. Is our driving behavior unique? In *DSP for in-vehicle and mobile systems*, pages 257–274. Springer, 2005.

[112] Rishi Pal Singh. Application of graph theory in computer science and engineering. *International Journal of Computer Applications*, 104(1), 2014.

[113] Karsten Borgwardt and Oliver Stegle. Computational approaches for analysing complex biological systems, 2010.

[114] Charu C Aggarwal and Haixun Wang. Graph data management and mining: A survey of algorithms and applications. In *Managing and mining graph data*, pages 13–68. Springer, 2010.

[115] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[116] Julien Lerouge, Zeina Abu-Aisheh, Romain Raveaux, Pierre Héroux, and Sébastien Adam. Graph edit distance: a new binary linear programming formulation. *arXiv preprint arXiv:1505.05740*, 2015.

[117] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pages 163–172. Springer, 2006.

[118] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7):950–959, 2009.

[119] Vincenzo Carletti, Benoit Gaüzere, Luc Brun, and Mario Vento. Approximate graph edit distance computation combining bipartite matching and exact neighborhood substructure distance. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 188–197. Springer, 2015.

[120] Kaspar Riesen, Sandro Emmenegger, and Horst Bunke. A novel software toolkit for graph edit distance computation. In *Graph-Based Representations in Pattern Recognition*, pages 142–151. Springer, 2013.

[121] The ford developer program: Openxc. URL https://developer.ford.com/pages/openxc. Accessed: 2018-01-18.

[122] *OpenXC VI Firmware Documentation.*

[123] Vehicle trace files. URL http://openxcplatform.com/resources/traces.html. Accessed: 2016-05-30.

[124] Stefan Schneegass, Bastian Pfleging, Nora Broy, Frederik Heinrich, and Albrecht Schmidt. A data set of real world driving to assess driver workload. In *Proceedings of the 5th international conference on automotive user interfaces and interactive vehicular applications*, pages 150–157. ACM, 2013.

Résumé détaillé en Français

1 Introduction

L'émergence rapide des nouvelles technologies de l'information et de la communication durant la dernière décennie a permis l'introduction de nouveaux concepts promettant aux populations une amélioration de la qualité de vie. *Smart city*, *smart transportation* et voiture connectée sont parmi les concepts récemment proposés pour faire face aux besoins actuels des villes en termes de mobilité, d'énergie et de sécurité. Appliquées au domaine de transport, les nouvelles technologies ont permis l'avènement des systèmes de transport intelligents (ITS). Le terme ITS est utilisé pour définir des systèmes de transport utilisant une vaste gamme de technologies et de services évolués pour remédier aux problèmes du trafic routier, causés par la croissance accrue des utilisateurs des routes. C'est un concept pluridisciplinaire qui regroupe différent domaines de recherche, depuis l'électronique aux systèmes d'information et à l'analyse des données, et qui couvre différent champs d'applications (e.g. information aux voyageurs, aide à la conduite, gestion de flotte) dans le but de fournir aux utilisateurs des solutions efficaces de mobilité.

Partie intégrante d'un réseau de transport, les nouvelles voitures disposent de composants électroniques et de capteurs capables de mesurer et traiter les différentes grandeurs physiques (e.g. vitesse de la voiture/ aux roues, accélération, angle de direction/de roues). Cette interaction entre l'électronique et le physique fait de la voiture un système cyber-physique (CPS) pouvant bénéficier des approches formelles de conception et d'analyse pour la création d'un système automobile sûr et efficient. Cette informatisation de la voiture a non seulement changé les attentes de ses usagers, qui cherchaient de plus en plus de services connectés, mais a apporté une transformation remarquable au secteur de l'automobile. Portés par cette transformation digitale, de nouveaux types de voitures connectées, autonomes et semi autonomes sont apparues. Bien que cette transformation de l'automobile offre de grandes opportunités pour le développement des solutions de gestion du trafic routier et d'aide à la conduite, le facteur humain reste toujours un élément déterminant dans la recherche de l'amélioration de la sécurité routière. L'analyse du comportement des conducteurs se présente ainsi comme étape indispensable pour la création des routes plus sécurisées.

En effet, les voitures intelligentes génèrent une quantité énorme d'informations, en collectant de façon continue des données sur le conducteur, l'environnement de conduite et la voiture elle-même. L'analyse de ces données recueillies par les voitures permet d'étudier le comportement des conducteurs dans des situations réelles, d'analyser leurs habitudes ainsi que la performance de leur conduite. Plus récemment, de multiple solutions d'analyse du comportement des conducteurs sont commercialisées ([18][19][20][21]), en tant que produits indépendants à usage professionnel aussi bien que personnel. Pour beaucoup de secteurs où le conducteur humain est de grande importance, tels que l'assurance automobile et la gestion de flotte, les données du véhicule sont analysées pour qualifier le comportement de conduite et "profiler" les conducteurs (*Driver profiling*), donnant naissance à de nouveaux types de solutions innovantes. À titre d'exemple, l'*usage-based insurance* est un nouveau concept d'assurance automobile qui promet à ses clients un paiement personnel adapté à leur comportement de conduite.

L'objectif du profiling est de caractériser de façon générale le comportement du conducteur plutôt que de modéliser des manœuvres spécifiques [45]. Ceci est généralement accomplit en appliquant des méthodes statistiques et/ou des méthodes d'apprentissage automatique et d'intelligence artificielle pour trouver des patterns dans les données de conduite et définir par la suite des profiles des conducteurs. De plus, établir des profiles des conducteurs permet de prédire et déterminer la sécurité de leurs comportements de conduite. Ils constituent ainsi une étape indispensable dans le processus de conception des systèmes d'assistance adaptatifs, surtout pour les véhicules à plusieurs conducteurs.

Les méthodes d'identification des caractéristiques de conduite peuvent être divisées en deux classes, méthodes directes et méthodes indirectes. Les méthodes indirectes, dites aussi *model-based* méthodes commencent par établir un modèle du conducteur qui décrit son comportement de conduite, puis, en se basant sur le modèle proposé, identifient des caractéristiques de conduite propre au conducteur. Les méthodes directes, quant à eux, consistent à utiliser une méthode d'analyse de données ou de reconnaissance de modèles pour analyser directement les données de conduite, sans qu'un modèle de conducteur soit établi.

Dans cette thèse, nous considérons une méthode indirecte pour la caractérisation des conducteurs. Pour cela, nous étudions les approches de modélisation du comportement des conducteurs humains et discutons leurs contributions au renforcement de la sécurité du comportement de conduite en permettant des analyses avancées. Les travaux menés sont axés sur l'application des méthodes formelles utilisées en informatique fondamentale (l'abstraction numérique, les automates et les graphes) pour la modélisation et l'analyse des comportements du conducteur. L'objectif principal est de proposer un framework pour une modélisation des données du comportement du conducteur dans un environnement connecté, en mettant l'accent sur deux analyses, (1) analyse de conformité des conducteurs au code de la route, (2) analyse de similarité des comportements des conducteurs, en recherche à l'identification des conducteurs. Cette thèse consiste ainsi en quatre contributions majeures : (1) l'abstraction numérique comme méthode de réduction des données de conduite, (2) les automates, graphes et apprentissage comme outils de modélisation du comportement du conducteur, (3) la vérification de modèles (*model checking*) et son application pour la vérification formelle de la conformité du comportement du conducteur aux exigences du code de la route, (4) l'appariement de graphes (*graph matching*) et son application pour l'analyse de similarité des comportements des conducteurs.

Cette dissertation se compose de 6 chapitres:

Le chapitre 1 décrit le contexte général et les motivations de ce sujet de thèse puis présente brièvement les travaux élaborés.

Le chapitre 2 vise à donner un aperçu global des différents aspects relatifs aux études du comportement des conducteurs automobile. Il commence par une présentation des véhicules intelligents, ses composants sensoriels et les différents types de systèmes d'assistance à la conduite ainsi que les différents rôles que l'humain peut avoir dans cette version améliorée de l'automobile. Dans sa deuxième partie, le chapitre se concentre sur le comportement du conducteur humain, comment peut il être mesuré et les différentes approches considérées dans la littérature pour sa modélisation.

Le chapitre 3 est consacré aux deux premières contributions de cette thèse. Dans ce chapitre, nous proposons une méthodologie pour la modélisation du comportement du conducteur, dont le but est de créer des modèles caractérisant de manière personnelle un conducteur. Le chapitre présente d'abord un état de l'art des approches pour la caractérisation des conducteurs et donne un aperçu sur l'approche de modélisation proposée. Cette dernière comprend deux étapes. La première consiste en une abstraction des données de conduite en utilisant le domaine abstrait des intervalles, précédemment proposé pour l'analyse par interprétation abstraite. La deuxième est une étape de construction qui utilise un algorithme d'apprentissage par renforcement pour construire un modèle du comportement du conducteur. Nous choisissons les modèles graphiques, notamment les automates probabilistes et les graphes étiquetés comme outils de modélisation. Une nouvelle approche pour la représentation de l'environnement de conduite est aussi présentée.

Le chapitre 4 décrit notre troisième contribution qui consiste à une vérification du comportement du conducteur par *model checking*. Il commence par un rappel sur les notions de vérification formelle, *model checking* et logique temporelle, puis présente la méthode de vérification proposée. L'utilisation de cette dernière est démontrée par une analyse des traces de conduite appartenant à un conducteur. L'impact de l'abstraction sur les résultats de la vérification est également étudié.

Le chapitre 5 présente notre quatrième contribution. Elle consiste à introduire l'appariement de graphes comme méthode d'analyse de similarité des conducteurs.

Le chapitre 6 conclut ce rapport de thèse. Il fait le bilan de nos contributions et propose des perspectives à ce travail.

2 Modélisation du comportement du conducteur

Conduire un véhicule est une activité de haute complexité où les conducteurs interagissent avec leur véhicule ainsi qu'avec leur environnent. Le conducteur, le véhicule et l'environnement constituent ensemble ce que l'on appelle un système DVE (*Driver, Vehicle, Environment*). C'est un système en boucle fermé dans lequel : le conducteur perçoit son environnement de conduite et agit sur le véhicule; qui de sa part exécute les taches réalisées par le conducteur et se déplace en conséquence dans l'environnement; le feedback du véhicule et l'information de l'environnement influenceront par la suite la prochaine décision du conducteur. Un comportement de conduite peut ainsi être défini comme une succession d'actions réalisées par un conducteur en réponse à un état particulier du système conducteur-véhicule-environnement, et où chaque action entraîne un changement dans l'état dynamique du véhicule.

Pour explorer la façon dont les personnes conduisent, les chercheurs ont mis au point une variété de modèles pour expliquer et simuler le comportement du conducteur, certains de ces modèles sont des modèles conceptuels qui aident principalement à comprendre les éléments de représentation de la tâche de conduite. D'autres sont des modèles informatiques qui calculent, simulent, et prédisent les différents aspects du comportement du conducteur. Ces modèles de calcul ont émergé comme des outils puissants, à la fois à l'étude théorique, et au développement des systèmes de véhicules intelligents. Boyraz, Pinar et al. ont proposé une classification des approches de modélisation en quatre groupes [44]. L'approche basée sur la théorie du contrôle modélise le comportement du conducteur du point de vue contrôle; le comportement de conduite latéral (e.g. maintien de voie) et longitudinal (e.g. freinage, prévention des collisions) est modélisé par une équation qui représente explicitement la relation physique entre les variables d'entrée et de sortie. L'approche basée sur les facteurs humains prend en considération les caractéristiques physiques du conducteur humain (e.g. perception visuelle, cognition, charge mentale). L'approche stochastique / non linéaire utilise des outils mathématiques puissants tels que les réseaux bayésien, les réseaux de neurones et les modèles de Markov cachés pour faire face à la nature incertaine et non linéaire du comportement du conducteur. L'approche hybride combine deux ou plusieurs approches mentionnées ci-dessus.

L'approche de modélisation que nous proposons consiste à utiliser les différentes informations issues des capteurs de la voiture pour la construction d'un modèle adapté au style de conduite du conducteur, et qui peut être utilisé par la suite pour l'analyse et la prédiction de son comportement. Cependant, le fonctionnement en continue des capteurs engendre des données de grandes tailles rendant leur analyse de plus en plus difficile. Des approches pour l'optimisation des ensembles de données ainsi que les algorithmes de collecte de données d'une manière qui peut être facilement applicables pour les mesures de sécurité routière sont donc indispensables. La particularité de notre approche est qu'elle permet une modélisation du comportement du conducteur humain à un haut niveau d'abstraction. Le concept de l'abstraction a été utilisé dans de nombreux domaines, d'une manière générale, l'abstraction est un processus de généralisation effectué consistant à garder que les caractéristiques essentielles d'un objet en négligeant les détails inutiles. Dans notre approche, nous utilisons une abstraction numérique des données par laquelle l'ensemble des données est approché par un ensemble bien réduit et abstrait, en ignorant les détails des mesures instantanées.

2.1 L'abstraction numérique appliquée aux données de conduite

L'approche d'abstraction que nous proposons repose sur les fondements théoriques de l'abstraction des structures mathématiques proposée par Patrick and Radhia Cousot. Selon Patrick and Radhia Cousot, l'abstraction est définie comme une approximation d'un ensemble concret ordonné $(C, \sqsubseteq_C)$, possiblement infini, par un ensemble abstrait $(A, \sqsubseteq_A)$. Un ensemble abstrait représente en fait une sur-approximation de l'ensemble concret, et devrait être le plus affinée afin d'inclure moins d'éléments supplémentaires.

Pour l'approximation des données de conduite nous utilisons le domaine abstrait des intervalles. Nous choisissons le domaine des intervalles pour son adaptabilité aux modèles et aux analyses que nous allons effectué par la suite. Les données de conduite contient un ensemble de variables (table 3.1). La phase d'abstraction consiste à définir pour chaque variable, de notre ensemble des données, son propre domaine abstrait. Afin d'utiliser le domaine non relationnel des intervalles, et par souci de simplicité, nous avons ignoré toute corrélation entre les variables. Toutefois, une analyse en composantes principales peut être utilisée pour extraire des variables non corrélées.

Soit $\mathcal{I}_{\bar{\mathbb{R}}} = \{\bot\} \cup \{[a, b[\mid a \in \mathbb{R} \cup \{-\infty\},\ b \in \mathbb{R} \cup \{+\infty\}\ and\ a < b\}$ l'ensemble des intervalles réels semi-ouvert à droite, où $\bot$ désigne un intervalle vide. Le domaine concret d'une variable x est défini par le poset $C_x = \langle \bar{\mathbb{R}}, \subseteq \rangle$ où $\subseteq$ est l'opérateur usuel d'inclusion. Le domaine abstrait, le poset $A_x = \langle \mathcal{I}_{\bar{\mathbb{R}}}, \sqsubseteq \rangle$ défini le domaine des intervalles $\mathcal{I}_{\bar{\mathbb{R}}}$ muni de la relation d'ordre $\sqsubseteq$. $\sqsubseteq$ est définie par $[a, b[\sqsubseteq [a', b'[\iff a \geq a'\ and\ b \leq b'$, on définit les opérateurs $\sqcap$ et $\sqcup$ par:

$$X \sqcap Y = \bot\ \boldsymbol{if}\ Y = \bot\ \boldsymbol{or}\ X = \bot$$

$$X \sqcap Y = \begin{cases} [max(a, a'), min(b, b')[& \boldsymbol{if}\ X = [a, b[\ \boldsymbol{and}\ Y = [a', b'[\\ & \boldsymbol{and}\ max(a, a') \leq min(b, b') \\ \bot & \boldsymbol{otherwise} \end{cases}$$

$$X \sqcup Y = \begin{cases} [min(a, a'), max(b, b')[& \boldsymbol{if}\ X = [a, b[\ \boldsymbol{and}\ Y = [a', b'[\\ X & \boldsymbol{if}\ Y = \bot \\ Y & \boldsymbol{if}\ X = \bot \end{cases}$$

Suivant cette abstraction en intervalles, un ensemble c de réels est approximé par l'intervalle $[a, b[$ où $a = min(c)$ et $b = max(c)$. La fonction d'abstraction α_x est alors définie comme suit:

$$\begin{array}{rccc} \alpha_x : & \bar{\mathbb{R}} & \rightarrow & \mathcal{I}_{\bar{\mathbb{R}}} \\ & c & \mapsto & [min(c), max(c)[\end{array} \quad , if\ c \neq \emptyset$$

$$\alpha_x(\emptyset) = \bot$$

Inversement, la fonction de concrétisation est définie comme:

$$\gamma_x([a, b[) = \{y \in \mathbb{R} \mid a \leq y < b\} \quad and\ \gamma_x(\bot) = \emptyset.$$

Il est aisé de voir que les deux fonctions (α_x, γ_x) forment une connexion de Galois $\langle C_x, \subseteq \rangle \underset{\gamma}{\overset{\alpha}{\leftrightarrows}} \langle A_x, \sqsubseteq \rangle$. Cela garantit qu'aucune information du concret ne serait perdu, ainsi on peut dire que A_x est approximation correcte (sound) de C_x.

Pour définir les intervalles qui approximent les données de conduite, nous distinguons entre deux approches, une approche statique et une approche dynamique. Dans l'approche statique, les intervalles sont définis à l'avance en fonction du type et des objectifs de l'analyse à effectuer. Prenons par exemple la variable "*vehicle speed*", noté *vel*, dont les valeurs varient de 0 à 240km/h (le domaine concret de *vel*). Dans le cas d'une analyse étudiant le conformité d'un conducteur aux panneaux de limitations de vitesse, les intervalles du domaine abstrait de *vel* peuvent être définis en fonction des limites de vitesse qui existent dans le code de la route. Le code français, par exemple, contient 6 panneaux de limitations de vitesse [93]: 30, 50, 70, 90, 110, 130. En se basant sur ces limites, on définit le domaine abstrait de *vel*, $A_{vel} = \{[0, 30[, [30, 50[, [50, 70[, [70, 90[, [90, 110[,$

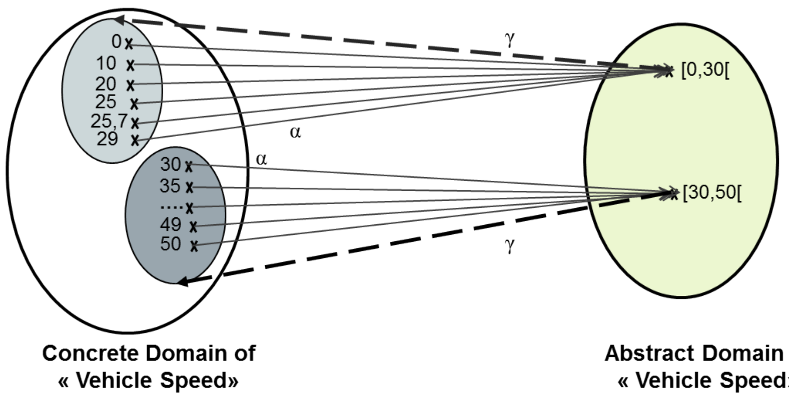

0
10
20
25
25,7
29
30
35
....
49
50
α
α
γ
γ
[0,30[
[30,50[
Concrete Domain of « Vehicle Speed»
Abstract Domain of « Vehicle Speed»

FIGURE 1: Abstraction numérique par domaine d'intervalles. (α : fonction d'abstraction, γ : fonction de concrétisation)

$[110, 130[$, $[130, 240[$, $[240, +\infty)\}$, signifiant que toutes les valeurs de vitesse entre 0 et 30 sont représentées par un seul intervalle $[30, 50[$, de même pour les autres intervalles. Une représentation graphique du domaine concret et abstrait de *vel* est illustrée dans Figure 1. Cela permet d'ignorer les valeurs de vitesses comprises dans les intervalles, en gardant que les changements entre les valeurs abstraites (les intervalles), information pertinente pour l'analyse considérée. De manière plus formelle, en utilisant les fonctions d'abstraction et de concrétisation, cette abstraction est formulée comme suit: $\forall y \in \mathbb{R}$ tel que $0 \leq y < 30, \alpha_{vel}(y) = [0, 30[$, $\gamma_{vel}([0, 30[) = \{y \in \mathbb{R} | 0 \leq y < 30\}$.

Pour sa part, l'approche dynamique utilise des algorithmes d'apprentissage pour trouver, à partir des données, les intervalles les plus appropriés. En utilisant la méthode du k-means, par exemple, l'abstraction numérique consiste à partitionner les valeurs d'une variable en k intervalles où le k est estimé à partir des données. Figure 2 illustre un exemple de segmentation des valeurs de vitesse d'un conducteur D1, du dataset publique UAH-DriveSet[1][95]. L'utilisation du k-means permet de classer les valeurs de vitesse en 9 intervalles, représentés dans la figure par différentes couleurs. L'abstraction de la variable vitesse se fait alors à base de l'ensemble: $A_{speed} = \{[0, 47.1[$, $[47.1, 56.2[$, $[56.2, 65.7[$, $[65.7, 78.0[$, $[78.0, 90.6[$, $[90.6, 97.0[$, $[97.0, 113.7[$, $[113.7, 131.3[$, $[131.3, 148.8[$, $[148.8, 255[$, $[255, \infty)\}$.

2.2 Les modèles proposés pour la représentation du comportement des conducteurs

Un comportement de conduite peut être défini comme un changement de l'état dynamique du véhicule initié par le conducteur en réponse à une situation spécifique de conduite. Les données de conduite collectées permettent de déterminer des états précédents par lesquels un véhicule a passé durant un trajet. Le comportement du

[1]available at http://www.robesafe.uah.es/personal/eduardo.romera/uah-driveset/

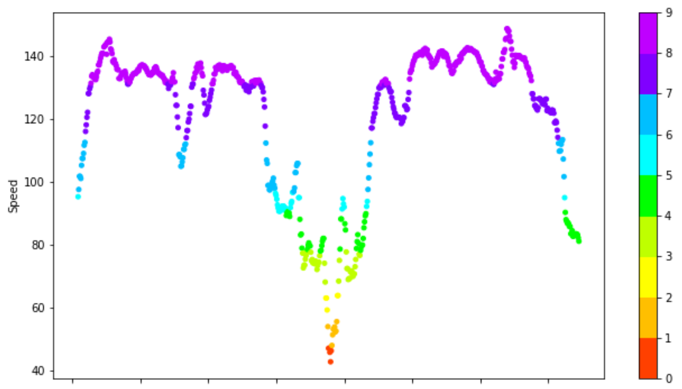


FIGURE 2: Abstraction dynamique de vitesse, en utilisant l'algorithme k-means. Le nombre des clusters considérés est $k = 9$.

conducteur durant un trajet peut être alors considéré comme des transitions entre ces états.

Pour la modélisation du comportement d'un conducteur, nous utilisons des formalismes de la famille des modèles graphiques probabilistes. Les modèles graphiques probabilistes sont connus comme des outils puissants de modélisation, combinant efficacement deux théories, la théorie de graphes et la théorie de probabilité. Ils fournissent ainsi un cadre utile pour la modélisation du comportement de conduite: le langage des graphes facilitera la représentation des relations entre conducteur, véhicule, et environnement de conduite, tandis que la probabilité permettra la représentation de l'incertitude. Nous proposons donc deux formalismes de modélisation, PRHIOA (Probabilistic Rectangular Hybrid Input/Output Automata) et ADG (Attributed Directed Graphs) , qui, en plus de leur puissance représentative, vont permettre d'effectuer des analyses approfondies se basant sur les techniques de vérification formelle et d'appariement de graphes (graph matching).

2.2.1 Représentation par Probabilistic Rectangular Hybrid Input/Output Automata (PRHIOA)

Le formalisme PRHIOA est une combinaison des notions des théories des automates input output hybrides [96], et des automates rectangulaires [97] pour lesquels le problème d'accessibilité (*reachability*) est prouvé décidable [98]. Un PRHIOA est définit, formellement, comme un tuple $(H,U,Y,X,Q,\Theta,inv,E,I,O,D,\mu)$, où:

- H est un ensemble de variables internes, U est un ensemble des variables d'entrée et Y est un ensemble de variables de sortie. H, U et Y sont disjoints l'un des autres, nous écrivons l'ensemble de variables $X \triangleq H \cup U \cup Y$.

- $Q \subseteq val(H)$ est un ensemble d'états, où $val(H)$ est l'ensemble des valeurs de H.

- Θ est l'ensemble des états initiaux.

- $inv : Q \rightarrow Rect(H)$ est une fonction invariant, où $Rect(H)$ est l'ensemble des prédicats rectangulaire défini sur H. Un prédicat rectangulaire ϕ sur H est une conjonction des inégalités rectangulaires, il définit l'ensemble des vecteurs $[\![\phi]\!] = \{z \in \mathbb{R}^n \mid \phi\ [H := z]\ is\ true\}$. Une inégalité rectangulaire définit sur H est une formule $h_i \sim c$, où $h_i \in H$, c est un entier constant et $\sim$ est un des opérateurs $<, \leq, >, \geq$. La fonction *inv* associe un invariant à chaque état.

- E, I et O sont des ensembles d'actions internes, entrée et de sortie, respectivement. Les actions internes seront par la suite dénotées par le symbole "?".

- D est un ensemble de transitions discrètes. Une transition discrète est étiquetée par une action et définie comme (q, o, g, q') où q est un état source, o est une action, $g \in Rect(H)$ est une garde sur les transitions, et q' est l'état cible. Pour simplifier, nous référons à une transition comme triplet (q, o, q').

- μ est une fonction de probabilité de transition définie sur les actions sorties O tel que: $\sum_{q' \in Q} \sum_{a \in O} \mu(q, a, q') = 1$ pour $q \in Q$ et $a \in O$.

Le comportement du conducteur modélisé par un automate correspond à ce qui suit :

- *Variables internes.* L'ensemble des variables relatives à la conduite, tel que *vitesse*, *accélération*, *angle de braquage.* Afin de représenter, dans chaque état, les informations contextuelles sur l'environnement de conduite (des règles de la conduite principalement définies), nous définissons la variable *driving context (cxt).* Il s'agit d'une représentation des règles de la route sous forme de contraintes sur les variables, mises à jour de façon continue suite aux changements de l'environnement. En effet, pour faciliter l'analyse et la mise à jour du contexte de conduite, *driving context (cxt)* est représentée par un ensemble de piles de contraintes, où chaque pile est associée à une des variables. Des stratégies de mise à jour ont été définies par la suite tenant compte de la différence entre les différentes situations de la route.

- *États.* L'ensemble des états correspond aux valuations possibles des variables.

- *Invariants.* Un invariant d'état est défini comme étant une conjonction de conditions sur les variables internes. Une condition sur une variable x prend la forme d'un intervalle $[a, b[$ du domaine d'abstraction de x.
- *Actions d'entrée (input actions).* Il s'agit d'informations contextuelles sur les situations de conduite, reçues par les capteurs du véhicules. Des exemples comprennent, entre autres, des panneaux de signalisation routière, des avertissements des dangers, de congestion. Ces entrées sont utilisées pour mettre à jour la variable *driving context (cxt).* La mise à jour suit des règles prédéterminées conçues en fonction des différentes entrées.
- *Actions de sortie (output actions).* Elles modélisent les actions du conducteur par rapport aux tâches de conduite. Des actions comme accélérer, tourner à droite, s'arrêter sont représentées comme actions de sortie.
- *Transitions.* Nous distinguons entre deux types de transitions : input-enabled transitions se produisent suite aux changements de l'environnement (réception des entrées), et output transitions activées par les actions du conducteur.

Le comportement du conducteur est modélisé par un automate probabiliste construit à base des données de conduite, dont les états représentent les états observables du véhicule et les transitions pondérées par des probabilités qui reflètent la fréquence de leur occurrence. Le nombre des états de l'automate dépend de la cardinalité des domaines abstraits utilisés pour l'abstraction des données, dont les éléments sont utilisés pour la définition des invariants. La construction de l'automate se fait par un algorithme d'apprentissage par renforcement, basé sur l'algorithme learning automata [105], et qui consiste à apprendre et mettre à jour, à partir des données, les probabilités des transitions entre les états de l'automate. Le pseudo-code de l'algorithme de construction est illustré dans Algorithm.1. L'algorithme s'exécute sur l'ensemble des données considérées pour la modélisation. A chaque occurrence d'une action d'entrée ou de sortie, des transitions sont ajoutées et les probabilités mises à jour, selon le schéma de renforcement suivant:

$$P(T) = \begin{cases} 1 & \text{if } r = 0 \\ P(T) + \frac{1-P(T)}{r} & \text{if } r \neq 0 \end{cases} \tag{3.1}$$

$$P(T') = P(T') - \frac{P(T')}{r}, \; for\ all\ T' \neq T \tag{3.2}$$

Où $P(T)$ est la probabilité de la transition renforcée $T \in D$ sortante de l'état q, et r le nombre de toutes les transitions T' sortante de q.

Un exemple d'un automate de comportement de conduite est illustré dans Figure 3. La vitesse et l'angle de braquage, dénotés respectivement *vel* et θ sont considérés comme

Algorithm 1 L'algorithme de construction de l'automate du comportement du conducteur.

```
Require: $Q$: states , $I$: input action set, $O$: output action set, $q_{initial}$: initial state,
    $Data$: driving data set
Ensure: $D$: transition set, $P$ : transition probabilities
  $q_{current} = q_{previous} = q_{initial}$
  while Data do
    convert $H \in Data$ to its corresponding abstract value $\alpha(H)$
    pick a state $q$ from $Q$ such that $inv(q) = \alpha(H)$
    $q_{current} = q$
    if (input action $I_j \in I$ exists in $Data$) then
      $q_{previous} = q_{current}$
      Update current driving context $cxt$ in $q_{current}$: $q_{current}.cxt = cxt_j$
      if $(q_{previous}, q_{current}, I_j) \notin D$ then
        Add input-enabled transition $T$ to $D$
      end if
      Update transitions probabilities using (3.1) and (3.2)
    end if
    if (output action $O_j \in O$ exists in $Data$) then
      if $(q_{previous}, q_{current}, O_j) \notin D$ then
        Add output transition $T$ to $D$
      end if
      $q_{previous} = q_{current}$
      Update transitions probabilities using (3.1) and (3.2)
    end if
  end while
```

variables internes. La variable *cxt* représente le contexte de conduite, et est initialisé à C_0, représentant les exigences implicite de l'environnement de conduite tels que la vitesse limite autorisée sur une route, l'obligation de conduire à droite sur les routes à deux voies. Les arêtes étiquetées par des nombres réels représentent les transitions initiées par des actions du conducteurs. Les étiquettes des arêtes de l'automate correspondent aux probabilités de l'occurrence des transitions, calculées dans la phase d'apprentissage du modèle. Les autres arêtes, en bleu, représentent les input-enabled transitions initiées par un changement dans l'environnement et qui entraine une mise à jour du contexte de conduite *cxt*.

2.2.2 Représentation par Attributed Directed Graphs (ADG)

Une autre alternative à la modélisation par automates consiste en une modélisation par graphes du comportement du conducteur. La modélisation par graphes a été largement utilisée dans de nombreux domaines de l'informatique [112], y compris, le génie logiciel, data mining et réseaux. Il existe différents types de graphes dont la structure et les propriétés varient en fonction du domaine d'application. Dans notre contexte, nous

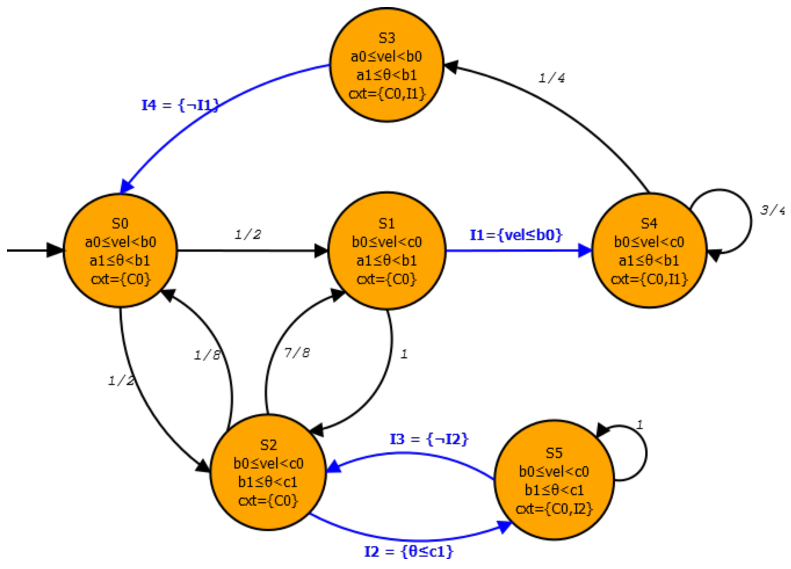


FIGURE 3: Exemple d'automate représentant le comportement d'un conducteur

choisissons l'utilisation des *attributed directed graphs* qui nous semble convenable pour représenter des nœuds avec attributs, des relations récursives entre ces nœuds, ainsi que la valorisation des relations. Un *attributed directed graph* G [99][100] est défini de façon formelle par le tuple (V, E, μ, δ) où:

- V est un ensemble fini de nœuds.
- $E \subseteq V \times V$ est un ensemble d'arcs orientés. Un arc est défini par le paire $e = (s, d)$ où $s \in V$ est le nœud source et $d \in V$ est le nœud destination.
- $\mu : V \to L_V$ est une fonction d'étiquetage de nœuds, qui associe à chaque nœud de V un label (étiquette) de L_V.
- $\delta : E \to L_E$ est une fonction d'étiquetage des arcs, qui associe à chaque arc de E un label de L_E.

L_V et L_E correspondent respectivement aux ensembles des labels des nœuds et des arcs, où les labels sont des tuples de taille fixe.

Suivant la même logique de modélisation par automates, le comportement de conducteur est représenté par un ADG comme suit:

- *Les nœuds* du graphe représentent les états du véhicule.
- *Les arcs* du graphe représentent les transitions entre les états du véhicule.
- *La fonction d'étiquetage des nœuds* attribut à chaque nœud v un label l_v. Le label d'un nœud est défini comme une séquence de labels associés aux valeurs abstraites

des données de conduite. Soit $X = \{x_1, x_2, \dots, x_n\}$ l'ensemble des variables de conduites utilisés pour définir les nœuds, et L_{x_i} l'ensemble des labels de x_i. Nous définissons la fonction d'étiquetage ϵ_{x_i} de la variable x_i comme:

$$\begin{array}{llll} \epsilon_{x_i}: & I_{\overline{\mathcal{R}}} & \rightarrow & L_{x_i} \\ & [a_i, b_i[& \mapsto & l_{i,x_i} \end{array}$$

La fonction d'étiquetage μ est par la suite définie comme:

$$\begin{array}{llll} \mu: & V & \rightarrow & L_V = L_{x_1} \times \ldots \times L_{x_i} \times \ldots \times L_{x_n} \\ & v & \mapsto & l_v \end{array}$$

- *La fonction d'étiquetage des arcs* attribut à chaque arc e un couple $(action, weight)$, où *action* correspond à l'action du conducteur et $weight \in \mathbb{R}_+^*$ est un poids qui représente l'occurrence de *action*. La fonction d'étiquetage des arcs est définie comme:

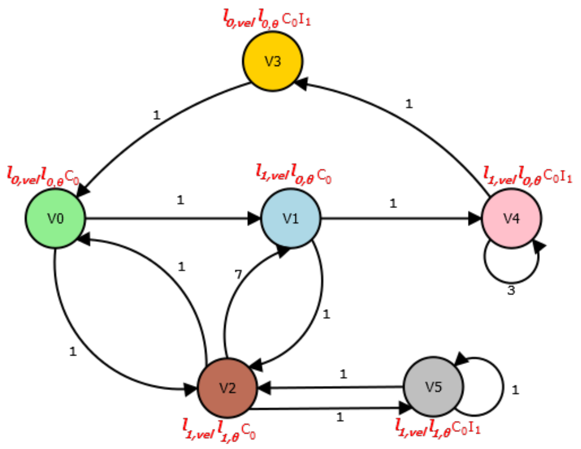
$$\begin{array}{llll} \gamma: & E & \rightarrow & L_E \\ & e = (s, d) & \mapsto & l_e = (action, weight) \end{array}$$

L'algorithme de construction du graphe du conducteur, dont le pseudo code est présenté dans Algorithm.2, s'exécute sur l'ensemble des données de conduite. A chaque timestamp, les valeurs des données sont d'abord mappées à un label de L_v (correspondant à un état du véhicule), les arcs sont ensuite créés et leurs poids mis à jour selon les données du comportement du conducteur. Ainsi, le graphe résultant est une représentation personnalisée du comportement du conducteur.

A titre d'exemple, Figure 4 montre le graphe du comportement du conducteur, modélisé par l'automate dans Figure 3. Les variables vitesse (vel) et angle de braquage (θ) sont considérées pour la définition des états du véhicule; et leurs valeurs utilisées pour l'étiquetage des nœuds. Soit A_{vel} et A_θ des domaines abstraits définis pour l'abstraction de vel et θ. la définition des labels pour les valeurs de vel consiste à attribuer à chaque intervalle $[a_i, b_i[$ de A_{vel} un label $l_{i,vel}$ de type entier ou string. Par exemple, $l_{0,vel}$ et $l_{0,\theta}$ correspondent respectivement aux labels attribués aux premiers intervalles de A_{vel} et A_θ. Pour le contexte de conduite (cxt), les labels sont définis à base des notations attribuées aux entrées du modèle (inputs). Ainsi, $C_0 I_1$ dans le label du nœud V_3 correspond au label attribué à cxt représentant un contexte de conduite régi par les exigences implicites de l'environnement de conduite C_0 et l'input I_1. En comparaison avec l'automate de Figure 3.9, les labels des nœuds du graphe correspondent aux invariants des états, tandis que les poids des arcs correspondent aux probabilités des transitions. Les actions du conducteur peuvent aussi être ajoutées aux labels des arcs.

Algorithm 2 L'algorithme de construction du graphe du comportement du conducteur.

```
Require: $V$: A set of labeled vertices, $V_0$: initial state, $Data$: driving data set.
Ensure: $G$: A graph
  $V_i = V_0$, $action =' none'$
  while $Data$ do
    map $H \in Data$ to the corresponding label $lbl = \mu(H)$
    while $lbl = \mu(V_i)$ do
      wait
    end while
    Pick a vertex $V_j$ with $\mu(V_j) = lbl$
    create an edge $e = (V_i, V_j)$
    if driver action $a_i$ known then
      add $a_i$ to the label of $e$: $\delta(e).action = a_i$
    end if
    if $e \in E$ then
      $\delta(e).weight = \delta(e).weight + 1$
    else
      add edge $e$ to $E$
      $\delta(e).weight = 1$
    end if
    $V_i = V_j$
  end while
```



$l_{0,vel} l_{0,\theta} C_0 I_1$
V3
1
1
$l_{0,vel} l_{0,\theta} C_0$
$l_{1,vel} l_{0,\theta} C_0$
$l_{1,vel} l_{0,\theta} C_0 I_1$
V0
1
V1
1
V4
1
7
3
1
1
V2
1
V5
1
1
$l_{1,vel} l_{1,\theta} C_0$
$l_{1,vel} l_{1,\theta} C_0 I_1$

FIGURE 4: Exemple d'un graphe de comportement du conducteur.

2.2.3 Abstraction numérique des données de conduite et performance de modélisation

L'abstraction numérique des données de conduite telle que présentée ci-dessous permet de réduire la taille des données collectées et ainsi du modèle construit. Afin d'étudier la relation entre le modèle du comportement du conducteur et l'abstraction, nous utilisons les données de conduite d'un conducteur collectées par *the behavior signal processing laboratory of Nagoya University* [106] dont les signaux sont présentés dans Figure 5.

Nous utilisons 5 abstractions différentes, présentées dans la table 1, qui diffèrent en terme

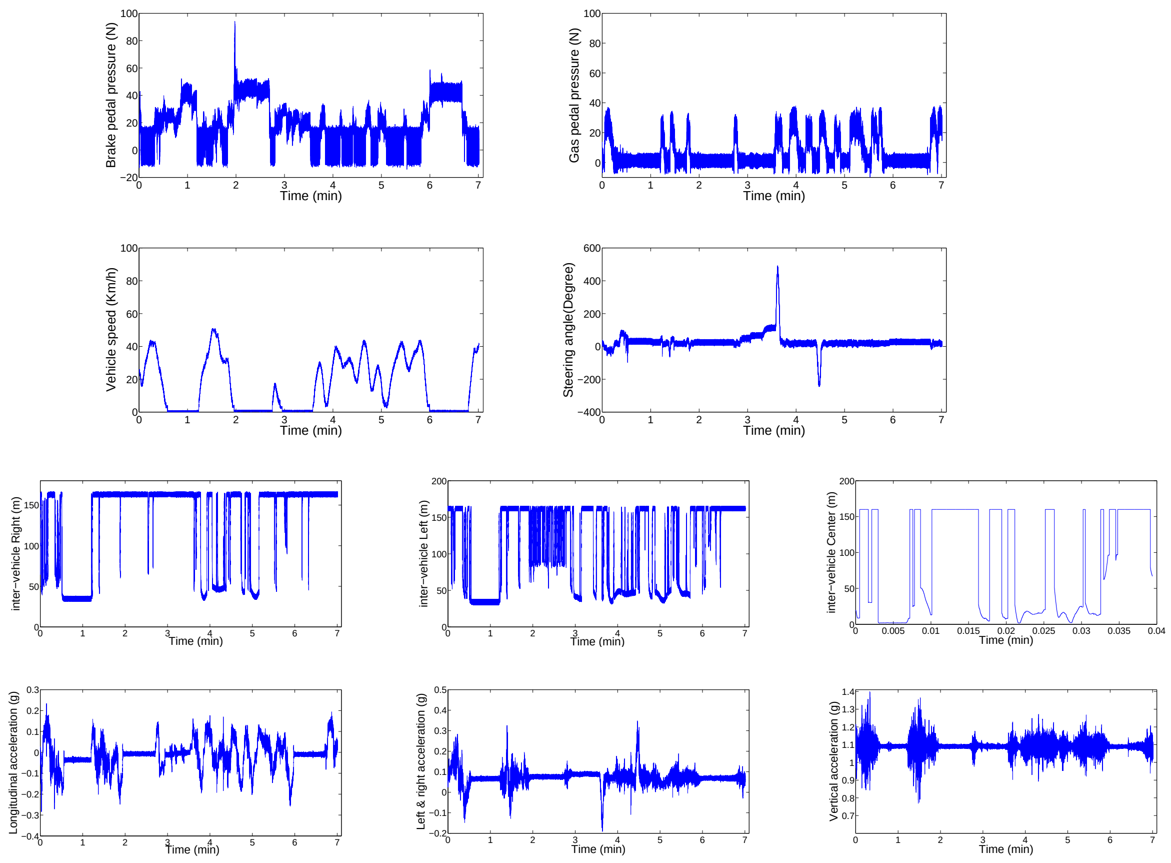


FIGURE 5: Distributions des signaux de conduite d'une conductrice, collectés pendant un trajet de 7 minutes. La fréquence d'enregistrement est 16 kHz. (Données fournies par CIAIR, Nagoya, Japan [5]

de précision d'abstraction, définie par la taille des intervalles considérés. Les bornes des intervalles dans table 1 sont fixées manuellement après analyse des distributions des données, et nous varions les tailles des intervalles en fonction des variations du signal qu'on veut détecter.

Figure 6 compare les différentes abstractions considérées en termes de temps du traitement des données et de taille du modèle (taille de l'espace d'états généré). Un modèle de grande taille et de long temps de traitement, comme prévu, résulte d'une abstraction très précise i.e. utilisant des intervalles étroits. C'est le cas pour l'abstraction 'A1', définie pour être très proche des mesures réelles, et qui, par rapport aux autres abstractions, a généré le modèle le plus grand (6635520 états) avec un temps de traitement plus long. Alors que l'abstraction 'A2' la moins précise i.e. utilisant des intervalles plus larges, génère un modèle de taille inférieure avec un temps de traitement plus court. Nous concluons ainsi que pour aboutir à une meilleure abstraction de données de conduite,

TABLE 1: Abstraction numérique des données de conduite illustrées dans Figure 5

Driving Signals	**Abstractions**				
	Abstraction 1 (A1)	**Abstraction 2 (A2)**	**Abstraction 3 (A3)**	**Abstraction 4 (A4)**	**Abstraction 5 (A5)**
Brake Pedal Pressure (BP)	$\{[-20,0[,[0,20[,[20,40[,[40,60[,[60,80[,[80,100[,[100,\infty[\}$	$\{[-20,20[,[20,60[,[60,100[,[100,\infty[\}$	$\{[-20,0[,[0,20[,[20,40[,[40,100[,[100,\infty[\}$	$\{[-20,0[,[0,40[,[40,100[,[100,\infty[\}$	$\{[-20,20[,[20,50[,[50,100[,[100,\infty[\}$
Gas Pedal Pressure (GP)	$\{[-10,0[,[0,10[,[10,20[,[20,30[,[30,40[,[40,\infty[\}$	$\{[-10,10[,[10,30[,[30,50[,[50,\infty[\}$	$\{[-10,0[,[0,10[,[10,20[,[20,30[,[30,40[,[40,\infty[\}$	$\{[-10,0[,[0,20[,[20,40[,[40,\infty[\}$	$\{[-10,10[,[10,30[,[30,40[,[40,\infty[\}$
Vehicle speed (Vel)	$\{[0,20[,[20,40[,[40,60[,[60,\infty[\}$	$\{[0,30[,[30,60[,[60,\infty[\}$	$\{[0,20[,[20,40[,[40,60[,[60,\infty[\}$	$\{[0,30[,[30,60[,[60,\infty[\}$	$\{[0,30[,[30,60[[60,\infty[\}$
Steering wheel angle (θ)	$\{[-300,-100[,[-100,100[,[100,300[,[300,500[,[500,\infty[\}$	$\{[-300,100[,[100,500[,[500,\infty[\}$	$\{[-300,-100[,[-100,100[,[100,300[,[300,500[,[500,\infty[\}$	$\{[-300,-100[,[-100,100[,[100,500[,[500,\infty[\}$	$\{[-300,-50[,[-50,50[,[50,500[,[500,\infty[\}$
Right front distance (d_{rf})	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[20,40[,[40,160[,[160,200[,[200,\infty[\}$
Left front distance(d_{lf})	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[20,40[,[40,160[,[160,200[,[200,\infty[\}$
Center distance (d_c)	$\{[0,50[,[50,100[,[100,150[,[150,200[,[200,\infty[\}$	$\{[0,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,100[,[100,200[,[200,\infty[\}$	$\{[0,50[,[50,170[,[170,\infty[\}$	$\{[0,40[,[40,160[,[160,200[,[200,\infty[\}$
Longitudinal acceleration (a_l)	$\{[-0.3,-0.2[,[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.1[,[-0.1,0.1[,[0.1,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.2[,[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.2[,[-0.2,0[,[0,0.2[,[0.2,0.3[,[0.3,\infty[\}$	$\{[-0.3,-0.05[,[-0.05,0[,[0,0.2[,[0.2,0.3[,[0.3,\infty[\}$
Directional acceleration (a_d)	$\{[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.2[,[0.2,0.4[,[0.4,\infty[\}$	$\{[-0.2,-0.1[,[-0.1,0[,[0,0.1[,[0.1,0.2[,[0.2,0.3[,[0.3,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.2[,[0.2,0.4[,[0.4,\infty[\}$	$\{[-0.2,0[,[0,0.1[,[0.1,0.4[,[0.4,\infty[\}$
Vertical acceleration (a_v)	$\{[0.7,0.9[,[0.9,1.1[,[1.1,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.1[,[1.1,1.4[,[1.4,\infty[\}$	$\{[0.7,1.0[,[1.0,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.0[,[1.0,1.3[,[1.3,1.5[,[1.5,\infty[\}$	$\{[0.7,1.09[,[1.09,1.13[,[1.13,1.5[,[1.5,\infty[\}$

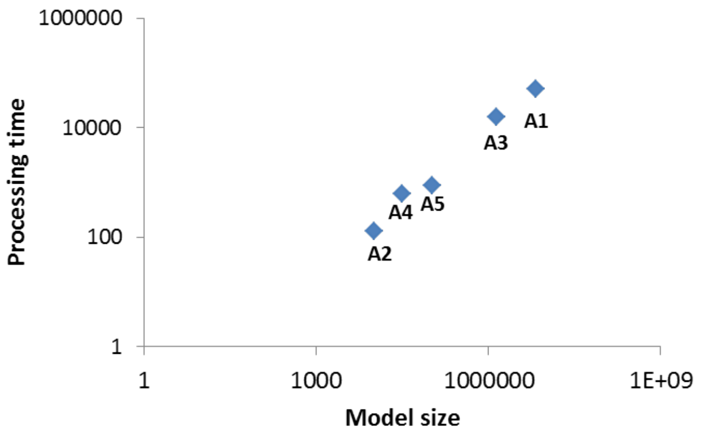


FIGURE 6: Comparaison des abstractions définies dans table 1 en termes de temps du traitement des données et de taille du modèle

un compromis entre la précision et le temps de traitement doit être trouvé.

3 Vérification formelle de la sécurité du comportement du conducteur

Une application intéressante de notre approche de modélisation et qui, à notre connaissance, n'a pas été proposé avant est la vérification formelle (par des formules logiques) de

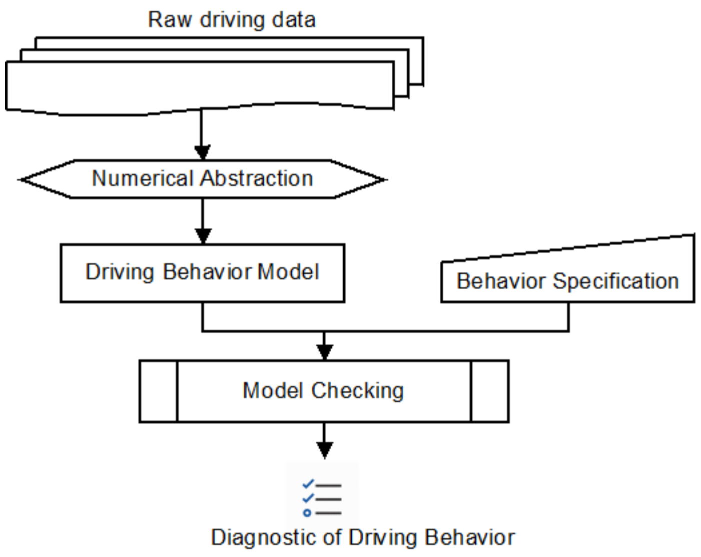


FIGURE 7: Vérification par model checking du comportement du conducteur

la conformité du comportement du conducteur humain aux règles de la route. C'est un des éléments clés de la sécurité routière, permettant de distinguer les bons conducteurs, qui respectent le plus souvent le code de la route dans toute les situations de conduite, des mauvais conducteurs, qui ignorent la plupart des règles édictées par le code la route. Cette évaluation du comportement de conducteur peut être d'une grande importance pour les gestionnaires de trafic, les assurances automobile, les auto-écoles ou aussi pour des conducteurs voulant améliorer la sécurité de leur conduite. Nous proposons une approche de vérification qui permet de vérifier automatiquement si le conducteur effectue certains comportements prédéfinis. Il s'agit d'une vérification offline dont les principaux objectifs sont l'évaluation et la quantification des comportements de conduite, de leur conformité avec les règles de circulation ainsi que le calcul des probabilités de leur violation. Pour cela, nous utilisons une représentation du comportement du conducteur en automate probabiliste (PRHIOA) et nous appliquons la technique du probabilistic model checking pour vérifier, de façon formelle, si le modèle du conducteur satisfait les propriétés relatives aux comportements désirés.

La méthodologie de cette vérification est illustrée graphiquement dans Figure 7. *Behavior specification* consiste en une formalisation des comportements de conduite i.e. que le conducteur doit satisfaire pour une conduite sûre, en formules PCTL (Probabilistic computational tree logic). La logique PCTL est une des logiques largement utilisées pour la spécification des systèmes probabilistes, qui en plus de l'analyse qualitative, permet la quantification probabiliste des propriétés. Un outil du model checking , le model checker est ensuite utilisé pour vérifier si le modèle du comportement du conducteur i.e. l'automate généré à partir des traces de conduite, satisfait les propriétés spécifiées. Nous utilisons, comme outil de vérification, le logiciel PRISM, c'est un logiciel open source permettant la modélisation et l'analyse de plusieurs types de modèles probabilistes : discrete and continuous-time Markov chains, Markov decision processes, probabilistic automata, probabilistic timed automata ainsi que des extensions de ces modèles avec récompenses [110].

Les variables mesurées par le bus CAN, ou par autre moyen de collecte de données de conduite, sont largement utilisées pour l'étude du comportement du conducteur. Il est donc possible à base de ces variables d'exprimer différents comportements de conduite sous forme de propriétés logiques. A titre de démonstration, et comme nous ne disposons pas de données avec des informations contextuelles (existent sous forme de vidéos qui nécessite des techniques dédiées pour l'extraction des exigences), nous nous contentons de vérifier certaines habitudes de conduite telles que le talonnage, les manœuvres de changement de direction, sans aucune considération des signalisations. Afin d'utiliser PRISM pour la vérification du modèle du comportement du conducteur, une modélisation de l'automate en langage de PRISM est indispensable. Ainsi pour représenter les invariants de l'état dans PRISM, nous nous définissons pour chaque variable de conduite V_i deux variables locales $V_{i_{min}}$ et $V_{i_{max}}$ représentant les bornes des intervalles utilisés pour définir les conditions sur V_i. La condition $a \leq V_i < b$, par exemple, sera modélisée dans PRISM par $V_{i_{min}} = a \wedge V_{i_{max}} = b$.

Nous distinguons entre deux types de propriétés, exprimant les comportements de conduite, selon la nature des valeurs utilisées dans l'expression des propriétés. Dans le cas où ces valeurs sont exprimées en tant que paramètres qui peuvent être fournis au moment de la vérification, les propriétés sont dites des propriétés paramétrées. Par contre, si les propriétés contiennent des valeurs déjà fixées, en se basant par exemple sur des règles de la route, elles sont dites propriétés non paramétrées. Soit la table 2 présentant les propriétés, que nous utilisons dans cette analyse, exprimées en PCTL. Les deux propriétés P_1 et P_2 vérifiant le comportement de talonnage sont des propriétés non paramétrées puisque la distance de sécurité est déjà fixée par le code de la route. Les autres propriétés présentées dans la table sont des propriétés paramétrées. Les comportements de conduite considérés dans la présente analyse sont présentés ci dessous.

- ***Le talonnage (Tailgating)***. Afin de diminuer les risques de collision, un conducteur doit maintenir une distance de sécurité entre son véhicule et celui qui le précède. Cette distance est définie (en mètres) par le code de trafic par $\frac{5}{9} \times vel$ (vel exprimée en km/h). Ce comportement est évalué par les propriétés P_1 et P_2. La propriété P_2 n'est vraie que si le conducteur respecte toujours la distance de sécurité. La propriété P_1 quant à elle calcule la probabilité que cette distance de sécurité soit satisfaite.

TABLE 2: La liste des propriétés logiques utilisées dans la vérification

		Property ϕ	Properties parameters
Non parameterized	$\mathbf{P_1}$:	$P = ?[F^{<100} d_{c_{min}} > 5/9 \times vel_{max}]$	
	$\mathbf{P_2}$:	$A[G\ d_{c_{min}} > 5/9 \times vel_{max}]$	
Parameterized	$\mathbf{P_3}$:	$A[G\ d_{lf_{min}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r\ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0\ \&\ \theta_{min} \geq \theta_l))]$	d_{safe}: the distance considered as safe.
	$\mathbf{P_4}$:	$A[G\ d_{rf_{max}} \leq d_{safe} \rightarrow ((\theta_{max} \leq \theta_r\ \&\ \theta_{min} > 0) \mid (\theta_{max} < 0\ \&\ \theta_{min} \geq \theta_l))]$	θ_r: the right turning angle.
	$\mathbf{P_5}$:	$A[\ G(a_{l_{min}} \geq a \rightarrow GP_{min} \geq 0)]$	θ_l: the left turning angle.
	$\mathbf{P_6}$:	$A[G\ !(GP_{min} > p\ \&\ BP_{min} > p)]$	a: the value of acceleration to not be exceeded
	$\mathbf{P_8}$:	$A[\ G(a_{l_{min}} \geq a \rightarrow (\theta_{max} < \theta_r\ \&\ \theta_{min} > \theta_l))]$	p: the tolerable value of pressure
	$\mathbf{P_9}$:	$P = \ ?[a_{l_{min}} > 0\ \&\ (\theta_{min} > \theta_r \mid \theta_{max} < \theta_l)]]$	

- ***Comportement en accélération/freinage (Braking/accelerating behavior)***. Le conducteur ne doit pas appuyer simultanément sur la pédale de frein et celle d'accélérateur. C'est un des comportements déconseillés, en conduite normale, qui peut même endommagé le système mécanique de la voiture. Ce comportement est exprimé par la propriété P_6. Cette dernière est vraie s'il existe aucun état, dans le modèle du comportement du conducteur, où BP et GP sont supérieur à 10. Il est aussi possible d'exprimer des propriétés vérifiant la réponse du véhicule aux commandes du conducteur. On peut vérifier par exemple que l'accélération du véhicule est positive quand le conducteur appuie sur l'accélérateur (Propriété P_5). La propriété P_5 est vraie si le modèle du comportement du conducteur ne contient aucun état où GP est négative et l'accélération longitudinale est supérieure à une valeur a.

- ***Comportement lors des changements de direction (Behavior when turning)***. Pour être plus prudent et éviter d'éventuels dérapages, un conducteur, effectuant une manœuvre de changement de direction, ne doit pas appuyer sur l'accélérateur. Ce comportement est exprimé par les propriétés P_8 et P_9. Nous utilisons aussi les propriétés P_3 et P_4 pour vérifier que le conducteur garde des distances de sécurité des deux côtés du véhicule en tournant à gauche/droite.

Nous utilisons ces propriétés pour vérifier le comportement du conducteur dont les signaux sont présentés dans Figure 5. Pour analyser l'impact de l'abstraction sur les résultats de vérification, nous vérifions des modèles générés pour les 4 abstractions présentées dans Table 1 pendant 1, 3 et 7 minutes de conduite. Les résultats de vérification sont rapportés dans Table 3. Les détails sur les résultats obtenus sont présentés dans la section 4.3 du chapitre 4.

Table 3: Résultats de vérification

Abs	**A2**			**A3**			**A4**			**A5**			**Parameters**
T	**1**	**3**	**7**	**1**	**3**	**7**	**1**	**3**	**7**	**1**	**3**	**7**	
P_1	0.999949	0.999949	0.999949	0.999999	0.999999	0.999999	0.999652	0.999652	0.999652	0.999793	0.999793	0.999793	
P_2	False	False	False	False	False	False	False	False	False	False	False	False	
P_3	False	False	False	False	False	False	False	False	False	False	False	False	$d_{safe} = 30$, $\theta_r = 30$, $\theta_l = -30$
P_4	True	True	True	True	True	True	True	True	True	True	True	True	
P_5	False	False	False	True	False	False	True	True	True	True	True	True	$a = 1$
P_6	True	True	True	True	True	True	True	True	True	True	True	True	$p = 10$
P_8	True	True	True	False	False	False	False	False	False	True	True	True	$a = 2$, $\theta_r = 60$, $\theta_l = -60$
P_9	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

Abs: Abstraction

T: Recording Time in minutes

Après comparaison avec les signaux d'origine (Figure 5), nous constatons que les résultats obtenus jusqu'à présent correspondent bien au comportement réel du conducteur, initialement représenté par les plots dans Figure 5. La comparaison des résultats pour les différentes abstractions révèle que les résultats de vérification dépendent principalement de l'abstraction utilisée ainsi que des paramètres des propriétés. Nous concluons ainsi qu'afin d'obtenir des résultats de vérification corrects, l'abstraction doit être définie en tenant compte des paramètres des propriétés.

4 Analyse de similarité des comportements des conducteurs

La deuxième analyse que nous proposons repose sur la théorie des graphes, principalement l'appariement des graphes (graph matching), pour étudier la similarité des comportements des conducteurs. Les graphes sont des outils de modélisation largement utilisés dans un grand nombre de domaines différents. Cette popularité est due principalement à leur capacité à représenter les propriétés des entités et les relations entre eux en même temps. Leur application pour la modélisation du comportement du conducteur permet d'avoir une représentation graphique de ce dernier. De plus, elle permet l'utilisation des méthodes et concepts mathématiques de la théorie des graphes pour l'analyse du comportement du conducteur.

En théorie des graphes, le problème d'appariement de graphes fait référence au processus d'évaluation de la similarité structurale des graphes. Il revient généralement à trouver un mapping $s : \mathcal{G} \times \mathcal{G} \rightarrow \mathcal{R}$, étant donné deux graphes G et G' appartenant à l'espace des graphes $\mathcal{G}$, tel que $s(G, G')$ mesure la similarité (ou dis-similarité) entre G et G' [113]. En raison de sa nature combinatoire, l'appariement des graphes est considéré comme l'un des problèmes les plus complexes. Sa classe de complexité reste encore incertaine et dépend de l'approche de correspondance considérée. Différentes méthodes d'appariement de graphes ont été proposées ces dernières années; elles peuvent être classées, de façon générale, en deux catégories : appariement exact et appariement inexact. Les méthodes d'appariement exact visent une correspondance stricte entre les deux graphes comparés (isomorphisme), ou entre ses sous-parties, alors que dans l'appariement inexact, une correspondance peut se produire même si les deux graphes comparés sont structurellement différents, dans une certaine mesure[115]. En effet, l'appariement inexact des graphes consiste à trouver la meilleure correspondance entre les graphes, définie comme étant la valeur optimale d'une fonction objective qui mesure la similarité entre les sommets et les arêtes des graphes comparés.

Dans l'analyse proposée, le comportement du conducteur est modélisé par un graphe, tel que présenté dans la section 2.2.2, et nous utilisons la distance d'édition de graphes (Graph Edit Distance), une des techniques les plus flexibles et les plus tolérantes aux erreurs, pour mesurer la dis-similarité des comportements. Il s'agit d'une analyse dans le domaine des graphes du comportement des conducteurs.

Nous étudions la similarité des conducteurs dans deux data sets publiques, OpenXC vehicle traces [2] et hiclab driving data set[3]. OpenXC met à disposition des développeurs un ensemble de traces collectées dans différents scénarios. Nous nous contentons des fichiers présentés dans Table 4 collectés sur les routes de New York City, USA. Chaque fichier contient les données issues des capteurs d'un seul véhicule, circulant dans une zone donnée (utilisée comme nom de fichier). Nous supposons que les fichiers appartiennent à des conducteurs différents. hcilab driving data set, quant à elle, contient les données

TABLE 4: *OpenXC* Vehicle trace files used in the analysis

Vehicle Trace Files for New York City, USA	
Symbol	File name
DC	*Downtown, Crosstown*
DE	*East Downtown*
DW	*West Downtown*
DW2	*West Downtown 2*
UC	*Uptown Crosstown*
UC2	*Uptown Crosstown 2*
UW	*West Uptown*
UW2	*West Uptown 2*

de conduite de 10 conducteurs.

[2]http://openxcplatform.com/resources/traces.html
[3]https://www.hcilab.org/research/hcilab-driving-dataset/

Les variables considérées pour la construction des modèles du comportement des conducteurs ainsi que leurs domaines d'abstraction sont présentés dans Table 5 et Table 6. Pour étiqueter les nœuds des graphes, nous attribuons des numéros aux intervalles des domaines d'abstraction. Nous présentons à titre d'exemple dans Table 7 les labels

Table 5: *OpenXC* driving signals used in the analysis

Signal	**Range**	**Abstract Domain Used**
engine_speed	0 to 16382 RPM	{[0,500[, [500,1000[, [1000,1500[, [1500,2000[, [2000,2500[, [2500,3000[, [3000,4000[, [4000,16382[, [16382,∞)}
fuel_level	0-100%	{[0,20[, [20,40[, [40,60[, [60,80[, [80,100[, [100,∞)}
torque_at_transmission	-500 to 1500 Nm	{[-500,-100[, [-100,0[, [0,50[, [50,100[, [100,150[, [150,200[, [200,500[, [500,1500[, [1500,∞[}
transmission_gear_position	{ first, second, third, fourth, fifth, sixth, seventh, eighth, reverse, neutral}	N/A*
vehicle_speed	0 to 655 km/h	{[0,10[, [10,20[, [20,30[, [30,40[, [40,50[, [50,60[, [60,655[,[655,∞[}

* N/A: Non abstraction applicable.

Table 6: hcilab driving signals used in the analysis

Signal	**Abstract Domain Used**
SPEED_GPS	{ [0, 1.07155[, [1.07155, 3.64651[, [3.64651, 6.92086[, [6.92086, 10.11305[, [10.11305, 13.08576[, [13.08576, 16.29244[, [16.29244, 19.33357[, [19.33357, 21.43441[, [21.43441, 24.52137[, [24.52137, 26.77736[, [26.77736, 28.96064[, [28.96064, 32.94277[, [32.94277, 37.47226[, [37.47226, 80] }
ACCELX	{ [-180, -7.50822[, [-7.50822, -6.30154[, [-6.30154, -5.51624[, [-5.51624, -4.59687[, [-4.59687, -4.07972[, [-4.07972, -3.21781[, [-3.21781, -2.37505[, [-2.37505, -1.58975[, [-1.58975, -0.93853[, [-0.93853, -0.55545[, [-0.55545, 0.01915[, [0.01915, 0.38307[, [0.38307, 0.74699[, [0.74699, 1.18752[, [1.18752, 1.83875[, [1.83875, 2.79643[, [2.79643, 4.09887[, [4.09887, 4.84586[, [4.84586, 5.55455[, [5.55455, 6.24408[, [6.24408, 8.02536[, [8.02536, 180]}
ACCELY	{[-180, -0.42138[, [-0.42138, 1.95367[, [1.95367, 3.83072[, [3.83072, 4.97994[, [4.97994, 5.66947[, [5.66947, 6.22492[, [6.22492, 8.00621[, [8.00621, 8.54251[, [8.54251, 9.30866[, [9.30866, 9.71088[, [9.71088, 10.05565[, [10.05565, 10.41957[, [10.41957, 10.91756[, [10.91756, 11.99016[, [11.99016, 12.85207[, [12.85207, 14.11621[, [14.11621, 15.07389[, [15.07389, 15.8975[, [15.8975, 16.81687[, [16.81687, 180]}
ACCELZ	{ [-180, -8.48505[, [-8.48505, -6.95276[, [-6.95276, -5.42047[, [-5.42047, -4.53941[, [-4.53941, -4.00311[, [-4.00311, -2.52828[, [-2.52828, -2.01113[, [-2.01113, -1.16837[, [-1.16837, -0.68953[, [-0.68953, -0.2873[, [-0.2873, 0.09577[, [0.09577, 0.47884[, [0.47884, 0.88107[, [0.88107, 1.37906[, [1.37906, 2.04944[, [2.04944, 3.12204[, [3.12204, 4.29041[, [4.29041, 5.51624[, [5.51624, 6.53138[, [6.53138, 8.40844[, [8.40844, 180] }

utilisés pour étiqueter les graphes du OpenXC data set. Le label d'un nœud est défini comme une séquence des labels des variables (suivant le même ordre dans Table 7). Par exemple, le nœud étiqueté 02000 représente un état où la valeur de *engine_speed* appartient à l'intervalle *[500, 1000 [* et *transmission_gear _ position* est en position ”*neutre*”. Nous utilisons l'algorithme 2 pour la construction des graphes représentants les comportements des conducteurs. Afin d'étudier le potentiel de l'appariement des graphes pour la reconnaissance des conducteurs, nous générons également des graphes pour des

Table 7: OpenXC: Matching the vertices labels to abstract values

label	**fuel_level**	**engine_speed**	**torque_at_transmission**	**vehicle_speed**	**transmission_gear_position**
-1	N/A**	N/A	"reverse"	N/A	N/A
0	NO_DATA*	NO_DATA	NO_DATA	NO_DATA	"neutral"
1	[0, 20[	[0, 500[	[-500, -100[	[0, 10[	"first"
2	[20, 40[	[500, 1000[	[-100, 0[	[10, 20[	"second"
3	[40, 60[	[1000, 1500[	[0, 50[	[20, 30[	"third"
4	[60, 80[	[1500, 2000[	[50, 100[	[30, 40[	"fourth"
5	[80, 100[	[2000, 2500[	[100, 150[	[40, 50[	"fifth"
6	[100,∞[	[2500, 3000[	[150, 200[	[50, 60[	"sixth"
7	N/A	[3000, 4000[	[200, 500[	[60, 655[	"seventh"
8	N/A	[4000, 16382[	[500, 1500[	[655,∞[	N/A
9	N/A	[16382,∞[	[1500,∞[	N/A	N/A

*NO_DATA : No data received at that time

**N/A : No corresponding valued

sous-parties des traces, que nous appellerons des sous-graphes dans la suite. La table 8 affiche les tailles des graphes et des sous-graphes générés. Un graphe conducteur (driver graph) fait référence au graphe résultant du traitement du fichier de trace entier d'un conducteur. Le sous-graphe 1 et le sous-graphe 2 sont des graphes résultant du traitement des sous-parties des données d'un conducteur. Nous calculons ensuite la distance de similarité entre les graphes conducteurs, ainsi qu'entre les sous-graphes et les graphes conducteurs. Les résultats relatifs aux OpenXC and hcilab dataset sont présentés dans les tables 9 et 10 respectivement. Les détails sur les expérimentations menées ainsi que les résultats obtenus sont présentés dans la section 5.2.3 du chapitre 5. Les résultats

Table 8: Sizes of the generated graphs

Data set		Driver graph size	Subgraph 1	Subgraph 2
OpenXC	nodes	618	326	484
	edges	2356	865	1852
hciLab	nodes	1544	488	999
	edges	3090	1247	2285

obtenus montrent (1) la grande dis-similarité entre le comportement des conducteurs, comme indiquée par les valeurs de distance d'édition des graphes dans les premiers sous-tableaux des tables 9 et 10. D'où, la potentielle application de l'appariement des graphes (graph matching) pour la reconnaissance et l'identification des conducteurs. (2) les conducteurs peuvent être distingués même avec des petits extraits de leur conduite, comme le montre la faible dis-similarité (les valeurs sur la diagonale) entre les sous-graphes et les graphes conducteurs qui leurs correspondent, dans les deux derniers sous-tableaux des tables 9 et 10. Ces deux sous-tableaux montrent aussi l'applicabilité de l'approche proposée pour analyser l'occurrence de certains patterns de conduite dans les comportements des conducteurs.

Table 9: Similarity distances of OpenXC driver behavior graphs

	Data graph							
Model graph	DC	DE	DW	DW2	UC	UC2	UW	UW2
DC	0.00000	2056.50257	2073.95075	2093.10844	2059.03047	2113.25615	2114.93765	2091.94170
DE	2054.00257	0.00000	1957.75123	1997.88723	1913.30240	1982.78196	2044.35497	2019.79011
DW	2057.45075	1960.75123	0.00000	1943.03156	1945.19991	1966.45163	2001.86077	1992.70756
DW2	2094.10844	1995.88723	1951.53156	0.00000	1990.67642	1904.84429	1974.17923	1988.26766
UC	2056.53047	1906.30240	1952.19991	1991.17642	0.00000	1958.65330	2014.52274	2043.65429
UC2	2115.75615	1979.78196	1956.45163	1908.84429	1956.65330	0.00000	1982.60797	1987.54391
UW	2113.93765	2044.35497	2000.86077	1965.17923	2016.52274	1981.60797	0.00000	1848.89301
UW2	2092.94170	2022.29011	1996.20756	1986.26766	2043.65429	1995.54391	1846.89301	0.00000

	Data graph							
Model graph	DC-2	DE-2	DW-2	DW2-2	UC-2	UC2-2	UW-2	UW2-2
DC	754.96510	1935.70923	1982.41664	1955.68725	1962,.7052	1956.69554	1962.65151	1947.01850
DE	1911.32963	319.00000	1924.39812	1908.50837	1907.93881	1917.20529	1951.87485	1951.14876
DW	1971.69568	1895.55784	319.00000	1920.98218	1910.75277	1910.33333	1951.67162	1940.71512
DW2	1947.12534	1921.20609	1878.74870	319.00000	1916.23672	1903.47287	1916.93548	1927.01432
UC	1957.16451	1874.00143	1946.14779	1931.75607	319.00000	1921.80100	1964.43565	1948.77117
UC2	1945.61971	1876.60017	1909.64123	1876.08197	1920.78026	319.00000	1919.93998	193.,05971
UW	1969.46870	1938.19506	1887.72817	1938.52178	1925,.2737	1919.33962	319.00000	1906.90397
UW2	1948.99027	1939.33800	1943.62392	1903.53000	1931.70679	1942.36092	1890.36821	319.00000

Data graphs generated for sub part 2

	Data graph							
Model graph	DC-1	DE-1	DW-1	DW2-1	UC-1	UC2-2	UW-1	UW2-1
DC	1591.53882	1640.43967	1640.18677	1637.12164	1619.11469	1645.83702	1632.18826	1636.43894
DE	1623.81021	1459.82527	1637.34044	1637.06409	1607.52039	1637.84527	1644.11703	1641.72188
DW	1630.31406	1635.05615	941.33640	1599.64984	1619.89528	1612.20249	1630.78967	1624.19915
DW2	1630.67968	1625.27924	1608.49581	1334.16718	1624.20151	1617.24393	1617.12146	1619.83041
UC	1639.95031	1623.50301	1645.96672	1641.73681	1423.55414	1659.25441	1636.67786	1645.07085
UC2	1630.29116	1654.13184	1624.13223	1620.62960	1623.33770	1418.99023	1617.70019	1620.76252
UW	1626.17448	1630.44310	1621.41008	1632.72611	1637.43513	1631.73166	1586.03848	1631.16784
UW2	1630.49693	1643.19935	1631.95487	1609.40638	1636.77580	1625.31303	1628.78667	1547.33329

Data graphs generated for sub part 1

5 Conclusions et perspectives

L'étude et la caractérisation du comportement du conducteur ont suscité un vif intérêt dans la communauté scientifique ainsi que de la part des industriels. Leur contribution au renforcement et à l'amélioration de la sécurité routière a été largement prouvée à travers de nombreuses expériences et solutions de systèmes d'analyse et d'aide à la conduite. De plus, l'informatisation de la voiture a permis le développement de nombreuses méthodes d'analyse et de modélisation, en permettant l'accès aux données relatives au comportement de conduite (e.g. vitesse, accélération, GPS).

Table 10: Results for dissimilarity measurement of driving behavior graphs

	Data graph									
Model graph	Driver 1	Driver 2	Driver 3	Driver 4	Driver 5	Driver 6	Driver 7	Driver 8	Driver 9	Driver 10
Driver 1	0.00000	2667.9073	2616.13573	2669.14375	2653.68632	2637.25759	2661.75518	2710.32218	2676.1233	2637.3065
Driver 2	2686.49064	0.00000	2694.25277	2686.16298	2683.98214	2677.66216	2760.20485	2659.14922	2704.80207	2714.45177
Driver 3	2614.46906	2688.91944	0.00000	2718.34636	2687.89817	2576.34997	2685.76627	2624.34392	2692.77529	2690.7667
Driver 4	2678.64375	2690.66298	2713.34636	0.00000	2640.82925	2598.30617	2698.75724	2628.72022	2684.88464	2703.04625
Driver 5	2648.68632	2702.48214	2712.89817	2652.32925	0.00000	2653.33582	2637.85617	2672.41027	2667.53709	2560.87778
Driver 6	2617.75759	2659.16216	2547.34997	2618.55617	2634.83582	0.00000	2586.90357	2656.67801	2666.49394	2669.88293
Driver 7	2646.75518	2755.20485	2683.93294	2698.75724	2638.6895	2583.7369	0.00000	2682.26608	2723.00671	2672.3643
Driver 8	2702.98885	2658.64922	2621.34392	2619.72022	2652.91027	2628.84468	2659.43275	0.00000	2722.19666	2604.90962
Driver 9	2667.6233	2699.96874	2671.60862	2683.38464	2686.53709	2647.49394	2726.00671	2712.69666	0.00000	2719.16406
Driver 10	2624.8065	2725.45177	2683.10003	2697.04625	2590.62778	2676.38293	2668.95716	2620.40962	2721.91406	0.000000

	Data graph									
Model graph	Driver 1-2	Driver 2-2	Driver 3-2	Driver 4-2	Driver 5-2	Driver 6-2	Driver 7-2	Driver 8-2	Driver 9-2	Driver 10-2
Driver 1	713.22500	2605.45131	2603.38918	2602.50371	2617.54314	2580.84493	2580.42148	2583.55197	2595.53163	2572.20725
Driver 2	2593.57754	677.43333	2605.46276	2631.66398	2641.74516	2624.37151	2649.65437	2603.58968	2640.94722	2614.85026
Driver 3	2578.4289	2608.02063	714.75000	2607.80187	2597.57674	2572.54419	2584.27630	2608.67788	2573.69969	2587.17741
Driver 4	2637.88701	2626.07691	2600.21215	675.00000	2626.50458	2597.95216	2593.39431	2617.03762	2602.57058	2594.36093
Driver 5	2640.10147	2644.62756	2624.79041	2588.16854	675.50000	2605.69495	2581.58573	2582.03631	2606.38727	2592.05518
Driver 6	2614.77235	2617.23776	2545.22096	2594.63236	2613.45504	750.23493	2579.15088	2575.21094	2570.91860	2604.10185
Driver 7	2623.67602	2650.56190	2605.96448	2623.11524	2612.35935	2600.50066	702.55000	2632.59960	2644.20713	2623.32978
Driver 8	2609.98305	2639.16561	2541.25596	2606.23308	2600.69879	2576.31503	2591.33763	781.79167	2623.59755	2590.62971
Driver 9	2601.39237	2614.41667	2608.27233	2575.73592	2623.17206	2615.62885	2634.19960	2620.89214	687.43810	2619.26264
Driver 10	2596.41322	2632.71364	2623.99021	2560.73535	2599.19492	2583.31645	2620.50965	2588.41342	2633.36127	675.50000

	Data graph									
Model graph	Driver 1-1	Driver 2-1	Driver 3-1	Driver 4-1	Driver 5-1	Driver 6-1	Driver 7-1	Driver 8-1	Driver 9-1	Driver 10-1
Driver 1	1450.33333	2447.95915	2462.14831	2466.78137	2441.30388	2378.96910	2454.09820	2457.45627	2446.36975	2412.81182
Driver 2	2432.84421	1502.00000	2449.71237	2435.97083	2439.62571	2421.91012	2467.74327	2470.52302	2461.80000	2455.81560
Driver 3	2398.04457	2447.86040	1480.12341	2436.20827	2408.04582	2395.61786	2435.98883	2418.00964	2456.66397	2421.91792
Driver 4	2447.74937	2466.12500	2418.71296	1449.96667	2429.02652	2431.58474	2458.30823	2406.87102	2431.66508	2400.98891
Driver 5	2412.69692	2445.55842	2443.08242	2424.73203	1454.08681	2406.92939	2461.30566	2432.35672	2457.14353	2443.34068
Driver 6	2425.90615	2450.42514	2405.94466	2413.72572	2409.11848	1472.59087	2462.62734	2444.05536	2429.90672	2448.19715
Driver 7	2467.12207	2465.40043	2435.28348	2447.41462	2447.36751	2455.91987	1470.54064	2453.09295	2449.76267	2441.14434
Driver 8	2436.39287	2437.65833	2436.41550	2415.76441	2415.23323	2396.93515	2457.08281	1496.66667	2442.19548	2402.07264
Driver 9	2437.46142	2452.00000	2417.43595	2399.11964	2395.49706	2398.13854	2462.82698	2433.32821	1471.97500	2444.32468
Driver 10	2443.82327	2446.45741	2430.06606	2406.64242	2421.61128	2432.52104	2449.72782	2440.33655	2447.73688	1451.50000

Dans cette thèse, nous nous sommes intéressé à la modélisation et à l'analyse du comportement général de conduite des conducteurs automobiles, en se basant sur des données générées par la voiture. Nous avons utilisé les modèles graphiques, en particulier les automates probabilistes rectangulaire et les graphes orientés étiquetés, pour représenter le comportement de conduite -défini par des interactions entre le conducteur, le véhicule et l'environnement, et un algorithme d'apprentissage par renforcement pour construire le modèle du conducteur à partir des données. Ce dernier permet la construction des modèles personnalisés des conducteurs pouvant être utilisés pour la prédiction de leur comportement mais aussi pour offrir une assistance adaptative. De plus, l'approche que nous avons proposé pour le traitement des données de conduite permet la modélisation du comportement du conducteur à un niveau d'abstraction élevé.

En terme d'analyse, nous avons proposé, d'une part, le *model checking* comme méthode de vérification du comportement du conducteur. Nous avons démontré, à travers des exemples, comment des comportements de conduite peuvent être exprimés en expressions logiques (spécification) que nous avons utilisé après pour la vérification du modèle du conducteur. Après comparaison des résultats de vérification pour des modèles avec différentes abstractions, nous avons conclut que la justesse des résultats de vérification dépend fortement du niveau d'abstraction utilisé. D'une autre part, nous avons proposé une analyse de similarité entre les comportements des conducteurs par *graph matching*. Les expérimentations réalisées confirment la dis-similarité entre les comportements des conducteurs, et supportent ainsi l'idée d'utiliser le *graph matching* pour la reconnaissance et l'identification du conducteur.

Comme champs d'application des contributions de cette thèse, nous proposons une intégration des méthodes et analyses proposées dans, entre autres, des systèmes de monitoring, des services personnalisées d'assurance, des systèmes pour la gestion des infractions routières, etc.

6 Bibliographie

Pour la bibliographie concernant ce résumé, nous référons à la section "Bibliography" du rapport de thèse.