Two-staged local trajectory planning based on optimal pre-planned curves interpolation for human-like driving in urban areas

Fernando José Garrido Carpio

To cite this version:

Fernando José Garrido Carpio. Two-staged local trajectory planning based on optimal pre-planned curves interpolation for human-like driving in urban areas. Automatic Control Engineering. Université Paris sciences et lettres, 2018. English. NNT : 2018PSLEM065 . tel-02194633

HAL Id: tel-02194633
https://pastel.hal.science/tel-02194633
Submitted on 25 Jul 2019

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

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

THÈSE DE DOCTORAT

de l'Université de recherche Paris Sciences et Lettres
PSL Research University

Préparée à MINES ParisTech

PLANIFICATION LOCALE DE TRAJECTOIRES À DEUX ÉTAPES BASÉE SUR L'INTERPOLATION DES COURBES OPTIMALES PRE-PLANIFIÉES POUR UNE CONDUITE HUMAINE EN MILIEU URBAIN

TWO-STAGED LOCAL TRAJECTORY PLANNING BASED ON OPTIMAL PRE-PLANNED CURVES INTERPOLATION FOR HUMAN-LIKE DRIVING IN URBAN AREAS

École doctorale n°432

SCIENCES DES MÉTIERS DE L'INGÉNIEUR (SMI)

Spécialité INFORMATIQUE TEMPS-RÉEL, ROBOTIQUE ET AUTOMATIQUE

COMPOSITION DU JURY :

M. Paul HONEINE
Université de Rouen Normandie,
Rapporteur

M. José Eugenio NARANJO
Universidad Politécnica de Madrid,
Rapporteur

M. Abdelaziz BENSHRAIR
INSA Rouen, Président

M. Fawzi NASHASHIBI
INRIA, Examinateur

M. Vicente MILANÉS
Renault, Examniateur

M. Joshué PÉREZ RASTELLI
Tecnalia, Examinateur

M. Mohamed-Cherif RAHAL
Vedecom, Examinateur

Soutenue par
Fernando José GARRIDO CARPIO
Le 4 décembre 2018

Dirigée par **Fawzi NASHASHIBI**

MINES ParisTech
PSL RESEARCH UNIVERSITY PARIS

This dissertation is dedicated to my parents, José Luis and Almudena,
and to my brother, Miguel,
who educated me in the values of hard work and humility,
encouraging me to strive for excellence,
always being a pillar of support in my life.

Acknowledgment

At the end of this thesis that reflects the effort and work of these last almost four years, it is necessary to look back and remember all the people who have helped me along this hard way. Since I arrived that Day of All Saints of 2014, I have met many people with great humanity, many of whom I can call now friends. Some of them are close, and others, unfortunately, are far away from here. My apologies to all those who are not mentioned but at some point helped me, and they know they are in my thoughts.

First of all, I want to thank my parents. They supported me to accept this challenge and have always supported me every single day of this adventure. Thank you for teaching me never to give up, to fight for achieving the goals, and to seek excellence, like our beloved club Real Madrid. Thanks to Miguel, my brother, for all your support in the distance and your patience, for all the football talks that made me forget the bad moments during the thesis and for making me always look for the positive side of things. I also apologize for not having been able to help you enough in your studies during this time, and not being able to share enough time together with you at home in Madrid.

I want to express my sincere gratitude to VEDECOM and INRIA research institutes for giving me the opportunity to make my PhD in such an interesting and rising field as the automated vehicles one is. I would like to specially thank Fawzi for accepting me to grow not only academically and profesionally but also personally. Thank you for letting me learn from a large number of experts in different areas, showing me how is working both on research and on the automated vehicles field in two of the leading centers in Europe. Thank you for the technical discussions, the thesis advices and the good talks.

Thanks to my supervisors Vicente and Joshué, who are responsible for the successful completion of this PhD thesis. Thanks for all your effort and dedication and your patience, for always being willing to give me the best advice, despite my stubbornness. It has been an honor to have worked with you, and I apologize for not having learned enough from you. Besides thanking you on a professional level, I also thank you on a personal level. From my arrival at the Orly airport, you made my integration in the team and the country as easy as possible, making me consider you part of my family in France. I want to thank Mohamed as well for all his time and dedication, for making me easier the relationship with the Vedecom members and his support in the bad moments, always available to support me.

Thanks to all the teammates with whom I have met these years. Thank you for all the talks, discussions and advice, both about the doctorate and life in Paris. I would like to especially thank David, who has always been very supportive, particularly during my first year, teaching me everything he knew about motion planning and making my adaption process smoother. I would also like to especially thank Fran and Carlos, who have been there to help me on a professional and personal level and have endured me with much patience particularly during the writing period. Thanks to all the rest of colleagues: PhD students, interns and permanent teammates.

Thanks to all my friends: From my friends in Madrid, especially to *Kalstorm* team members: Jose, Mikel, and Álex. Thanks for supporting me on the distance and sharing with me good moments of laugh and joy every time I travel to Madrid. Thank you for not forgetting our origins, the *Universidad Politécnica de Madrid*, to which I will be eternally grateful.

Up to all the new friends that I have met during this period. Both the ones that stay here right now as well as all the ones that are not in Paris anymore and continue their lives all over the world. I hope someday I can meet again with you all. I have to especially thank Pablo for putting up and encourage me for over two years allowing me to have a piece of Madrid in Paris.

Thanks to all the rest of my family, especially my uncles and my grandmothers. Thank you for thinking about me, encouraging me, desiring me good luck and providing me with some Spanish products, which are always welcome.

Finally, thanks to all those people that I have not named but who have contributed to the realization of this doctoral thesis.

Contents

Acknowledgment **iii**

1 Introduction **3**
- 1.1 Motivation . . . 3
- 1.2 Objectives . . . 5
- 1.3 Manuscript organization . . . 5
- 1.4 Contributions . . . 6
- 1.5 Publications . . . 7

2 State of the art **11**
- 2.1 Evolution of the ITS to the automated vehicle . . . 13
  - 2.1.1 Economic and social impact of automated driving . . . 13
  - 2.1.2 Historical overview of the ITS - research and industrial projects, demonstrations and competitions . . . 15
  - 2.1.3 Advanced Driver Assistance Systems (ADAS) . . . 18
  - 2.1.4 Levels of driving automation for on-road vehicles . . . 21
- 2.2 Functional architecture for automated vehicles . . . 27
  - 2.2.1 Advanced automated vehicle systems . . . 27
  - 2.2.2 A reference automated vehicle architecture . . . 29
- 2.3 Path Planning Techniques for Automated Vehicles . . . 32
  - 2.3.1 Graph search based algorithms . . . 32
    - 2.3.1.a Dijkstra . . . 32
    - 2.3.1.b A-star (A*) based algorithms . . . 33
    - 2.3.1.c State lattices . . . 35
  - 2.3.2 Sampling-based algorithms . . . 37
    - 2.3.2.a Probabilistic RoadMaps (PRM) . . . 38
    - 2.3.2.b Artificial potential fields . . . 39
    - 2.3.2.c Rapid Exploring Random Tree (RRT) and Enhanced RRT (RRT*) 40
  - 2.3.3 Interpolating curves algorithms . . . 43
    - 2.3.3.a Straight lines and circular arcs . . . 43
    - 2.3.3.b Clothoids . . . 44
    - 2.3.3.c Polynomial curves . . . 47
    - 2.3.3.d Splines . . . 47
    - 2.3.3.e Nurbs . . . 48
    - 2.3.3.f Bézier curves . . . 49
- 2.4 Discussion . . . 52

3 Path planning in static environments 57
3.1 Problem description 58
3.1.1 Assumptions and constraints 59
3.2 Global planning 60
3.3 Local planning 63
3.3.1 Pre-planning stage 63
3.3.1.a Bézier curves based path planning 65
3.3.1.b Algorithm description 67
3.3.1.c Optimality criteria 70
3.3.1.d Validation of the proposed optimality criteria 73
3.3.1.e Checking the optimality of the evaluated curves 75
3.3.1.f Human-like driving behavior 78
3.3.1.g Databases resolution 79
3.3.1.h Vehicle model 81
3.3.2 Real-time planning stage 82
3.4 Conclusions 89

4 Path planning in dynamic environments 93
4.1 Overview of dynamic path planning strategies 93
4.2 Problem formulation 96
4.3 Dynamic local planning algorithm based on a virtual lane generation and a grid discretization 97
4.3.1 Grid-based discretization 98
4.3.2 Virtual lane generation 100
4.3.3 Re-planning method for dynamic scenarios 102
4.3.4 Safety avoidance path 105
4.4 Conclusions 107

5 Valdiation Tests 111
5.1 Validation platforms: simulator and vehicles 111
5.1.1 Simulation tools 111
5.1.2 Vehicles 112
5.2 Validation tests for the static path planner 117
5.2.1 Results on simulation platforms 117
5.2.2 Results on real platforms 120
5.2.2.a Cycab experiments 121
5.2.2.b Citroën C1 experiments 124
5.3 Validation tests for the dynamic path planner 127
5.3.1 Results on simulation platforms 128
5.3.1.a Static obstacles scenario 128
5.3.1.b Dynamic obstacles scenario 129

6 Conclusion 137
6.1 Conclusions and remarks 137
6.2 Contributions to the state of the art 138
6.2.1 Static environments 138
6.2.2 Dynamic environments 138
6.3 Future work 139

A Appendix Title **141**

List of Figures

2.1 Transport of goods and passengers in the EU in 2015 (from data at [European Commission, 2017]) . . . 11
2.2 EU Road fatalites and targets 2001-2020 (from data at [European Commission, 2016b]) . . . 12
2.3 Cost to consumer and consumers willingness to pay for ADAS features [Mosquet et al., 2015] . . . 13
2.4 Global ADAS market expectation [Woodside Capital Partners (WCP), 2016] . . . 14
2.5 Economic and comprehensive cost estimates in billions, 2010 [NHTSA National Center for Statistics and Analysis, 2017b] . . . 14
2.6 Economic and comprehensive cost estimates in billions, 2010 [NHTSA National Center for Statistics and Analysis, 2017a] . . . 15
2.7 History of first ITS developments (1970-1994) [Nowacki, 2012] . . . 15
2.8 The holistic view of safety [European Commission, 2016a] . . . 19
2.9 SAE Automation levels. Adapted from [SAE International, 2016] . . . 22
2.10 Deployment of automated vehicles according to the vehicle type. Adapted from [ERTRAC, 2017] . . . 24
2.11 Current automated cars: Waymo's (left), Uber's (right) . . . 27
2.12 Automated passenger vehicles: Lyft-Aptiv (left) and Audi-Nvidia's AI (right) . . . 28
2.13 Robotaxis: Navya's *AUTONOM CAB* (left) and EasyMile's EZ10 (right) . . . 28
2.14 INRIA's Cybercars: Cycab (left) and Cybus (right) . . . 29
2.15 INRIA RITS vehicles architecture . . . 30
2.16 Dijkstra algorithm [LaValle, 2006] . . . 33
2.17 Comparison of A*, D* and Hybrid A* search algorithms [Montemerlo et al., 2008] 35
2.18 State Lattices [Pivtoraiko and Kelly, 2008] . . . 36
2.19 PRM planning. Figure adapted from [Saha, 2006] . . . 39
2.20 Artificial potential fields applied into a stay-in-lane scenario [Wolf and Burdick, 2008] 40
2.21 RRT and RRT* algorithms performance [Karaman et al., 2011] . . . 42
2.22 Combination of straight lines and circular arcs for path planning [Kanayama and Hartman, 1989] . . . 45
2.23 Clothoids: double end spiral (left), line-arc-clotoid interpolation (middle) [Girbés et al., 2011], line-clothoid-circle interpolation (right) . . . 46
2.24 Polynomial curves with different weightings to tune the path [Gu et al., 2013] . . . 47
2.25 η-Splines interpolating given points [Piazzi et al., 2002] . . . 48
2.26 Cubic NURBS curve [Piegl and Tiller, 1996] . . . 49
2.27 Bézier curves [Sederberg, 2007] . . . 49
2.28 Path planning timeline on automated vehicles greatest hits . . . 54

3.1 Path planning flowchart 58
3.2 Global planning flowchart 61
3.3 Global path example at INRIA-Rocquencourt facilities 62
3.4 Local planning for static environments flowchart 64
3.5 Single-turn scenario with optimal curves search [Garrido et al., 2016a] 65
3.6 Bézier curve [Garrido et al., 2016a] 66
3.7 Bézier's Convex Hull and concavity change 67
3.8 Comparison of the different cost functions on 60°, 90° and 120° turns 73
3.9 Comparison of the proposed optimality criteria with respect to curvature-dependent approaches placing the control points both statically and dynamically 75
3.10 Validation of the proposed optimality criteria with respect to a curvature-dependent approach 76
3.11 Path databases for consecutive intersections scenarios 79
3.12 Matlab-Simulink model for comparing the steps of variation for the databases generated in the pre-planning stage 81
3.13 Bicycle model (re-make own figure with described nomenclature) 82
3.14 Real-time static local path planning flowchart 83
3.15 Segmentation process of the consecutive turns shared segment to search the optimal junction point between curves: flowchart (a) and use-case example (b) 86
3.16 Real-time path planning process: from loading the curves from the databases to get the whole path 88

4.1 Path planning system architecture 94
4.2 Dynamic local planning system architecture 97
4.3 Trajectory planned with the proposed algorithm for obstacle avoidance 98
4.4 Grid representation of the different vehicle types 99
4.5 Virtual lane generation with the proposed algorithm for obstacle avoidance 100
4.6 Representation of the slope φ_1 and φ_2 angles for the lane change maneuver 102
4.7 Virtual lane generation for dynamic obstacles considering the obstacle's occupied space prediction 102
4.8 Re-computation of the virtual lane in real-time for re-planning on scenario with changing dynamics of the obstacle 105
4.9 Emergency return to the lane maneuver under unexpected vehicle found situation 106
4.10 Emergency return to the lane maneuvers under unexpected vehicle found situation 106

5.1 RTMaps and Pro-Sivic interfaces for testing on both simulation and real platforms 113
5.2 Experimental platforms of the INRIA RITS team used for on the validation tests . 114
5.3 RTMaps architecture for testing on Cycab platforms 115
5.4 High-level cascade lateral controller for the Citroën C1 117
5.5 Comparison of planned and tracked paths on the simulation scenario 118
5.6 Comparison of planned and tracked paths on the simulation scenario 120
5.7 INRIA RITS platforms running on automated way on the test track 121
5.8 Path following itinerary performed by the Cycab platform at Inria-Rocquencourt . 122
5.9 Curvature details of the path in Figure 5.8a 123
5.10 Path following itinerary performed by the Citroën C1 platform at Inria-Rocquencourt 124
5.11 Path following itinerary from Figure 5.10: A-B area 126
5.12 Path following itinerary from Figure 5.10: C-D area 127
5.13 Dynamic path on a straight stretch scenario with static obstacles 130

5.14 Dynamic path on a straight stretch scenario with dynamic obstacles 131
5.15 Virtual lane re-computation and path replanning applied to a wrong prediction of the obstacle dimensions 132
5.16 Emergency return to the lane re-computation of the virtual lane, aborting the original avoiding maneuver 133

List of Tables

2.1 Comparison of graph search-based techniques for path planning 38
2.2 Comparison of sampling-based techniques for path planning 44
2.3 Comparison of interpolating curves based techniques for path planning 51

3.1 Main physical parameters of the vehicles 64
3.2 Validation experiments of the optimality function 77
3.3 Paths databases according to the different directions of rotation (d.o.r) 79
3.4 Comparison of the steps of variation of the curve parameters on the different databases 81

4.1 Obstacle types classification based on the width 99

5.1 Comparison of comfort variables: curvature and curvature derivative 119

Chapitre 1

Introduction

Below is a French summary of the following chapter "Introduction".

Le nombre de victimes de la route dans le monde a diminué de 42% au cours de la dernière décennie [ITF, 2016]. En dépit de ces chiffres prometteurs, les accidents de la route sont la 9^{th} cause principale de décès [Woodside Capital Partners (WCP), 2016]. Sur la base de cette tendance, les accidents de la route pourraient devenir la principale cause de décès évitables d'ici 2020 [Al-Dweik et al., 2017]. Un autre gros problème dans le transport routier est la congestion. Étant donné que le nombre de véhicules dans les grandes villes continue d'augmenter, cela représente un accroissement de la pollution de l'air et du temps de transport, augmentant la probabilité d'erreurs humaines entraînant des accidents de la route [Zhang et al., 2011].

Les systèmes de transport intelligents (STI) ont émergé en réponse à ces problèmes. Ces systèmes utilisent les télécommunications, l'électronique et les technologies de l'information pour planifier, concevoir, exploiter et entretenir un tel système de transport [Nowacki, 2012]. Les premiers développements intelligents sur les systèmes ITS dans le secteur automobile sont connus sous le nom de systèmes ADAS (Advanced Driver Assistance Systems). Ces systèmes sont équipés dans les véhicules en tant que fonctions d'aide, d'avertissement ou d'assistance dans la tâche de conduite. Le nombre d'ADAS intégrés dans les véhicules utilitaires ne cesse d'augmenter, de même que le niveau d'automatisation des véhicules, le développement de véhicules entièrement automatisés étant l'objectif final de ces ADAS. Par exemple, des systèmes tels que l'alarme d'angle mort, l'alerte de collision avant, l'alerte de sortie de voie, le régulateur de vitesse adaptatif (ACC), l'ABS, Stop & Go, l'assistance pour le maintien de la voie et même l'aide au stationnement ont déjà été commercialisés.

Toutefois, les normes ADAS établies pour les véhicules de tourisme couvrent le niveau deux d'automatisation sur cinq (automatisation complète). Cela signifie que nous sommes encore loin d'avoir des véhicules entièrement automatisés qui roulent sur les routes, en particulier dans les environnements urbains, ce qui suppose un défi technologique plus important. Là, les véhicules doivent réagir de façon réactive aux situations imprévues, et ces capacités ne sont pas encore disponibles. Sous ces considérations, l'objectif de cette thèse est de développer une approche de planification des trajectoires capable de traiter ces scénarios urbains complexes, pouvant apporter une solution au problème de suivi de trajectoire, aussi bien en environnement statique qu'en environnement dynamique, en traitant les obstacles rencontrés dans l'itinéraire et adaptable à différentes plates-formes.

Chapter 1

Introduction

The number of road fatalities in the word has decreased by 42% in the last decade [ITF, 2016]. In spite of these promising numbers, road traffic crashes rank as the 9^{th} leading cause of death [Woodside Capital Partners (WCP), 2016]. Based on this tendency, traffic accidents could become the leading cause of preventable deaths by 2020 [Al-Dweik et al., 2017]. Another big issue in road transport is congestion. Since the number of vehicles in big cities continues increasing, this represents an enlargement of air pollution and in transportation time, incrementing the probability of human errors leading to traffic accidents [Zhang et al., 2011].

Intelligent Transportation Systems (ITS) emerged as a response to these problems. These systems apply telecommunications, electronics and information technology to plan, design, operate and maintain such transportation system [Nowacki, 2012]. The first intelligent developments on ITS systems in the automotive field are knowns as Advanced Driver Assistance Systems (ADAS). These systems are equipped in the vehicles as aiding, warning or assisting features in the driving task. The number of ADAS integrated in commercial vehicles keeps increasing, as well as the automation level of the vehicles, where the development of fully automated vehicles is the final goal of these ADAS. For instance, systems such as the blind spot warning, forward collision warning, lane departure warning, Adaptive Cruise Control (ACC), ABS, Stop&Go, Lane keeping assist and even parking assist have already been commercialized.

However, established ADAS for passenger vehicles cover up to the level two of automoation out of five (full automation). It means that we are still far away for having fully automated vehicles running on roads, specially on urban environments, which suppose a bigger technological challenge. There, the vehicles have to react to the unexpected situations in a reactive way, and those capabilities are not available yet. Under these considerations, the goal of this PhD thesis is to develop a motion planning approach able to deal with these complex urban scenarios, being able to provide a solution to the path following problem both on static and dynamic environments, dealing with the obstacles found in the itinerary and adaptable to different platforms.

This research work has been accomplished between the Robotics and Intelligent Transportation Systems team at the French national institute for research in computer science and automatics (from french *Institut National de Recherche en Informatique et en Automatique*), in collaboration with Vedecom institute (from french *Véhicule Décarboné Communicant et sa Mobilité*). The motivation of the thesis as well as the objectives are presented below.

1.1 Motivation

Research and development of ADAS can improve transportation from an economic, environmental and safety driving points of view. Since production vehicles started to commercialize these features, a significant reduction on the number of accidents on road transport was produced, with the corresponding reduction of expenses derived from those accidents [European Commission, 2016b], [Mosquet et al., 2015]. It also contributes to the reduction of the CO_2 emissions [International Council on Clean Transportation (ICCT), 2017]. The increasing tendency of electric and hybrid vehicles in the market worldwide, surpassing the two million vehicles in 2016 after achieving one million in 2015 [International Energy Agency (IEA), 2017]. Combined with the development of ADAS, they contribute to *ecodriving*, making the vehicle operate in the most efficient manner, reducing the emissions a 15%, with a great potential in urban environments where accelerations and decelerations occur more frequently [Breemersch, n.d.]. Finally, they improve the day-life of road users, producing a significant reduction of transportation time thanks to systems such as the traffic jam assist or the ACC [Verband der Automobilindustrie e. V. (VDA), 2015].

Recent demonstrations worldwide have presented the great potential of these systems. Waymo recently made a demonstration showing their vehicle riding in city streets using a 360° vision system. They also intend to predict the behavior of road users, and gathering data from their vehicles running on automated way for more than six years [Google, 2015]. Recently, they have racked up eight millions of kilometers traveled on automated way, taking less than three months to achieve last million. The development of on-demand automated vehicles for car-sharing providing robo-taxi features have raised last years. Both Navya, who presented their *Autonom CAB* vehicle in the CES 2018, and Easymile, with their EZ10 driverless shuttle tested for mobility on airports and on cities as demonstrated in the University of Laussane in 2016, have started to show the great potential of these platforms, mostly in urban environments. Audi, together with Nvidia, area one of the leading companies introducing machine-learning on automated vehicles, as shown with their BB8 during the CES 2018 congress in a path following application on a non-structured environment with obstacles. Some other important demos, such as the recreation of the first trip made by a car but in automated way on the Mercedes Benz memorial route in 2013, the Public Road Urban Driverless (PROUD) car test of University of Parma, driving on the International VisLab Intercontinental Autonomous Challenge (2010) from Parma to Shangai, the first USA coast to coast automated journey in 2015, or the Grand Cooperative Driving Challenge, have shown the improvements in the field after the two DARPA Challenge competitions.

However, recent accidents of automated vehicles running on normal roads with human driver supervision demonstrate that we are not that close to achieve the fully-automation level. Among them, the first death where an automated vehicle was responsible was produced in the USA with an Uber car due to a failure in recognizing a pedestrian crossing in front of the vehicle at night. Previously, first death where one of these vehicles was involved was due to a driver relaxation. The driver assumed the Tesla autopilot was able to react to every situation, expecting a maximum level of automation, using the ride time for leisure without supervising the scene. Similar accidents have been produced because of other human drivers could not understand the behavior of the automate cars, which was not natural (such as the crash produced in a intersection by an Uber vehicle, or the accidents of Navya in Las Vegas). Thus, more robust systems able to compensate the malfunctions of perception systems are needed before achieving a higher automation level.

Path planning is one of the most important elements on the vehicle navigation. The ability to deal with the unexpected circumstances offering alternative paths to arrive at the destination place safely is critical to achieving a semi-automation level. Development of planning algorithms applied to automated vehicles started ro rise since the DARPA Grand Challenge in 2005, where graph-search based algorithms were firstly used to find a path on the desert scenario. Later, in the DARPA Urban Challenge in 2007, combination of graph-search based algorithms and parametric curves were used dealing with urban environments, and sampling based solutions were introduced. A classification of path planning methods has been considered in this thesis, namely: graph search based algorithms, sampling based algorithms, and interpolating curve based algorithms.

This thesis presents a path planning approach where parametric curves are explored, proposing an approach where quartic Bézier curves are used searching comfort as planning criteria, due to the ability of these curves to adapt to the different environments thanks to be defined by control points. This allows to offer a solution where both constraints of road and vehicle are considered in a pre-planning stage, and later optimal curves are loaded in real-time according to human-like driving style. That way, a fast and natural planning solution is intended to solve the navigation problem in urban environments, being able to deal with obstacles in the path generating a comfortable trajectory. This two-staged planning architecture is validated both in simulation (Pro-Sivic&RTMaps, Matlab-Simulink) and in real platforms (Cybercars and and a robotised electric car), showing its performance in such a different scenarios, adapting to the scene conditions and to the type of ego-vehicle.

1.2 Objectives

The main goal of this PhD thesis is to design a functional local path planning modular architecture for automated vehicles running on urban environments. It generates a continuous path in real-time, providing a solution to the navigation problem in such changing scenarios. A two-staged algorithm is presented, where the pre-planning stage allows to consider the physical constraints of both road and vehicle to pre-compute parametric curves to fit the best to any turn configuration. Smoothness is the main criteria for the optimality function in the real-time planning stage, where an extended planning horizon is envisaged to optimize not only the upcoming cuve but also the next one, providing a human-like driving style. Different paths are studied where the starting and ending position on the curves depend on the changes of concavity and the available space between curves.

In addition to dealing with static environments, a dynamic planning system for adapting the path to avoid the obstacles found in the way is addressed. There, the problem is solved by building a virtual lane on a grid based discretization of the scene. This virtual lane modifies if necessary the global path to avoid the obstacle performing two lane changes, allowing the system to target the new global path as a static environment where each lane change consists of two curves. Thus, it searches a fast response benefiting from the static planning stage, and the dynamic algorithm only search in real-time the best slope for changing lanes generating an smooth path.

In order to validate the proposed local planning approach, different experiments have been carried out, both in simulation as in real platforms running at different speeds (both at low and at medium speeds). This PhD work aims the missing gaps on the state of the art, providing human-like paths, generated in real-time as fast as possible and considering both road and vehicle constraints, adapting to the changes on the environment.

1.3 Manuscript organization

This PhD thesis is organized in six chapters. A brief description of the contents of each chapter is presented below.

Chapter 2, State of the Art: First, this chapter presents an overview of the development and in-market penetration of ADAS. It shows the evolution of these systems, focusing on the economical, technological and the safety contribution to road transport, up to the appearance of the automated vehicles and the different levels of automating driving. Then, a generalization of the functional architecture for automated vehicles in the literature is described. Finally, a review of the different path planning methods used in the main demonstrations involving vehicles automation by both research laboratories and industrial companies is detailed. Based on this review, path planning methods are divided in three main groups: graph search-based methods, sampling-based methods and interpolating curve methods.

Chapter 3, Path planning in static environments: The proposed planning strategy is presented here. First, it is divided in global and local planning. The proposed local planning approach consists of a pre-planning stage and a real-time planning stage. This chapter presents both stages for solving the navigation problem on static environments. The pre-planning stage benefits from the static information of road and vehicle to pre-compute the optimal curves that the vehicle might encounter for any turn configuration on urban roads. This allows the real-time planner to generate smoother paths by interpolating in real-time the optimal curves, fitting better to the road layout. The algorithm considers both the sharpness of the road and the limited space between turns to provide an extended planning horizon where two curves are optimized concurrently. Quartic Bézier curves are considered, with curvature constraints at the beginning and at the end of the curves for ensuring path continuity.

Chapter 4, Path planning in dynamic environments: Once the planning approach for static environments has been presented, dynamic environments are addressed in this chapter. A method to modify the planned static path avoiding the obstacles the automated vehicle could found in the itinerary is presented. It is based on a grid discretization of the scene, a classification of the obstacle type from the perception information and the construction of a virtual lane allowing to solve the avoiding problem benefiting from the former knowledge of the static environment.

Chapter 5, Validation tests: This chapter presents the experiments carried out to validate the path planning system, both in simulated and in real environments. Simulation experiments have been performed emulating urban environments and platform on ProSivic and RTMaps software to validate the real-time planning, whereas Matlab-simulink is used for validating the pre-planning stage. Real experiments have been performed on the platforms of the INRIA RITS team, more specifically on a Cycab and on a robotised electric Citroën C1. The planning strategy has been validated both on low-speed and medium-speed vehicles. The different tested scenarios show the planned and traked paths on different situations where road layout combines both consecutive curves and straight stretches, presenting the turns different sharpness and with different space available among turns.

Chapter 6, Conclusion, future work and research perspective: Final chapter presents the conclusions extracted after this PhD work, presenting some possible future works as well. The perspective on motion planning research is also provided.

1.4 Contributions

The main contribution of this thesis can be briefly described below.

1. Local path planning approach based on a two-staged architecture (pre-planning and real-time planning) for automated vehicles on urban environments. It provides a real-time performance, generating enhanced G_1 continuous paths minimizing both curvature and abrupt changes on it. The novelty of this approach is the introduction of a pre-planning phase where both road phyisical constraints and vehicle kinematic constraints are considered to pre-compute the optimal curves for any possible turn scenario, generating different databases where different starting and ending position of the vehicle on the lane are considered.
2. Then, the real-time algorithm generates a continuous path by joining the pre-computed optimal curves by considering the actual road layout (sharpness, available distance and concavity changes). Thus, the only parameter to be optimized in real-time is the junction point between curves, making the algorithm work with a low computational burden.
3. Planning approach working on both static and dynamic environments, where obstacles are found in the path. A virtual road generation based algorithm is used to modify the planned static path adapting it to the changing scene. It benefits from the static local planner to address the avoidance problem as the performance of two additional curves for each lane change, considering the new itinerary as a static scenario. This allows to generate faster avoidance paths. A prediction of the motion of the obstacles is done thanks to a grid-based discretization of the scene, where vehicles are first classified and after adding the security distances the virtual road is computed by the dynamic algorithm. There, the dynamic planning algorithm searches solutions minimizing the slope of the lane changes to generate smoother avoiding paths.

1.5 Publications

Title: Optimized trajectory planning for Cybernetic Transportation Systems
Authors: F. Garrido, D. González, J. Pérez, V. Milanés, and F. Nashashibi
Conference: 9th IFAC Symposium on Intelligent Transportation Systems
Place: Leipzig, Germany **Date:** July, 2016

Title: Real-time planning for adjacent consecutive intersections
Authors: F. Garrido, D. González, J. Pérez, V. Milanés, and F. Nashashibi
Conference: 19th International IEEE Conference on Intelligent Transportation Systems (ITSC)
Place: Rio de Janeiro, Brazil **Date:** November, 2016

Title: Human-like Based Real-Time Path Planning for Dynamic Environments
Authors: F. Garrido, V. Milanés, J. Pérez, and F. Nashashibi
Journal: IEEE Intelligent Transportation Systems Magazine
Status: To be submitted

Chapitre 2

État de l'art

Below is a French summary of the following chapter "State of the art".

L'état de l'art est divisé en trois blocs principaux: Tout d'abord, un examen important de l'évolution du domaine des STI, allant de l'ADAS de production aux projets de recherche mondiaux les plus pertinents, est réalisé dans la section 2.1. Deuxièmement, l'architecture fonctionnelle des véhicules automatisés proposée dans cette thèse est présentée dans la section 2.2. Là, les sept étapes qui composent l'architecture sont introduites, en se concentrant sur l'étape de la planification. De plus, les hypothèses et contraintes prises en compte pour le système de planification sont résumées dans la section. La troisième partie de l'état de la technique est spécifiquement axée sur les algorithmes de planification des trajectoires, qui sont la cible de cette thèse 2.3.

Chapter 2

State of the art

During the last decades, road transport has remained as the primary way of moving both people and goods in the world. For instance, it accounted for almost 50% of the total goods transport and over 80% of the passenger transport in the European Union during 2015 (see Figure 2.1), according to the last statistics [European Commission, 2017].

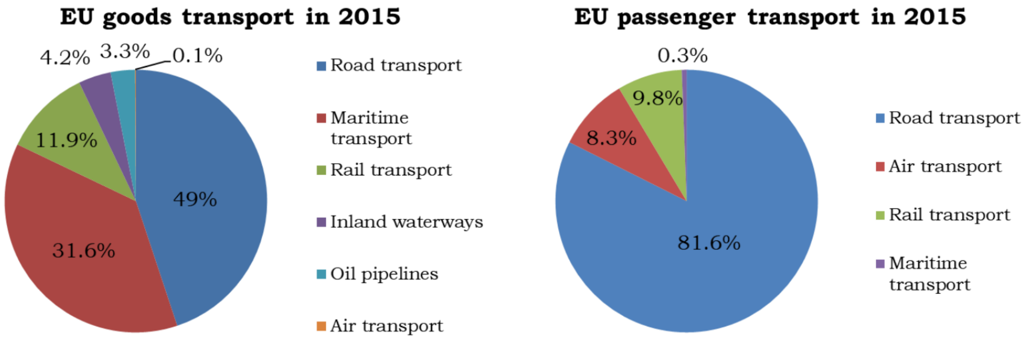


Figure 2.1 – Transport of goods and passengers in the EU in 2015 (from data at [European Commission, 2017])

Improving road safety is one of the top priorities for governments and institutions worldwide. Every day, 70 people die on European roads, and 370 people suffer serious injuries, which is equivalent to a large airplane [European Commission, 2016b]. This makes 135 thousand people seriously injured per year, which is equivalent to two large football stadiums. Therefore, as 26 thousand road fatalities have been registered each year since 2013, on average there are five severe injuries for each road fatality. Besides, in the United States, motor vehicle crashes were the leading cause of death for young people between 16 and 23 years old in 2015 [NHTSA National Center for Statistics and Analysis, 2017b].

Organizations and institutions worldwide, such as the National Highway Traffic Safety Administration, and the European Commission, are putting their efforts to reduce deaths, injuries and economic losses on roads. New frameworks for improving road safety have been created, including legislation and recommendations. For instance, in 2015, an agreement for the deployment of innovative technology that can save lives [European Commission, 2010], which entails that all the new vehicles from March 2018 have to be fitted with the eCall system.

Although the inclusion of technology in both vehicles and infrastructure, as well as the legislative plans, have reduced the number of fatalities this decade, the decrease rate has been stalled, and it even grew slightly in 2015, showing that the tendency is not as good as expected, as shown in Figure 2.2. Between 2001 and 2010, it was reduced by 43%, and between 2010 and 2015 by another 17%. However, last year only a 17% of reduction compared to 2010 was registered, and from 2013 to 2014 the reduction was close to zero, increasing on 2015. It means that the efforts must be stepped up to reach the strategic target of halving the number of road deaths by 2020. Thus, this great interest in improving this means of transport for both governments and enterprises led first to the birth of the Intelligent Transportation Systems (ITS). Then, this interest went through the inclusion of the Advanced Driver Assistance Systems (ADAS) on the roads, and the further appearance of the automated features on the driving task, i.e., the automated vehicles.

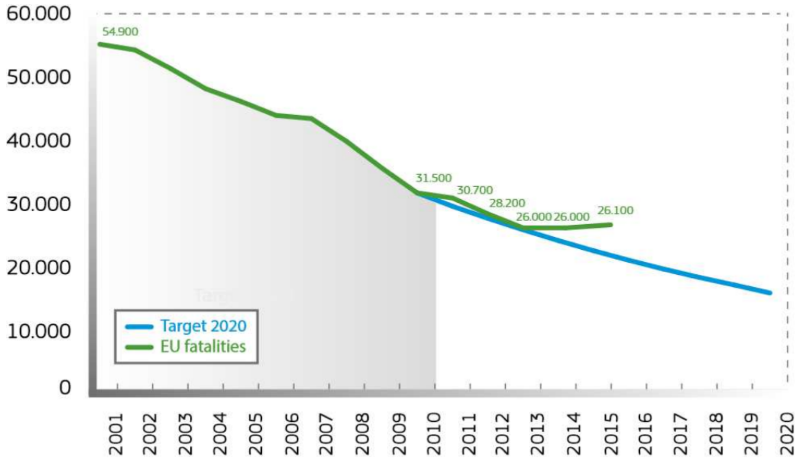


Figure 2.2 – EU Road fatalites and targets 2001-2020 (from data at [European Commission, 2016b])

The development of ADAS contributing to better and safer driving experience, providing a partial level of automation, is demonstrated thanks to systems such as the Adaptive Cruise Control, Lane Keeping Assist, or the Autonomous Emergency Braking system. Nevertheless, the greater complexity of urban environments makes them still represent an unsolved challenge. Automated vehicles have to interact with other vehicles and vulnerable road users, dealing with unexpected situations to perform collision-free trajectories through the desired itinerary. Path planning systems have an essential role since they allow the generation of trajectories where passengers comfort is a design parameter, avoiding the possible obstacles in the route in real-time, providing a smooth trajectory to the vehicle controller searching the best tracking on the vehicles. Although motion planning problems have been largely studied in robotics during last decades [Latombe, 1991], cars and robots present different constraints making necessary to consider different path planning strategies to deal with them. Among these constraints, the following can be highlighted: structured against unstructured environments, non-holonomic against holonomic systems, and different kinematics and dynamics of vehicles). Thus, a classification of the different path planning approaches in the literature has been done to consider the proper path generation algorithm on the path planning system presented in this thesis.

The state of the art is divided into three main blocks: first, a significant review of the ITS field evolution ranging from production ADAS up to most relevant worldwide research projects is carried out in Section 2.1. Second, the functional architecture for the automated vehicles proposed in this thesis is presented in Section 2.2. There, the seven stages that compose the architecture are introduced, focusing the planning stage as the target of this research. In addition, the assumptions and constraints considered for the planning system are summarized in Section 3.1.1. The third part of the state of the art is specifically focused on the path planning algorithms, which is the target of this thesis.

2.1 Evolution of the ITS to the automated vehicle

2.1.1 Economic and social impact of automated driving

This subsection presents some remarkable figures and facts about the social and economic impact of the ADAS and the automated vehicles, based on their current status of penetration on the automotive market, as well as on short-term predictions.

The cumulative safety contribution of available ADAS technologies works out to $16,307 per vehicle over a vehicle's 20-year life [Mosquet et al., 2015]. If all new-car buyers made an investment of $8,240, which is the price of these features, it would reduce by 30% the number of crashes and by 9,900 the number of fatalities in the United States. Furthermore, the motor crashes cost the USA $910B or 6% of the real gross domestic product each year. Current ADAS features could save $251B annually. Currently, it would represent a 98% of safety return delivered over vehicle's lifetime, which could become a 439% with the future fully automated cars [Mosquet et al., 2015].

Despite their enormous potential to improve transportation systems, ADAS features have a slow adoption curve, related to the economic cost for consumers. They are unwilling to pay as much for ADAS features as they cost to make and market, as can be seen in Figure 2.3. For example, most consumers appointed that they would pay on average $270 and as much $400 for the Blind Spot Detection system (BSD) when the actual cost is $595 per vehicle.

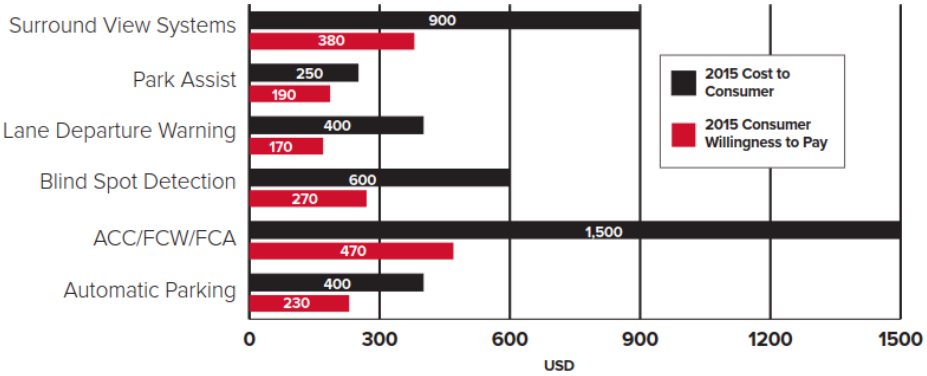


Figure 2.3 – Cost to consumer and consumers willingness to pay for ADAS features [Mosquet et al., 2015]

Figure 2.4 shows the evolution of the ADAS market up to 2016 and the expected evolution up to the horizon of 2020. It would reach up to $60.14 billion by then, registering a Compound Annual Growth Rate (CAGR) of 22.8% during the period 2014-2020. The growing trend for comfort and safety while driving, along with favorable government initiatives has contributed to this growth.

Improving the transportation systems is not only relevant for safety, but also from the economic point of view. Although the highest severity crashes decreased by 16.8% in the last decade in the United States, the number of motor-powered vehicle fatal crashes have increased 7% from 2014 to 2015, with an increment of 4.1% of non-fatal injury crashes and a 3.7% increase in property-damage-only crashes [NHTSA National Center for Statistics and Analysis, 2017b]. On average, 96 people died each day, and one person was killed every 15 minutes in motor vehicle accidents in the

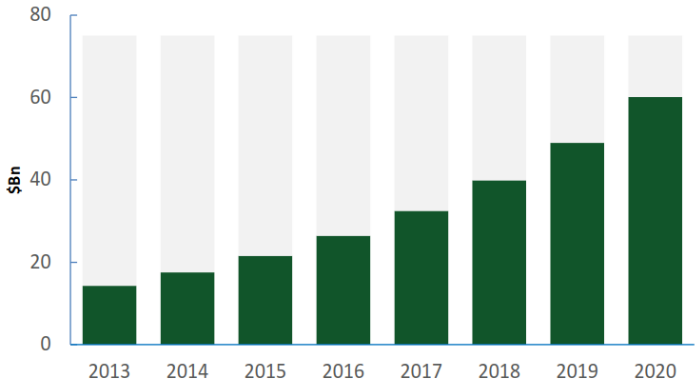
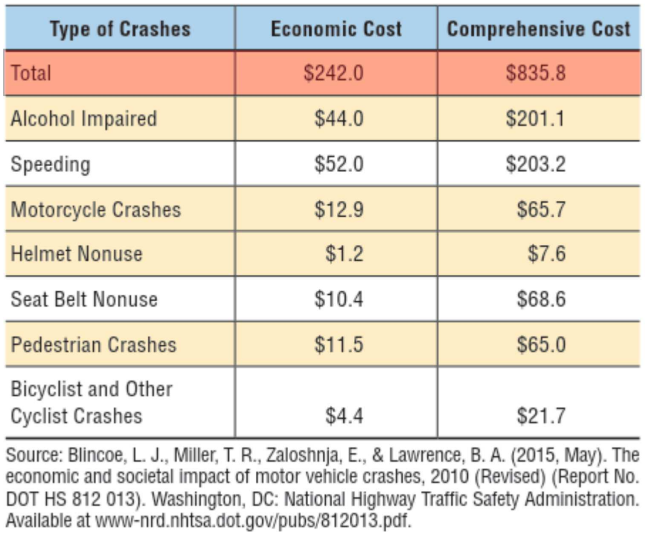


Figure 2.4 – Global ADAS market expectation [Woodside Capital Partners (WCP), 2016]

United States. Therefore, the estimated economic cost for the material losses that result from all motor vehicle traffic crashes in the USA in 2010 was $242 billion (see Figure 2.5).

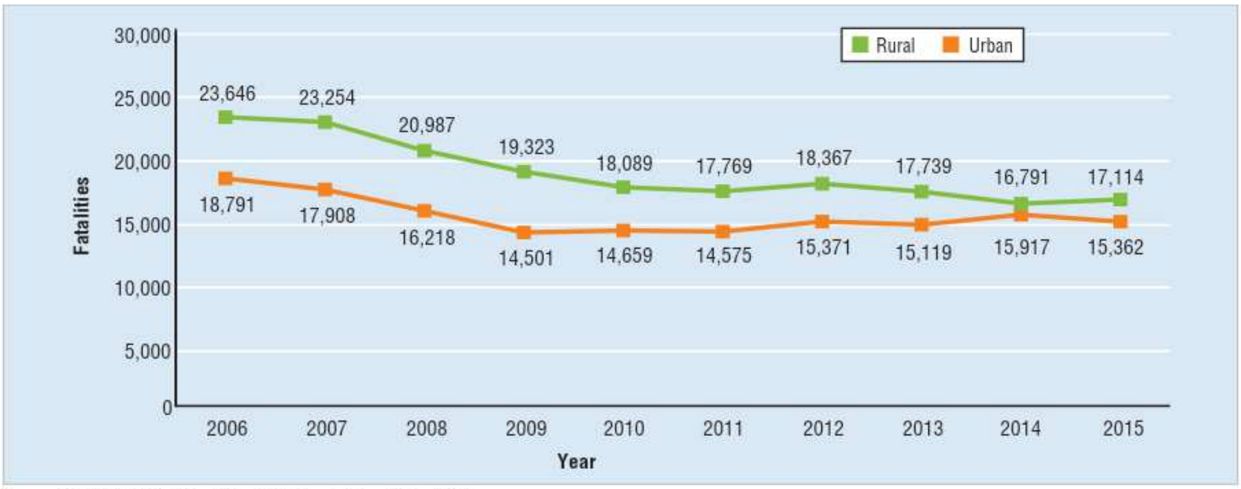


Type of Crashes	Economic Cost	Comprehensive Cost
Total	$242.0	$835.8
Alcohol Impaired	$44.0	$201.1
Speeding	$52.0	$203.2
Motorcycle Crashes	$12.9	$65.7
Helmet Nonuse	$1.2	$7.6
Seat Belt Nonuse	$10.4	$68.6
Pedestrian Crashes	$11.5	$65.0
Bicyclist and Other Cyclist Crashes	$4.4	$21.7

Source: Blincoe, L. J., Miller, T. R., Zaloshnja, E., & Lawrence, B. A. (2015, May). The economic and societal impact of motor vehicle crashes, 2010 (Revised) (Report No. DOT HS 812 013). Washington, DC: National Highway Traffic Safety Administration. Available at www-nrd.nhtsa.dot.gov/pubs/812013.pdf.

Figure 2.5 – Economic and comprehensive cost estimates in billions, 2010 [NHTSA National Center for Statistics and Analysis, 2017b]

The National Highway Traffic Safety Administration (NHTSA) of the U.S Department of Transportation (USDoT) pointed out that in 2015 there were more than 32 thousands of fatal motor vehicle traffic crashes, resulting in more than 35 thousands of fatalities [NHTSA National Center for Statistics and Analysis, 2017a]. Indeed, 45% of these accidents and 44% of the fatalities occurred in urban areas. Although the rate of urban fatalities has declined by 18% from 2006 up to 2015 (see Figure 2.6), these figures confirm that despite both industry and research have been working on the integration of in-vehicles ADAS it has not been enough to improve transport safety. Additionally, urban areas still suppose a big challenge regarding safety because of the interaction between cars and other vulnerable road users (VRU) such as pedestrian, cyclists or other two-wheeled motorized vehicles. Besides, cities would continue growing, reaching a percentage of 70% of the people living in cities and only 30% in the countryside [Verband der Automobilindustrie e. V. (VDA), 2015]. It will require a greater effort in the development of ITS solutions for these areas to reduce the associated problems such as long traffic lines or too few parking spaces.



Source: FARS 2006-2014 Final File, 2015 Annual Report File (ARF)

Figure 2.6 – Economic and comprehensive cost estimates in billions, 2010 [NHTSA National Center for Statistics and Analysis, 2017a]

Connected vehicles should lead to a reduction in the number of accidents on the roads, achieving a 90% of reduction by deploying more applications in domains such as the road design, traffic management, vehicle design, information and communications technologies, and human systems integration [Barbaresso et al., 2015].

2.1.2 Historical overview of the ITS - research and industrial projects, demonstrations and competitions

Intelligent Transportation Systems (ITS) emerged in the 1970s willing to facilitate the safe, clean, efficient and comfortable mobility of people and goods, saving lives, time and money. ITS are defined as systems that apply telecommunications, electronics and information technologies into road transport to plan, design, operate and maintain such transport systems [Nowacki, 2012].

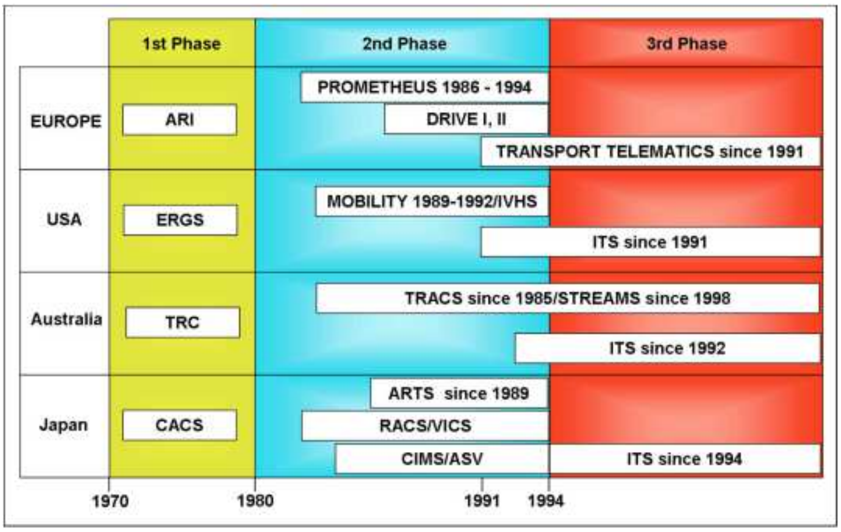

1st Phase
2nd Phase
3rd Phase
EUROPE
ARI
PROMETHEUS 1986 - 1994
DRIVE I, II
TRANSPORT TELEMATICS since 1991
USA
ERGS
MOBILITY 1989-1992/IVHS
ITS since 1991
Australia
TRC
TRACS since 1985/STREAMS since 1998
ITS since 1992
Japan
CACS
ARTS since 1989
RACS/VICS
CIMS/ASV
ITS since 1994
1970
1980
1991
1994

Figure 2.7 – History of first ITS developments (1970-1994) [Nowacki, 2012]

Figure 2.7 shows the timeline of the main developments in Europe, United States, Australia, and Japan, from their birth up to the acceptance of the ITS term. The first developments in the ITS appeared in the 1960s in the United States with the Advanced Vehicle Control System (AVCS) of the General Motors research group, which provided both automated lateral and longitudinal control. Besides, the MIT launched the METRAN (MEtropolitan TRANsportation) project, whose aim was applying new control techniques to urban transportation. This led to the further conceptualization of the ITS [Dingus et al., 1996], and the foundation of the CACS program (Comprehensive Automobile Control System) in Japan in the 1970s, to test an interactive route guidance system with an in-vehicle display unit in urban areas.

In the 1980s and beginning of the 1990s, the conditions for the developments of ITS were determined. The technological development of mass memories made possible a cheaper information process, which encouraged both manufacturers and the European Community to develop concurrently two projects in Europe: (i) the Eureka PROMETHEUS project (PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety, 1987-1995) [Eureka, 1987-1995], to improve the competitive strength of Europe by simulating developments in information technology, telecommunications, robotics, and transport technology. (ii) And the DRIVE project (Dedicated Road Infrastructure for Vehicle Safety in Europe, 1988-1991) [European Commission, 1988-1991], a European Commission project which looked forward to a Europe in which drivers would be better informed and in which intelligent vehicles would interact with their surroundings. The European Road Transport Telematics Implementation Coordination Organization (ERTICO) was funded in 1991 for all European and international organizations to work together for the sustainability of transport through the ITS.

At the same time, there were other projects to deploy the ITS worldwide. In Japan, the RACS (Road/Automobile Communication System) project [Takada et al., 1989] in 1984 formed the basis for the current car navigation system. In Australia, the project TRACS (Traffic Responsive Adaptive Control System) appeared as a pioneering project to evolve transportation concerning traffic management systems. In the United States, the Mobility 2000 group was the precursor of the IVHS (Intelligent Vehicle Highway Systems) program, a Federal Advisory Committee for the US Department of Transportation (USDoT).

The ITS America was established as a non-profit organization, and the term ITS was accepted in 1994 [Auer et al., 2016]. Since then, the prior programs started to be implemented and telematics was settled as a major topic of research, intending to develop new ITS applications and defining its standards, as promoted in the IV EU Framework program.

In 1997, the California PATH group, in collaboration with General Motors, presented an eight cars platoon on a highway scenario during the National Automated Highway Systems Consortium (NAHSC) Proof of Technical Feasibility Demonstration held in San Diego [Shladover, 1997]. The platoon operated with an inter-vehicle distance of 6.5 meters, accelerating, decelerating and performing coordinate stops, at speeds as high as the full highway speed (around 105 km/h) to prove the feasibility of improving the throughput of the transportation in highways.

Cybercars concept appeared in the 1990s [Parent and de La Fortelle, 2005]. These are fully automated road vehicles designed for passengers or goods transport, operating on-demand and with door-to-door capabilities for short trips at low speeds in urban areas. In 1997, they were operated for the first time in long-term parking at the Amsterdam Schiphol airport. Since then, several European projects (such as Cybercars and Cybercars-2) and different demonstrations have been done in the 2000s and 2010s to introduce cybercars in the cities public transport (such as the ones in La Rochelle or Antibes), considering a fleet of these cars operating together on platoon configurations.

Regarding international competitions, in 2004 the Defense Advanced Research Projects Agency (DARPA) proposed a Grand Challenge in the Mojave Desert region in the USA. It was the first long-distance competition for automated cars in the world, whose goal was to encourage the research and development of technologies needed to create the first fully automated ground vehicles capable of completing an off-road course. None of the vehicles participating in the first edition of 2004 finished the route. A second edition of the competition was held in 2005. Then, five vehicles completed the route. In 2007, an urban version of the challenge was held in Victorville, California. There, a mock urban scenario comprising four-way stop intersections, U-turns, and parking areas. There, six of the 11 participating vehicles completed the 90 km course, where vehicles had to respect all traffic regulations, dealing with other traffic and obstacles, and merging into traffic.

Since interoperability is a critical factor for a deeper development of ITS, new challenges sought to boost the development of cooperative vehicles, able to operate together efficiently by exchanging and interpreting data, providing information and services to other systems in real-time, such as the state of the roads, allowing better traffic management. For instance, it would allow an ambulance to arrive faster to the hospital by changing the timing of the traffic lights after notifying the accident. The Grand Cooperative Driving Challenge [Lauer, 2011] took place as an important competition to deepen the cooperative automated driving. It was held on a highway closed to traffic between Helmond and Eindhoven (Holland), in 2011. There, the nine European teams participating had to develop the longitudinal controller for a platoon configuration of heterogeneous vehicles, where they were exchanging their positions, velocities, and accelerations through wireless communication. There, research on new algorithms for sensor fusion, vehicle-to-vehicle communication, and cooperative control was tested [Geiger et al., 2012]. A second edition of the GCDC was held in 2016 as part of the European project i-GAME. On that occasion, three challenging cooperative scenarios were the focus of the competition: automated platoon merge, automated crossing and turn at an intersection, and automated space-making for emergency vehicles in traffic jam [Englund et al., 2016].

In 2010, another remarkable demonstration was carried out by the Vislab group (University of Parma), as part of the VIAC project. It consisted of an international journey with the PROUD automated car from Parma to Shangai. The course combined rural, freeway and urban open roads, where the vehicle was capable of dealing with the public traffic [Broggi et al., 2014].

There exist some other relevant projects that appeared between the 2000s and the 2010s such as HAVEit, SPITS or DESERVE. HAVEit project (Highly Automated Vehicles for Intelligent Transport) aimed to contribute to higher traffic safety and efficiency by designing a task repartition scheme between the driver and the co-driving system, a failure tolerant vehicle architecture and developing and validation the next generation of ADAS directed towards a higher level of automation. The SPITS project (Strategic Platform for Intelligent Traffic Systems) was a Dutch project that aimed to improve mobility and safety, focusing on three main areas: traffic management through cooperative driving and mobility, development of an upgradeable in-vehicle platform to deploy the different in-vehicle systems, and a service download and management solution. DESERVE (DEvelopment platform for Safe and Efficient dRIVE) European project (2012-2015) aimed to establish a new embedded software and hardware design by using a more efficient development process, overcoming challenges in reducing component costs and development time of future ADAS functions [Kutila et al., 2014].

Finally, from the Strategic Plan for IVHS, six functional areas can be identified in the development of the ITS [Sussman, 2008]:

- Advanced Traffic Management Systems (ATMS), to predict traffic congestion, offer alternative routing instructions improving the efficiency of the highway network, being able to control the traffic dynamically and performing an incident detection to reduce the road traffic.

- Advanced Traveler Information Systems (ATIS), to provide data both to road users at their vehicles or their workplaces and to transit users, such as the location of incidents, weather problems, road conditions, optimal routing and lane restrictions.
- Advanced Vehicle Control Systems (AVCS) -now Advanced Vehicle Safety Systems (AVSS)-, to make the travel safer and more efficient, which comprises collision warning features and emergency brake assist. In the longer term, those systems would imply a higher infrastructure information treatment to improve the efficiency of the roads, concept known as Automated Highway System (AHS).
- Commercial Vehicle Operations (CVO), improving the productivity of trucks, vans and taxi fleets.
- Advanced Public Transport Systems (APTS), to enhance the accessibility of information to users of public transport and the scheduling of public transport vehicles.
- Advanced Rural Transportation Systems (ARTS), to face the economic constraints in low-density roads.

Nowadays, these functional areas of the ITS are covered by 31 user services, which surged as an evolution of the National ITS Program Plan in 1995, providing a comprehensive planning reference for ITS, illustrating how the goals of ITS could be addressed through the development of these inter-related user services [Walton et al., 2000].

2.1.3 Advanced Driver Assistance Systems (ADAS)

Advanced Driver Assistance Systems (ADAS) are vehicle-based ITS designed to improve road safety concerning crash avoidance and injury prevention (primary safety), reduction of injury in the event of a crash (crash protection or secondary safety) and post-impact care assistance (to reduce the consequences of injury) [European Commission, 2016a]. They were born as an evolution of the first systems applied for the safety or convenience, such as the cruise control (1958), the seatbelt reminders (the 1970s), anti-lock braking systems (1971) and electronic stability control (1987) [Mosquet et al., 2015]. Nevertheless, as the European Commission pointed out, not all the in-vehicle systems are used for safety purposes but are also intended to improve the comfort or to manage the traffic.

ADAS can also be defined as electronic components installed in modern vehicles that present an intelligent driving experience to the driver. Their main challenge is the green, safe and supportive transportation, in particular to the accident-free mobility scenarios. Thus, the ADAS have three fundamental functions: aid, warn and assist the driver [Mosquet et al., 2015]. These systems began to be commercialized in the 2000s. So far, we can distinguish the following.

(i) First, as **aiding features**, we can find visual aids such as the night vision (2000), rear-mounted cameras (2002), adaptive front headlights (2006) and surround view systems.

(ii) Second, as **warning features**, there exist systems to alert the driver to potential dangers through sensory signals (auditory, light or vibrations). Some examples are the park assist (2002), the forward collision warning (2000), the lane departure warning (2005), the blind-spot and the rear cross-traffic detectors (2006), and the driving monitoring systems (2006) such as the driver fatigue or drowsiness monitoring.

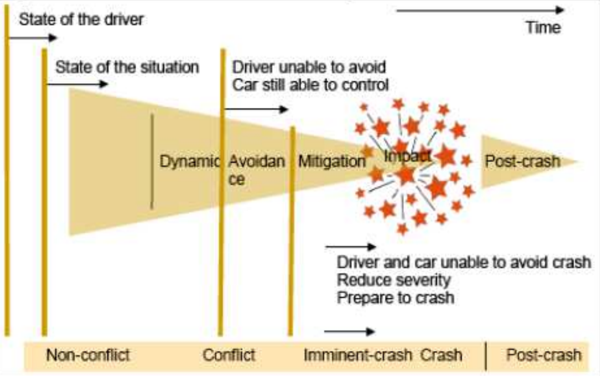


Figure 2.8 – The holistic view of safety [European Commission, 2016a]

(iii) Third, as **assisting features**, the ones that directly act on the vehicle are found, i.e., on the steering, accelerator or brakes to ensure the vehicle's safe operation. They include forward collision assist (2008), adaptive cruise control (2008), self-parking (2006), lane keep assist (2010) and pedestrian avoidance (2014). Some other features as the intelligent speed adaptation are envisaged to enter the market in the coming years.

In recent years, the number of ADAS integrated into commercial vehicles has increased. Based on the safety criteria described above (primary safety, secondary safety or post-impact assistance), the following ADAS can be found:

- **Crash/collision avoidance/warning systems (primary safety)**: These systems are thought to prevent and avoid the accidents, studying the causality of the accidents reducing them to the maximum level. These systems are placed at the beginning of the conflict part in Figure 2.8, where the car is still able to avoid the collision.
  - Autonomous emergency braking (AEB) systems (assisting feature): This emergency system either prevents collisions with approaching road users detected or reduces their severity in the event of a crash. The emergency brake assist applies the needed pressure to the break if the driver has not applied enough in an emergency situation. The study made by [Fildes et al., 2015] reflects that vehicles equipped with the AEB system reduced a 38% the front to rear crashes.

    Among the different systems and technologies which are under research and implementation we can highlight the following:
  - Forward Collision Warning (FCW) (warning feature): It warns visually and acoustically to the driver if the car is too close to the vehicle in front. Combining FCW with AEB has shown in [Cicchino, 2016] that can reduce a 39% the rear-end crashes without injuries, a 42% with direct injuries and a 44% with third-party injuries.
  - Reverse Collision Warning (RCW) (warning feature): This system warns visually and acoustically to the driver about the likelihood of collision with an object or person behind the vehicle by the use of sensors in the rear bumper, intensifying the magnitude when the distance between the vehicle's rear and the object decreases.
  - Adaptive Cruise Control (ACC) (assisting feature): It is the enhancement of the well-known Cruise Control (CC). It automatically maintains a fixed distance with the vehicle

  in front, thanks to the use of radar or lidar sensors, adapting the vehicle speed according to that of the vehicle in front.

  - Attention assist (aiding/warning feature): These systems monitor the driving behavior, such as the level of attention or drowsiness and the use of the steering wheel, alerting in case he or her seems to be sleepy.
  - Vision enhancement (aiding feature): Vision systems contribute to better detection of objects or other road users when driving, especially during night-time. For instance: the adaptive headlight system adjusts the headlight beam by moving it to optimize its operation with different weather or visibility conditions, whereas the night-vision cameras provide additional visual information to the driver.
  - Lane support systems (warning/assisting feature): The Lane Departure Warning System (LDWS) is a device that warns the driver if the vehicle is veering off the lane or the road, in case of having visibility of the road markings. On the other hand, the Lane Keeping Assist System (LKAS) is an automatic system which keeps the vehicle in its lane thanks to the lane marking recognition, except if some blinker is activated. Although it is a newly implemented technology in certain commercial vehicles and it is still early to measure its impact on safety, some figures can be highlighted: Nodine et al. [Nodine et al., 2011] analyzed the impact of the LDWS built-in 16 passenger cars on the safety, founding a 33% reduction in the rate of lane-change near crashes and a 19% reduction in road-departure near-crashes.
  - Blind spot monitor (aiding/warning feature): This system warns the driver about the presence of other road users in the blind spots of the vehicle. They may provide an additional warning if using the turn blinker when there is a vehicle next to the vehicle in another lane. The warning signals can be visual (in the side-view mirrors or the windshield frame), audible, vibrating or tactile.
  - Intelligent Speed Adaptation (ISA) (warning/assisting feature): It constitutes a primary safety system which warns the driver not to exceed the maximum speed limit of the road or other desired speed thresholds below this limit at safety-critical points, establishing the in-vehicle speed limit automatically according to that of the road. There are three types of ISA systems: (i) Informative or advisory: Gives the driver feedback through a visual or audio signal. (ii) Supportive or warning: increases the upward pressure on the accelerator pedal. (iii) Intervening or mandatory: prevents any speeding by reducing fuel injection or by requiring a reduction by the driver.
    As stated in [Elvik, 2009], speed is a key factor for the road safety. An increase of 5% on the mean speed leads to a 20% increase in fatal accidents. Decreasing the number of accidents would be translated into some other benefits such as a reduction in the cost, fuel saving, decrease in CO_2 emissions and potential to reduce the travel time.

- **Crash mitigation systems (primary safety)**: These are the active in-vehicle systems which aim to mitigate the severity of the crash when it is imminent and cannot be avoided. These systems are placed in the conflict part of Figure 2.8, just before the impact occurs.

  - Advanced Braking Systems (assisting feature): The Anti-lock Braking Systems (ABS) is a safety system conceived in the 1950s to prevent skidding where loss of steering and control results from locked wheels when braking hard, providing additional steering in an emergency situation. It contributes to reducing the collisions with vulnerable road users and collisions involving turning vehicles. However, ABS seems not to contribute

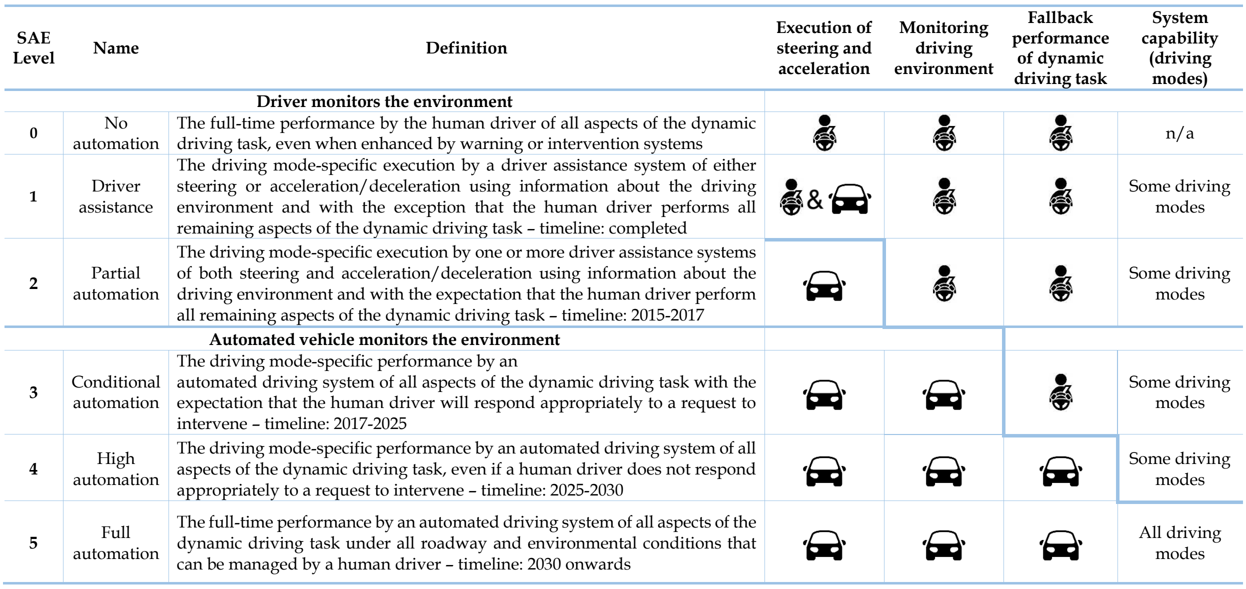
  to reducing the rear-end collisions. This system has been enhanced with the Electronic Stability Control (ESC) to control the front-to-rear brake bias electronically. It has speed sensors and independent braking for each wheel, identifying a critical driving situation and applying specific brake pressure on one or more wheels. Its implementation in the industry has meant a reduction of 22% in the number of crashes within the vehicles where this system is integrated. In addition, they had around 35% fewer accidents in wet and snowy conditions respectively.
  - Seat belt reminders (aiding feature): These devices constitute a warning system that detects if any of the seatbelt are not in use, emitting both a visual and an increasing acoustic signal until the belts are used. A combination of this system with airbags could reduce the death risk substantially in a collision.
  - Alcohol interlock systems (aiding feature): Alcohol interlocks are automatic control systems which are designed to prevent driving with excess alcohol by requiring the driver to blow into an in-car breathalyzer before starting the ignition. Introducing those systems would contribute to reducing the road deaths caused by an excess of alcohol in Europe, which currently comprises a 25% of all road deaths.

- **Crash protection systems (secondary safety)**: These systems have been developed during the last 20 years to reduce the injury severity during the impact phase.
  - Improvements in occupant restraint systems which better reflect the different human tolerance thresholds of male and female occupants and different age groups. In this category systems as the following can be mentioned: airbag, seatbelt, latch, head restraint or child safety seats. Those systems limit the injuries by restraining the occupants as quickly as possible after a collision. Their purpose is to slow the occupant over the longest possible time and distribute the crash forces over the largest area possible.
  - Multi-collision brake: These systems apply full braking and activate the hazard lights following a collision that has deployed the airbag. It aims to mitigate a subsequent collision with another vehicle or obstacle. If the driver considers that the risk is likely to increase, the system can be defeated by pressing the accelerator.

- **Post-crash response systems**: These systems aim to alert and advance emergency medical support in the event of a crash.
  - eCall: This system sends an automatic message to the emergency services right after a road crash including on it the accident location. This aims to reduce the time between the collision and the medical services are deployed, reducing the consequences of the accident, trying to prevent fatalities and disabilities. From April 2018, the new vehicles in the European Union must integrate the 112-eCall in-vehicle system [European Parliament and Council of the European Union, 2015].
  - Electronic driving licenses: which have an embedded smart card containing personal information about the driver, including which vehicles he or she is authorized to drive, serving as an ignition key access, allowing the driver to start the car if there is a correspondence between the card and the vehicle unit.
  - In-vehicle crash data or event data recorders and journey data recorders: The first ones are devices that collect data over a period before and after the crash and critical events, to monitor or validate new safety technology, to establish human tolerance limits or to record impact speeds. They are often based on the airbag control module and will

cease to store information once the airbag has deployed. The second ones collect data to provide information regarding driving behavior and low infringements. This data can be used for studying the traffic management.

2.1.4 Levels of driving automation for on-road vehicles



SAE Level	Name	Definition	Execution of steering and acceleration	Monitoring driving environment	Fallback performance of dynamic driving task	System capability (driving modes)
Driver monitors the 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	🧑	🧑	🧑	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 exception that the human driver performs all remaining aspects of the dynamic driving task – timeline: completed	🧑 & 🚗	🧑	🧑	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 – timeline: 2015-2017	🚗	🧑	🧑	Some driving modes
Automated vehicle monitors the 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 – timeline: 2017-2025	🚗	🚗	🧑	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 – timeline: 2025-2030	🚗	🚗	🚗	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 – timeline: 2030 onwards	🚗	🚗	🚗	All driving modes

Figure 2.9 – SAE Automation levels. Adapted from [SAE International, 2016]

SAE international published in 2014 a taxonomy for motor vehicle driving automation systems [SAE International, 2014], represented in Figure 2.9. This classification is based on the performance of the driving task, that can be done either by the driver, by the system or by both of them. This standardization body provides a classification of six levels of driving automation with detailed definitions for each of them. From no automation (level 0 or fully manual), where the driver performs all the driving task; to full driving automation (fifth level or fully automated), where is the vehicle which performs the driving task entirely.

This standardization took as a basis the two previous documents describing the levels of driving automation, namely: the US National Highway Traffic Safety Administration (NHTSA)'s "Preliminary Statement of Policy Concerning Automated Vehicles", published in May 2013 [National Highway Traffic Safety Administration (NHTSA), 2013], and the German Federal Highway Research Institute's (Bundesanstalt für Strassenwesen, a.k.a. BASt) "Legal consequences of an increase in vehicle automation", published in July 2013 [Gasser et al., 2013]. After it was published in January 2014, the International Organization of Motor Vehicle Manufacturers adopted the BASt levels and aligned them with the SAE's one, including the addition of the sixth level representing the full driving automation. The NHTSA adopted the SAE standard in 2016, and SAE published an update of the standardization [SAE International, 2016], which is the current version.

According to this classification, the described ADAS belong to either the first (driver assistance) or the second (partial automation) SAE levels. There, the human driver is in charge of monitoring the environment, and the system executes either the steering or the accelerator/brake pedal (first level), or both (second level).

The term automated vehicles can be applied moving forward to the third SAE level, also called conditional automation. There, is the system itself that monitors the driving environment, but the driver must respond to the request to intervene, whereas at the fourth level is the system that responds in any case. The fifth level corresponds to the fully automated car, where the system controls all driving modes. Although we are still far away from reaching the maximum level of automation, expected from 2025 [Mosquet et al., 2015], some features belonging to the partially automated driving (second level) and conditional automation levels (third level) are already commercial. They include the single-lane highway autopilot, introduced in 2014 by Tesla in their Model S. This system takes control of the steering and the pedals during the lane keeping operation while adjusting the speed to the road limits thanks to the recognition of signals on the way. Other features are being developed, such as the highway autopilot with lane change-assist (third level) and the automated valet parking (fourth level). Some others are expected for a more distant future (from 2025), as the urban autopilot.

A classification of the already established and the future ADAS can be done according to their level of automation, considering the passenger cars, the freight cars, and the urban mobility vehicles separately, as depicted in Figure 2.10.

(i) **Systems for Automated Passenger Car**: This kind of vehicles is the most extended, and thus, the one where both research and industries have put more efforts to continue developing automated driving. Among the systems developed, we can distinguish between the parking use cases and the driving scenarios.

- Automated Parking Assistance:
  (a) Parking Assist (second level): This system can accomplish the parking maneuver by itself both in private or in public parking places, but always under the supervision of the human driver, who may had to take control of the car to stop the maneuver if needed. It is already present in some commercial vehicles, such as in most of the BMW's X models, Mercedes-Benz, Volvo, Cadillac, Chevrolet, Ford, Infiniti, Land Rover, Lincoln and Buick cars.
  (b) Parking Garage Pilot (fourth level): The highly automated parking system is currently in the development phase. It is designed to control the parking maneuver to and from the parking place remotely (by smartphone or key) without human supervision.
  Mercedes has developed a variant of the parking assist that includes a 360°camera as well as the new remote parking pilot, both systems built in its E-Class. It intends to make easier the tasks of finding, choosing and leaving the parking places, braking if an obstacle is detected while doing the maneuver. Additionally, Mercedes-Benz and Bosch have promised a joint pilot project to deploy the automated parking. Their system claims to pick up the car from the drop-off area of the parking garage by the on-demand application. Then, it moves off into the parking garage and maneuvers into a free parking space, being able to park and unpark without any further human intervention.
- Automated Driving Assistance:
  (a) Traffic Jam Assist (second level): This system controls the vehicle longitudinally and laterally to follow the traffic flow at low or medium speeds (<60 km/h). The system can be seen as an extension of the ACC with Stop&Go functionality (i.e., no lane change support). Cars like the Volkswagen Passat and the Audi A4 have

	SAE Automation Level	**Established**	**Future**
Driver monitors the environment	0: **No automation**	**Function**: warning or support by active safety systems Lane Departure Warning System: 🚗 🚚 🚌 Blind-spot warning: 🚗 🚚 🚌 Forward collision warning: 🚗 🚚 🚌 ABS, ESC: 🚗 🚚 🚌 Emergency brake: 🚗 🚚 🚌	
	1: **Driver assistance**	**Advanced Driver Assistance Systems (ADAS):** Adaptive Cruise Control: 🚗 🚚 🚌 Stop & Go: 🚗 🚚 🚌 Lane Keeping Assist: 🚗 🚚 🚌 Lane Change Assist: 🚗 🚚 🚌 Parking Assist: 🚗 🚌	CACC Truck platooning: 🚚
	2: **Partial automation**	Traffic jam assist: 🚗 Parking assist: 🚗	Traffic jam assist: 🚚 🚌 Parking assist: 🚌 Automated truck platooning: 🚚 Urban bus assist: 🚌
Automated vehicle monitors the environment	3: **Conditional automation**		Traffic Jam Chauffeur: 🚗 🚚 Highway Chauffeur: 🚗 🚚 Automated Urban bus chauffeur: 🚌
	4: **High automation**		Highway autopilot: 🚗 Highly automated vehicles in confined areas 🚚 Highly automated vehicles on dedicated areas: 🚚 Highway pilot platooning: 🚚 Highly automated vehicles on open roads 🚚 Automated PRT/Shuttles on dedicated roads 🚌 Automated buses on dedicated roads 🚌 Automated PRT/Shuttles in Mixed Traffic 🚌 Automated Buses in Mixed Traffic 🚌
	5: **Full automation**		Fully automated passenger cars 🚌 Fully automated freight vehicles 🚌 Fully automated urban Mobility vehicles 🚌

Figure 2.10 – Deployment of automated vehicles according to the vehicle type. Adapted from [ERTRAC, 2017]

deployed this system in their vehicles. There, the car follows the vehicle ahead and automatically operate the steering, accelerator and brakes within the limits of the system, keeping this way the vehicle within the limits of the lane.

(b) Traffic Jam Chauffeur (third level): Conditional automated driving in traffic jams up to 60 km/h on motorways and motorway similar roads. Once the system is activated in a traffic jam situation, it detects slow-driving vehicles in front. Then, it takes control of both longitudinal and lateral commands, It does not require any supervision from the human driver. It adapts its speed to that of the surrounding traffic and the road speed limit. In order to make the system work, the speed at the traffic jam has to be under 50-70 km/h.

(c) Highway Chauffeur (third level): It is an enhancement of the traffic jam assist. It not only adapts the speed according to the other vehicles and road but also decides to change the lane to either overtake or return to the slower lane or leaving the highway. In the end, it consists of a lane keeping assist together with a lane change assist.

(d) Urban and Suburban Pilot (fourth level): Highly Automated Driving up to limitation speed, in urban and suburban areas. The system can be activated by the driver on defined road segments, in all traffic conditions. The driver can at all time override or switch off the system.

(e) Highway Autopilot including Highway Convoy (fourth level): Highly Automated Driving up to 130 km/h on highways or similar road from the entrance to exit, on all lanes, including overtaking and lane change. The driver must activate the system, but does not have to monitor the system constantly.

- Automated private vehicles on public roads: The fully automated vehicle must be able in the mid-term future to make an A-B itinerary without any human intervention. The driver will be able at all time to retake the control of the vehicle.

(ii) **Systems for Automated Freight Vehicles**: Automation of commercial vehicles for long-distance freight transport has a particular interest for manufacturers due to the lower technological gap that exists for deploying the systems since these vehicles operate in restricted lanes, generally in highways.

- Highway applications:

  Platooning is a kind of vehicles formation that will be deployed in scenarios where fuel saving and improvement of safety and traffic flow are sought. From mono-fleet platoons to multi-fleet platoons, from low-level platoons with driver involvement to high-level platoons with driver involvement.

(a) CACC Platooning (first level): Cooperative ACC (CACC) will be applied in trucks forming a partially automated truck platooning. There, the system makes the speed control keeping a short but safe distance to the leading vehicle, while the drivers remain responsible for all driving functions.

(b) Automated Truck Platooning (second level): It enables platooning in both dedicated lanes and on open roads in mixed traffic. The vehicle should be able to keep its position in the platoon with a safe distance between the vehicles. The leading vehicle transmits speed through V2V communication to the following vehicle. It should also take into consideration changes in the platoon, such as merging and dissolving platoons and the interaction with other road users.

    (c) Highway Pilot Platooning (fourth level): Automated driving on highways from entrance to exit on all lanes, including overtaking and lane changes. The driver will have to activate the system but is not in charge of monitoring the system.

  From a single vehicle point of view, besides the traffic jam assist, the traffic jam chauffeur and the highway chauffeur, the following systems are envisaged:

    (d) Highly Automated Trucks on Open Roads (fourth level): Highly automated trucks for automated operation on public roads in mixed traffic handling all typical scenarios without driver intervention on planned freight transport operation. Remote fleet and transport management and monitoring are required.
    (e) Fully automated freight vehicles (fifth level): The fully automated vehicle should be able to handle all driving from point A to B, without any input from the driver or passenger in all operating environments.

- Confined areas and dedicated roads:
    (a) Highly automated freight vehicles in confined areas (fourth level): Automated freight transport carriers in confined areas for potentially unmanned freight transport.
    (b) Highly automated freight vehicles in dedicated lanes/roads/areas (fourth level): Automated freight transport carriers on dedicated and controlled lanes/roads/areas and for potentially unmanned freight transport. A fuel reduction would be possible by operating at night.
    (c) Highly automated freight vehicles on open roads (fourth level): Automated freight transport carriers on public roads and for unmanned freight transport.

(iii) **Systems for Urban Mobility Vehicles**: The current automated systems running on the European urban environments require low speeds or dedicated infrastructure.

- Urban driving assist and chauffeur applications
    (a) Parking Assistance (second level): Idem than for system for automated passenger cars.
    (b) Traffic Jam Assist (second level): Idem than for system for automated passenger cars, but for low speeds up to 30 km/h.
    (c) Urban Bus Assist (second level): Automated assist functions for city-buses to increase productivity and safety for city bus operation such as bus stop maneuvering, short distance following, and maneuvering on narrow lanes.
    (d) Automated Bus Chauffeur (third level): Conditional automated driving in traffic jams up to 60 km/h on highways. The system can be activated in case of a traffic jam scenario exists. It detects slow-driving vehicles in front and then handles the vehicle both longitudinally and laterally.
- Highly automated urban applications
    (a) Automated Personal Rapid Transit (PRT)/Shuttles on dedicated roads (fourth level): The automated PRT/Shuttle drives in designated lanes/dedicated infrastructure, with a maximum speed of 40 km/h.
    (b) Automated Personal Rapid Transit (PRT)/Shuttles in mixed traffic (fourth level): The automated PRT/Shuttle drives in mixed traffic at the same speed as other traffic.

   (c) Automated buses on dedicated lanes (fourth level): The automated bus operates in dedicated bus lanes together with non-automated buses in normal city bus speed. Functions may include bus platoons, path following, and bus-stops automation.
   (d) Automated buses in mixed traffic (fourth level): The automated bus operates in mixed traffic on open roads together in normal city traffic speed. Functions may include bus platoons, path following, and bus-stops automation.

- Fully automated urban vehicles: Fully automated vehicles that can bring passengers to any destination as robotaxis, cybercars, automated shuttles and automated passenger cars.

2.2 Functional architecture for automated vehicles

Having the fully-automated vehicle running in both urban and highway roads is the final goal of both research and industry on the automotive field. Although we have not achieved this fifth automation level so far, expected from 2025 [Mosquet et al., 2015], the number of ADAS that are equipped in commercial vehicles has increased, leading to up to automation level three in commercial vehicles and four in prototypes for research.

Based on the category of application of the automated vehicles described in the previous section, the following already commercial systems can be highlighted.

2.2.1 Advanced automated vehicle systems

As automated passenger vehicles: Waymo presented a 360° video showing how the vehicle recognizes the environment and takes decisions based on it, operating as a taxi. Other companies which envisage offering this service soon are Uber and Tesla. Uber's cars are based on a hybrid perception system combining multi-camera (up to 20) with LiDAR and GPS for localization. Tesla became famous thanks to their highway autopilot which was commercialized in 2014 in their Model S. This system takes control of the steering and the pedals during the lane keeping operation while adjusting the speed to the road limits thanks to the recognition of signals on the way.

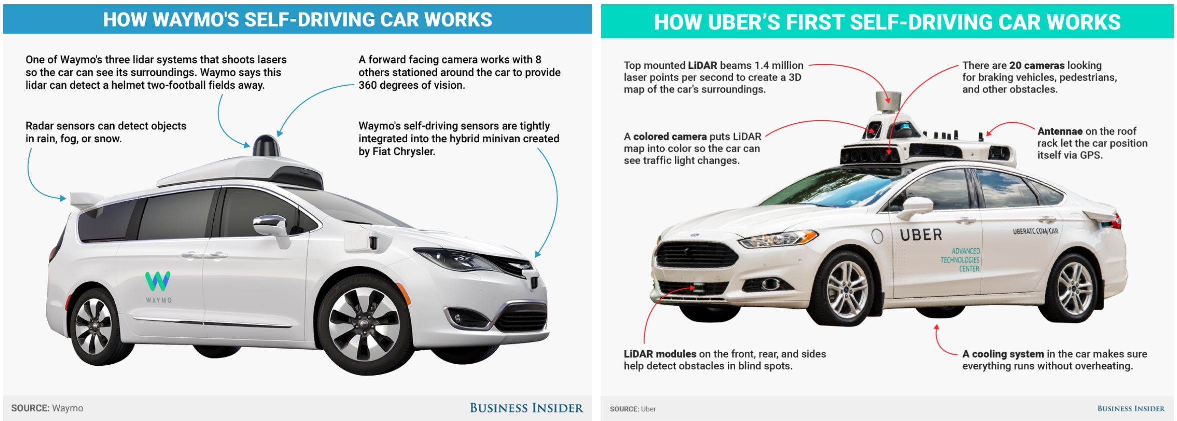

HOW WAYMO'S SELF-DRIVING CAR WORKS
One of Waymo's three lidar systems that shoots lasers so the car can see its surroundings. Waymo says this lidar can detect a helmet two-football fields away.
A forward facing camera works with 8 others stationed around the car to provide 360 degrees of vision.
Radar sensors can detect objects in rain, fog, or snow.
Waymo's self-driving sensors are tightly integrated into the hybrid minivan created by Fiat Chrysler.
WAYMO
SOURCE: Waymo
BUSINESS INSIDER
HOW UBER'S FIRST SELF-DRIVING CAR WORKS
Top mounted LiDAR beams 1.4 million laser points per second to create a 3D map of the car's surroundings.
There are 20 cameras looking for braking vehicles, pedestrians, and other obstacles.
A colored camera puts LiDAR map into color so the car can see traffic light changes.
Antennae on the roof rack let the car position itself via GPS.
UBER
ADVANCED TECHNOLOGIES CENTER
UBERATC.COM/CAR
LiDAR modules on the front, rear, and sides help detect obstacles in blind spots.
A cooling system in the car makes sure everything runs without overheating.
SOURCE: Uber
BUSINESS INSIDER

Figure 2.11 – Current automated cars: Waymo's (left), Uber's (right)

Lyft-Aptiv presented in the Consumer Electronic Show of 2018 (CES2018) their BMW semi-automated car operating as a taxi service on Las Vegas. Audi, in cooperation with Nvidia, was a pioneer in introducing the Artificial Intelligence in the automated vehicles. Some other features are being developed, such as the highway autopilot with lane change-assist (third level) and the automated valet parking (fourth level). Some others are expected for a more distant future (from 2025), as the urban autopilot.

Figure 2.12 – Automated passenger vehicles: Lyft-Aptiv (left) and Audi-Nvidia's AI (right)

Navya and EasyMile (France) are the leading companies in the deployment of *robotaxis*. Figure 2.13 shows the Navya's AUTONOM CAB and the EasyMile's EZ10. The first one (Navya's) was presented as the first 100% automated robotaxi. They envisage with this shuttle to complement the mass transport systems for long distances and becoming the automated taxi for short distances. It presents neither steering wheel nor pedals, and it can accommodate six passengers. It works on-demand, being able to be requested through a mobile application. Navya claims that, unlike other companies working such as Uber and Lyft, they do not add new vehicles to the roads making more significant the congestion on cities and augmenting the pollution as well. In contrast, they promote the car sharing, either as a private service (taxi) or as a public service (shuttle). The second one (EasyMile) has shown the application of these platforms as a mobility service in airports, moving people from the parking space to the terminal, presenting obstacle detection and emergency braking capabilities. They also showed its versatility on a demonstration in Lausanne University, where the vehicle operates in three different modes (metro, bus, and on-demand), with a capacity of 12 passengers. It uses hybrid localization (laser, vision, and GPS) making unnecessary the lane marking.

Crédit photo : EasyMile

Figure 2.13 – Robotaxis: Navya's *AUTONOM CAB* (left) and EasyMile's EZ10 (right)

Cybercar platforms are urban vehicle prototypes designed to operate as a fully automated city shuttle system. They have been largely used at research centers such as Inria and Mines ParisTech to develop the concept of Cybernetic Transportation Systems (CTS). European projects such as Cybercars, Cybercars2, CityMobil, and CityMobil2 have targeted the research and development of these vehicles.

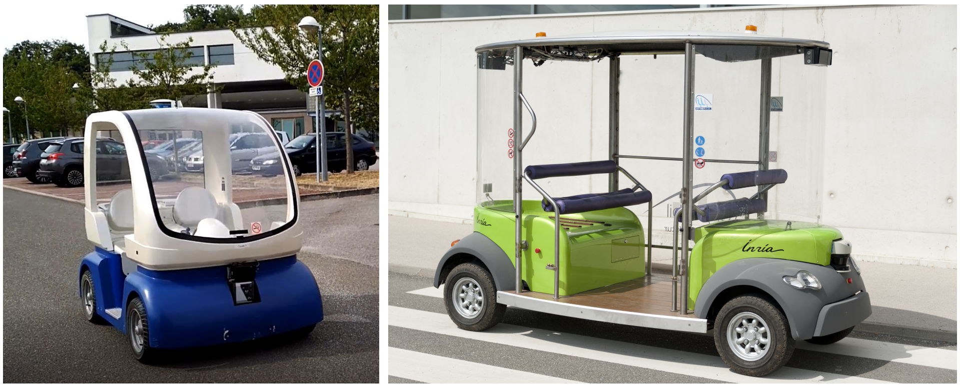


Figure 2.14 – INRIA's Cybercars: Cycab (left) and Cybus (right)

The project team RITS (Robotics and Intelligent Transportation Systems), previously IMARA, is a multidisciplinary project at INRIA (from the French *Institut National de Recherche en Informatique et en Automatique*), working on robotics for intelligent transportation systems towards the design of advanced intelligent robotic systems for autonomous and sustainable mobility. They cover a wide range of research topics within the ITS: multi-sensor signal processing and data fusion, advanced perception for environment modeling and understanding, route planning, vehicle control, wireless communications, large-scale traffic modeling and simulation, and the development of automated vehicles. The goal of these studies is to improve transportation in terms of safety, efficiency, comfort and minimize nuisances.

Together with Mines ParisTech, the RITS team is part of the *La Route Automatisée* consortium, associated to the *Drive4U* International Chair, involving the École Polytechnique Fédérale de Lausanne (EPFL), the University of Berkeley and the Shanghai Jiao Tong Univ. (SJTU) as academic partners. It is also a partner of the euRobotics European network, as well as the french MOV'EO cluster, being involved in numerous European projects on automated navigation, ADAS, cooperative driving and traffic management.

Furthermore, the team is deeply involved in the development of the cybercars. These platforms, also called Cybernetic Transportation Systems (CTS), are low-speed vehicles designed to operate on urban roads, both for passengers and goods transport [Roldao et al., 2015]. A cooperation is underway with Robosoft, Valeo, Akka, Renault (France), Yamaha (Japan) and the SwRI (USA) to develop and test these vehicle prototypes. Inria is then one of the major test sites for CTS, having a fleet of Cycabs and Cybuses in the Inria-Rocquencourt site. Together with a recently robotised Citroën C1, they constitute the platforms of the team.

2.2.2 A reference automated vehicle architecture

The operation of all automated vehicle is based on a modular architecture such as the one proposed in Figure 2.15. It constitutes the enhanced and up-to-date version of the architecture for cybercars presented first in [González and Pérez, 2013] and then improved in [Roldao et al., 2015] with the description of the technical specifications, as well as in [González et al., 2016b] with the inclusion of a control sharing stage. This architecture is composed of seven main stages, which allows describing the whole behavior of the vehicle according to the functions that it accomplishes, namely acquisition, perception, localization, communication, decision, control and actuation, finding some auxiliary stages to manage the system as well. Each of these stages is described below.

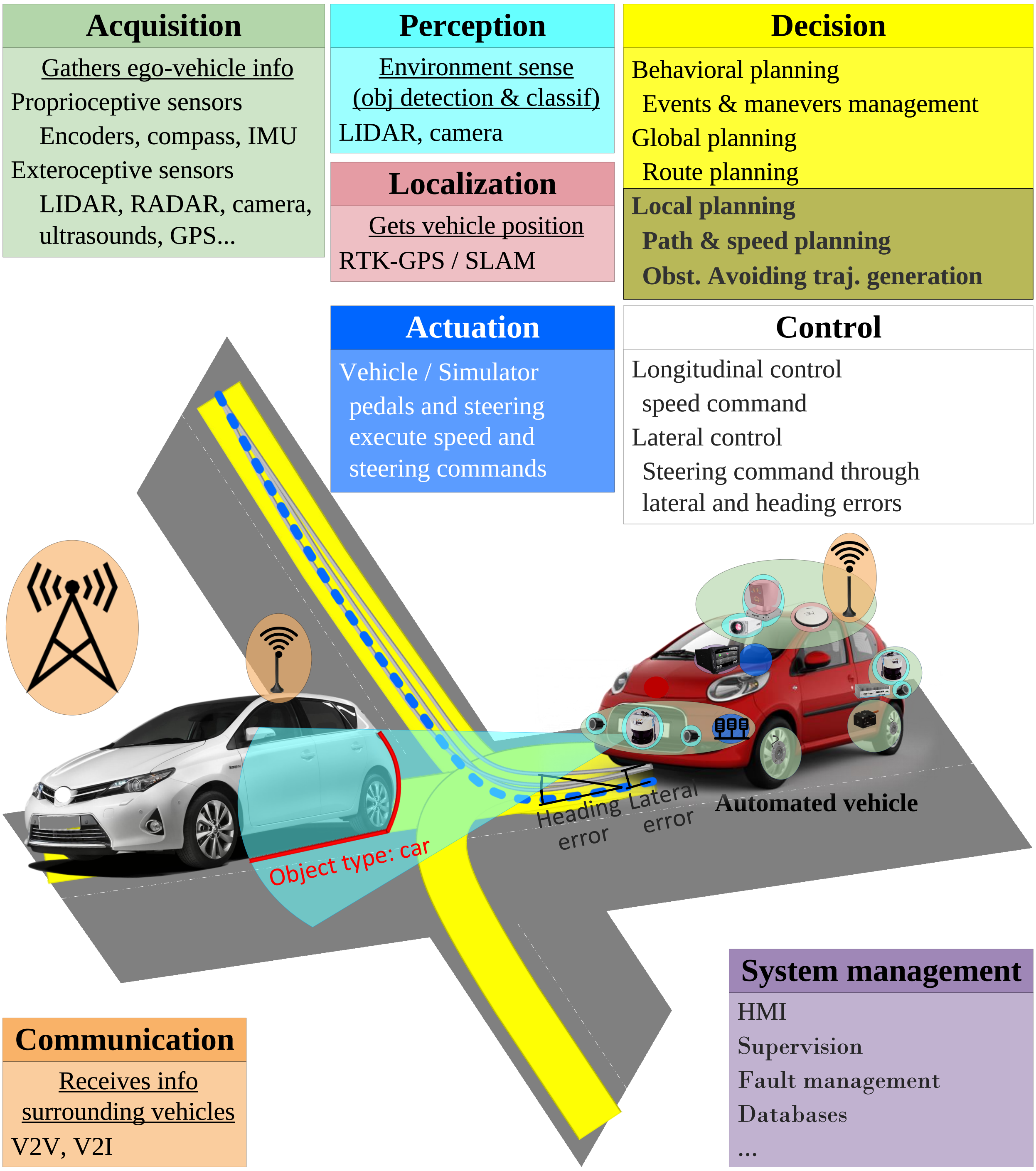

Acquisition
Gathers ego-vehicle info
Proprioceptive sensors
Encoders, compass, IMU
Exteroceptive sensors
LIDAR, RADAR, camera, ultrasounds, GPS...
Perception
Environment sense (obj detection & classif)
LIDAR, camera
Localization
Gets vehicle position
RTK-GPS / SLAM
Decision
Behavioral planning
Events & manevers management
Global planning
Route planning
Local planning
Path & speed planning
Obst. Avoiding traj. generation
Actuation
Vehicle / Simulator
pedals and steering execute speed and steering commands
Control
Longitudinal control
speed command
Lateral control
Steering command through lateral and heading errors
Object type: car
Heading error
Lateral error
Automated vehicle
Communication
Receives info surrounding vehicles
V2V, V2I
System management
HMI
Supervision
Fault management
Databases
...

Figure 2.15 – INRIA RITS vehicles architecture

Acquisition (in green in Figure 2.15): This stage is in charge of retrieving the information from the environment with both the proprioceptive and the exteroceptive sensors. On the one hand, the proprioceptive sensors measure the internal values of the vehicle, such as the encoders for the speed, compass for the orientation, GPS for the position and IMU for altitude, velocity and position, among others. On the other hand, the exteroceptive sensors obtain the information from the vehicle environment. Some examples are the LIDAR (Light Detection and Ranging) and RADAR (Radio Detection and Ranging) lasers for obstacles detection and classification, as well as for the vehicle localization, allowing to generate a map of the surrounding environment where both ego-vehicle and obstacles are localized. Other exteroceptive sensors are the ultrasonic sensors, for short-term obstacles detection, used mostly in parking maneuvers as a warning system; and cameras, to perceive the signals, the lane marking or the nearby vehicles and pedestrians in the scene.

Perception (in light-blue in Figure 2.15): It fuses the information extracted from the acquisition and makes the environment sensing, detecting and classifying the objects on it. Hence, it provides the state of the vehicle to the upcoming stages as well as the numerical information about the surrounding, built from the data received.

Localization (in red in Figure 2.15): In this stage the vehicle position is obtained by one of the following methods: either with Real Time Kinematic Differential Global Positioning System (RTK-GPS) or a fusion of GPS data with the Inertial Measurement Unit (IMU) data, or by the Simultaneous Localization and Mapping (SLAM) algorithm [Trehard et al., 2014]. The last one creates a map of the surrounding fusing the data of the LIDAR lasers.

Communication (in orange in Figure 2.15): It allows the information exchange between vehicles (V2V), and between vehicle and infrastructure (V2I), such as the position, speed, and acceleration of the vehicles, or the road information such as the traffic state or the presence of accidents.

Decision (in yellow in Figure 2.15): This stage is considered the core of the architecture. It receives the information of the environment coming from the perception and communication stages and decides which maneuver perform, generating the proper trajectory. Accordingly, it is divided into three sub-stages. (i) Behavioral planning, in charge of the maneuver decision (i.e., vehicle stop, overtaking, or yielding at intersections, among others). (ii) Global planning, receiving the mission order, as going from an origin to a destination point, and computing a first route consisting of way-points defining the itinerary. (iii) Local planning, which generates a collision-free trajectory to reach the destination, smoothing the itinerary.

Control (in white in Figure 2.15): It receives the planned trajectory (i.e. the path to be followed and the reference speed profile), and combines longitudinal and lateral control methods, for both steering wheel and throttle and brake pedals, to minimize errors in the tracking of the path, ensuring the stability of the system and preserving the passenger comfort.

Actuation (in blue in Figure 2.15): This is the final stage of the architecture, which acts directly through the vehicle pedals and steering wheel, applying the speed and steering commands received from the control stage, respectively.

System management (in purple in Figure 2.15): There are some auxiliary modules to manage the system (in purple), such as the HMI, the supervision sensors, the fault management, and the databases, among others.

Based on the described architecture for automated vehicles (see Figure 2.15), trajectory planning composes the core of the decision-making stage. It is performed at different levels, namely global and local planning, which form the phases in which the planning stage is divided. Global planning (also called route planning) is in the highest level of the planning phase. It is in charge of generating a first route from the origin to the destination. Local planning is in the lowest level of the planning phase. It considers the dynamism of the scene to generate a feasible continuous trajectory, respecting the different constraints regarding vehicle kinematics, motion and road, whereas providing a dynamic behavior, being able to avoid dynamic obstacles and re-compute the trajectory if an unexpected situation occurs.

A review of the state of the art on path planning techniques is presented in Section 2.3.

2.3 Path Planning Techniques for Automated Vehicles

Motion planning algorithms can be classified according to two different criteria: the completeness (exact or heuristic) and the scope (global or local) [Hwang and Ahuja, 1992]. Since exact algorithms always find a solution or prove its non-existence, these algorithms are usually computationally expensive, whereas heuristic algorithms are aimed to generate a solution in a short time. On the one hand, the resolution completeness is related to discretization. When continuous quantities are discretized, the associated algorithm is approximate, and its accuracy depends on the resolution of the discretization. Thus, an algorithm is resolution complete if an algorithm is exact in the limit as the discretization approaches a continuum.

There are several techniques for generating trajectories that can be used in the global and local planning phases. The ones more used in the literature for automated ones are classified into the following four categories: based on graph search, based on sampling, based on curves interpolation, and based on numerical optimization. The most important algorithms of each category are introduced below.

2.3.1 Graph search based algorithms

The main objective of path planning algorithms is to make the target (i.e., the robot or the vehicle) arrive from an initial position to a destination point. The road space through which the vehicle passes can be discretized employing grids or lattices, which represent the space as occupied or free, depending on the obstacles found on the road and the road restrictions. Thus, the path can be understood as a set of states through which the vehicle passes to arrive at the destination. Path planning can be generated therefore using a graph search algorithm. These algorithms not always find a solution, and the solution found might not be the optimal one [González et al., 2016b].

2.3.1.a Dijkstra

This algorithm finds single-source shortest paths on a weighted directed graph [LaValle, 2006]. There, all edge weights are non-negative [Bestaoui Sebbane, 2014]. It was developed by Edsger Dijkstra in 1959 with the purpose of finding the shortest path through a set of interconnected nodes. Since Dijkstra's algorithm always chooses the lightest or closest vertex, it is considered a greedy algorithm, which means that it does not provide optimal solutions, but computes shortest paths. However, the graph needed to represent a road network can be formed by millions of edges, making this algorithm impractical [Paden et al., 2016].

This algorithm has been implemented in the following automated vehicles: The Ben Franklin Racing Team's vehicle LittleBen, which was one of the six vehicles that finished the DARPA Urban Challenge. Its motion planning algorithm computes an alternative sequence of way-points using Dijkstra to adapt the response to the traffic conditions [Bohren et al., 2008]. In addition, the VictorTango's team vehicle Odin, which finished in the third position the DARPA Urban Challenge, used the algorithm to perform a parking maneuver, selecting the control points for navigating toward the parking spot and reversing out of the spot [Bacha et al., 2008]. Li et al. [Li et al., 2009] also implemented Dijkstra's algorithm to calculate the shortest path between the source node and target node. It has been applied to taxis GPS trajectories, reflecting a hierarchical cognition of road network. They performed three programs which respectively are Dijkstra algorithm based on original graph, hierarchical route planning algorithm based on a hierarchical graph by road class and hierarchical route planning based on a hierarchical graph by taxi experience. They use the Dijkstra's algorithm on a flat graph of the road network.

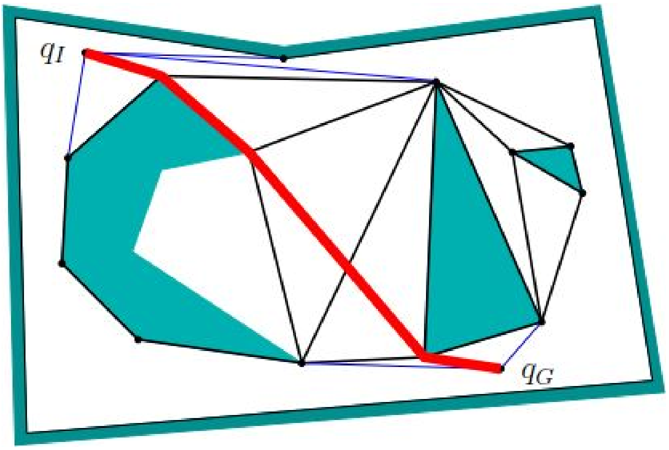


Figure 2.16 – Dijkstra algorithm [LaValle, 2006]

2.3.1.b A-star (A*) based algorithms

It is described as an extension of the Dijkstra algorithm [Delling et al., 2009]. It improves the Dijkstra's running time in practice if a heuristic is available, by focusing the search towards the goal [Bestaoui Sebbane, 2014]. It is used to compute minimum cost paths in graphs, searching a graph efficiently for a chosen heuristic. Since it returns an optimal path in the heuristic is optimistic, i.e., it always returns a value less than or equal to the cost of the shortest path from the current node to the goal node. The output of the A* algorithm is a back pointer path, which is a sequence of nodes starting from the goal and going back to the start.

This algorithm tries to reduce the total number of states explored by incorporating a heuristic estimate of the cost to get to the goal from a given state. It works in the same way as Dijkstra's algorithm, but it guarantees to find optimal plans [LaValle, 2006]. Since it is a classical shortest path search algorithm like Dijkstra, it results impractical if the graph representing the road network is not small, which may contain for example millions of edges [Paden et al., 2016].

KIT AnnieWAY's team used this algorithm on its automated system in the 2007 DARPA Urban Challenge [Kammel et al., 2008]. Since the zones where the Urban Challenge took place were parking lots and off-road areas, there a graph for path planning is not available. Therefore, an implicit search graph in which all paths are feasible was defined. Employing this algorithm, they expanded the search graph, accelerating the exploration of the search space by using a combination of two cost functions for both kinematic constraints and for evaluating the graph free and occupied space due to the obstacles encountered, which allows finding the least-cost path.

Ziegler et al. [Ziegler and Werling, 2008] used the A* algorithm to find the least-cost path onboard the AnnieWay team vehicle, which participated in the DARPA Urban Challenge. This algorithm provided good results for static environments, and combined with Voronoi heuristics guided the search towards the target even in unstructured environments.

There exist some variations of the A* algorithm, where the following can be highlighted:

Weighted A*: It is an extension of A* that trades-off running time and solution quality [Bestaoui Sebbane, 2014]. The main difference with respect to A* is that it inflates the heuristic by a factor greater or equal than 1. It makes the higher it is, the greedier the search and the sooner a solution is found, offering a sub-optimality bound, i.e., is no costlier than factor times the cost of the optimal solution.

Dynamic A*: Stentz, from CMU, introduced the Dynamic A* as a dynamic approach of A* [Stentz, 1994]. The main difference with respect to the A* is that the arc cost parameters can change during the execution of the algorithm. It was tested on sensor-equipped robots as the path planning algorithm capable of work under unknown, partially known and changing environments, in an efficient, optimal and complete manner.

Field D*: Ferguson and Stentz presented the Field D* and the Multi-resolution Field D* algorithms [Ferguson and Stentz, 2006b], two interpolation-based algorithms that extend D* using linear interpolation to produce low-cost paths that eliminate unnecessary turning efficiently. A significant limitation of current planners arises from the use of uniform resolution grids to present the environment. The use of this kind of grids is usually unfeasible due to the large amount of memory and computation required to store and plan over these structures. Therefore, they propose this multi-resolution approach to overcome these inconveniences.

θ*: Another extension of the A* is the θ* [Bestaoui Sebbane, 2014]. This algorithm does not restrict the search to the neighboring nodes, but it allows to search through the nodes in the line of sight with the expanded node. Daniel et al. presented two different approaches to this algorithm [Daniel et al., 2014]: the Basic Theta* and the Angle-Propagation Theta*. Both are intended to face the issue of the artificially constrained headings of the paths formed by grid edges whose lengths can be longer than true shortest paths in the terrain. They proved the correctness and completeness of these Theta* approaches and compared them to other any-angle path planning algorithms, namely A* with post-smoothed paths (A* PS), A* on visibility graphs and Field D* (FD*). It was tested for path planning in video games maps formed by considerably large grids (100x100 and 500x500), showing that Theta* finds shorter paths than both A* and Field D* with a runtime comparable to that of A* on grids.

Anytime repairing A* (ARA*) and Anytime D* (AD*): Likhachev et al. introduced the Anytime Repairing A* algorithm [Likhachev et al., 2008], which is an efficient anytime heuristic search that also runs a series of A* searches with inflated heuristics but satisfying the sub-optimality bounds. They demonstrate the efficiency of the algorithm on a motion planning application involving a simulated robotic arm with several degrees of freedom. Anytime planning algorithms are useful when the environment is known in advance, but they are not appropriate when the environment is barely known or dynamic. Additionally, they present the Anytime D* (AD*) algorithm, a search algorithm that is both anytime and incremental. It enhances the ARA* algorithm, which stops the execution when it finds a solution even if more planning time is available. On the contrary, the AD* algorithm re-uses its old search efforts while concurrently improving its previous solution. This algorithm (AD*) was also used by the Boss vehicle of the Tartan Racing team of the Carnegie Mellon University, the winners of the DARPA Urban Challenge [Likhachev et al., 2008]. It quickly generates an initial, suboptimal plan for the vehicle and then improves the quality of this solution while deliberation time allows to incrementally repair its solutions when changes in the environment occur. The algorithm is able to efficiently repair its existing solution to account for the new information.

Hybrid A*: An evolution of the A* algorithm, called Hybrid A*, was implemented by the Stanford University on its Junior vehicle in the DARPA Urban Challenge [Montemerlo et al., 2008]. There, the vehicle state is represented in a four-dimensional discrete grid, where the dimensions correspond to the (x,y) vehicle's position, its direction and the direction of motion. Hybrid A* is more convenient than the non-hybrid A* for vehicles running on unstructured environments, since the A* states are discrete and the environment is continuous and therefore big enough to be represented with discrete states. There, two cost functions are used to guide the search process. First one considers the kinematic constraints of the vehicle, and the second one is derived from the Voronoi's graph of the vehicle's free space and thus incorporates knowledge of shape and position of the obstacles.

Montemerlo et al. compared A* (left) to Field D* (center) and Hybrid A* (right) , as depicted in Figure 2.17. A* associates cost with centers of cells and visits only states that correspond to grid-cell centers. Meanwhile, Field D* associates cost with cell corners allowing arbitrary linear paths from cell to cell, and Hybrid A* associates a continuous state with each cell, where the score of the cell is the cost of its associated continuous state. As can be deduced from the figure, the paths generated by A* and Field D* cannot be easily tracked by vehicles, whereas the one generated by Hybrid A* it is thanks to associating continuous coordinates with each grid cell.

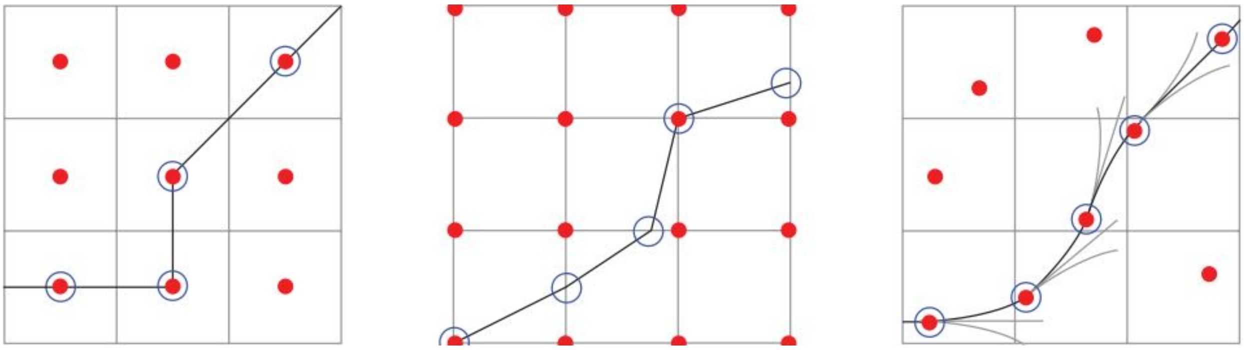


Figure 2.17 – Comparison of A*, D* and Hybrid A* search algorithms [Montemerlo et al., 2008]

2.3.1.c State lattices

This algorithm searches the discretization of the vehicle's states space as a direct graph to find a motion plan that satisfies the constraints [Pivtoraiko and Kelly, 2008]. Therefore, a state lattice can be defined as a discretization of the configuration space into a set of states, representing configurations, and connections between these states, where every connection represents a feasible path [Likhachev and Ferguson, 2008].

A state lattice is built from a discretization or sampling strategy to represent the states in the lattice, and the action space or control set for the inter-state connections, which has to be dense enough that every feasible path through the lattice can be constructed by combining sequences of these actions. There, the state space is formed by its position, orientation, and maximum forward and reverse velocity. Lattices provide a method for motion planning problems to be formulated as graph searches. The feasibility requirement of lattice connections guarantees that any solutions found using a lattice will also be feasible. Thus, they are suitable for being used for motion planning with non-holonomic vehicles.

The state lattice is represented as a cyclic directed graph, where nodes are discrete values of state [Pivtoraiko and Kelly, 2008]. Finding a motion plan that satisfies differential constraints is reduced to finding a path (a sequence of nodes and edges) in the state lattices graph. Each edge can be assigned a cost. Since the state lattice is a directed graph, any standard systematic search algorithm can be utilized to find the shortest path in it, which would correspond to a minimum-cost feasible motion that drives the system from the initial to the goal state. Figure 2.18 shows a state lattice formed by a repeated and regular pattern of vertices and edges.

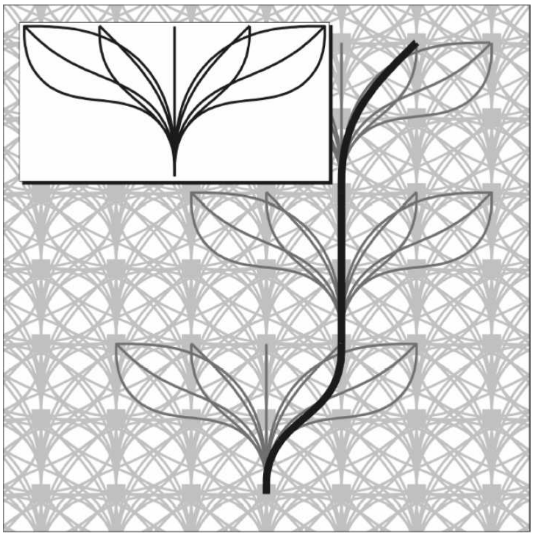


Figure 2.18 – State Lattices [Pivtoraiko and Kelly, 2008]

Howard et al. from CMU presented an algorithm for wheeled mobile robot trajectory generation in rough terrains for space stations that achieves a high degree of generality and efficiency [Howard and Kelly, 2007]. There, numerical optimization methods are required to solve the nonlinear equations of motion for such arbitrary terrain. They applied state lattices for creating an inherently feasible search space for global motion planning, connecting each state with feasible motions that serve as edges to make the states transitions, where the edges are adapted to the shape of the terrain.

The lattice is assumed to contain all feasible paths up to a given resolution, allowing resolution complete planning queries [Pivtoraiko and Kelly, 2005]. Like a grid, it converts the problem of planning into a continuous space in the decision making in a discretized space, but unlike a grid, the connections represent feasible paths, where mobility constraints have to be considered. There, each node has a four-dimensional space, comprising position, heading and curvature. For constructing the state lattice, they use an inverse path generation method to find paths between any node in the grid and the arbitrarily chosen origin. The algorithm was tested on the Reeds-Shepp car.

Later, Pitvoraiko et al. presented a differentially constrained mobile robot motion planning in arbitrary cost fields [Pivtoraiko et al., 2009]. There, the state lattice used to discretize the search space permits fast full configuration space cost evaluation and collision detection to avoid the arbitrary obstacles. Since the state lattice is a directed graph, any systematic graph search algorithm is appropriate for finding a path on it. They use them together with A* and D* search algorithms because they return optimal paths for the desired cost function, even while the topology of the space is dynamically changing. They validated the approach on simulation test and prototype rovers, showing that state lattices are resolution complete, optimal and smooth, satisfying the differential constraints and the efficiency. Although they are not as fast as grid planners in the presence of obstacles, they prevent problems related to the non-feasibility of the paths produced with grid planners.

Ziegler and Stiller presented a motion planner algorithm that combines spatio-temporal state lattices with quintic splines for generating continuous and smooth paths on dynamic on-road driving scenarios, considering the motion of the obstacles [Ziegler and Stiller, 2009]. In conventional state lattices, the configuration space is a graph where the nodes represent the samples, and the geometric connection between nodes is done with the local planning algorithm. They have been used either to generate high-speed trajectories in real-time [Ziegler and Stiller, 2009] or to calculate highly optimized paths between destinations [Andreasson et al., 2015].

Unlike the conventional lattices, the spatio-temporal state lattices allow the prediction of the obstacles motion by combining both configuration space and time. Afterward, a sampling phase is applied on the discrete subset at equidistant positions, using quintic polynomials to represent edges, which leads to second order continuous paths and boundary conditions for position, velocity, acceleration and time. Besides, using spatio-temporal state lattices does not require using shortest path algorithms such as Dijkstra or A* to maintain the visited vertices in a partially ordered data structure.

In McNaughton et al. a motion planner using spatio-temporal state lattices for highway driving is presented [McNaughton et al., 2011]. Unlike the work of Ziegler and Stiller in [Ziegler and Stiller, 2009], they use a more efficient method of augmenting the state lattice with time and velocity dimensions, using realistic vehicle kinematics to construct the actions, reducing the modeling error inherent to hierarchically decomposed planners.

Likhachev and Ferguson presented an algorithm using anytime incremental search on a multi-resolution lattice state space for generating dynamically-feasible maneuvers with automated vehicles at high speeds, testing the approach on the Tartan Racing team's vehicle [Likhachev and Ferguson, 2008]. The multi-resolution lattice allows planning long complex maneuvers over lattices, which can be expensive concerning both computation and memory using normal lattices. It consists of using a high-resolution action space close to the vicinity of the vehicle and a low-resolution action space elsewhere.

A global planning approach using multi-resolution state lattice search space has been used in [Likhachev and Ferguson, 2008] to reduce the complexity of the global search, which is a drawback of the purely global-planning based approaches. Indeed, an efficient anytime incremental search is used to quickly generate bounded suboptimal solutions, solving one of the issues of the purely based local planning approaches. Since planning complex maneuvers over lattices can be expensive regarding computation and memory, it is not convenient to explore the whole spectrum of paths between vehicle and goal configurations. On the contrary, a combination of a high-resolution action space in the vicinity of the vehicle and a low-resolution action space elsewhere can be applied.

They implemented the approach on an automated passenger vehicle, generating dynamically-feasible maneuvers at high speeds on the Tartan Racing team's vehicle, which drove over 3000 kilometers in urban environments, including the DARPA Urban Challenge. The multi-resolution state lattice planner was used for planning through parking lots and into parking spots, as well as for geometric road following in off-road areas, and in error recovery scenarios. There, the vehicle ran at speeds around 25 km/h, performing complex maneuvers avoiding static and dynamic obstacles. The multi-resolution lattice planner searches backward out from the global pose and generates a path consisting of a sequence of feasible high-fidelity maneuvers that are collision-free considering the static obstacles observed in the environment. The lattice used in this application does not explicitly represent the curvature. They showed the performance of the system on a parking lot when planning between two parking spots, using the AD* algorithm to compute the cost of the path. The algorithm is three times faster than without multi-resolution lattices. The algorithm was validated on multiple scenarios, such as obstacle-laden parking lots, and narrow and highly-constrained parking lots.

Table 2.1 – Comparison of graph search-based techniques for path planning

Technique	**Advantages**	**Disadvantages**
Dijkstra's algorithm	Finds the shortest path on weighted direct graphs. Suitable for global planning in structured and unstructured environments Adaptable to dynamic environments	Impractical for large graphs representing the road network. The generated paths are neither optimal nor continuous. The search is not heuristic.
A* family	Extension of Dijkstra's algorithm. Heuristic search, returning an optimal path. Kinematic and dynamic constraints considered through cost functions for dynamic environments. Computational time and memory use are lower.	Impractical for large graphs representing the road network. The generated path is not continuous. Shortest and lower cost path lie on the heuristic applied
State lattices	Suitable for dynamic on-road scenarios. Allow space-time planning, including time and velocity dimensions. Lane adapted workspace parameterization. Suitable for global planning. Incremental search algorithm: suitable for unknown or partially known environments. Unlike the graph-based algorithms, they consider the feasibility of the path	Problems with curvature continuity. Motion could be restricted. Slower than grid planners in the presence of obstacles. Resolution completeness translated into computation cost. The planning precision depends directly on the lattice resolution.

2.3.2 Sampling-based algorithms

Motion planning methods based on sampling the configuration space can be classified in two sub-types according to the way of sampling: random sampling-based methods and deterministic sampling-based methods [Ziegler and Stiller, 2009].

State lattices can be considered as deterministic sampling-based algorithms, where each sample becomes a vertex in a graph, and edges connect each vertex to a finite number of neighbors with an associated cost, forming a geometric path that is used by a graph searching method to search the shortest path.

Since deterministic sampling-based methods have already been presented in Section 2.3.1.c, this subsection focuses on the probabilistic sampling-based methods such as Probabilistic RoadMaps

(PRM), Artificial Potential Fields (APFs) and the random sampling-based algorithms, specifically in the RRT family algorithms.

2.3.2.a Probabilistic RoadMaps (PRM)

Probabilistic roadmaps (PRM) methods emerged in the late 1990s as effective methods to solve complex motion planning problems [Saha, 2006]. This algorithm finds a path from a starting configuration to a goal configuration while avoiding obstacles in static spaces.

It consists of two phases: a construction and a query phase [Kavraki et al., 1996]. First, a learning phase where a probabilistic roadmap is built and stored as a graph whose nodes correspond to collision-free configurations and whose edges correspond to feasible paths between these configurations. Second, a query phase where the start and goal configurations are connected to the roadmap (graph), and then a path is sought that joins these two nodes by applying a graph search algorithm such the Dijkstra's shortest path.

PRM methods are capable of computing the motion both for single-robot as for multiple-robot systems operating in complex geometric scenarios and considering motion constraints such as collision avoidance, stability, visibility, and contact constraints [Saha, 2006].

PRM planning approaches work by constructing a simplified sample-based approximate representation of the free space, called a probabilistic roadmap. This roadmap is a graph whose nodes are configurations sampled from the free space according to some suitable probability measure, reflecting the uncertainty on the actual shape of the free space. Meanwhile, an edge or local path connects a pair of nodes in the roadmap. Once the roadmap is constructed, a path connecting origin and destination nodes is extracted using standard graph-searching techniques. The presence of narrow passages in the free space makes this approach to be computationally expensive.

The planning steps of the PRM method are depicted in Figure 2.19. There, the planning is done in the vehicle's free space. The nodes represent the different configurations, and the edges represent the collision-free links to the k-nearest neighbors. Finally, the start (s) and goal (g) configurations are added to the PRM, which is searched for a path from s to g, represented by the red polyline.

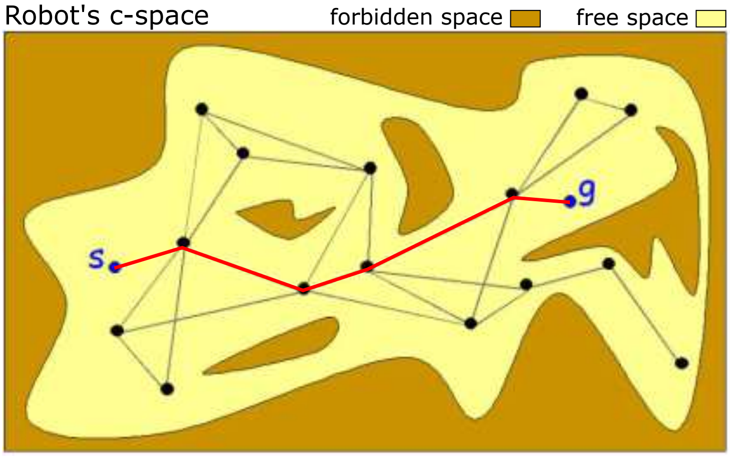


Figure 2.19 – PRM planning. Figure adapted from [Saha, 2006]

The described method has also been applied to non-holonomic car-like robots. Although the PRM algorithm works well on static environments, however, for planning applications involving dynamic and rapidly changing environments is not convenient because building a roadmap a priori may not be feasible [Fiore, 2008]. On the contrary, the Rapid Exploring Random Tree (RRT) sampling-based motion-planning approach targets this weakness, as will be shown in Subsection 2.3.2.c. Thus, techniques as the RRT are more convenient for being used with non-holonomic systems such as the automated vehicles.

For addressing the limitations of the sampling-based path planning algorithms available in the literature, Karaman and Frazzoli proposed a new algorithm called Optimal Probabilistic RoadMaps (PRM*) [Karaman and Frazzoli, 2011]. It is a batch variable-radius PRM, in which the radius is scaled with the number of samples ensuring both asymptotic optimality and computational efficiency. This algorithm has shown a performance with a configuration space of dimensions up to five.

2.3.2.b Artificial potential fields

Potential fields represent a heuristic approach to motion planning where the configuration space is discretized into a fine regular grid of configurations to search a free path through it. There, the robot represented as a point in the configuration space can be seen as a particle moving under the influence of an attractive artificial potential produced by the goal configuration and a repulsive one produced by each obstacle [Latombe, 1991]. Although these methods can be very efficient, their main drawback is linked to the fact that they are descent optimization methods, and therefore they can get trapped into local minima of the potential function other than the goal configuration.

Unlike roadmap and cell decomposition methods, which are considered global methods, potential field methods can be used as global [Warren, 1989] or as local methods [Wolf and Burdick, 2008], since at each step they compute the potential gradient to make the transition between configuration states.

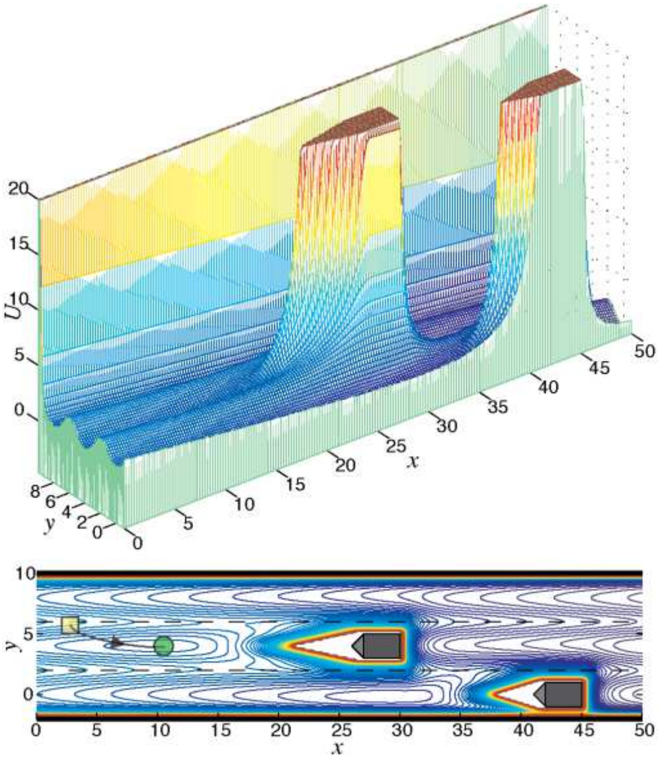


Figure 2.20 – Artificial potential fields applied into a stay-in-lane scenario [Wolf and Burdick, 2008]

Although this method has been mostly applied in robotics, it has also been applied in the vehicles automation field on simulation to assist an automated vehicle in a multi-lane populated highway, concretely for lane-keeping, road-staying, speed preference, vehicle avoidance and passing scenarios. There, potentials for both lane and road are used to keep the vehicle on the road, preferably in the center of the lane. Figure 2.20 shows an application of APF into a scenario where the vehicle must stay in the lane. It can be used not only for navigation, but it can also be applied for decision making purposes, as well as an input in driver assistance devices.

APF have also been applied to vehicle control, as in [Gerdes and Rossetter, 1999], where various assistance systems can be integrated through superposition of individual potential and dampling functions.

2.3.2.c Rapid Exploring Random Tree (RRT) and Enhanced RRT (RRT*)

Rapid Exploring Random Tree (RRT) is a random sampling based method which builds an incremental tree for searching in large spaces. LaValle et al. introduced the RRTs as randomized data structures designed for a broad class of path planning problems [Lavalle, 1998].

Unlike the previous randomized techniques (potential fields and probabilistic roadmaps), this algorithm was specifically designed to handle nonholonomic constraints, including dynamics, and high degrees of freedom (being tested up to 12 degrees of freedom). Besides, it has also been successfully applied to holonomic and kinodynamic planning problems [LaValle and Kuffner, 1999].

Since the probabilistic roadmap technique might require thousands of connections among configuration states fo find a solution, it results impractical for nonholonomic and kinodynamic problems that arise in robotic and automatic fields. In contrast, thanks to RRTs do not require any connections to be made between pairs of configurations (or states), this method is suitable for nonholonomic constraints.

RRT, like most of the current motion planning algorithms in the state of the art, prioritize providing a quick solution instead of an optimal one, due to the real-time constraint dominating the navigation applications. On the other hand, old motion planning algorithms tried had difficulties to finish the execution, using all the computation time to find a better solution, for different criteria such as time, path length, and fuel consumption. The shortcoming of these algorithms is that they cannot ensure to converge to optimal trajectories.

Karaman and Frazzoli presented an enhanced version of the RRT algorithm, called RRT* [Karaman and Frazzoli, 2010], which targets the weakness of RRT, consisting of the non-existence of theoretical bounds on the quality of the solution obtained by these algorithms, i.e., the optimality of the solution cannot be ensured. This algorithm is a tree version of the Rapidly-exploring Random Graph (RRG), which preserves the asymptotic optimality of RRG while maintaining a tree structure like RRT, which allows to deal with differential constraints or cope with modeling errors presented in the motion planning problems. It essentially rewires the tree as it discovers new lower-cost paths reaching the nodes that are already in the tree. Nevertheless, the asymptotic computational complexity for the RRT* algorithm remains the same as that of the RRTs and the RRG. These strengths of RRT* compared to RRT can be shown in [Karaman and Frazzoli, 2010], presenting an environment cluttered with static obstacles on simulation, where the cost of the RRT* solution is lower when augmenting the number of iterations.

An anytime algorithm based on the RRT* was subsequently presented. This anytime algorithm finds an initial feasible solution quickly, but unlike RRT, it almost surely converges to an optimal solution [Karaman et al., 2011]. Additionally, two extensions of the RRT* were introduced: committed trajectories and branch-and-bound tree adaptation, allowing the algorithm to make more efficient use of computation time online, resulting in an anytime algorithm for real-time implementation. They evaluated the method using a series of Monte Carlo runs, comparing it to the RRT method in simulation and on an outdoor robot. Figure 2.21 depicts the resulting paths from the simulation tests with both RRT (on the left) and RRT* (on the right) algorithms under an environment consisting of a bounded region with two polygonal obstacles. The vehicle dynamics correspond to those of a rear wheel-steered nonholonomic ground vehicle. There, it is shown that the RRT frequently produces trajectories that are unnecessarily long, whereas RRT* provides shorter paths to the goal thanks to the correct identification of the route between the two obstacles.

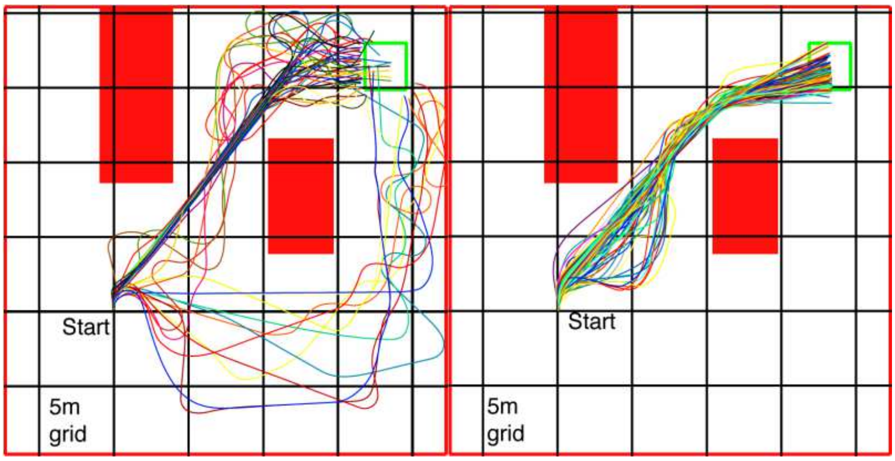


Figure 2.21 – RRT and RRT* algorithms performance [Karaman et al., 2011]

The committed trajectory extension consists of an initial planning phase and an iterative planning phase. In the 1^{st} one, the RRT* runs until the robot must start moving towards its goal. In the 2^{nd} one, the robot starts to execute the motion plan generated by the RRT* algorithm until a given commit time, deleting each of its branches and declaring the end of the committed trajectory (initial path) as the new root. Meanwhile, the RRT* continues to improve the remaining part of the trajectory within the new and communicated tree of trajectories. Once the robot reaches the end of the committed trajectory, the iterative phase is re-executed until the robot reaches the goal region, using the initial portion of what is currently the best path in the RRT* tree to define a new committed trajectory. In addition to considering a committed trajectory, they employ a branch-and-bound strategy to build the tree more efficiently. It removes periodically from the tree the set of vertices whose cost to get to them plus the lower-bound *cost to go* from the node to the goal region is higher than the upper-bound cost trajectory that reaches the goal node.

An extension of the RRT algorithm named closed-loop RRT (CL-RRT) was implemented on the MIT's vehicle finishing in fourth place the DARPA Urban Challenge [Kuwata et al., 2009], where the vehicle completed a 60 mile (around 96.5 km) simulated military supply mission, while safely interacting with other automated and human-driven vehicles. This algorithm presented there extends the RRT by making use of a low-level controller and planning over the closed-loop dynamics. It enables the online use of RRTs on robotic vehicles with complex, unstable dynamics and significant drift, while preserving safety. As stated in [Kuwata et al., 2008], the use of RRTs had never been used in online planning systems for robotic vehicles. They implemented the algorithm on the MIT's Landrover LR3 used for the DARPA Urban Challenge, where the algorithm was validated on different scenarios such as on an obstacles field, a parking-lot area, and a U-turn.

A threat-aware approach using the CL-RRT method for path planning was presented in [Aoude et al., 2010]. There, they put the focus on negotiating the traffic intersections, where several vehicles were involved in collisions or near-collisions, wherein the motion planner evaluates the risk of potential trajectories considering the obstacle uncertainty. This threat assessment module was validated through simulation and experiments performed in the MIT Real-time indoor Autonomous Vehicle test ENvironemnt (RAVEN), a testbed equipped with motion-capture cameras to provide high-fidelity vehicle state data, to identify and avoid an errant human-driver vehicle at intersections.

Another RRT based implementation was implemented by the Korea University of Technology and Education for high-speed on off-road scenarios and with capabilities to avoid obstacles of different patterns [Ryu et al., 2013]. It was implemented on the Pharos automated vehicle and tested during the Hyundai-Kia motors Autonomous Vehicle Competition. The RRT algorithm was modified to increase the computation efficiency of the planner to provide a faster response to the environment changes, allowing to run at higher speeds.

Robust RRT was presented in [Fiore, 2008] as another enhancement of the RRT algorithm, designed for large robotic vehicles and uncertain, dynamic environments. One fundamental difference with the original RRT algorithm is that the Robust RRT samples the space of inputs to the vehicle controller, as opposed to sampling the space of inputs to the vehicle directly. This algorithm was employed by the MIT's team on the DARPA Urban Challenge, being validated on the six different scenarios conforming the event, i.e., a full parking lot, blocked road, rectangular track with obstacles, dense obstacle field, narrow passage and dead ends.

The RRT* algorithm was implemented in [hwan Jeon et al., 2013] for the half-car dynamical model to enable automated high-speed driving. They provide a fast local steering algorithm for the half-car dynamic model separating geometric path planning step from optimal time parameterization step, providing a significant advantage, allowing to make the collision checking after the geometric path planning step.

RRT has also been used for planning the motion of multiple vehicles, like in [Kala and Warwick, 2011]. There, different traffic scenarios were simulated (such as curved roads and straight roads leading to avoidance maneuvers), and the generated trajectories are smoothed with spline curves, finding the maximum feasible speed associated to the paths. Unlike [Kuwata et al., 2008], they separated the planning and control modules, where RRT is only in charge of the planning task, making the planner more adaptive to any vehicles whose dynamics are initially unknown.

An extension of RRT has also been tested in car-like robots, more precisely in the INRIA cybercars, under simulation experiments [Fulgenzi et al., 2009]. They present a path planning algorithm where the probability of both obstacles future trajectory and collision are considered for the motion of the cycab on a simulated environment among multiple dynamic obstacles.

An anytime algorithm for planning paths through high-dimensional, non-uniform cost search spaces was presented in [Ferguson and Stentz, 2006a]. It generates a series of RRTs where each tree reuses information from previous trees to improve its growth and the quality of the resulting path. The approach was tested on both single-robot and multi-robot planning domains.

A heuristically-guided implementation of the RRT (hRRT) was presented in [Urmson and Simmons, 2003]. There, the search is guided by a heuristic quality function in order to improve the path produced through the RRT like search. It was tested on simulation, showing a lower computational cost than the classical RRT in its performance with obstacles.

An RRT based global planner called RTR (rotate-translate-rotate) has been used in [Nagy et al., 2015] to generate a path consisting of straight lines and circular arcs primitives, which is then transmitted to the C*CS steering method, which improves the path by generating lower bounded raddi arcs that are feasible to be tracked not only by a differential robot but also by a non-holonomic car-like robot.

The uncertainty of both the environment and the initial and final configuration spaces are tackled in [Panchea et al., 2017], where the tBoxRRT* algorithm based on RRT* is presented as a robust planner to guarantee safe paths while avoiding static obstacles.

Table 2.2 – Comparison of sampling-based techniques for path planning

Technique	**Advantages**	**Disadvantages**
Probabilistic RoadMaps	Roadmap stored as a graph, where a search algorithm is applied. Efficient on static environments. Works for avoiding static obstacles.	Not convenient for dynamic and changing spaces. Impractical for nonholonomic and kinodynamic problems. Computationally expensive in road-like areas. Depends on graph-search based algorithms.
Artificial Potential Fields	Relatively fast and effective for generating safe paths around obstacles in dynamic environments. Suitable for both global and local planning. Suitable for nonholonomic vehicles	The workspace must be known in advance. Trapped into local minima since they are descent optimization methods.
RRT family	Handle nonholonomic and dynamic constraints. Deals with differential constraints and copes with modeling errors. Real-time operation, providing a quick solution. Probabilistically complete. Works on dynamic and changing scenarios. Works on high-speed driving conditions. Even on off-road scenarios.	Non-existence of theoretical bounds on the quality (RRT), i.e., no guarantee of finding an optimal solution. The path generated is not continuous. Asymptotic computational complexity

2.3.3 Interpolating curves algorithms

These methods generate a smooth path from a set of way-points that describe the route. Unlike the previous methods that are mostly used for global planning, these are well-known as local planning approaches. The most significant methods are described below.

Furthermore, these methods allow an easy adaptation to the dynamic conditions of roads, where only a path to avoid the obstacle and another one to come back to the original path needs to be computed.

2.3.3.a Straight lines and circular arcs

A combination of straight line segments, and circular arcs was also used to model both path and lane to generate smooth trajectories for a car-like on-road vehicle [Horst and Barbera, 2006].

Circular arc-based approaches are used for path planning on parking maneuvers, where a combination of either straight-lines and circular arcs or several circular arcs is used, depending on the parking configuration. For example, a method using a simple arc circle together with a rectilinear movement was used for parking on perpendicular spots on the OSU-ACT vehicle during the DARPA Urban Challenge [Hsieh and Ozguner, 2008]. In addition, it was applied to parallel parking for cybercars in [Marouf et al., 2014] and [Petrov and Nashashibi, 2014b].

Another application of this combination method was used in [Kanayama and Hartman, 1989] to describe a path as a sequence of postures (positions with their correspondent orientation) that is later smoothed by applying a cost function that tries to minimize the maximum curvature by using cubic splines.

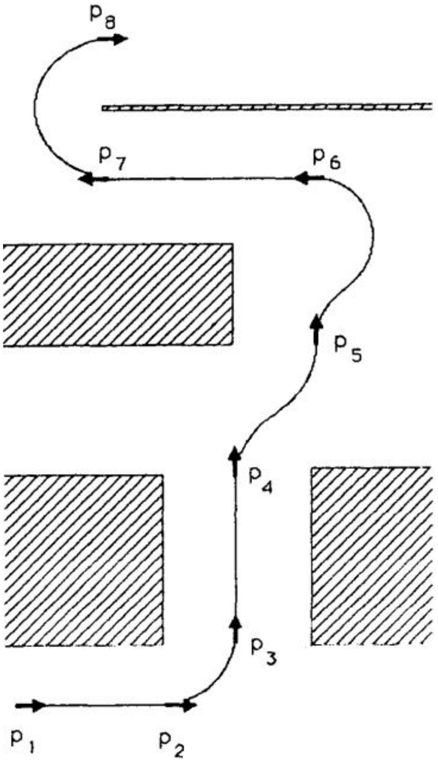


Figure 2.22 – Combination of straight lines and circular arcs for path planning [Kanayama and Hartman, 1989]

Dubins showed that the shortest possible path that meets a maximum curvature bound between a starting and an ending position with predefined orientations consists of at most three pieces, being either straight lines or arc circles [Dubins, 1957]. This algorithm was extended to car-like vehicles by considering the backward motion [Reeds and Shepp, 1990]. However, the main disadvantage of these methods is the curvature discontinuity which occurs at the joint point of two consecutive path segments [Tsourdos et al., 2010]. This problem can be solved by applying a clothoid arc between a straight line and a circular arc [Fraichard and Scheuer, 2004].

2.3.3.b Clothoids

These are curves whose curvature varies linearly with the length of the curve. Their main advantage is that they are used in the design of road highways, where some of them are constructed joining clothoids with circular arcs and straight-line segments [Walton and Meek, 2005]. They are used as the transition from straight segments to circular arcs in roads and highways due to the constant change of acceleration that the vehicle would have following the curve at constant speed [Meidenbauer, 2007].

Additionally, they can also be used for modeling the urban road lanes [Gackstatter et al., 2010]. However, they are not applied in urban areas due to the dynamism of the environment with various actors that the vehicle may find when driving, which would increase the computation cost to process them in real-time. In addition, since they are defined in terms of Fresnel integrals which cannot be solved analytically, their use in real-time applications is difficult in comparison with non-linear curvature methods that are easier to compute [Brezak and Petrović, 2014].

Although some authors do not consider clothoids as an interpolation method [Labakhua et al., 2008], they have been placed in this category because they can be defined by parametric equations, depending on the initial curvature, the curvature derivative, the orientation angle and the radius of the clothoid.

Several approximation of clothoids have been made, using power series [Press et al., 1992], numerical integration algorithms, other analytical curves such as Bézier or B-splines [Wang et al., 2001], s-power series [Sánchez-Reyes and Chacón, 2003], rational function approximations [Heald, 1985], continuous function approximation [Mielenz, 2000] or arc splines [Meek and Walton, 2004]. However, none of these methods guarantees bounded error of clothoid approximation over a broad range of clothoid parameters.

Stanford University used clothoids for pre-compute the path to be followed at high-speed by an Audi TTS using a highly accurate differential GPS [Funke et al., 2012]. There, the turns have been modeled with a combination of a straight stretch, an entry clothoid, a circular arc segment and an exit clothoid. The system was validated on the Pikes Peak Hill Climb course, which combines paved and unpaved road surfaces.

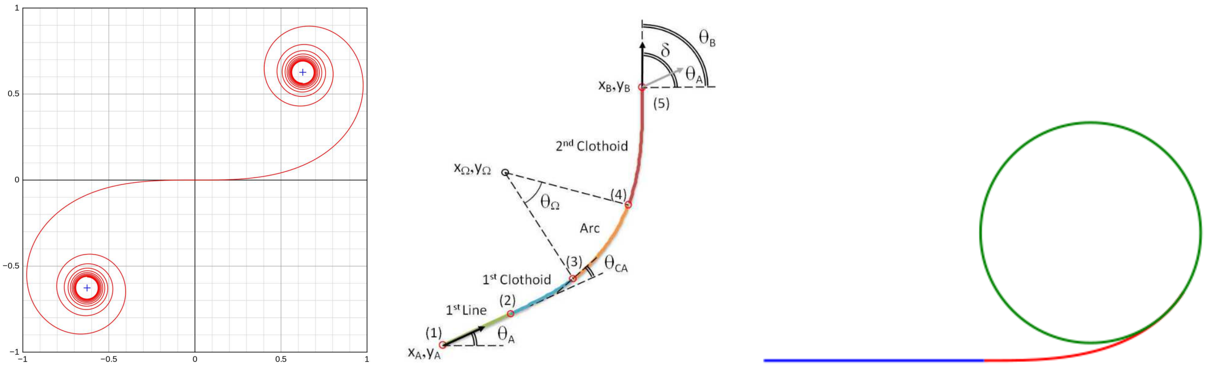


Figure 2.23 – Clothoids: double end spiral (left), line-arc-clotoid interpolation (middle) [Girbés et al., 2011], line-clothoid-circle interpolation (right)

Clothoids were used in [Bertolazzi and Frego, 2012], [Bertolazzi and Frego, 2015] to solve the problem of Hermite G^1 interpolation, which allows a curve to interpolate two given points in a plane with assigned tangent directions. However, in real-time applications, this interpolation is cost-effective, in particular when the discontinuity of the curvature is acceptable.

Piecewise clothoids have been extensively used in road design and robot path planning. A mechanical model called super-clothoids allowing to compute the dynamics of an elastic, inextensible piecewise clothoid is presented in [Bertails-Descoubes, 2012].

Although clothoids are mathematically and algorithmically more complex than polynomial splines, Baran et al. state that the quality of the generated path weights the cost [Baran et al., 2010].

A combination of straight lines and clothoids for generating G^2 continuous paths is proposed in [Lekkas, 2014]. There, they state that one of the main drawbacks of clothoids is that their coordinates do not have a closed-form expression since they involve computation of the Fresnel integrals.

A combination of clothoids, line segments and circular arcs was also used in [Girbés et al., 2011] to generate continuous curvature paths for the kinematic control of a wheeled mobile robot. They present two alternatives: the Single Continuous Curvature path (SCC) and the Double Continuous Curvature path (DCC). First ones (SCC paths) are composed of two clothoids and a circular arc in-between (as shown in the mid-part of Figure 2.23). Second ones (DCC paths) consist of two SCC paths plus an additional final straight line segment.

Broggi et al. used clothoids during the Vislab Intercontinental Autonomous Challenge (VIAC) combining them with circular arcs, according to the curvature conditions of the terrain for approximating the vehicle's motion [Broggi et al., 2012]. Thus, circular arcs were used for continuous curvature, whereas clothoids were used in any other case, where a non-negligible curvature is introduced.

A local planning approach based on the combination of clothoid tentacles with the generation of an ego-vehicle centered occupancy grid was presented in [Alia et al., 2015]. The algorithm discretizes the environment through the occupancy grid and classifies the tentacles as navigable or not navigable. The algorithm was tested off-line on simulation with the data extracted from the Lidar embedded in the non-robotized vehicle, on a real scenario where obstacles were present, comparing the path obtained by clothoid tentacles with another solution consisting on circular tentacles.

A clothoid-based solution has also been applied to a Linear Time-Varying Model Predictive Control (LTV-MPC) approach to address the problem of path following by a non-holonomic vehicle traveling at low-speeds. There, a set of clothoids is sent to the LTV-MPC to describe a reference path [Lima et al., 2015].

2.3.3.c Polynomial curves

These are curves defined by polynomial equations. They are used to fit a series of data points, possibly subject to constraints, where either interpolation or smoothing is involved. As advantages, they have a low computational cost, and they make possible the concatenation of several curves in a continuous fashion. However, with curves of less than fourth degree is hard to find the coefficients to build continuous curves. Therefore there is an increase in the calculation cost.

A trajectory planning algorithm divided into path generation, search of the optimal path and post-optimization, where the path is formed by connecting cubic and quartic curvature polynomials has been proposed in [Xu et al., 2012]. After the path generation, they apply a path and speed optimization step using lattices for discretize both sets.

Quintic polynomial splines are used by CMU in [Piazzi et al., 2002] proving that the continuity of both curvature's rate of change and its derivative is guaranteed with this order, leading to smoother paths allowing higher speed tracking. They also applied cubic polynomial curvature spirals combined with lattices sampling [Gu et al., 2013].

Fourth and fifth-degree polynomials were used in [Glaser et al., 2010] for lane changes. There, a set of fixed polynomial paths is generated with a corresponding target speed and a time lapse. The considered input constraints are the vehicle position, speed, and acceleration. Besides, the calculation of the coefficients for both the fourth order longitudinal and the fifth order lateral polynomials are described, according to the continuity constraints. Fifth order polynomials were used previously in [Simon and Becker, 1999] to describe both trajectory and road boundaries, considering the position, orientation, and curvature as system constraints. In addition, quintic polynomial curves were used in [Resende and Nashashibi, 2010] to search the optimal path building a spatio-temporal set of trajectories to be performed in a dynamic highway environment.

Interpolation of cubic polynomial curves has been applied in [Petrov and Nashashibi, 2014a] for performing lane change maneuvers in overtaking scenarios, since the trajectory generated by the polynomial curves constitutes a reference path for the adaptive controller, which solves the overtaking as a tracking problem.

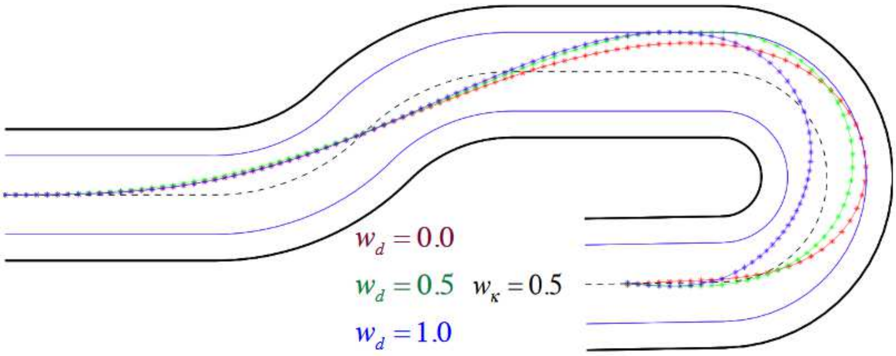


Figure 2.24 – Polynomial curves with different weightings to tune the path [Gu et al., 2013]

2.3.3.d Splines

Splines are piecewise differentiable curves defined by parametric equations. These can be either polynomial curves ([Bacha et al., 2008], [Ghilardelli et al., 2014]), b-splines (quartic b-splines [Berglund et al., 2010], cubic b-splines [Trepagnier et al., 2007]), i.e., a generalization of Bezier curves, Bézier curves ([Romani and Sabin, 2004]) or even clothoids. Their knots or polynomial joints pose a high degree of smoothness. For instance, quartic B-splines used in [Berglund et al., 2010] ensure a continuous derivative of the curvature, which benefits smoothness. In addition, cubic b-splines were implemented on a motion planning algorithm running on the Team Gray's KAT-5 vehicle participating on the DARPA Grand Challenge [Trepagnier et al., 2007]. They were used to follow the center of the route while ensuring the path is feasible for a non-holonomic vehicle, respecting the kinematic constraints (maximum feasible curvature) and allowing to avoid obstacles by adjusting the control points in real-time. Later on, cubic splines were applied in the DARPA Urban Challenge by the VictorTango's team to generate the path that fits the more with the physical lanes of the road, given the geo-referenced aerial imagery provided by the competition.

Splines have also been used to approximate clothoid spirals, as was proposed in [Meek and Walton, 2004], where they claim that spline arcs are very easy to lay out to fit with the clothoid arcs defining the highways. Furthermore, pairs of clothoids were used as blending curves to construct clothoid splines as control polylines, which facilitates the design and modification of the clothoids to adjust to the road definition [Walton and Meek, 2005].

Furthermore, splines have been largely used together with some spatio-temporal search space discretization algorithm as state lattices. Quintic splines were used by the motion planning algorithm of KIT Institute in [Ziegler and Stiller, 2009] to generate continuous and smooth paths on dynamic on-road driving scenarios.

A parameterized curve primitive, the η^4-spline, was proposed by the University of Parma for the generation of high-quality drive paths for a truck and trailer automated vehicle [Ghilardelli et al., 2014]. This primitive consists of a ninth order polynomial curve. This spline can be systematically used to generate smooth paths for the automation of articulated vehicles. Indeed, η^4 splines can be used to solve problems of dynamic articulated vehicles. The η-spline primitive was presented in [Piazzi et al., 2002].

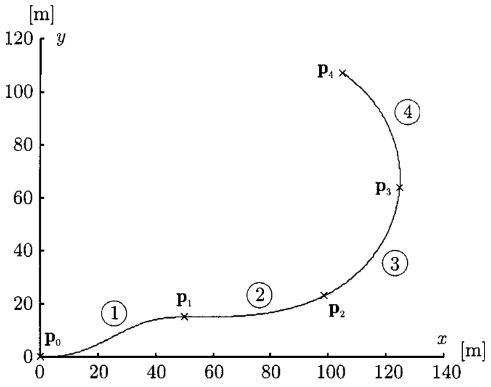


Figure 2.25 – η-Splines interpolating given points [Piazzi et al., 2002]

2.3.3.e Nurbs

The term *nurbs* is an acronym of non-uniform rational B-spline. It is a mathematical model widely used in computer graphics for generating and representing curves and surfaces, thanks to presenting the following characteristics [Piegl and Tiller, 1996].

- A unified mathematical basis for representing both analytic as well as free-form shapes.
- Intuitive design.
- Fast and numerically stable algorithms.
- Invariant curves and surfaces under common geometric transformations.
- They are generalizations of non-rational B-splines and rational Bézier curves and surfaces.

Figure 2.26 shows a cubic Nurbs curve.

2.3.3.f Bézier curves

Bézier curves are parametric curves based on the Bernstein polynomials and ruled by control points [Prautzsch et al., 2002]. Although they were created for designing automobile bodies, they have been widely used in computer graphics and animation [Choi et al., 2008]. Additionally, they present some properties that make them appropriate for path planning purposes [Han et al., 2010]: (i) Bézier curves always pass through the first and last control points defining them. (ii) The vectors with the two control points defining the beginning and the end of the curve respectively are tangent to the curve. (iii) The curves will always be framed within the convex hull defined by the outermost control points. (iv) The behavior of the curve concavity is consistent with the concavity formed by the control points (as shown in Figure 2.27).

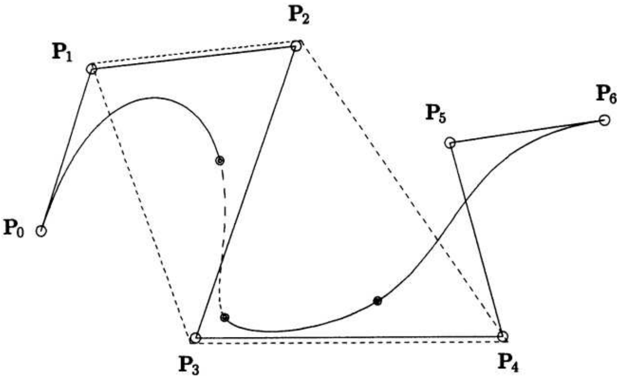


Figure 2.26 – Cubic NURBS curve [Piegl and Tiller, 1996]

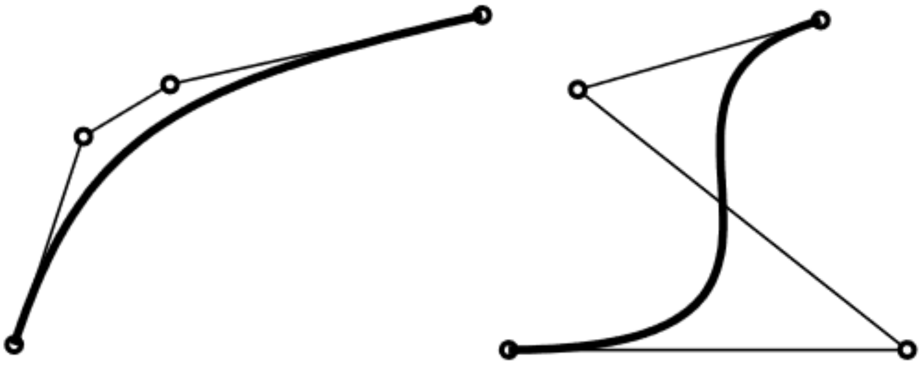


Figure 2.27 – Bézier curves [Sederberg, 2007]

Thanks to their properties, they are used to satisfy the path constraints generating a reference trajectory feasible for automated vehicles, as in [Choi et al., 2008], where the path planning algorithm considered the corridor constraints for the operation of an automated ground vehicle. In addition, since these curves are defined by control points, they can be easily modified in real-time to adapt the path to avoid possible obstacles in the itinerary, while maintaining a smooth path tracking [Han et al., 2010]. Thus, it shows the good behavior of this method with real-time constraints, generating a fast and low-computational cost. Since these curves have been widely used for surface design, the continuous transition between them has been studied. For instance, in [Walton et al., 2003] a planar G^2 transition between Bézier segments that ensure geometric continuity in the joint point where segments share a common curvature center.

It has also been applied to unmanned aerial vehicles [Yang and Sukkarieh, 2008], where cubic Bézier spirals where used to generate a continuous curvature path, smoothing a previously generated path with RRT, respecting the non-holonomic constraints and considering the angle between the way-points. A seventh order Bezier curve based local planning algorithm is used in [Neto et al., 2010] to connect the vertexes of the tree generated by an RRT global planning approach for an aircraft, i.e., a holonomic robot, on a well-known non-structured space with static obstacles.

Thanks to the low-computation load of this method, it can be used for the generation of lane change trajectories on-road environments, as in [Chen et al., 2013], where quartic curves are used for generating a continuous curvature path respecting the non-holonomic constraints of the vehicle. Quartic Bézier curves were also used in [Chen et al., 2014] for generating continuous and bounded curvature profiles shaping the trajectory, and at the same time, generating a linear velocity profile to execute the trajectory.

Table 2.3 – Comparison of interpolating curves based techniques for path planning

Technique	**Advantages**	**Disadvantages**
Lines and arcs	Low computational cost. Generates shortest possible path that meets a maximum curvature with at most three pieces. Suitable for parking maneuvers.	Path generated is not continuous. Not suitable for non-holonomic constraints. It requires a local planning post-process to smooth the path.
Clothoids	Curvature varies linearly with arc length. Smooth transitions between straight lines to circle arcs. Used for highways road design. High-quality of the generated paths. Clothoid tentacles can be combined with road discretization methods. Suitable for nonholonomic systems. Applied to high-speed path tracking when the environment is static and well-known.	High computational since they are defined by Fresnel integrals and cannot be solved analytically. Not suitable for real-time. Interpolation of clothoid curves in real-time is computationally expensive. Approximated methods to implement them not guaranteeing bounded errors.
Polynomial curves	Trajectories meet kinematic constraints. Feasible to be used in real-time. Concatenation of curves in a continuous fashion is feasible. Suitable for dynamic environments. Modeling of both road boundaries and path.	Optimization process might be required to generate a smooth trajectory with low degree curves. Forth or fifth-degree curves are needed to ensure path continuity, augmenting the computational cost.
Splines	Real-time implementation. Consecutive curves can be interpolated in a continuous way thanks to knots-based definition. Allow respecting nonholonomic constraints.	Do not guarantee finding an optimal solution. Meeting smoothness and road constraints require using higher degree polynomials, or an optimization phase.
Nurbs	Generalization of non-rational b-splines and rational Bézier curves. Intuitive design. Low computational load.	The smoothness of the path depends strongly on the degree of the curves used.
Bézier curves	Low-computation cost. Real-time concatenation of multiple curves. Curves are malleable (i.e. ease to control the shape to meet the kinematic and road constraints). Real-time adaptability to the dynamism of the scene.	Curvature continuity requires either at least quartic degree curves or some optimality evaluation. Curve points do not pass through all the control points, making necessary an algorithm to define a polygon where they should be placed respecting the lane borders.

2.4 Discussion

The increasing adoption of ADAS in commercial vehicles leads to a positive impact in the road transport, both from the safety and the environmental impact. Systems such as the lane keeping assist, the lane change assist or the parking assist are some examples of already commercialized ADAS. Automated vehicles have emerged in last decade as a promising line of research to contribute on those aspects as well, aiming to integrate the developed technology on the on-road vehicles as soon as possible. Both private companies and public research institutions envisage deploying vehicles able to achieve the fourth automation level in the short term. It means to develop vehicles able to travel autonomously under specific scenarios. The development of this technology is making possible this transition between a fully manual and a fully automated experience, where the control of the car would be shared between human driver and the system.

One of the most challenging scenarios in driving is the urban environment, where the vehicle has to consider all the dynamic changes on the scene, interacting with the other vehicles and vulnerable road users such as bikes and pedestrians, following traffic rules. Additionally, the ability to drive in a natural way (i.e., as a human driver does) plays an important role for users acceptability. Automated vehicles are expected to deliver smooth driving, increasing ride comfort to the passengers.

Trajectory planning plays a key role in achieving such goals as part of the decision-making stage. These systems generate the route to arrive from the departure position to the destination safely, which implies the modification of the route to deal with unexpected situations such as obstacles in the path, either to avoid them or to perform an emergency braking if needed. Although the path planning problem has been largely studied in robotics [Latombe, 1991], the operating conditions for robots and automated vehicles are not equivalent. Constraints from both the vehicle and the infrastructure such as the non-holonomicity, the structured roads, and the traffic rules, lead to the appearance of different planning solutions. Solutions applied in robotics worked well on unknown or partially known environments, where the robot moves in a free-space non-structured environment. However, different solutions were needed for automated vehicles, traveling through roads which are structured environments and in most of the cases are well-known thanks to the information provided by the digital maps. Unlike robots, the movement of automated vehicles on the lanes is not only limited by the physical layout, but also by more restrictive kinematic and dynamic constraints. In the same way, path planning approaches for highways are not suitable either for urban driving due to the complexity of the scene, where the vehicle has to interact with the different VRU, generating a collision-free trajectory keeping passengers comfort as a design variable.

A review of the path planning approaches that have been used or are appropriate for automated vehicles has been done in this chapter. The different algorithms have been classified into three main groups, according to the base of the algorithm, namely: graph search based, sampling-based and interpolating curves based algorithms. Tables 2.1, 2.2 and 2.3 summarize their main advantages and disadvantages. A representation of the most relevant events for automated vehicles where path planning methods have been applied is presented in the timeline of Figure 2.28.

After reviewing the state of the art, the following conclusions can be drawn:

- Methods based on curves interpolation are the ones mostly used for urban areas. This results both from their flexibility, adapting the path to the dynamic conditions to avoid every possible collision with VRU; and from their low computational cost, allowing a real-time operation and re-planning.

- Graph search-based algorithms were the methods mostly used at first, thanks to the prior know-how on robotic applications. Since the research and development on this domain were promoted by the DARPA Urban Challenge, those algorithms have been studied combined with another interpolating curve based algorithm. Specifically, state lattices have been combined with splines for path planning on structured areas, both in static and dynamic environments. Lattices allow space-time planning and the adaptability to the lane workspace and are suitable for unknown or partially known environments. Splines allow the real-time evaluation of several path alternatives keeping a low computational burden and respecting both kinematic and dynamic constraints.

Motion planning systems have to provide a fast response to work efficiently together with the other modules of the architecture, especially with perception and control, being the input and output components respectively for the planning system. Since perception algorithms might be heavy algorithms requiring memory resources to represent the environment and a higher time consumption [Pendleton et al., 2017], the planning algorithms must be as fast as possible. They have to be reactive and fast enough to decide what to do and re-compute the trajectory on unexpected situations [Pivtoraiko and Kelly, 2009], such as when a pedestrian is detected crossing in the middle of the road, a vehicle is blocking the lane, or the human driver leaves the steering wheel control to the vehicle when traveling through a destination.

Robust algorithms to predict the behavior of the other road users in a dynamic environment such as road intersections, pedestrian crossing or parking lots have to be developed. Additionally, creating fault-tolerant planning systems is a must to react when failures occur in the perception and vision algorithms, avoiding possible accidents such as the one occurred in Arizona where a woman who was crossing the road was fatally run over by an automated vehicle. Learning algorithms may be considered on the path planning strategies in the short term. Research on this line is trying to study traffic scenarios, such as in [Fridman et al., 2018], where the perception, planning, and control systems are handled by a single neural network in the reinforcement learning process in a micro-traffic simulation. A review of emerging trends in this field, precisely in perception, planning and decision making is presented in [Schwarting et al., 2018]. As an open challenge, planning methods will have to provide safe and system compliant performance in complex, cluttered environments while modeling the uncertain motion of other traffic participants. So far, most approaches are rule-based, i.e., use a state machine to switch between predefined behaviors. It supposes a lack of generalization to unknown situations and to deal with uncertainties. Therefore, an integrated perception and planning solution is expected, where the control input for the vehicle is generated directly from sensory information relying on machine learning. There, deep-learning based algorithms have a high potential to improve planning algorithms by learning how the other vehicles react according to the traffic situation achieving the fourth automation level.

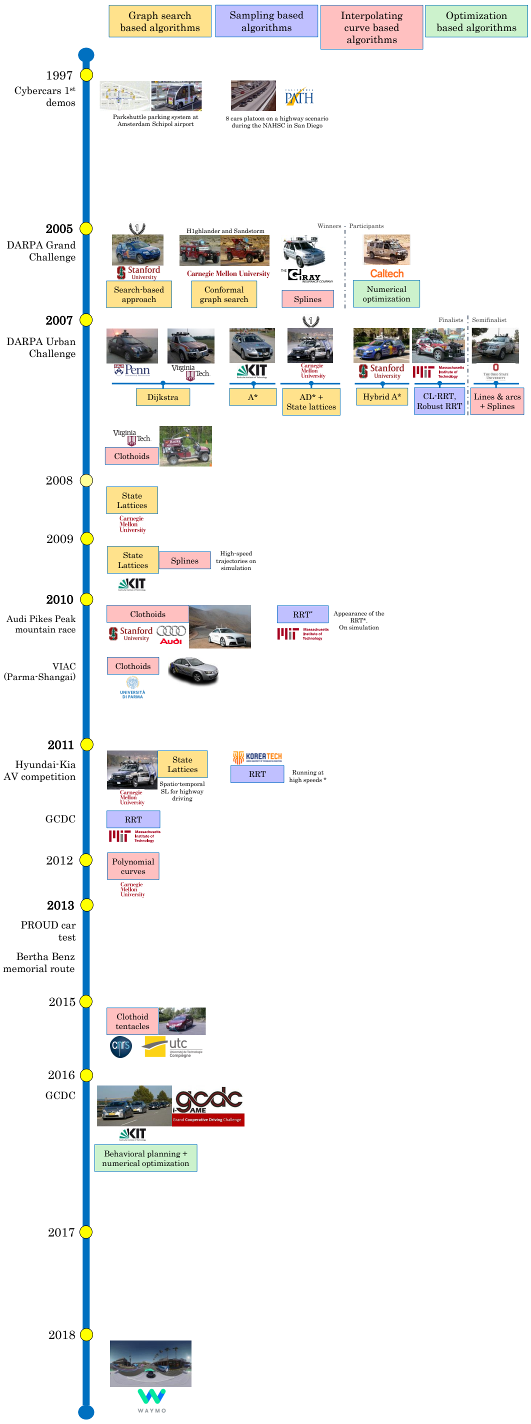

Graph search based algorithms
Sampling based algorithms
Interpolating curve based algorithms
Optimization based algorithms
1997
Cybercars 1st demos
Parkshuttle parking system at Amsterdam Schipol airport
8 cars platoon on a highway scenario during the NAHSC in San Diego
PATH
2005
DARPA Grand Challenge
Highlander and Sandstorm
Winners
Participants
Stanford University
Carnegie Mellon University
Caltech
Search-based approach
Conformal graph search
Splines
Numerical optimization
2007
DARPA Urban Challenge
Finalists
Semifinalist
Penn
Virginia Tech
KIT
Carnegie Mellon University
Stanford University
Massachusetts Institute of Technology
Dijkstra
A*
AD* + State lattices
Hybrid A*
CL-RRT, Robust RRT
Lines & arcs + Splines
Virginia Tech
Clothoids
2008
State Lattices
Carnegie Mellon University
2009
State Lattices
Splines
High-speed trajectories on simulation
KIT
2010
Audi Pikes Peak mountain race
Clothoids
Stanford University
Audi
RRT*
Appearance of the RRT*. On simulation
Massachusetts Institute of Technology
VIAC (Parma-Shangai)
Clothoids
Università di Parma
2011
Hyundai-Kia AV competition
State Lattices
Spatio-temporal SL for highway driving
Carnegie Mellon University
KOREATECH
RRT
Running at high speeds *
GCDC
RRT
Massachusetts Institute of Technology
2012
Polynomial curves
Carnegie Mellon University
2013
PROUD car test
Bertha Benz memorial route
2015
Clothoid tentacles
utc
2016
GCDC
gcdc
GAME
Grand Cooperative Driving Challenge
KIT
Behavioral planning + numerical optimization
2017
2018
WAYMO

Figure 2.28 – Path planning timeline on automated vehicles greatest hits

Chapitre 3

Planification des trajectoires pour des environements statiques

Below is a French summary of the following chapter "Path planning in static environments".

Ce chapitre présente l'algorithme de planification des trajectoires pour les environnements statiques. Il se compose de deux étapes:

- Étape de pré-planification, où les courbes optimales pour tout scénario de virage sont générées en tenant compte des informations statiques de la cinématique du véhicule et de la disposition de la route.
- Étape de planification en temps réel, où un trajet continu est construit en joignant la courbe pré-planifiée optimale pour chaque virage, en tenant compte de la configuration réelle de la route.

Ce planificateur local à deux étages offre un style de conduite humain grâce aux fonctionnalités suivantes:

- Il bénéficie de l'étape de pré-planification [Garrido et al., 2016a] pour générer des chemins plus lisses, où seul le point de jonction entre les courbes optimales est évalué en temps réel
- Un horizon de planification étendu de deux courbes consécutives est envisagé, optimisant simultanément deux courbes (la seconde à l'avance), grâce à la connaissance de la carte numérique fournie par le planificateur global [Garrido et al., 2016b].

Le but de cette étape est de générer un chemin qui s'adapte le plus à l'environnement, avec les moindres changements de courbure afin de faciliter le suivi de trajectoire en cherchant un voyage confortable pour les passagers. Ainsi, ce chemin statique est envoyé au planificateur local dynamique, qui le modifie en fonction des changements dynamiques sur la scène. Ce planificateur dynamique est présenté dans le chapitre suivant.

Chapter 3

Path planning in static environments

Automated vehicles are intelligent systems that use decision making to process the observations obtained from the onboard sensors and, together with the knowledge about the road and vehicle, are used to control the motion of the vehicle [Paden et al., 2016]. The decision-making system of automated cars can be decomposed into several layers (see Figure 2.15): behavioral planning, global planning, and local planning, which is connected to the control of the vehicle. Global planner selects a route through the road network from its current position to the requested destination. Behavioral planner considers the road conditions and the vehicles behavior to decide which motion specification carry out during the progress along the route. This route is smoothed on the local planning layer to make it feasible, and finally, the control system adjusts the variables to command the actuators to correct the lateral and longitudinal errors in the tracking of the reference trajectory.

The rest of the chapter is structured as follows: The motivation for studying the path planning problem for automated vehicles is discussed in Section 3.1. Then, the planning system is presented with a modular architecture, where the planning problem is divided first in global and local planning. The global planning stage is described in Section 3.2, where a route to the destination consisting of a set of way-points is generated. This route is transmitted to the local planner stage, which generates a collision-free continuous path smoothing the route, as explained in Section 3.3. Finally, some remarks are given in Section 3.4.

The main contribution of the thesis relies on a new planning architecture for automated vehicles, depicted in Figure 3.1. The different stages of the architecture are described below.

1. The global planner is composed of the following main blocks:
    (a) Digital map: it is the database with all the road layout information. In our case, it corresponds to the coordinates of the singular points (or way-points) and additional information such as the type of singular point or the maximum speed. These data is recovered from localization maps such as Google, Open Street Maps or from routing engines such as Open Source Routing Machine (OSRM). These coordinates are converted from the World Geodetic System (WGS84) to the coordinates system defined on the standard of the dynamics of road vehicles (ISO8855).
    (b) Route generation: This module receives from the HMI the desired destination point of the route. Then, it localizes both the vehicle and the destination point on the digital map, generating a set of way-points defining the itinerary to follow, which is finally sent to the local planner.
2. Local planner: It consists of two phases.

(a) First, a pre-planning phase presented in Section 3.3.1, where the goal is to pre-compute the optimal curve for every feasible isolated turn, taking advantage of the static information about vehicle and infrastructure. These optimal pre-computed curves for every feasible turn configuration are saved into several databases where the initial and final position of the vehicle changes on the lane according to the design parameters, allowing to join the curves later in the real-time stage providing a human-like driving.

(b) Second, a real-time planning stage presented in Section 3.3.2. The environment is considered as static, and a smooth and continuous path is guaranteed on real-time using the pre-planned information, evaluating the sharpness of the road bends and the available space among them to provide a human-like driving style. In addition, it presents an extended planning horizon, optimizing two consecutive curves concurrently, where the only parameter that is evaluated in real-time is the junction point between the loaded optimal curves. Finally, the resulting path is transmitted through a buffer to the dynamic planner. It is in charge of adapting the static path to the changing scenario, using a grid-based discretization of the scene, where obstacles are classified, and a virtual lane is built to target the dynamic scenario as a static one by joining two curves for each lane change, using the static planner. A re-planning is only made if the estimated motion of the obstacles is not as expected, or if an unexpected situation arises, generating a safe return to the lane maneuver aborting the former avoidance one.

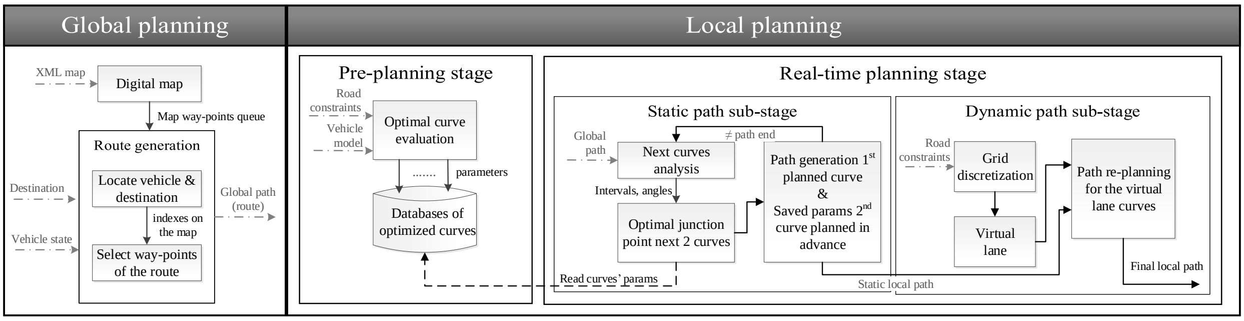

Global planning
Local planning
XML map
Digital map
Map way-points queue
Route generation
Destination
Locate vehicle & destination
indexes on the map
Vehicle state
Select way-points of the route
Global path (route)
Pre-planning stage
Road constraints
Vehicle model
Optimal curve evaluation
....
parameters
Databases of optimized curves
Real-time planning stage
Static path sub-stage
Global path
≠ path end
Next curves analysis
Intervals, angles
Optimal junction point next 2 curves
Path generation 1st planned curve & Saved params 2nd curve planned in advance
Read curves' params
Static local path
Dynamic path sub-stage
Road constraints
Grid discretization
Virtual lane
Path re-planning for the virtual lane curves
Final local path

Figure 3.1 – Path planning flowchart

3.1 Problem description

Urban areas represent the most challenging environment for automated vehicles because of the interaction with different agents such as pedestrians, cyclists, and other vehicles, especially in unexpected situations, which highly increase the complexity of the driving task. These dynamic environments require a proper real-time trajectory planning, being able to adapt the route according to the obstacles found in the changing scene, in a safe and comfortable way. As stated in Section 2.2, decision making is one of the biggest unsolved challenges in the field, and thus research on motion planning methods as part of the decision-making system is essential. Those systems will be able to model the behavior of other traffic participants, predicting their motion and planning their paths according to that. In addition, they will have to integrate perception to consider the uncertainties of the sensors seeking a robust solution, even with variable weather conditions [Schwarting et al., 2018].

The main focus of this thesis is to improve the existing path planning systems on urban transport vehicles. All the factors mentioned above demand algorithms able to run in real-time under these changing scenarios, performing a collision-free route from an origin to a destination point autonomously in a comfortable way. In this way, by making a proper trajectory generation, it would improve transportation by reducing the traffic flow as well as the fuel consumption, increasing the safety on the roads, reducing the likelihood of accidents.

In the field of automated vehicles control, path planning plays a key role to improve the efficiency and safety of transportation [Gu and Dolan, 2012], increasing the driving stability [Fu et al., 2015]. For open spaces such as parking lots, where automated vehicles deal with different road actors, it must determine the best possible collision-free path.

3.1.1 Assumptions and constraints

Over recent decades numerous planning systems have been developed, specially in the field of mobile robotics [Latombe, 1991], [LaValle, 2006], [Bestaoui Sebbane, 2014]. When planning trajectories for automated vehicles, it is important to consider different constraints that are not compatible with standard robotics systems. This has a significant impact on path planning. This section includes both the constraints and the assumptions set for the implementation of the presented path planning system, as well as their justification.

A series of assumptions and constraints concerning road, vehicle kinematics, motion and time performance are considered for the development of the local planning algorithm.

Regarding the road, information about road and obstacles coming from the perception and the global planner is assumed to be accurate, as well as the vehicle localization. Thus, the global planner provides the way-points forming the itinerary to follow, along with the lane width, as inputs for the local planner. Since the typical value of the lane width in urban roads is between 3 and 3.7 meters, a 3 meters lane width is assumed. As the vehicle circulates through structured environments, a constraint is set to respect the limits of the road, not invading the sidewalk. Thanks to the Convex Hull property of the curves, this constraint is met. It guarantees that the curve is defined inside the polygon formed by its outermost control points. Thus, by considering an internal separation of half the width of the vehicle at both sides of the road borders, the vehicle will always be inside the road limits.

Regarding the vehicle kinematics, the vehicle model is well-known concerning the physical characteristics such as width, length, and maximum steering angle. Since the vehicles are non-holonomic systems, they present a maximum steering angle and, therefore, a maximum associated curvature. Hence, the generated curves forming the path must comply with the maximum curvature requirement, which means that the curvature in all the points of the curves must be lower or equal than the maximum curvature of the vehicle, as will be further explained in Section 3.3. In addition, a kinematic model with drift, also known as bicycle model, is considered for the cycabs, due to the low-speed limitation presented on these platforms (up to 5 m/s), assuming that small lateral forces are generated, and the angle of drift is negligible at such speeds.

Regarding the motion, the generated trajectories have to be continuous to search the smoothness of the path and, consequently, the comfortability for the passengers. A constraint in the curvature at the beginning and the end of each curve is introduced forcing it to be approximately zero to ensure smooth transitions. There are several techniques that can be used for the trajectory generation for the local path planning [Katrakazas et al., 2015, González et al., 2016b]. Among them, the interpolating curve methods meet our requirements, and Bézier curves have been chosen due to the ease manipulation of the curves, since they are defined by control point. It allows to fulfill all the described requirements.

Regarding the execution time, the developed algorithm has to be implemented in real-time to work together with the perception and control algorithms. It means that the frequency of the system has to be at least of 10 Hz to be considered a real-time application since 100 milliseconds is the maximum response time perceived as instantaneous [Miller, 1968]. However, most of the human drivers react in a range of 0.3 to 1.2 seconds, depending on factors such as the speed of the vehicle or the visibility conditions [Ruhai et al., 2010], and brake reaction times on unexpected and surprise situations are between 1.25 and 1.5 seconds, varying in function of the level of risk and the time to collision [Summala, 2000]. According to these time constraints, quartic Bézier curves are considered, as they ensure the path continuity by introducing the curvature constraint, presenting a lower computational cost than quintic Bézier curves. Unlike cubic Bézier curves, quartic curves present a central control point allowing to push away or attract the curve to the center of the turn, providing more flexibility to the path generation. It also allows the generation of paths whose curvature profiles are less curvy and more comfortable for the passengers. Although they are costlier than cubic Bézier curves, since the function definition is well-known, there is almost no impact on the computational burden. Time to generate the optimal cubic curves for a specific turn angle on the databases is around 2.9 seconds, whereas it takes around 4.5 seconds for the quartic curves ones, which present five times more iterations. Therefore, the time to generate a single curve is around 1.35 milliseconds for quartic curves, and 1.2 milliseconds for cubic ones. Indeed, the narrowed the turn, the more difficult to find a cubic curve that fulfills the design requirements, such for example with the 60 degrees turns where there is no feasible solution with cubic curves.

3.2 Global planning

Global planning has been widely explored in the robotics fields, where the environments are usually non-structured, and the robots do not present non-holonomic constraints. However, global planning methods for automated vehicles have to be adapted and complemented with local planning strategies due to the limitations of either the environment (since vehicles usually operate on structured environments), the vehicle (as non-holonomic systems) or due to comfort constraints (paths should not be jerky to preserve comfort of passengers).

Among the different methods for global planning, the following have been tested with automated vehicles or car-like robots: graph search based algorithms, i.e., Dijkstra, A* based algorithms and state lattices, and the RRTs sampling-based algorithms, as shown in Section 2.3.

Since the goal of these global planners is to design routes minimizing travel time, they consider the shortest path generated by the underlying graph search based algorithm, together with the information of the road, such as the traffic conditions, accidents, or works.

Although these systems make use of the GPS sensors to get the vehicle location during the route, an accurate digital map of the road is required. Different navigation systems such as Waze and Google Maps provide an accurate digital map which considers the real-time traffic information to generate several options to the user through an HMI to arrive at the destination, using GPS as the localization source.

However, this localization does not consider the position of the vehicle on the lane, the borders of the lanes or the road layout, mostly on urban scenarios, where both road and marking may not be properly defined. The research trend is the real-time lane marking detection on roads based on vision algorithms [Liu et al., 2008], [Bauda et al., 2017], [Lee and Moon, 2018].

Localizing vehicles with respect to the lane marking is still an unsolved challenge, but a combination of the information coming from the GPS-RTK and the IMU sensors is still an accurate way of being localized on open environments, and a digital map based solution can be considered.

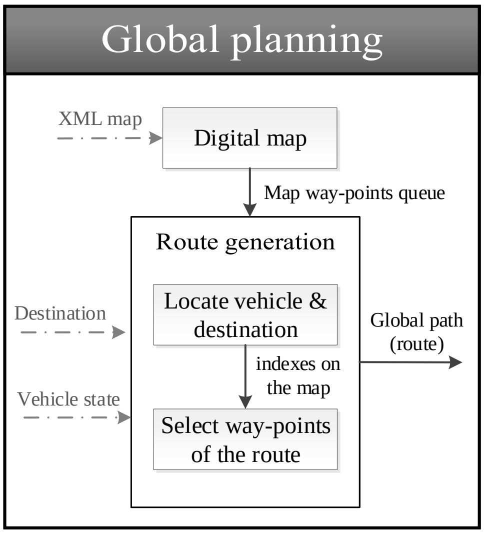


Figure 3.2 – Global planning flowchart

The global planning strategy of the team is depicted in the flowchart of Figure 3.2. First, the global planner reads an XML file representing the way-points describing the route. These way-points consist of the (x,y) GPS coordinates describing the center of the road most significant points, as well as its maximum speed and the type of road shape. Thus, these points need to be taken by hand based on the digital map of the area, as in Figure 3.3, where a global path formed by the way-points in red is shown at the INRIA-Rocquencourt facilities, for a path following operation from an origin point O to a destination point D. Automatic global planning approaches can be used there to modify the itinerary in real-time through an HMI, changing the destination point while traveling [Vaca et al., 2016]. It allows the vehicle to adapt the route in function of traffic congestion and users demand. Additionally, it allows to add stop points in the route or modifying the intermediate way-points choosing another itinerary to arrive at the destination. Automation of the global planning process is based on Open Source Routing Machine (OSRM)[1], a digital map tool that provides a set of singular points for the different requested destinations, taking into account the geographic information and road network, generating an itinerary to be followed in real-time.

These way-points are placed in the center of the lane. That way, as the road width is well known (either because of the localization maps or because of the knowledge of the usual urban road configuration, where the width is between 2.7 and 3.7 meters) the road limits are computed geometrically by applying half of the lane width to both sides of the physical representation resulting from the union of those way-points. Thus, the road is easily modeled with straight lines building a corridor through which the trajectory will be generated [w. Choi et al., 2010], transforming that way the real space into a lane space [Horst and Barbera, 2006].

The following XML code represents an example of a pre-defined global path, like the one shown in Figure 3.3.

```
<network>
    <link id="1" dir="0" >
        <node id="1" x="28.69" y="45.76" speed="3" ></node>
        <node id="2" x="19.15" y="24.50" speed="3" type="-2" ></node>
```

---

[1] http://project-osrm.org/

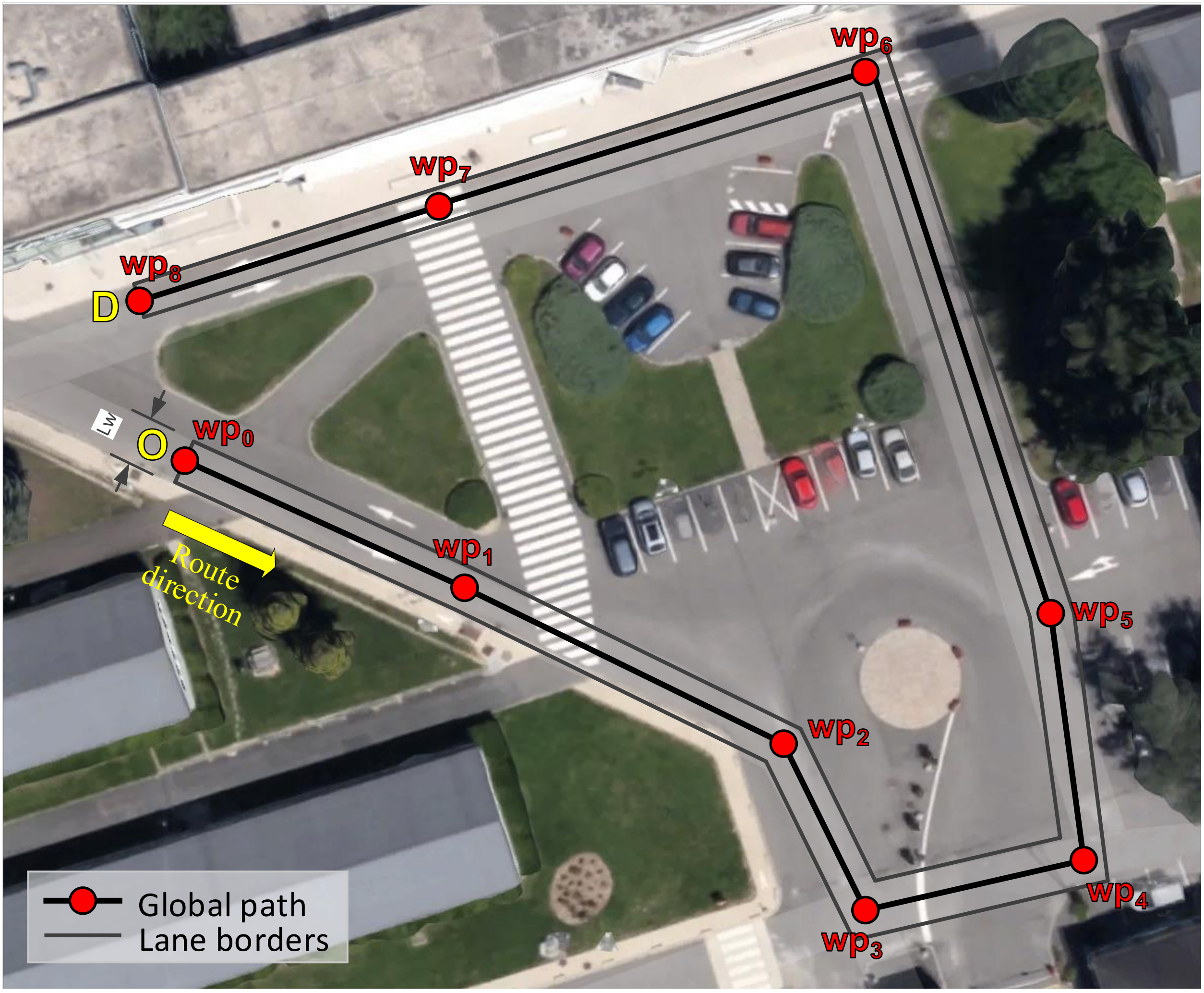

wp$_6$
wp$_7$
wp$_8$
D
LW
O
wp$_0$
Route direction
wp$_1$
wp$_5$
wp$_2$
wp$_4$
wp$_3$
Global path
Lane borders

Figure 3.3 – Global path example at INRIA-Rocquencourt facilities

```
        <node id="3" x="11.60" y="3.80" speed="3" type="-2" ></node>
        <node id="4" x="-6.13" y="-3.77" speed="3" type="-2" ></node>
        <node id="5" x="-2.42" y="-17.60" speed="3" type="-2" ></node>
        <node id="6" x="16.97" y="-16.53" speed="3" type="-2" ></node>
        <node id="7" x="56.18" y="-6.02" speed="3" type="-2" ></node>
        <node id="8" x="47.87" y="26.08" speed="3" type="-2"></node>
        <node id="9" x="41.47" y="47.74" speed="3" ></node>
    </link>
</network>
```

There, each node represents a way-point, containing the following information:

- Id: Identification number for each way-point in the XML
- x,y: Cartesian coordinates (in meters) describing its global position on the local map with respect to an origin point defined by the localization technique, usually either an RTK-GPS or a SLAM based algorithm.

- Speed: Maximum speed associated with the road segment between the current and the following way-point.
- Type: Type of singular point. It allows defining the road shape, where -2 represents a turn or intersection. Some other road shapes can be added, such as roundabouts or special road crosses. The first and last points do not have this attribute since both are origin and destination points respectively.

Some other parameters can be added to the XML map, such as the altitude of the points for considering the slope of the terrain, but it is out of the scope of the thesis.

Following the flowchart describing the global planning strategy of Figure 3.2, once the XML file is read, a data structure containing this map is saved. Then, the vehicle is located on the map, i.e., finding the future closer way-point on the map and considering the vehicle position as the origin way-point. After gathering the destination point chosen by the human from the HMI, this point is also located on the map in the same way. Then, the global path is created by adding to a list the way-points between origin and destination, as well as the origin point as the first one and the destination point as the last one. Finally, the global path list is sent to the local planner. Hence, the destination point acts as a trigger to activate the system, waiting for the human request to start the automated mode. In case a new destination request is made during the path execution, the global planner re-plans the new path in real-time, selecting the way-points that form the new itinerary. It would be equivalent and replaceable by a digital map like that of HERE[2] or any other manufacturer that gave the information of the road.

3.3 Local planning

Since global planner approaches provide a route formed by a set of way-points defining the itinerary, this global path only presents G^0 geometric continuity, where curves touch at the joint point, i.e., it is only geometrically continuous on its function definition, but not on its derivatives. It means that, as it is composed by straight lines, its stretches touch at the joint points, but they do not share a common tangent direction. The purpose of the local planner is to take the global path as a reference path and smooth it to comply with the kinematic and dynamic constraints of both vehicle and road, considering the non-holonomicity of the system.

In the rest of this section, the proposed local planning approach for static environments is presented as a two-stage process: pre-planning and real-time planning, as depicted in Figure 3.4. There, the left part shows the pre-planning stage [Garrido et al., 2016a]. It generates the databases which contain the optimal curve for each kind of turn feasible by the vehicle. Meanwhile, the right part represents the real-time stage [Garrido et al., 2016b]. It generates a smooth and continuous path by loading from the databases the appropriate curves, joining them according to the characteristics of the road and the map, coming from perception and global planner, respecting the vehicle constraints as well.

3.3.1 Pre-planning stage

Thanks to the accuracy provided by current perception systems and localization maps, together with the knowledge of the itinerary provided by global planers, we know in advance how is the environment through which the vehicle must drive. Hence, as the way-points defining the itinerary

---

[2] https://www.here.com/

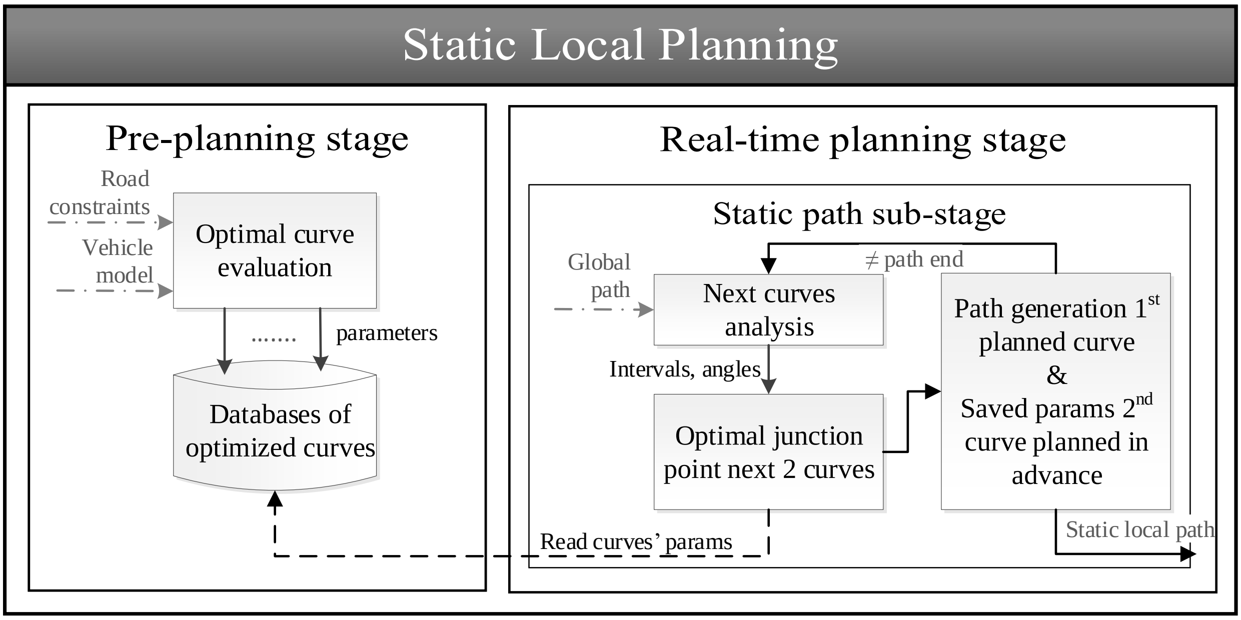

Static Local Planning
Pre-planning stage
Road constraints
Vehicle model
Optimal curve evaluation
....
parameters
Databases of optimized curves
Real-time planning stage
Static path sub-stage
Global path
≠ path end
Next curves analysis
Path generation 1st planned curve & Saved params 2nd curve planned in advance
Intervals, angles
Optimal junction point next 2 curves
Read curves' params
Static local path

Figure 3.4 – Local planning for static environments flowchart

to follow are received from the global planner, it is immediate to compute the available distance between the road way-points, as well as the sharpness of the bends defining the scenario.

One can easily identify the next static parameters: 1. the physical and kinematic characteristics of the vehicle, 2. road layout, in the sense of limits of the road and configuration of the lanes, distance between turns, or angles of the feasible turns. These parameters are well-known and do not change during the navigation (as seen in Sub-section 3.1.1), assuming low-speed vehicles ruled by a kinematic model. Among them, the following can be highlighted: length and width of the vehicle as physical parameters, maximum steering angle and the associated maximum feasible curvature as kinematic parameters of the vehicle, and road lanes width and turning angles for the road bends as infrastructure constraints. Table 3.1 shows these values for the team vehicles, which are first mentioned in Section 2.1 and detailed later in Section 5.1.

Table 3.1 – Main physical parameters of the vehicles

Vehicle	Physical parameters		
	Wheelbase	Width track	Max steering angle (deg)
Cycab	1.25 m	1.15 m	38.5
Cybus	2.15 m	1.05 m	38.5
Citroën C1	2.34 m	1.415 m	40.0

Computation cost for local path planning may be too high if the full process is done in real-time. We can get benefit from all the above to pre-compute the optimal curve the vehicle can perform for all possible single-turn scenario, regardless of the vehicle type. Figure 3.5 depicts a single-turn scenario, defined by three points, two segments and the angle of the turn that they form. The points are defined in the center of the lane: the origin of the turn (G_{n-1}), the central point of the turn (G_n) and the ending point of the turn (G_{n+1}). The distance between these points, i.e., the length of the segments formed by joining the points are dis_{seg1} and $dist_{seg2}$. These points represent the trio of way-points that constitute the itinerary received by the global planner. Finally, the angle of the turn is represented by α.

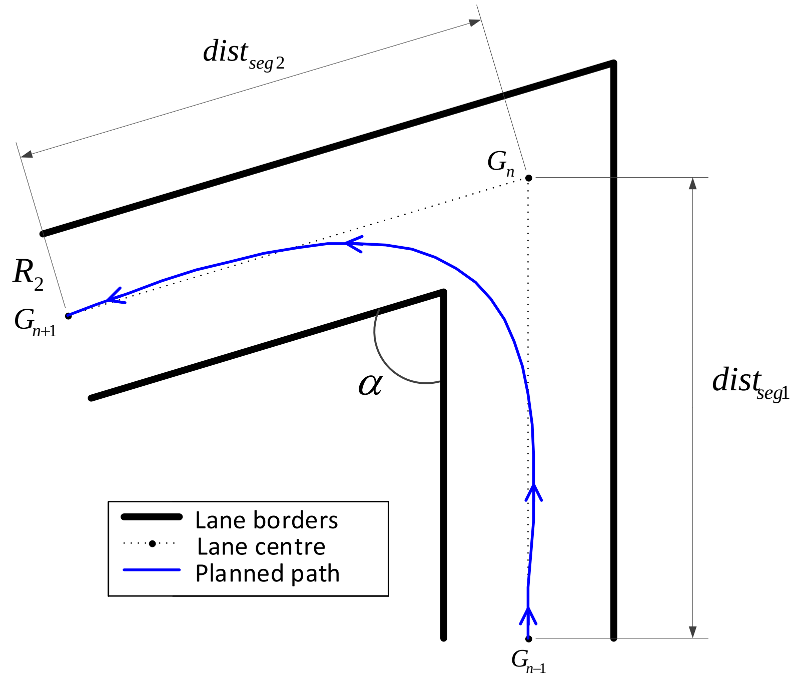


Figure 3.5 – Single-turn scenario with optimal curves search [Garrido et al., 2016a]

Thus, the algorithm goal is to find the optimal curve for each single-turn configuration, i.e. for each trio α, $dist_{seg1}$, $dist_{seg2}$.

To this end, the algorithm iterates over the different singular curves by changing the value of the three parameters. Then, for each iteration it evaluates the optimal curve and saves the relative location of the optimal curves' control points into a database indexed with the same three parameters. Finally, the generated databases are loaded later in the real-time stage, generating the whole path by joining the optimal curve for every single turn of the itinerary.

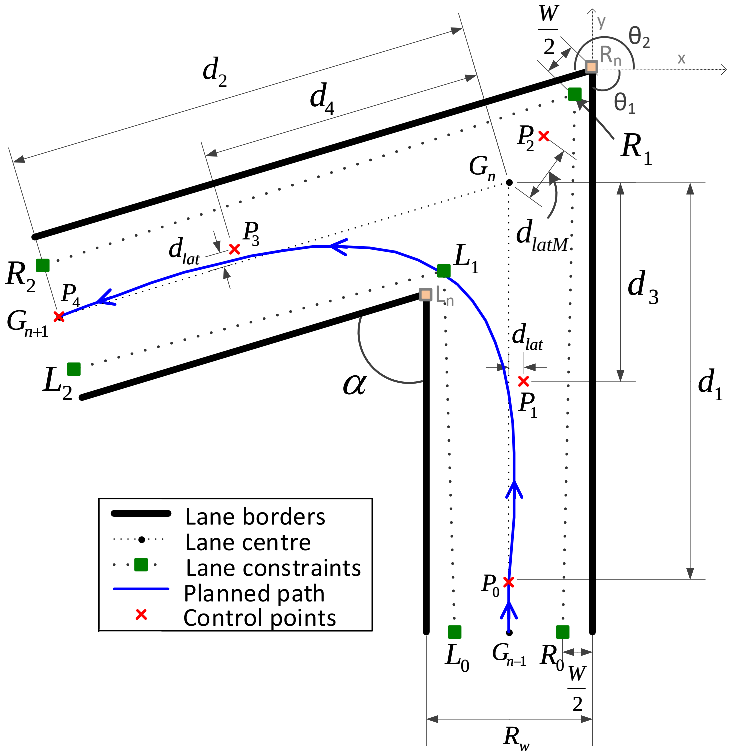
A pseudo-code description of the intelligent algorithm developed for this pre-planning stage is presented in Algorithm 1, where it is shown how the databases containing the parameters of the optimal curves are generated. Next sub-sections explain the algorithm in detail. Therefore, this pre-planning stage seeks for the curve that better fits in a pre-defined turn scenario.

3.3.1.a Bézier curves based path planning

Chapter 2.3 presented the different techniques for path planning, grouped in: graph search based algorithms, sampling based algorithms and interpolating curves algorithms. As studied there, the group of techniques that fit more our requirements is the one based on interpolating curves. Graph search based planners are not the most appropriate for local planning since they are based on a grid search, and the resulting path presents discontinuities. Thus, these methods are mostly applied to global planning together with a further smoothing approach, which could be a local planning method. Sampling-based planners could respond to the real-time constraint, but continuity there

Algorithm 1 Pre-planning algorithm for the generation of the databases containing the parameters of the optimal curves

```
for each turn configuration (changing $\alpha$, $d_1$, $d_2$) do
    Calculate road limits and road constraints
    Changing $d_3$, $d_4$, $d_{lat}$ and $d_{latM}$
    Calculate control points location
    Check if the curve is feasible by the vehicle
    Check if the continuity constraints are fulfilled
    while the curve is not totally generated do
        Generate curve points
        Calculate nearest point to internal constraint point
    Check if the car doesn't invade the sidewalk
    Evaluate curve's optimality with cost function
    Check if this curve improves the best one
Save the best curve's configuration in the database
```



d_2
d_4
$\frac{W}{2}$
y
θ2
Rn
x
θ1
R_1
P_2
G_n
d_{latM}
d_3
d_1
P_3
d_{lat}
L_1
Ln
R_2
P_4
G_{n+1}
L_2
α
d_{lat}
P_1
Lane borders
Lane centre
Lane constraints
Planned path
Control points
P_0
L_0
G_{n-1}
R_0
$\frac{W}{2}$
R_w

Figure 3.6 – Bézier curve [Garrido et al., 2016a]

is not guaranteed. However, interpolating curve planners allow a faster path generation while respecting the curvature continuity requirements for generating smooth trajectories. This is feasible thanks to the modularity of the curves (especially the polynomial based approaches). Among the different interpolating curve algorithms, we have therefore chosen Bézier since these curves accomplish the real-time and the continuous curvature path constraints, thanks to the flexibility provided by the control points defining them. Thus, since they are ruled by control points, positioning them through space is a simple task even when the degree of the curve is high.

The basis of these curves are the Bernstein polynomials. The generation of an n-degree Bézier curve is mathematically defined in Equation 3.1.

$$B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i \quad t \in [0,1] \tag{3.1}$$

where n is the degree of the polynomial equation, and P_i are the control points that define the curve in the time interval t, defined between 0 and 1.

As introduced in Chapter 2.3, these curves present several properties that make them suitable for path planning under the described constraints [Han et al., 2010]. The most important ones for the current work are stated below.

(i) The curves begin at the first control point (P_0) and end at the last control point (P_n).

(ii) The curves begin with a direction defined by the vector formed by the first two control points ($\overrightarrow{P_0P_1}$) and end with a direction defined by the last two control points ($\overrightarrow{P_{n-1}P_n}$).

(iii) The curves are framed within the convex hull defined by the outermost control points.

(iv) The behavior of the curve concavity is consistent with the concavity formed by the control points, as can be seen in Figure 3.7.

(v) The curves are fully symmetric, i.e., if the control points are reversed, the resulting curve is the same.

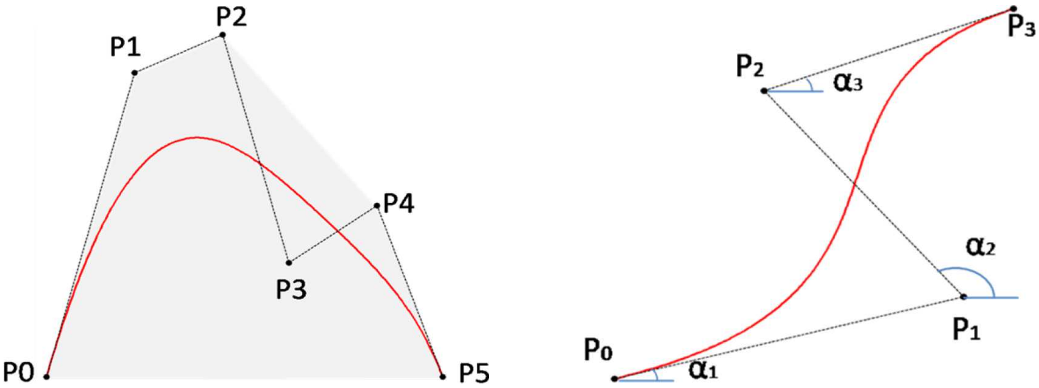


Figure 3.7 – Bézier's Convex Hull and concavity change

3.3.1.b Algorithm description

The goal of the pre-planning algorithm is to find the optimal curve for each single-turn configuration, i.e. for each trio α, $dist_{seg1}$, $dist_{seg2}$ (see Figure 3.5). Thus, the algorithm iterates over the different singular curves by changing the value of the three parameters. Then, for each iteration, it evaluates the optimal curve and saves the relative location of the optimal curves' control points into a database indexed with the same three parameters. Finally, the generated databases are loaded later in the real-time stage, generating the whole path by joining the optimal curve for each single turn of the itinerary.

Since the goal is to generate a database for each resolution type, firstly it is necessary to set the step of variation for the parameters defining the single-turn scenario, that is, for the angle of the turn (α), the distance to the turn mid-point ($dist_{seg1}$) and the distance from there to the end of the turn stretch ($dist_{seg2}$), as explained in Subsection 3.3.1.g. Then, the algorithm first iterates changing the angle of the turn (Algorithm 1, line 1), ranging from the sharpest turn that the vehicle can perform according to its maximum steering angle (about 35°) up to a straight line (180°). Thus, the sharpness of the turn is defined.

For each turn with a defined angle α, the algorithm iterates the other two parameters defining the turn, i.e., the *arrival distance* (distance up to the mid-point of the turn) and the *exit distance* (distance to leave the turn up to the next way-point). Since the databases are thought to consider realistic turn scenarios, a maximum distance of 40 meters is considered for both arriving and exit distances. Assuming that a human driver usually begins to signal a turn about three seconds ahead, a distance of 40 meters has been assumed as the maximum distance to start the maneuver, considering that 50 km/h (13.89 m/s) is the maximum speed in urban environments.

Once the parameters that define the turn are defined, the road limits are computed geometrically (Algorithm 1, line 2), assuming that the starting and ending way-points are located at the center of the lane and applying half of the lane width to both sides to build the lane representation with straight segments. In addition, to ensure that the vehicle stays in the lane limits without invading the sidewalk curbs, a set of points forming a corridor internal to the lane representation are computed. These points are separated an inner distance of half the vehicle width ($W/2$) from the lane boundaries (points L_0, L_1, L_2 and R_1, R_2, R_3 in Figure 3.6 for the left and right lane constraints, respectively). Thanks to the Convex Hull property of the Bézier curves described above, by placing the control points of the curve inside the polygon formed by these points, it ensures that the curve will be inside this polygon, respecting the road curbs.

Since Bézier curves are defined by control points, the algorithm has to compute their location. The algorithm tries to find first a third-order Bézier curve (cubic) that fits the requirements, and if it is not valid, a fourth order Bézier curve (quartic) is evaluated. Thus, the number of control points will be either four or five. The control points are placed in the defined corridor according to the degree of the Bézier curve, obtaining the following configurations:

1. On the one hand, in the case of evaluating cubic Bézier curves (four control points): the first two control points are placed in the first half of the corridor (between the way-points G_{n-1} and G_n). First (P_0) and last (P_n) control points are the first and last points of the curve. They are placed at a longitudinal distance of d_1 and $d2$ with respect to the turn mid-point, respectively, either in the center of the lane or with a lateral displacement, depending on the concavity change with respect to the previous and following turn, as will be explained in Subsection 3.3.1.f. However, the inner control points are moved not only longitudinally but also laterally, in order to find better curves by reducing the curvature. Longitudinally, they are moved from the turn mid-point up to the location of the external control points, being placed at a distance of d_3 and d_4, respectively. Laterally, they are moved from the lane center (no lateral displacement) to the border of the internal corridor separated half of the vehicle width. Thus, $d_3 < d_1$ and $d_4 < d_2$.

2. On the other hand, if quartic Bézier curves are evaluated (five control points), the fifth control point is placed parallel to the turn central way-point. The algorithm iterates the lateral displacement d_{latM} from the lane center (no lateral displacement, matching with G_n position) to the position of the road constraint R_1 in the case of a left-hand curve (or L_1 in the case of a right-hand curve), i.e. a distance of half of the lane width minus half of the

vehicle width ($R_w - W/2$), considering the angle of the turn.

One of the design parameters of the planning algorithm is continuity on the path. Generating paths with G_1 geometric continuity increases the smoothness on the path. It is achieved by joining cubic or quartic curves sharing a common tangent direction. Providing this behavior is a simple task thanks to the adaptability of the Bézier curves. Considering consecutive turns, where the joint point between curves corresponds to both last and first control points of the two curves, it ensures having the last two points of a curve aligned with the first two points of the next one. Ensuring G^2 continuity (where curves also share a common center of curvature at the joint point, i.e., three control points are aligned in the same vector) would require both first and second function derivatives continuous at the joint points, which is feasible from quintic Bézier curves [González et al., 2016a]. Since these curves are computationally more expensive than quartic ones, a constraint in the curvature at the joint points of the quartic curves is set, evaluating curves whose first and last points have a curvature approximately zero. This results in an improvement in the G^1 continuity of the path.

Once the algorithm has set these parameters to evaluate the curve, i.e., the order of the Bézier curve and the longitudinal and lateral distances to place the control points, the control points are computed as described in Equation 3.2 (see Figure 3.6).

$$P_{i_{x,y}} = \begin{pmatrix} G_{n_x} + d_1 \frac{G_{n-1_x} - G_{n_x}}{\|G_{n-1} - G_n\|} + sign \cdot d_{lat} \cdot cos(\theta - \frac{\pi}{2}), \\ G_{n_y} + d_1 \frac{G_{n-1_y} - G_{n_y}}{\|G_{n-1} - G_n\|} + sign \cdot d_{lat} \cdot sin(\theta - \frac{\pi}{2}) \end{pmatrix} \tag{3.2}$$

$$P_2 = \begin{cases} G_n - d_{latM} \cdot \frac{G_n - L_n}{\|G_n - L_n\|}, & \text{if right turn} \\ G_n + d_{latM} \cdot \frac{G_n - L_n}{\|G_n - L_n\|}, & \text{if left turn} \end{cases} \tag{3.3}$$

where:

- P_i are the (x,y) coordinates of the control points defined in Equation 3.2, except for the central control point in quartic Bézier curves, which is computed from Equation 3.3.

- G_n represents the (x,y) coordinates of the way-point defining the center of the turn, whereas G_{n-1} and G_n represent the previous and the next way-point, respectively.

- d_1 is the distance from the first way-point of the turn (G_{n-1}) to the central way-point (G_n) describing the turn, hereinafter called *arrival distance*. The first control point of the curve P_0 is placed at this distance from the turn mid-point G_n. In the same way, d_2 is the distance from the central way-point of the turn to the next way-point, hereafter called *exit distance*. The last control point of the curve P_4 (for quartic Bézier curves) is therefore placed at this distance from the turn mid-point G_n.

- Conversely, $d3$ and $d4$ represent the longitudinal distance to place the internal control points (P_1 and $P3$, respectively), with respect to G_n. These points also present a lateral displacement with respect to the center of the lane, which is described by d_{lat} in Figure 3.6.

- d_{latM} is the lateral displacement with respect to G_n for the central control point in quartic Bézier curves.

- L_0, L_1, L_2 and R_0, R_1, R_2 represent the points that constraint the lane on the left and right sides, respectively, allowing to build the polygon where the control points are placed. These points are placed at a distance of $W/2$, i.e., half of the vehicle width from the lane borders.

- L_n and R_n are the points defining the left and right corners of the turn.
- θ, which can be either θ_1 or θ_2, represent the angle used for computing the control points displaced from the center of the lane, considering θ_1 for those in the first segment and θ_2 for those in the second segment.
- *sign* represents the direction of the turn. It will be negative for right turns and positive for left turns.

For each turn iteration, another iteration process is done to generate the different Bézier curve trajectories modifying the location of the five control points P_n, both laterally and longitudinally in the mathematical definition of the curve (see Equation 3.1). This means changing the values of the following parameters to generate the control points: the longitudinal distances to place the control points (d_1, d_2, d_3, d_4), and the lateral displacements with respect to the lane center (d_{lat} and d_{latM}).

Once the control points are computed and the curve is generated, an analysis of its feasibility must be done (Algorithm 1, line 5). It will ensure that the vehicle will be able to track the curve due to the kinematic constraints are respected since the curves forming the planned path present a curvature profile smaller than the maximum curvature feasible by the vehicle. The maximum curvature feasible by the vehicle is given when the angle of the wheel is maximum using the Ackerman model [Ackermann, 1999]. The analytical method is used to calculate the curvature k at each point t of the Bézier curve $\overrightarrow{B}(t)$, defined in Equation 3.4 for two-dimensional curves [Walton et al., 2003]. This method calculates the curvature k at each point t of a curve $B(t)$ whose (x, y) coordinates depend on this variable t [Choi et al., 2010], [Walton et al., 2003]. Because of the simplicity of this equation, it makes its application in computational algorithms much faster, if the equation that defines the curve and if its derivative are known.

$$k(t) = \frac{\overrightarrow{B}'(t) \times \overrightarrow{B}''(t)}{\left\| \overrightarrow{B}'(t) \right\|^3} \tag{3.4}$$

Although there exist other methods to compute the curvature, the analytical computation allow us to get the curvature from the mathematical definition of the curve, whereas other methods compute the curvature by generating a circular arc with a set of passed (or future) planned points (minimum 3), which adds at every step an error in the measurement.

Each of the Bézier curves evaluated must comply with the maximum curvature constraint, as well as with the zero curvature constraint at the initial and final points of the curve (Algorithm 1, line 6). This last constraint allows to generate a continuous path, avoiding discontinuities in the joints between curves or curves and straights, and therefore it is translated into the generation of smoother paths.

3.3.1.c Optimality criteria

The curves are evaluated through optimality criteria determined by a cost function. The optimal curve for each turn configuration is the one that minimizes the cost (i.e., maximizes the fitness). In this work, several cost functions have been used in the intelligent algorithm, as presented below.

1. The first proposed approach takes into account the curvature, as it was considered in the previous work in the RITS team [González et al., 2014]. Specifically, it considers the measure of the curvature k in the three most critical points: at the beginning, in the middle and at

the end of the curve. On the one hand, the first and last point play a key role since they link with previous and next road stretches respectively. These points correspond to $t = 0$, $t = 0.5$ and $t = 1$ in Equation 3.4, respectively. On the other hand, since the Bézier curves are generated symmetrically there (i.e. placing the internal control points at the same distance, as well as the external ones). It leads to a curvature profile whose maximum value is reached at the mid-point of the curve. Therefore, this first cost function search the curves minimizing the curvature at these three points, as described in Equation 3.5

$$Q_1 = k_0 + k_{n/2} + k_n \tag{3.5}$$

where $n + 1$ is the number of points of the Bézier curve.

2. Coming up next, as a second approach, the cost function considers minimizing the derivative of the curvature (k') instead of the curvature (k) on those three points. Equation 3.6.

$$Q_2 = k'_0 + k'_{n/2} + k'_n \tag{3.6}$$

Since the derivative penalizes sudden changes on the curvature, the aim was obtaining a smoother path avoiding the abrupt changes on the curvature.

Equation 3.7 defined in [Walton et al., 2003] is used to calculate the derivative of the analytical curvature for each curve in the algorithm, where the parametric curve is defined by the set of points $\overrightarrow{B}(t) = (x(t), y(t))$ for a real $t \in [0, 1]$

$$k'(t) = \frac{r(t)}{\left\|\overrightarrow{B}'(t)\right\|^5} \tag{3.7}$$

where

$$r(t) = \left\{\overrightarrow{B}'(t) \cdot \overrightarrow{B}'(t)\right\}\left\{\overrightarrow{B}'(t) \times \overrightarrow{B}'''(t)\right\} - 3\left\{\overrightarrow{B}'(t) \times \overrightarrow{B}''(t)\right\}\left\{\overrightarrow{B}'(t) \cdot \overrightarrow{B}''(t)\right\} \tag{3.8}$$

Since the Bézier function is well-known, its derivatives can be pre-computed for the degrees of the curves that are used, in this case for cubic and quartic Bézier curves. Thus, the first, second and third derivatives of the cubic, quartic and quintic Bézier curves required for computing the curvature derivative are shown in Equations 3.9, 3.10 and 3.11, respectively.

$$\begin{aligned}
B'(t) &= -3(1-t)^2 \cdot P_0 + 3(3t^2 - 4t + 1) \cdot P_1 + (-3t^2 + 2t) \cdot P_2 + 3t^2 \cdot P_3 \\
B''(t) &= 6(1-t) \cdot P_0 + 3(6t - 4) \cdot P_1 + 3(-6t + 2) \cdot P_2 + 6t \cdot P_3 \\
B'''(t) &= -6 \cdot P_0 + 18 \cdot P_1 - 18 \cdot P_2 + 6 \cdot P_3
\end{aligned} \tag{3.9}$$

$$\begin{aligned}
B'(t) &= -4(1-t)^3 \cdot P_0 + 4((1-t)^3 - 3t(1-t)^2) \cdot P_1 + 6(2t(2t^2 - 3t + 1)) \cdot P_2 \\
&\quad +4((3 - 4t)t^2) \cdot P_3 + 4t^3 \cdot P_4 \\
B''(t) &= 12(1-t)^2 \cdot P_0 - 24(2t^2 - 3t + 1) \cdot P_1 + 12(6t^2 - 6t + 1) \cdot P_2 \\
&\quad +24t(1 - 2t) \cdot P_3 + 12t^2 \cdot P_4 \\
B'''(t) &= (24t - 24) \cdot P_0 + (-96t + 72) \cdot P_1 + (144t - 72) \cdot P_2 \\
&\quad +(96t + 24) \cdot P_3 + 24t \cdot P_4
\end{aligned} \tag{3.10}$$

$$\begin{aligned}
B'(t) &= -5(1-t)^4 \cdot P_0 + 5((1-t)^4 - 4t(1-t)^3) \cdot P_1 \\
&+10(-6t^2(1-t)^2 + 2(1-t)^3) \cdot P_2 + 10(-3t^2(1-t)^2 + 2t(1-t)^3) \cdot P_3 \\
&+5(-t^4 + 4t^3(1-t)) \cdot P_4 + 5t^4 \cdot P_5 \\
B''(t) &= 20(1-t)^3 \cdot P_0 - 20(2(1-t)^3 - 3t(1-t)^2) \cdot P_1 \\
&+20((1-t)^3 - 6t(1-t)^2 + 3t^2(1-t)) \cdot P_2 + 20(3t(1-t)^2 - 6t^2 + 7t^3) \cdot P_3 \\
&+20(3t^2 - 5t^3) \cdot P_4 + 20t^3 \cdot P_5 \\
B'''(t) &= (-60t^2 + 120t - 60) \cdot P_0 + (300t^2 - 480t + 180) \cdot P_1 \\
&+(-600t^2 + 720t - 180) \cdot P_2 + (600t^2 - 480t + 60) \cdot P_3 \\
&+(120t - 300t^2) \cdot P4 + 60t^2 \cdot P_5
\end{aligned} \tag{3.11}$$

3. Results from the prior cost function showed that considering the derivative of the curvature it is not enough for generating smooth paths due to the non-avoidance of high curvature peak profiles, especially at the mid-point of the curve. This led to consider a combination of both curvature and its derivative in the cost function as a third approach. It tries to find the curves that minimize the derivative of the curvature at these three significant points and the maximum curvature in the middle of the curve. Equation 3.12 describes the corresponding cost function.
$$Q_3 = k_{n/2} + k'_0 + k'_{n/2} + k'_n \tag{3.12}$$

4. With the previous cost function, which minimizes both the maximum curvature and its derivative in the three most significant points, a slight improvement regarding smoothness was achieved, but not as much as expected. Therefore, based on [Xu et al., 2012] and [Gu and Dolan, 2012] the cost function was redefined considering all the relevant static parameters regardless of the vehicle dynamics, as shown in Equation 3.13. This allows evaluating not only the comfort but also the efficiency, the energy consumption, and the driving behavior. Those parameters are the following: (i) First since the generated curves are not symmetrical (i.e. the control points are not necessarily equidistantly placed), both curvature in its absolute value $|k_i|$ and its change $|k'_i|$ are considered in the whole curve (i.e. in all its points). (ii) In addition, the lateral offset at every point of the curve $offset_i$ (that is, the displacement from the center of the lane) is also considered, since the displacement from the center of the lane can be translated into a lateral error. (iii) Then, since one of the goals of the planning is to generate the shortest paths between origin and destination, the length of the path l is considered as an optimization parameter, with an associated weight w_l. (iv) Finally, in addition to minimize the length of the path, another aim of the optimization function is to generate the shortest path as fast as possible. Thus, the time t to generate the curve is also included in the cost function.
$$Q_4 = \sum_{i=0}^{n} w_{ki}\,|k_i| + w_{k'i}\,|k'_i| + offset_i + w_l \cdot l + t \tag{3.13}$$

5. Finally, since the paths that minimize the curvature and its change are similar among them, and the less the curvature, the less the distance covered, this parameter can be removed from the cost function. In the same way, the time to generate the curve is not needed. Considering that the experimental vehicles (either the current automated vehicles on the roads or the vehicles on simulation) cannot travel to high speeds, due to either limitations of the infrastructure (the test track is limited to 30 km/h, and urban roads are limited to 50

km/h) or limitations of the vehicles (cybercars are limited to a maximum speed of about 20 km/h).

$$Q_5 = \sum\nolimits_{i=0}^{n} w_{ki} \left|k_i\right| + w_{k'i} \left|k'_i\right| \tag{3.14}$$

Thus, the optimal curve for each turn configuration is the one that minimizes the cost computed by applying the cost function Q_5, defined in Equation 3.14. There, n represents the number of points defining the curve, w_{ki} is the weight assigned to the curvature and $w_{k'i}$ the one assigned to the curvature derivative.

Hence, the goal of this cost function is to minimize both curvature and its derivative in all the points defining a curve, penalizing abrupt changes of the curvature and trying to find the minimum possible curvature to have a smooth and safe path.

3.3.1.d Validation of the proposed optimality criteria

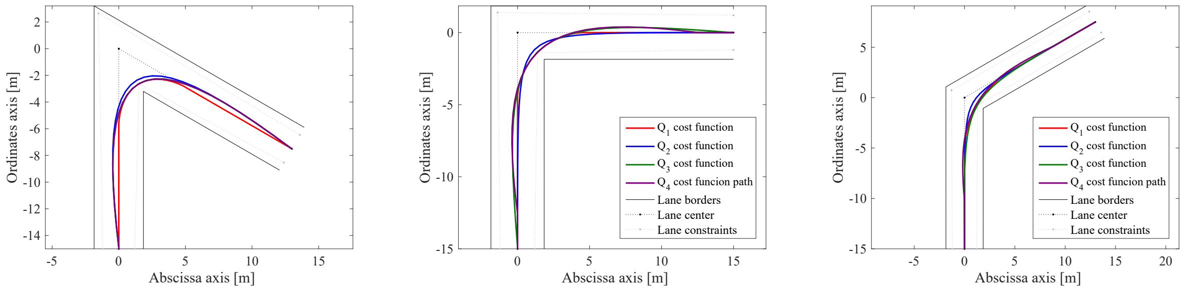
Several weights of the curvature and curvature derivative have been taken into account in the definition of the cost function, trying to give more importance either to the curvature or the curvature derivative. The results obtained with the several cost functions reflected that the best weighting was $w_{ki} = w_{k'i} = 1$.

Figure 3.8 shows the most significant subset of turn cases considered for comparing the cost functions described above. They try to represent the most common scenarios that can be found in urban environments. The angle of each turn is measured counter-clockwise in the internal side of the turn, i.e. the point where the curvature is maximum. That is a sharp turn (60°), a right-angle turn (90°), which can be considered the most common one, and a more opened turn (120°). For each turn scenario, the optimal curve planned for the path (Figure 3.8a), its curvature profile (Figure 3.8b), and its curvature derivative (Figure 3.8c) are represented.

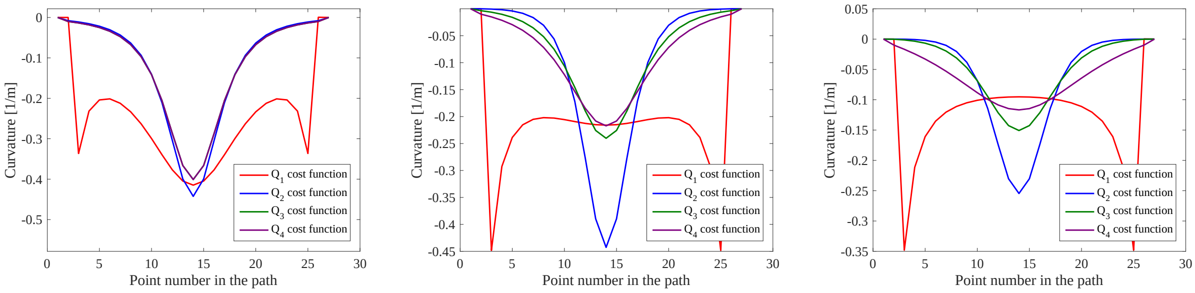
As stated before, by considering only either the curvature at the first, the midpoint and the last points of the curve (Q_1 cost function, in red), or its derivative (Q_2 cost function, in blue) is not enough for generating continuous curvature profiles with low maximum curvature. Additionally, minimizing both the curvature at the midpoint and the curvature change (curvature derivative) shows a significant improvement concerning smoothness (Q_3 cost function, in green), but assumes that the curves are symmetrical and the maximum curvature is in the midpoint, which is not the case in our algorithm. Thus, considering both curvature and its derivative at all the curve points constitutes a better approach, providing less curvy profiles (Q_4 cost function, in purple).

After presenting the optimality criteria based on evaluating the cost for generating the path on the single-turn scenario, the approach is compared with previous methods in the team [González and Pérez, 2013], [Pérez Rastelli et al., 2014]. As with the literature approaches, that cost function only considers minimizing the curvature in the three most significant points of the curve (first point, mid-point, and last point). The first approach used cubic curves where the control points are statically placed, i.e. considering a fixed distance with respect to the center of the turn. The second one makes a dynamic allocation of the control points, but only considers a short amount of positioning changes for the control points in real-time, resulting in a small set of different curves evaluated, thanks to the inclusion of the curvature derivative as an optimization parameter.

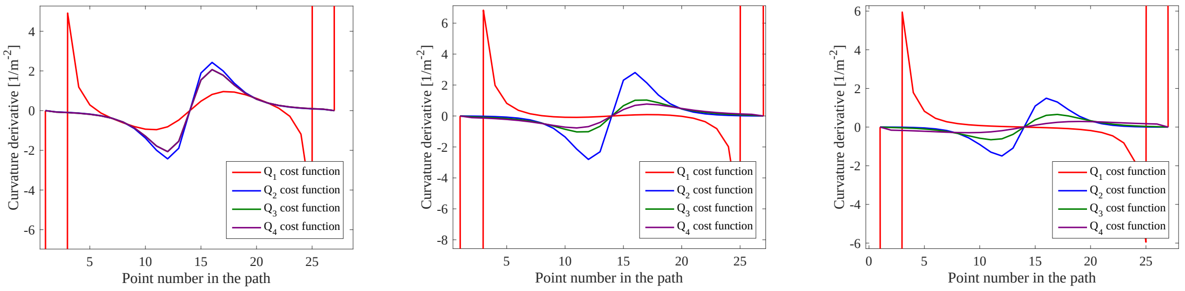
Figure 3.9 shows a comparison between the proposed algorithm using the cost functions Q_3 and Q_4, with respect to the algorithms on [González and Pérez, 2013] (static allocation of the control points) and [Pérez Rastelli et al., 2014] (dynamic allocation of control points but considering only few variations) using cubic Bézier curves. There, the algorithm which only considers a static allocation of the control points (in red) is not appropriate, since a bad allocation of the control



(a) Optimal curves for the different cost functions



(b) Curvature profiles for the different cost functions

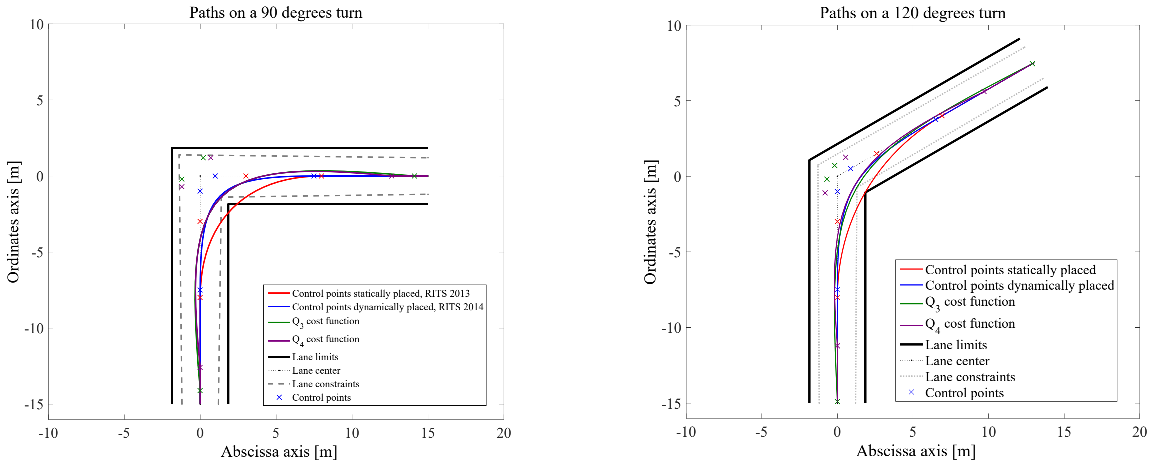


(c) Curvature derivative profiles for the different cost functions

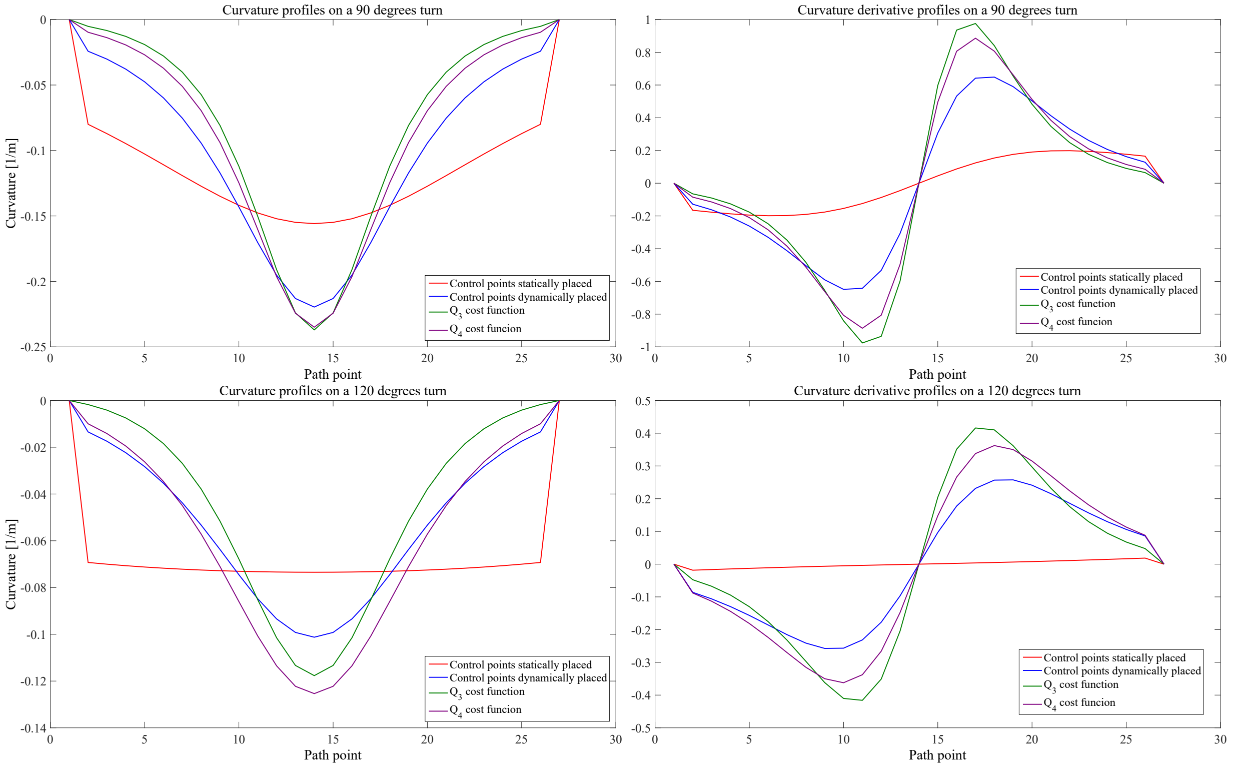
Figure 3.8 – Comparison of the different cost functions on 60°, 90° and 120° turns

points may generate a path not lying within the limits of the lane, invading the sidewalk, as can be noticed in Figure 3.9a. An improvement in terms of smoothness can be appreciated with respect to the dynamically allocated control points algorithm. Since the proposed algorithm with both cost functions Q_3 in green and Q_4 in purple present lower curvature profiles (Figure 3.9b), with a more continuous behavior than the algorithms in [González and Pérez, 2013] (in red) and in [Pérez Rastelli et al., 2014] (in blue) due to the lower curvature derivative profiles.

Furthermore, Figure 3.10 shows a more detailed set of experiments performed to validate the optimality approach comparing the proposed cost function with the curvature dependent method presented in [Pérez Rastelli et al., 2014]. There, four different turns are considered, namely 60°, 90°, 120° and 150° turns, which represent the most common cases can be found in urban roads. Here, quartic Bézier curves are used, since with cubic curves is harder to fulfill the curvature continuity constraint included in the algorithm, becoming quartic curves more convenient to ensure that the junction between consecutive curves is continuous. In Figures 3.10a the red line shows the path generated using the algorithm described in [Pérez Rastelli et al., 2014], while the garnet and dark blue lines show the paths tracked by the vehicle, respectively. Meanwhile, in Figures 3.10b the



(a) Optimal curves comparison between the proposed cost function and a curvature-dependent one

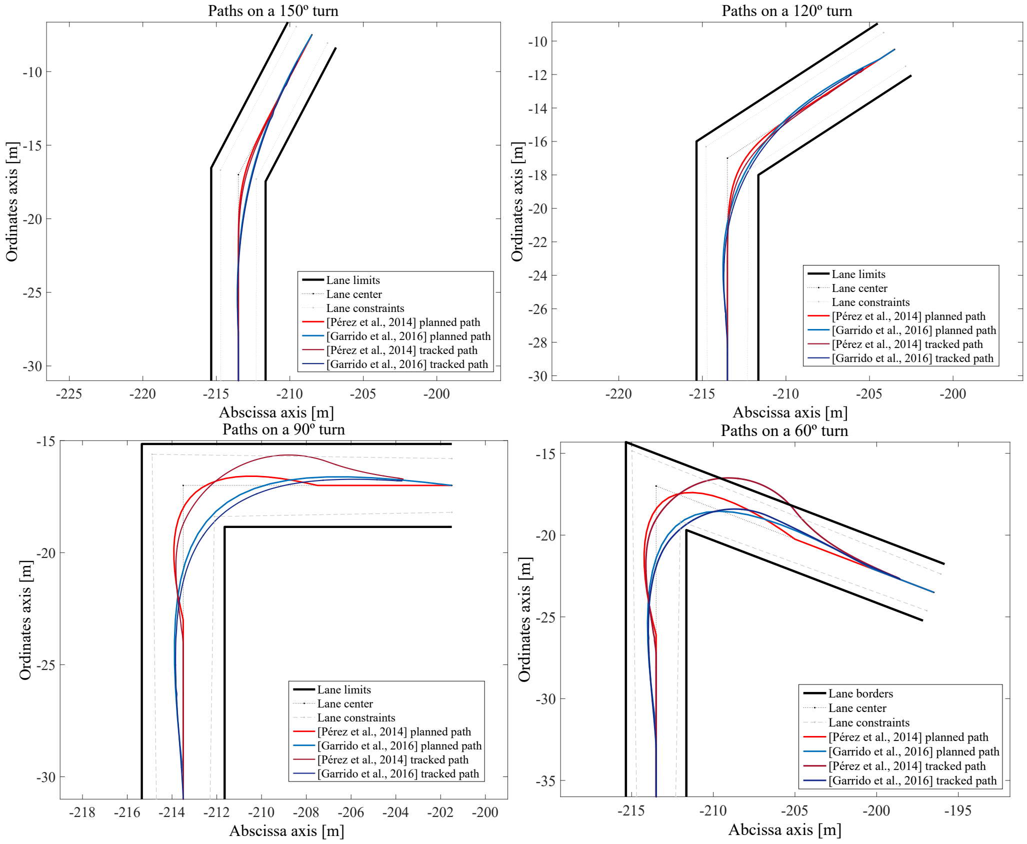


(b) Curvature and curvature derivative profiles

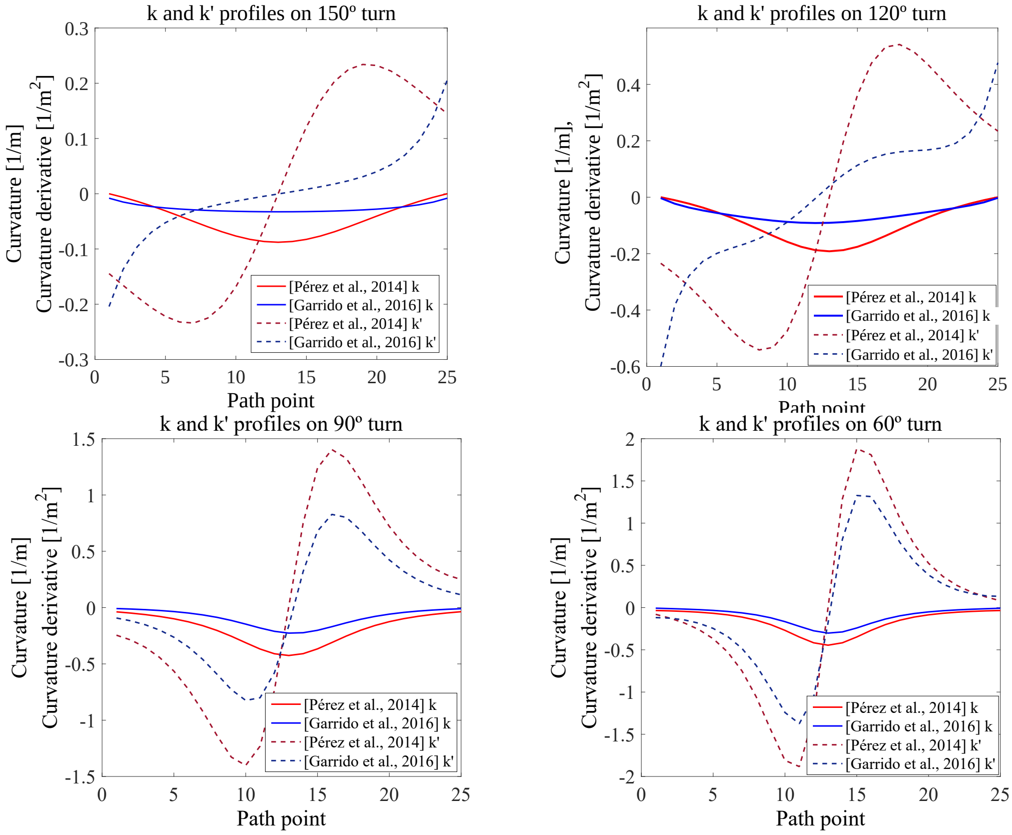
Figure 3.9 – Comparison of the proposed optimality criteria with respect to curvature-dependent approaches placing the control points both statically and dynamically

curvatures and curvature derivatives for both the [Pérez Rastelli et al., 2014] and the proposed approach [Garrido et al., 2016a] are represented respecting the same colors used in Figure 3.10a.

Table 3.2 summarizes the most relevant information of the prior experiments. There, a computationally less expensive behavior of the proposed algorithm is shown since the time to generate the curves with the proposed approach is lower. Considering that for each turn scenario the previous algorithm has around 2, 000 iterations and the proposed here has around 10, 400, 000 iterations, i.e. 5, 000 times more, the execution time has been reduced a 34% in the worst scenario. It can also be



(a) Optimal curves comparison between the proposed cost function and a curvature-dependent one



(b) Curvature and curvature derivative profiles

Figure 3.10 – Validation of the proposed optimality criteria with respect to a curvature-dependent approach

noticed an improvement on the curvature in all the scenarios. Regarding the maximum curvature, it improves from 32% for a 60° turn up to a 62% for a 150° turn. It can also be shown a reduction on the mean curvature difference. Furthermore, the derivative of the curvature presents the same behavior. These results reflect an improvement in the path continuity, mostly in the connection between curves, where the strength of the algorithm lies. In Figure 3.10b we can appreciate that the curvature and its derivative with the presented method (gray lines) have lower values than the obtained with the other method (black lines), improving the continuity and leading us to a smoother and more comfortable path planning approach.

Table 3.2 – Validation experiments of the optimality function

Turn	**Algorithm**	**Measurements**: time [ms], k [m^{-1}], k' [m^{-2}]				
		time	$\|\mu_k\|$	$\|k_{max}\|$	$\|\mu'_k\|$	$\|k'_{max}\|$
150°	Pérez et al., 2014	48	0.0466	0.0878	0.1738	0.2340
	Proposed approach	13	0.0259	0.0327	0.0560	0.2061
120°	Pérez et al., 2014	24	0.0925	0.1917	0.3759	0.5415
	Proposed approach	13	0.0583	0.0915	0.1936	0.5997
90°	Pérez et al., 2014	18	0.1893	0.4256	0.7504	1.4033
	Proposed approach	12	0.0909	0.2267	0.4247	0.8275
60°	Pérez et al., 2014	22	0.1637	0.4453	0.7815	1.8833
	Proposed approach	13	0.1020	0.3021	0.5709	1.3745

3.3.1.e Checking the optimality of the evaluated curves

After evaluating the optimality of each curve, the algorithm compares it with the current optimal curve and saves the curve configuration in case its cost is lower than the one of the current optimal curve (line 12, Algorithm 1). These parameters are the longitudinal and lateral displacement with respect to the center of the turn placed on the center of the lane, i.e. d_1 (longitudinal distance for P_0), d_2 (longitudinal distance for P_4), d_3 (longitudinal distance for P_1), $dLat_{P1}$ (lateral distance for P_1), d_4 (longitudinal distance for P_3), $dLat_{P3}$ (lateral distance for P_3), and $dLat_M$ (lateral distance for P_2).

This information is saved into the corresponding database in order to be loaded later in the real-time local planner to generate the path by interpolating the corresponding loaded curve for each turn (line 13, Algorithm 1).

The databases are generated from the inside out. This means that for a specific turn sharpness (with a defined turn angle α)) the other two indexes of the databases are changed from the center of the lane to the maximum distance considered, which is 40 meters as explained above. Thus, the algorithm starts to iterate changing the value of these two indexes defining the *arrival distance* and *exit distance* from the midpoint of the turn with the step change defined in the space discretization explained in Subsubsection 3.3.1.g. Thus, for instance, for a turn defined by a 90° angle, an arrival distance of 20 meters and an exit distance of 15 meters, the external control point defining the beginning of the curve can be at any point between the mid-point and the arrival distance. That is, it does not have to be necessarily located at the arrival distance of 20 meters from the mid-point of the turn. The algorithm starts to iterate from the turn mid-point with a distance equal to the step of variation. That way, since the algorithm has already iterated over the previous distances, it only has to evaluate the possible curves iterating the internal control points, comparing the evaluated curves with the already optimal one for the iterations with the external control points. It means that for this configuration (90°, 20 meters, 15 meters) the algorithm has already evaluated the curves iterating the external control points for 20 meters minus the step of variation and 15 meters minus the variation step. That way, assuming a variation step of 2 meters for the external control points, when starting to iterate for 20 meters of arrival distance, the algorithm has already computed the optimal curve for 18 meters. It implies that the current optimal solution to start to iterate could be any curve with an arrival distance less than 18 meters (for instance $d_1 = 10$ meters), where the external control point is placed.

The following XML code (Listing 3.1) shows an example of content belonging to the database of optimal curves generated in the pre-planning stage, and presented in A. Its structure is defined with the aforementioned indexes, which are: *angle* as the turn angle of *alpha* degrees, *segment1* as the arrival distance, and *segment2* as the exit distance, whose values are determined by the *length* parameter. For each specific turn scenario, i.e. for each angle *alpha*, distance of the first segment up to the midpoint of the turn $segment1$ and distance of the second segment up to the final way-point $segment2$, the database saves the parameters generating the optimal curve, which are the following: $d1_{seg1}$ and $d1_{seg2}$ are the longitudinal distances to place the external control points at both first and second segment, $d2_{seg1}$ and $d2_{seg2}$ are the distances to place the internal control points, respectively. Finally, $dLat_{P1}$ and $dLat_{P2}$ represent the lateral displacement of the control points with respect to the lane center for the internal control points in the first and second segments, respectively, whereas $dLat_{PMiddle1}$ represents the lateral displacement for the central control point. In addition, the cost of each curve is saved in the parameter *cost*.

```
<turn>
<angle alpha="30">
...
<angle alpha="90">
  <segment1 length="4">
    <segment2 Bezier_order="4" length="6" d1_seg1="4" d1_seg2="6"
        d2_seg1="2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="
        0" cost="26.682454851472151" />
    <segment2 Bezier_order="4" length="8" d1_seg1="4" d1_seg2="6"
        d2_seg1="2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="
        0" cost="26.682454851472151" />
     ...
  </segment1>
  <segment1 length="6">
    ...
</angle>
...
</turn>
```

Listing 3.1 – Database XML subset code

3.3.1.f Human-like driving behavior

One of the goals of the path planning approach is to provide a human-like driving style. Four database types have been defined with that purpose, as shown in Figure 3.11.

Table 3.3 explains the combination of all possible scenarios that can be found for interpolating curves later in the real-time planning stage, i.e. it summarizes the typical scenarios that can be

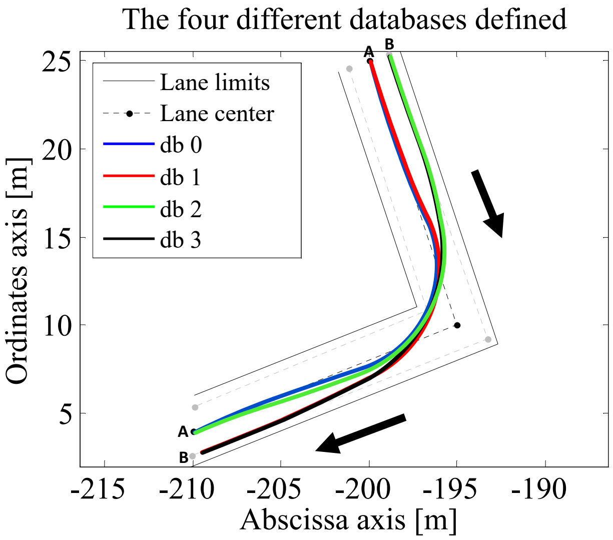


Figure 3.11 – Path databases for consecutive intersections scenarios

found to perform a turn depending on the direction of rotation of the previous and following curve. Thus, the scenarios where the initial or final position of the car in the curve are, either at the center of the lane or displaced towards the right border of the lane, are covered. The starting/ending points for generating the curve path can be either center in the lane (represented by A) or slightly moved to the border of the lane (represented by B).

This feature emulates the behavior of the human drivers, who use the entire width of the lane depending on the road layout. For instance, if consecutive curves are presented in the same direction, the database whose curves use the entire lane width will be loaded in the real-time phase for planning the curve for the upcoming turn, i.e. the ones whose ending points are displaced with respect to the lance center. Therefore, it allows the real-time local planner to load the curves that fit more with the road layout.

Table 3.3 – Paths databases according to the different directions of rotation (d.o.r)

Trajectory	**Starting point**	**Ending point**
Blue	A	A
Red	A	B
Green	B	A
Black	B	B

Bézier curves contribute to this human-like style thanks to the ease of manipulation of the curves, allowing to modify the initial and final position of the car on the lane thanks to the first and last control points. This permits to the whole trajectory meet both the road and vehicle constraints above in subsection 3.1.1, as well as an easy path modification to avoid obstacles. Apart from that, they present a low computational load since they are defined using the analytical method [Walton et al., 2003], where the curve derivatives are well-known in advance, which permits a fast trajectory generation [Han et al., 2010].

3.3.1.g Databases resolution

Thanks to the pre-planning stage the optimal curve for each turn scenario can be pre-computed. However, a trade-off between database resolution and database size is needed to find the minimum variation step for the three parameters defining the turn (angle, arrival distance and exit distance) that ensure generating paths that are continuous and smooth enough.

Obviously, the lower step of variation of the curve parameters, the higher the smoothness of the curves and the higher the resolution of the database. However, since the databases occupy a physical space into the disk, and they are read later on in the real-time planning stage, the bigger they are, the more time to access to them and thus the slower the algorithm it is. Thus, since the computational cost of evaluating in real-time a large enough set of curves would be high, a trade-off is needed.

Since both the sharpness of the turn and the available space considered to make a curve are limited and well-known, a discretization of the space can be done to define this variation step. There, the angle of the turn changes from a minimum value of 60°, which would represent the sharpest turn a vehicle could find when driving (according to the maximum steering of the vehicle, which normally is between 37 and 40 degrees), up to 180 degrees, which would represent a straight-line segment. Once the real-time planner receives the way-points of the global path, both the angles of the turns and the distances between way-points are computed.

The goal is generating the curves that fit more with the real scenario. The angle of the turn is rounded to the lower value applying the variation step, reproducing the worst case. For instance, if an angle of 93.5° is measured, considering a variation step of 5°, the turn is approximated to a 90° turn and not to a 95° turn, i.e. to the closest sharp turn.

First, the different curves have been evaluated considering a variation step of five degrees for the sharpness of the curves. After testing with different configurations, ranging from one degree up to ten degrees of variation step, the chosen five degrees was the one providing the best relationship between path quality and computational expense.

Second, for both the distances to the external control points (arriving and exit distances) as for the internal control points, a variation step must be chosen. In the same way as for the angle, an approximation is needed to reproduce the turn scenario as close as possible to the reality, considering the worst case. Thus, the values of the distances are approximated to the smaller distance, since it is more realistic consider having less free space than having more free space to perform the curve.

In order to search the best variation step for the distances, the same four characteristic turns already presented were considered: 60°, 90°, 120° and 150°. For each scenario, the pre-planning algorithm generated the optimal curve changing the values ranging from one meter (as the minimum variation step) up to five meters (as the maximum variation step).

We tested the different databases containing the pre-planned trajectories for the isolated curves with the model of the vehicles in Matlab-Simulink, as shown in Figure 3.12. This model has been set-up with the Cybercars configuration and is presented in the following subsection (Section 3.3.1.h). The planning module has been introduced to analyze the variation of the both the lateral and the angular error, as well as the variation of the curvature, with respect to the most accurate configuration, being the one where the distances are changed every meter (lower value considered as variation step).

If a significant error is detected when augmenting the variation step, the last variation step is considered to be the best one. Table 3.4 shows the comparison of three databases considering different steps of variation of the external control points (both for the arrival distance and the exit distance).

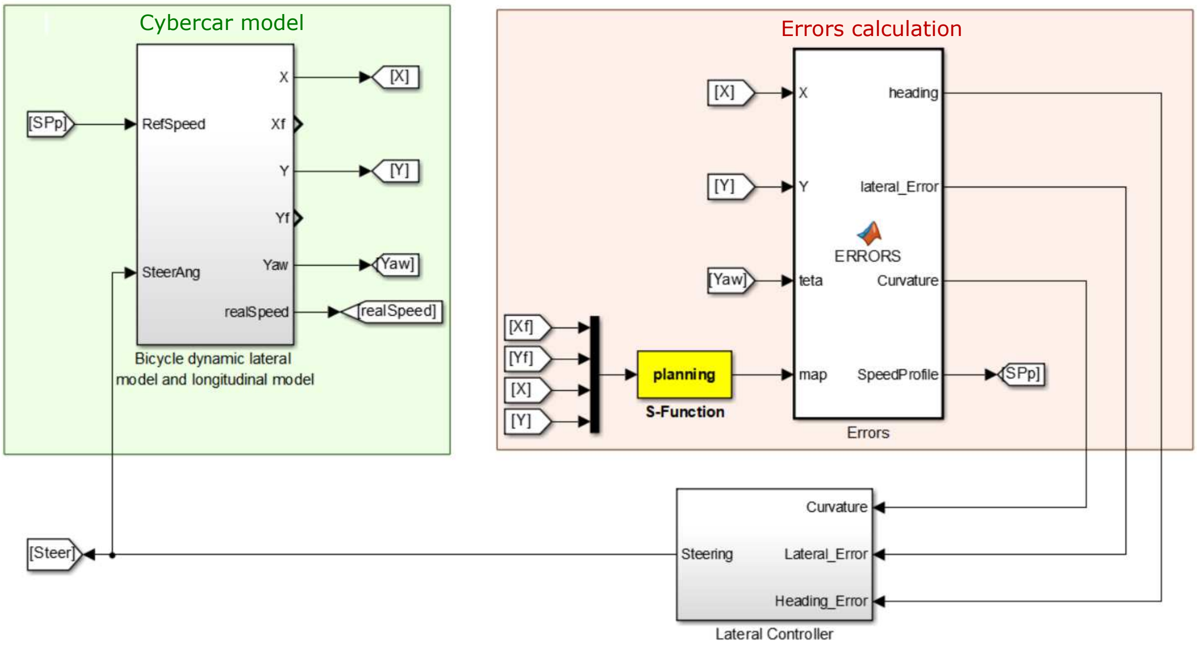

Cybercar model
[SPp]
RefSpeed
SteerAng
X
Xf
Y
Yf
Yaw
realSpeed
[X]
[Y]
[Yaw]
[realSpeed]
Bicycle dynamic lateral model and longitudinal model
Errors calculation
[X]
[Y]
[Yaw]
X
Y
teta
map
heading
lateral_Error
Curvature
SpeedProfile
ERRORS
[SPp]
Errors
[Xf]
[Yf]
[X]
[Y]
planning
S-Function
[Steer]
Steering
Curvature
Lateral_Error
Heading_Error
Lateral Controller

Figure 3.12 – Matlab-Simulink model for comparing the steps of variation for the databases generated in the pre-planning stage

Table 3.4 – Comparison of the steps of variation of the curve parameters on the different databases

Turn		**Lateral error [m]**		**Heading angle [deg]**		**Curvature [m^{-1}]**	
Angle	DB	mean(abs)	var (%)	mean(abs)	variation (%)	mean(abs)	variation(%)
60°	1m 1m	0.0530	0.00	8.1732	0.00	0.0581	0.00
	2m 1m	0.0575	8.49	8.4473	3.35	0.0607	4.48
	5m 1m	0.0586	10.57	8.4029	2.81	0.0587	1.03
90°	1m 1m	0.0592	0.00	9.9158	0.00	0.0676	0.00
	2m 1m	0.0510	-13.85	10.2236	3.08	0.0766	13.31
	5m 1m	0.0594	0.34	10.1939	2.78	0.0687	1.63
120°	1m 1m	0.0281	0.00	6.2424	0.00	0.0441	0.00
	2m 1m	0.0342	21.71	5.8505	-6.28	0.0409	-7.26
	5m 1m	0.0294	4.63	5.6943	-8.78	0.0419	-4.99
150°	1m 1m	0.0087	0.00	2.8803	0.00	0.0199	0.00
	2m 1m	0.0211	142.53	2.3402	-18.75	0.0174	-12.56
	5m 1m	0.0153	75.86	2.9614	2.82	0.0226	13.57

The databases are compared with respect to the more discrete one, i.e. the one where both external and internal control points are moved every meter, called "db 1m 1m". Although the difference of the mean value of the control variables (lateral and heading error, and curvature) is close between second and third databases, presenting both a difference of about 15 %, practical results show that in some cases there is no solution for the less discrete databases, which has not been reflected in the database. The intermediate database (db 2m 1m) has been the one chosen for the experiments since the discretization level is higher and with a good number of iterations (about 6.5 million iterations in the pre-planning stage, against 153 millions of the more discretized one and almost 4 millions of the most discretized). Considering the smallest database would not provide much more accuracy (a reduction of less than 10 % of improvement in the variables), but would mean a bigger increment in the time to access the databases and consequently in the computational cost of the algorithm.

3.3.1.h Vehicle model

The vehicle model considered for validating the pre-planning stage in simulation on Matlab-Simulink is a dynamic bicycle model [Rajamani, 2011] since this model is appropriate for vehicles operating at low-speeds.

Although the control of the vehicle is out of the scope in this thesis, some basic information about the algorithm used in the team architecture is provided. Equation 3.15 describes the motion of the vehicle in the Cartesian plane, according to the kinematic model of the vehicle, represented in Figure 3.13.

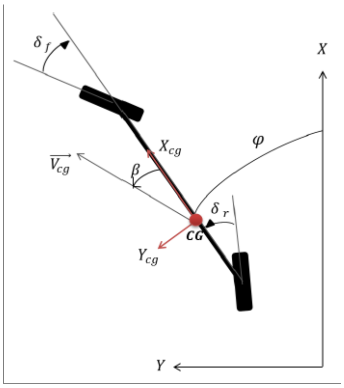


Figure 3.13 – Bicycle model (re-make own figure with described nomenclature)

Vehicle dynamics terminology from SAE [Society of Automotive Engineers. Vehicle Dynamics Committee, 1978] is used as the convention for the nomenclature. For describing this movement, three coordinates are needed: x and y for the position and ψ for the orientation of the vehicle. Additionally, L represents the wheelbase, ψ is the orientation angle with respect to the global frame, and δ is the steering angle.

$$\begin{aligned} x'(t) &= \frac{dx(t)}{dt} = v_x cos(\psi(t)) \\ y'(t) &= \frac{dy(t)}{dt} = v_x sin(\psi(t)) \\ \psi &= \frac{d\psi(t)}{dt} = \frac{v_x}{L} tan(\delta(t)) \end{aligned} \tag{3.15}$$

Since these vehicles operate at low speed, both the slip and the forces transferred between wheels of the same axle can be neglected. It leads to the assumption that the maximum curvature feasible for the vehicle is when the steering angle is maximum.

3.3.2 Real-time planning stage

As shown in Figure 3.14, this phase can be divided into three fundamental tasks. First, an analysis of the turns and straights of the itinerary through which the vehicle will pass are analyzed, in terms of available space and sharpness. Second, it makes a real-time evaluation to find the optimal location of the junction point between curves, evaluating the turns in pairs by loading from the databases the curves that fit more the road layout. Third, for each turn, once the junction point is found, the algorithm generates the optimal curve for the upcoming turn with the parameters loaded from the database. In addition, it keeps the optimal curve parameters for the following turn, which has been planned in advance. Finally, after repeating the process for all the turns on the itinerary, the full static path is transmitted to the dynamic path component, which modifies it in the presence of obstacles on the path and sends it to the controller.

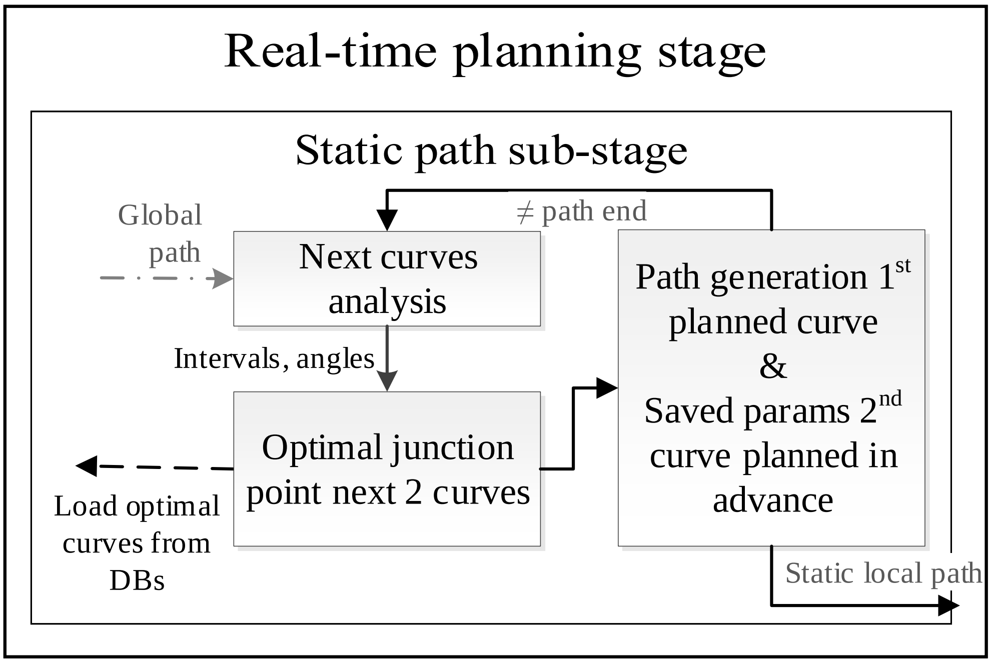

Real-time planning stage
Static path sub-stage
Global path
≠ path end
Next curves analysis
Path generation 1st planned curve & Saved params 2nd curve planned in advance
Intervals, angles
Optimal junction point next 2 curves
Load optimal curves from DBs
Static local path

Figure 3.14 – Real-time static local path planning flowchart

Urban environments usually consist of a series of road elements such as turns, intersections, and roundabouts, which can be found very consecutively and with limited space between one element and the following. Since the way-points defining the path following itinerary are received from the global planner, the algorithm can easily compute the distance between these elements, as well as the sharpness of the turns needed to pass through them. Thus, for every single turn, the real-time algorithm plans the curve that fits more by considering a pair of turns: the upcoming and the following one. By planning two curves concurrently, we try to distribute the Bézier curves on the road improving the comfort by reducing the cost function for the pairs of curves.

Thanks to the databases generated in the pre-planning stage, the real-time algorithm only has to determine which is the best junction point between them. Thanks to the pre-planning stage, the curves generated for each turn by loading its parameters from the databases are already the optimal ones. The only operation done in real-time is the evaluation of the junction point of the pair of consecutive curves (both upcoming and next one). This junction point is moved on the shared segment, and the cost of both curves is computed accordingly. Then, it analyzes if the evaluated pair is the optimal one by comparing with the cost of the current optimal one.

Algorithms 2 and 3 describe the procedure the real-time planning follows to generate the static path, where the first one represents the main algorithm, and the second one details the function for the path generation, making use of the mentioned pre-planning stage.

Algorithm 2 Real-time local path planning for static areas

```
 1: Init: load databases, read vehicle and road properties
 2: Read: veh. position and orientation, itinerary way-points
 3: if first time then
 4:     Locate vehicle on the itinerary
 5:     Add first itinerary point to the path list
 6:     Generate a $1^{st}$ path until reaching the horizon distance
 7: Locate vehicle on the path and remove past path points
 8: Calculate distance to travel from vehicle position to the next path point
 9: Add vehicle position point into the path list front
10: if distance to travel $<$ horizon distance then
11:     Generate path until horizon is reached
12: Send local path through a buffer to the controller
```

First of all (line 1, Algorithm 2), the algorithm has an initialization phase where the databases containing the optimized curves for the single-turn scenarios are loaded into main memory. In addition, the algorithm reads the information concerning both vehicle and road constraints, i.e. the length and width of the vehicle, its maximum steering angle and the width of the lane, as well as the number of lanes of the road.

Once the initialization phase has finished, the vehicle position and orientation are read from the onboard sensors or from the simulation component that provides the vehicle state at every time interval. Furthermore, the component receives from the global path the way-points defining the itinerary (line 2, Algorithm 2). This will let the algorithm compute the physical borders of the lane geometrically, as explained in previous sections.

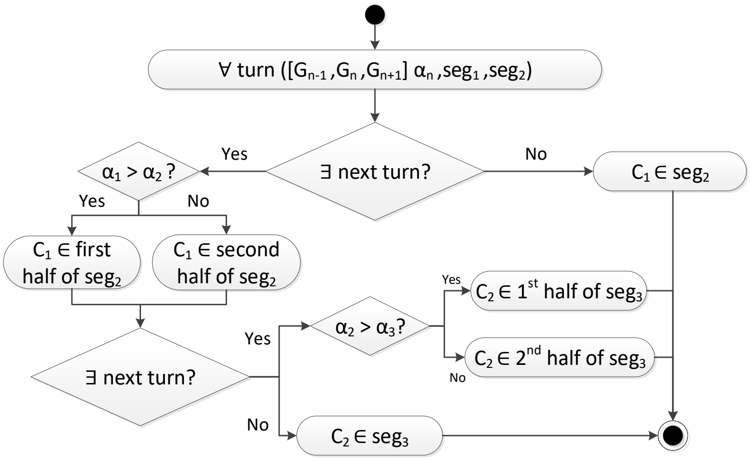
After receiving all the needed input data, the algorithm first must localize the vehicle on the itinerary (line 4, Algorithm 2). If the path following scenario has just started, the vehicle position at the beginning is added as the first point of the trajectory list containing the points of the local path. Then, a first trajectory until reaching the horizon distance can be computed (line 6, Algorithm 2). This path is generated using Algorithm 3, that will be further explained. If the vehicle has already started the path following maneuver, after localizing the vehicle on the global path and the local path, the past points are removed from the lists, i.e. the points describing positions where the vehicle has already passed through (line 7, Algorithm 2). The distance to travel from the vehicle position to the next path point is computed (line 8, Algorithm 2). If this distance less or equal than the planning horizon, then the algorithm computes the path until reaching the horizon (lines 10 and 11, Algorithm 2). Finally, the planned local path is sent to the controller through a buffer (line 12, Algorithm 2).

The path generation process is presented in Algorithm 3, and it is explained below.

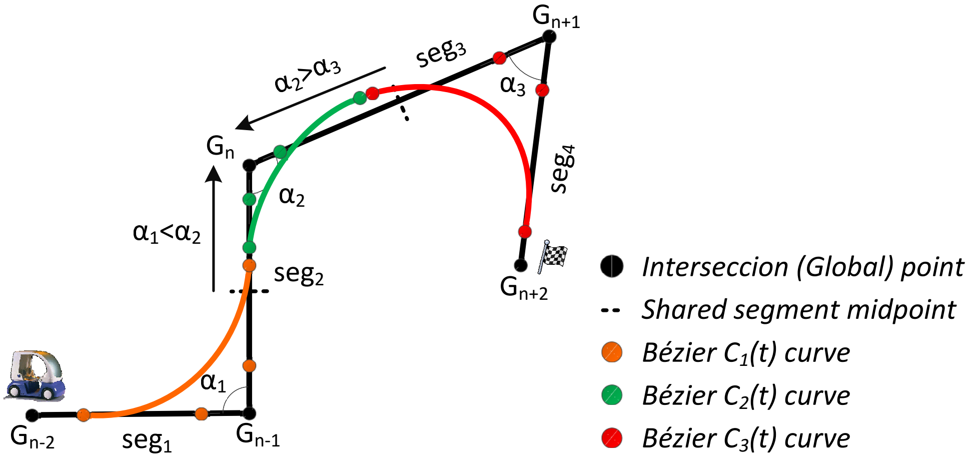
Algorithm 3 Function: Path generation

```
 1: for i=currentWayPoint; i<waypoints.length; i++ do
 2:     if ∃ i+3, i+4 (∃ $2^{nd}$ & $3^{rd}$ turns)) then
 3:         Define turns [i, i+2], [i+1, i+3], [i+2, i+4]
 4:         Get: distances between turn points, turn angles
 5:         Evaluate type: single, first, middle or last turn
 6:     Compute geometrically road limits and constraints
 7:     Evaluate way-point type: left or right turn, straight
 8:     Normalize values to the DBs discretization step
 9:     Evaluate curves junction location interval
10:     if i = pathBegin & i+2 ≠ path end (∃ a $1^{st}$ turn) then
11:         Iterate junction point between $1^{st}$ & $2^{nd}$ turns ($m$)
12:         Analyze direction of rotation $1^{st}$ & $2^{nd}$ curves
13:         to determine DB to load for $1^{st}$ curve
14:         Iterate junction point between $2^{nd}$ & $3^{rd}$ turns ($n$)
15:         Analyze direction of rotation $2^{nd}$ & $3^{rd}$ curves
16:         Load the optimal curve info for $2^{nd}$ curve
17:         if curve1.cost + curve2.cost < bestCost then
18:             bestCurve1 = curve1; bestCurve2 = curve2;
19:             best%SegmentCurve1 = $m$
20:             best%SegmentCurve2 = $n$
21:         LastCurveSegment2% = best%SegmentCurve1
22:         LastCurveSegment3% = best%SegmentCurve2
23:     else if i≠pathBegin & i+2≠path end (mid-turn) then
24:         curve1 = bestCurve2 (already computed $1^{st}$ curve)
25:         Iterate junction point $2^{nd}$ [& $3^{rd}$ turns] ($n$)
26:         Analyze direction of rotation $2^{nd}$ [& $3^{rd}$ curves]
27:         Load the optimal curve info for $2^{nd}$ curve
28:         if curve1.cost + curve2.cost < bestCost then
29:             bestCurve1 = curve1; bestCurve2 = curve2;
30:             best%SegCurve1 = LastCurveSeg3%
31:             best%SegCurve2 = $n$
32:         LastCurveSeg2% = best%SegCurve1
33:         LastCurveSeg3% = best%SegCurve2
34:     else // i+2=path end (last curve)
35:         consider whole length of segment2
36:         repeat process of line 23 only for the last curve
37:     Compute control points location from curve's info
38:     if curve was planned (control points generated) then
39:         Compute Bézier curve points from control points
```

First of all, the algorithm iterates over the way-points of the itinerary analyzing which kind of scenario they represent (turn, intersection, roundabout, etc). In this work, all of them are considered as turns. Then, since the way-points defining the itinerary are known in advance, the algorithm can profit from that information to optimize two curves in parallel. Since the urban scenarios may present consecutive turns in a short period, by planning the path considering an extended horizon of two curves allows the algorithm to generate smoother paths by considering the direction of rotation of the curves on the road.


∀ turn ([G$_{n-1}$,G$_n$,G$_{n+1}$] α$_n$,seg$_1$,seg$_2$)
∃ next turn?
Yes
No
C$_1$ ∈ seg$_2$
α$_1$ > α$_2$?
Yes
No
C$_1$ ∈ first half of seg$_2$
C$_1$ ∈ second half of seg$_2$
∃ next turn?
Yes
α$_2$ > α$_3$?
Yes
C$_2$ ∈ 1st half of seg$_3$
No
C$_2$ ∈ 2nd half of seg$_3$
No
C$_2$ ∈ seg$_3$

(a) Flowchart for the segmentation process of the consecutive turns shared segment to search the optimal junction point between curves


G$_{n+1}$
seg$_3$
α$_2$>α$_3$
α$_3$
G$_n$
seg$_4$
α$_2$
α$_1$<α$_2$
seg$_2$
G$_{n+2}$
α$_1$
G$_{n-2}$
seg$_1$
G$_{n-1}$
Interseccion (Global) point
Shared segment midpoint
Bézier C$_1$(t) curve
Bézier C$_2$(t) curve
Bézier C$_3$(t) curve

(b) Explanatory example of an adjacent consecutive turns scenario for the segmentation process strategy

Figure 3.15 – Segmentation process of the consecutive turns shared segment to search the optimal junction point between curves: flowchart (a) and use-case example (b)

Thus, a process to distribute the curves on the space is needed. Flowchart of Figure 3.15a describes the process to segment the shared space between curves in order to evaluate the position to place the junction point between them. Meanwhile, Figure 3.15b represents a scenario consisting of three consecutive turns. This Figure is used as support to explain the segmentation process. There, the three turns are named by their central way-points, namely G_{n-1}, G_n and G_{n+1}. The corresponding Bézier curves for the three turns are C_1 in orange for the first, C_2 in green for the second, and C_3 in red for the third turn. There, since the first turn is more narrowed than the second one, the junction point is searched in the half of the shared segment closer to the second turn (G_n). Similarly, the second turn is more opened than the third one, so the junction point between the curves is searched in the half of segment closer to G_n. That way, a natural driving style is searched for the system, where it tries to compensate the sharpness of the curves more difficult to track by using the whole shared space between curves. Additionally, the concavity changes are considered. For instance, since first is a curve to the left and second one to the right, this concavity change makes both curves join in the center of the lane, reducing that way the curvature profile. However, second and third turns are both to the right. Thus, no concavity change is produced, and the system get benefit from the developed databases to make both curves join using the available width of the lane, reducing that way the curvature and the curvature changes.

If in addition to the upcoming turn there exist a 2^{nd} and a 3^{rd} one (line 2, Algorithm 3), the algorithm takes them into account by:

- computing the distances between the way-points defining the center of the turns.
- getting the angles of the turns and the direction of rotation according to their concavity changes.
- evaluating the type of turn in the itinerary (first turn, last turn or a mid-turn).

In addition, road limits are obtained geometrically from the global path, whose points define the center of the lane (line 6, Algorithm 3).

Then, the direction of rotation of the upcoming turn is analyzed, checking if it is a left turn, a right turn or a straight segment. This further lets the algorithm place on the correct side the control points, as well as consider the correct internal lane constraint according to the concavity change of the curve.

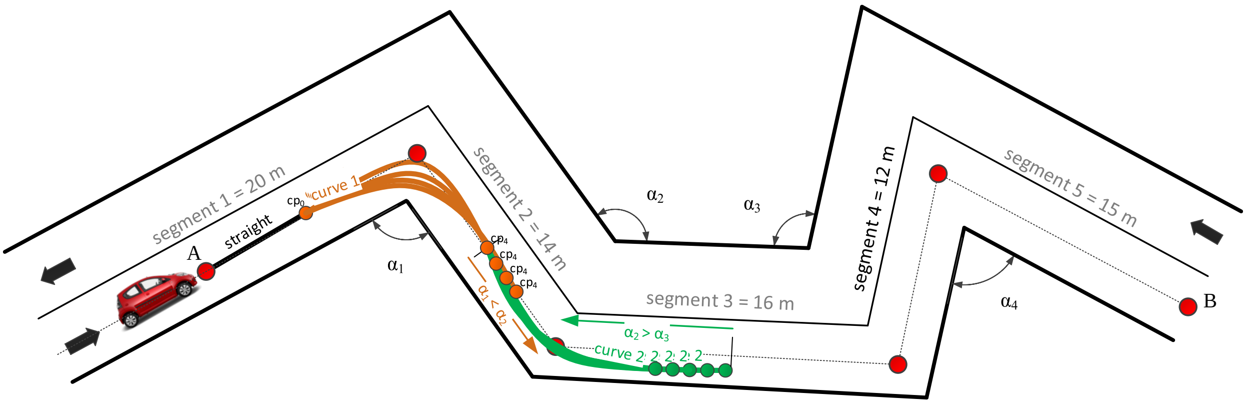
Since the databases containing the already optimized curves for the single turn scenarios have been discretized as explained in Section 3.3.1.g, the parameters that describe each turn are normalized, namely the angle of the turn, the arrival distance and the exit distance to the mid-point of the turn. Those parameters are normalized to the lower value to bring the scenario closer to the most realistic case, loading the nearest closed curve from the database. For instance, if the geometrically measured angle for the upcoming turn is 92^{o}, assuming a variation step of 5^{o} between turns as the database discretization, the optimal curve for a 90^{o} turn will be considered. In the same way, the arrival and exit distance are normalized according to the variation step for both segments in the database, which is not the same than the one defining its angle.

Since the algorithm aims to provide a human-like driving style, a segmentation of the shared segment between curves is proposed considering both the relationship between the turn sharpness and the physical space for performing the curves.

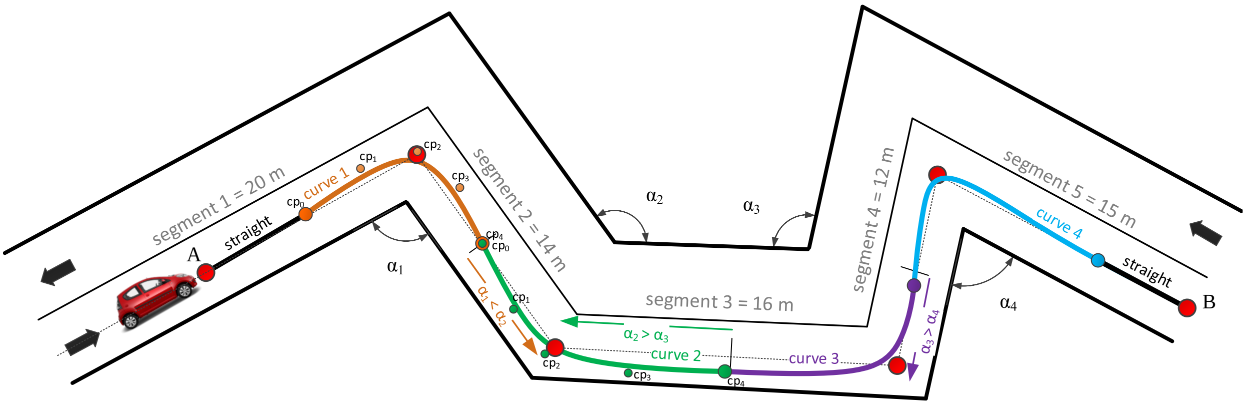
First of all, the position of the turn in the path is analyzed, verifying if the upcoming turn is either the first one of the path, the last one or an intermediate one, considering the three cases separately for the segmentation process.

Thus, the algorithm benefits the turn more difficult to perform, i.e. the junction point between curves is placed in the half of segment closer to the softer turn. Then, if the 1^{st} turn to perform (upcoming turn) is softer than the 2^{nd} one (next one), the junction between both curves is moved in the second half of the shared segment. On the contrary, the junction point is moved along the first half of the shared segment. In order to plan in advance the next curve (2^{nd} curve), an analysis of the relation between the 2^{nd} and the 3^{rd} turns is done similarly. Therefore, if the 2^{nd} turn is sharper than the 3^{rd} one, the junction point for the second curve will be placed on the 2^{nd} half of the shared segment, otherwise in the first one.

Once the analysis to place the junction point between curves has been done, the algorithm starts to iterate the position of the junction point between 1^{st} and 2^{nd} turns. Since the most logical situation would be starting to iterate from the middle of the shared segment, for the exit distance of the 1^{st} curve the algorithm first evaluates the curves with an exit distance of a 50% of the actual segment length. This exit distance will take values from the half of the segment until the beginning (or the end) of the segment, depending on the case, with a variable step of change depending on the chosen discretization. In this case, a step of 5% has been used.


segment 1 = 20 m
A
straight
cp₀
curve 1
α₁
segment 2 = 14 m
cp₄
cp₄
cp₄
cp₄
α₁ < α₂
α₂
α₂ > α₃
curve 2 2 2 2 2
segment 3 = 16 m
α₃
segment 4 = 12 m
α₄
segment 5 = 15 m
B

(a) The algorithm searches the best location for the junction point between curves in real-time


segment 1 = 20 m
A
straight
cp₀
curve 1
cp₁
cp₂
cp₃
α₁
segment 2 = 14 m
cp₄
cp₀
cp₁
α₁ < α₂
cp₂
α₂
α₂ > α₃
curve 2
cp₃
cp₄
curve 3
segment 3 = 16 m
α₃
segment 4 = 12 m
α₃ > α₄
α₄
segment 5 = 15 m
curve 4
straight
B

(b) Planned path after repeating the process for every curve where the best junction points are maintained

Figure 3.16 – Real-time path planning process: from loading the curves from the databases to get the whole path

Figure 3.16 shows a possible urban environment, consisting of a two-lane road where some consecutive turns can be found.

In the upper sub-figure, the red points represent the way-points defining the itinerary for the path-following scenario, received from the global planner. The point A shows the departure point and the point B the destination. Then, the algorithm computes the angles of the turns that will be found in the itinerary (α_1, α_2, α_3 and α_4), and the distances between them (segment$_1$, segment$_2$, segment$_3$ and segment$_4$). The upper sub-figure (3.16a) shows the process to find the best junction point between the turns α_1 and α_2 considering the sharpness of both. In that case, as α_1 is sharper, the algorithm searches the junction in the half of the segment$_2$ which is closer to α_2, giving that way more space to the curve which is more difficult to track. In addition, the direction of rotation of the curves is considered to decide whether to start and end the curve at the center of the lane or close to the lane boundary. Here, as α_1 is the first one the curve starts at the center of the lane. Besides, as α_1 is a right turn and α_2 is a left turn, the junction point between them is placed at the center of the lane, due to the previously described smoothness criteria. Therefore, the algorithm generates the optimal pair of curves for both turns by evaluating the curves loading them from the databases for each position of the junction point. The location where the sum of the cost for both curves is minimized is considered the best junction, and the curve parameters for both turn α_1 and α_2 are saved. In addition, the curve points of the first planned curve are added to the planned path in order to be tracked. The process is repeated for the following turns, getting the whole static path as depicted in the below sub-figure (3.16b). In the following iterations, the difference lies in the upcoming turn, which has already been planned and the algorithm only has to plan the following one, in case the first one remains feasible.

Once the upcoming curve and the following one are planned, the cost of both curves has been computed, and then an evaluation is done to verify if that configuration is the optimal one, i.e. if the sum of the cost of both curves is less than the cost of the optimal configuration. Then, the percentage of the best percentage for the second curve is saved.

Finally, after the iteration is saved, the optimal curve to perform the upcoming turn has already been planned, it can be generated loading it from the proper database with the control points location as indexes of the databases.

3.4 Conclusions

This chapter has presented the path planning algorithm for static environments. It consists of two stages: (i) pre-planning stage, where the optimal curves for any turn scenario are generated considering the static information of both the kinematics of the vehicle and the road layout. (ii) real-time planning stage, where a continuous path is built joining the optimal pre-planned curve for each turn, considering the actual road layout. This two-staged local planner provides a human-like driving style thanks to the following features:

- It benefits from the pre-planning stage [Garrido et al., 2016a] to generate smoother paths, where only the junction point between the optimal curves is evaluated in real-time
- An extended planning horizon of two consecutive curves is considered, optimizing two curves concurrently (the second one in advance), thanks to the knowledge of the digital map provided by the global planner [Garrido et al., 2016b].

The goal of this stage is to generate a path that adapts the more to the environment, with the least curvature changes in order to ease the path tracking searching comfortable travel for the passengers. Thus, this static path is sent to the dynamic local planner, which modifies it according to the dynamic changes on the scene. This dynamic planner is presented in the following chapter.

Chapitre 4

Planification des trajectoires pour des environements dynamiques

Below is a French summary of the following chapter "Path planning in dynamic environments".

Ce chapitre présente la démarche de planification illustrée à la Figure 4.1 pour adapter en temps réel le chemin statique généré en fonction des événements dynamiques survenant dans les environnements urbains, en modifiant le chemin en maintenant la continuité et le confort. Tout d'abord, Section 4.1 présente un bref aperçu des différentes stratégies permettant de surmonter les obstacles rencontrés dans la planification des parcours. Deuxièmement, le problème est formulé dans la section 4.2, où l'approche proposée est présentée. Il est basé sur une génération de route virtuelle, présentée dans la section 4.3, où un corridor est généré en temps réel pour traiter le problème dynamique en tant que problème statique (sous-section 4.3.2). combinée avec une discrétisation de l'espace basée sur une grille (sous-section 4.3.1). Enfin, quelques conclusions sont tirées de la section 4.4.

Chapter 4

Path planning in dynamic environments

Chapter 3 has shown the path planning strategy for static environments, where a continuous path is generated based on a two-stage approach, 1) combining a pre-planning stage, where the optimal curve for every single turn is precomputed; and 2) a real-time stage that evaluates the junction of the optimal curves loaded from the pre-planning stage by considering a human-like driving behavior and adjusting them the most to the road layout.

In order to make a path planner able to deal with dynamic environments, there are two main aspects to be considered: 1) Urban environments are complex scenarios. The interaction with the different road users causes challenging situations where the scene changes dynamically in short time intervals, requiring a fast response of the system being able to adapt the path avoiding the obstacles in a safe way. 2) Despite these difficulties, both vehicles and rest of VRU can be classified according to their dimensions which are well-known. These characteristics physically limit their speed and therefore their position. This assumption let us consider the path planning system developed in Chapter 3 as the basis to optimize the dynamic environment by modifying the path to address the dynamic scenarios as static.

This chapter presents the planning approach in Figure 4.1 to adapt in real-time the already generated static path according to the dynamic events that arise on urban environments, being able to modify the path maintaining the continuity and the comfort. First, Section 4.1 presents a short review about different strategies to deal with obstacles in path planning. Second, the problem is formulated in Section 4.2, where the proposed approach is presented. It is based on a virtual road generation, presented in Section 4.3, where a corridor is generated in real-time to treat the dynamic problem as a static problem (Subsection 4.3.2) combined with a grid-based discretization of the space (Subsection 4.3.1). Finally, some conclusions are drawn in Section 4.4.

4.1 Overview of dynamic path planning strategies

The problem of navigating through dynamic environments was first tackled on robotics, where mobile robots generate collision-free paths, avoiding any possible obstacle or physical barrier that may prevent the robot from reaching the destination point.

The path planning strategies for robots used in the literature range from grid-based A* algorithms, roadmaps built with Voronoi diagrams or visibility graphs, cell decomposition and artificial potential fields [Willms and Yang, 2006]. These algorithms are mainly used in static environments as global planning methods since they are computationally expensive for complex environments.

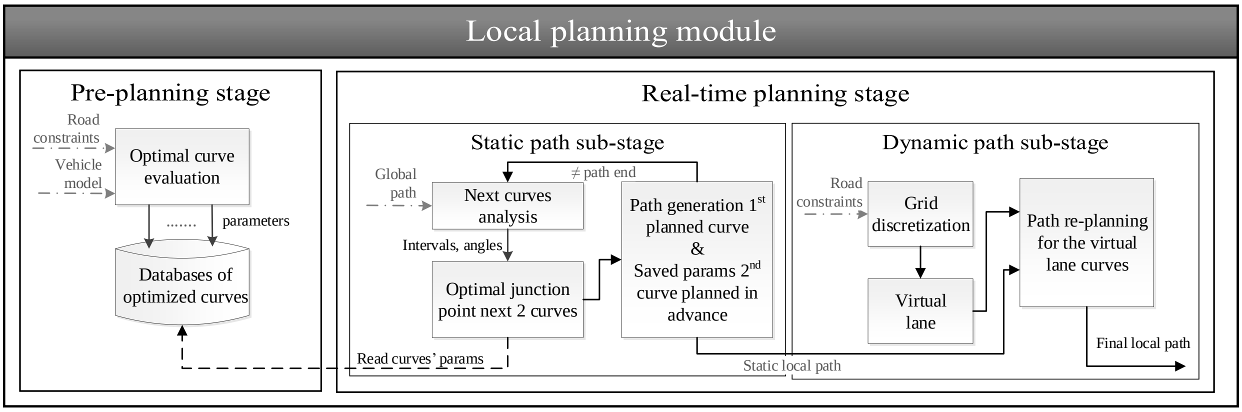

Local planning module
Pre-planning stage
Road constraints
Vehicle model
Optimal curve evaluation
....
parameters
Databases of optimized curves
Real-time planning stage
Static path sub-stage
Global path
Next curves analysis
≠ path end
Intervals, angles
Optimal junction point next 2 curves
Path generation 1st planned curve & Saved params 2nd curve planned in advance
Read curves' params
Dynamic path sub-stage
Road constraints
Grid discretization
Virtual lane
Path re-planning for the virtual lane curves
Static local path
Final local path

Figure 4.1 – Path planning system architecture

Additionally, they are not convenient for structured environments such as urban or peri-urban roads, where they can stuck in a local minima, for example in U-turns [Willms and Yang, 2006]. Lately, techniques based on artificial intelligence have been introduced, for example, based on neural networks or reinforcement learning based models. Sampling-based approaches such as RRT* have been recently applied to perform the path re-planning in the presence of random and unpredictable obstacles in the path. This algorithm is suitable for robotics since it provides a quick solution, but it is not optimal since the path generated is not continuous unless some local planning technique is applied [Connell and La, 2017]. In addition, the behavior of other road participants can be modeled more accurately than the behavior of moving robots or other objects in a non-structured space.

Generating collision-free trajectories on dynamic scenarios for non-holonomic vehicles involves complex decision making including the performance of obstacle avoidance maneuvers, if the obstacle is not moving, or overtaking maneuvers if the vehicle is traveling slower in the same lane. Unlike solutions for static environments, not only the information concerning the ego-vehicle is considered, but also that of the obstacles surrounding it.

Overtaking maneuvers have been treated in the literature as three-phase maneuvers, consisting of: first a left lane change to avoid the slower vehicle, then a lane keeping to pass it and finally another lane change, to return to the original lane [Shamir, 2004]. It is considered as one of the most challenging maneuvers due to the different factors that can lead to a collision between overtaking and overtaken vehicles. For instance, a wrong estimation of the distance to the in-front vehicle to overtake, a wrong estimation of its speed or the lack of vision beyond the obstacle in front, due to malfunctions in the perception system or the on-board sensors [Richter et al., 2016].

A classification of the different approaches has been drawn depending on which is the main of the application: from a safety point of view, from a geometric point of view, from a planning point of view, and from a control point of view.

First, from a safety point of view, emergency lane change maneuvers where studied in the late 90s. There, the minimum distance from which a static obstacle cannot be avoided given an initial speed, performing the sharpest feasible maneuver was determined in [Shiller and Sundar, 1998]. The generation of collision avoidance maneuvers in emergency scenarios where the automated vehicle may need to go up to their handling limits was treated in [Funke et al., 2016]. There, an MPC-based approach was presented, considering stabilization and collision avoidance as a single problem.

Second, from a geometric point of view, Shamir [Shamir, 2004] stated that the shape and time of the lane change trajectory do not depend on the velocity of the obstacle. He proposed an optimization problem to determine the time and distance of the trajectory considering the maximal acceleration during the maneuver as the only dynamic constraint, generating minimum jerk trajectories. Murgovski et al. [Murgovski and Sjöberg, 2015] also addressed the problem from an optimization point of view, minimizing the error on the reference velocity and position trajectory to plan the entire maneuver in one optimization step.

From a path planning point of view, a combination of quartic polynomial curves for the lane change maneuvers and cubic polynomial curves for the lane keeping maneuvers were employed in [Xu et al., 2012] for generating trajectories to avoid slower obstacles in the route, applying an iterative path and speed optimization which is less time-consuming than a simultaneous optimization. Cubic polynomials were also used in [Petrov and Nashashibi, 2014a] to define the lane change trajectory geometrically. In addition, from a control point of view a kinematic modeling of the relative inter-vehicle kinematics for the overtaking maneuver was proposed, considering no information is received from the infrastructure, and being able to adapt the maneuver to overtake multiple obstacles, assuming they travel performing a rectilinear movement. Similarly, Naranjo et al. [Naranjo et al., 2008] determined the point where the curve for changing the lane in the overtaking should finish, corresponding to the position where the rear part of the ego-vehicle is parallel to the front part of the overtaken vehicle. Besides, they propose to trigger the overtaking maneuver considering the longitudinal distance it would take to change the lane and estimating the distance traveled by the overtaken vehicle, assuming it is traveling at a constant speed. Also from a control based point of view, a fuzzy-logic decision system for the longitudinal control of the overtaking maneuvers, even under risky situations, was presented in [Pérez et al., 2011]. Although these last works present the lane change problem using geometric approaches, their main focus is on the control point of view, thus, comfort in the lane change paths is not a key element there. Different security lateral distances between overtaking vehicle and obstacles in the lane change maneuver were defined in [Milanes et al., 2012] to perform safer maneuvers. There, the dimensions of the obstacles are considered, thanks to a vision system that classify the obstacles according to their estimated width and length. Most of the strategies described above address the lane change problem from a spatial point of view. Nilsson et al. [Nilsson et al., 2017] evaluated the appropriate inter-vehicle traffic gap and time instance to perform a lane change maneuver. There, a reachability analysis is done to ensure safe margins and satisfying the physical limitations of the road.

Quintic polynomial curves have been considered in some works because they generate a smoother curve than quartic and cubic approaches, ensuring C^2 continuity. González et al. [González Bautista, 2017] used them not only to generate a smooth collision-free path but also to generate a continuous speed profile, where the quintic curves assure a smooth jerk and acceleration, improving the comfort. Qian et al. [Qian et al., 2016] combined a Subplex optimization of the quintic-curves path with a reference speed profile optimization using an MPC controller that considers some dynamic and energy consumption constraints. You et al. [You et al., 2015] also used quintic curves in a collaborative strategy where they apply the infinite dynamic circles approach for detecting possible collisions. An optimization method for the steering angle searching both vehicle performance and driving comfort by reducing the lateral acceleration during the lane change was proposed in Zhang et al. [Zhang et al., 2013]. This optimized steering angle is used afterward to generate a collision-free candidate trajectory based on a predictive kinematic model, where the movements of the other vehicles are predicted under traffic conditions in simulation. Trigonometric and exponential functions were used in [Ji et al., 2017] to geometrically describe the road and the dynamic constraints of the obstacles, respectively, building a 3D potential field for the collision avoidance path planning. Then, it uses a Multiconstrained MPC (MMPC) for the path-tracking, trying to minimize risk to the vehicle through evasive maneuvering and stabilizing the vehicle at high speeds.

The following works describe the use of grid-based planners for dynamic environments. An extended solution to consider the uncertainty and dynamism of the road on the trajectory generation is the use of grids as drivability maps. Fu et al. [Fu et al., 2015] combined the navigation circle method with cubic splines to analyze the collision risk and generate the collision-free trajectory but considering dynamic obstacles as if they were static. A road course estimation on a grid-based representation of the environment is done in [Tanzmeister et al., 2016]. It benefits from the maps of the static world to generate the collision-free paths combining A* and RRT graph-based algorithms. The first one to find the minimum cost path and the second one to explore the search space uniformly. However, the approach is computationally expensive and the grid does not consider the dynamic changes in the environment. A dynamic occupancy grid map based on a stereo-vision framework instead of the classical Lidar-based framework was proposed in [Li and Ruichek, 2014]. It combines the segmentation of the moving objects with an occupancy probability estimation to take into account their motion. A Bayesian grid holding both occupancy state and velocity was employed to represent the environment providing an enhanced estimation of the motion of the objects in the scene, especially when they are not describing rectilinear movements on the urban environment [Gindele et al., 2009] . This approach was tested using the Team AnnieWAY's automated vehicle which participated in the DARPA Urban Challenge. Hundelshaussen et al. presented an ego-centered occupancy grid-based method for drivability [Hundelshausen et al., 2009], combining it with the tentacles that are used for evaluating it and to perform the motion, integrating the method on the finalist Team AnnieWAY's vehicle of the DARPA Urban Challenge [Kammel et al., 2008]. A method to deal with uncertainties in the perception of the environment by combining Belief Functions and clothoid tentacles is proposed in [Mouhagir et al., 2017]. There, an evidential occupancy grid is built based on these functions to represent the uncertainties, whereas a set of clothoid tentacles is used for planning the local trajectories. The evidential grid considers the safety distances between the automated vehicle and the obstacles, and they enhance the binary occupancy grid by including the road limit and the longitudinal expansion of the dynamic obstacles.

4.2 Problem formulation

From the literature review, one can appreciate how the formation of collision-free paths handling the dynamics of the obstacles must combine a technique for exploring the space and another for the path generation considering the motion of the obstacles [Gindele et al., 2009], [Hundelshausen et al., 2009], [Mouhagir et al., 2017]. When dealing with complex urban environments, multiple traffic agents generation can occur in a really short period of time, underlying the importance of being able to adapt vehicle's trajectory in real time.

This thesis presents an approach that divides the motion planning problem for urban environments into two sub-problems. First, the optimal path to arrive at the desired destination is computed, considering the physical limitations of the vehicle as well as the road constraints, as presented in Chapter 3. There, a two-stage algorithm is proposed. The pre-planning stage (Section 3.3.1) aims to benefit from the static information of both vehicle and road to pre-compute the optimal curves that the vehicle may encounter in all urban scenario. Then, a real-time local planning stage considers the actual road layout to load the optimal curves, searching the best junction between them to provide a smooth trajectory.

This chapter presents the strategy to deal with dynamic environments. The strength of this algorithm lies in the benefit of the already planned path for the static environment to adapt it to the dynamic scene, providing a real-time response. Next sections present the developed algorithm, where a grid-based local planning algorithm is developed to represent the scene considering the obstacles that the ego-vehicle may find on the route. It predicts their future position according to the vehicle type, creating then a virtual lane to address the problem as a static environment. There, the obstacle avoidance path is built by joining two curves for each lane change according to the virtual lane configuration, loading them from the pre-planning stage. A verification process is done in the real-time planning stage to check if the path needs to be re-planned. A new avoiding path is computed if the prediction of the distance covered by either ego-vehicle or moving obstacle is far from those predicted. Furthermore, an auxiliary trajectory to return safely to the lane is computed at every planning step during the avoiding maneuver. It would allow to the automated vehicle to return safely to the original lane in case some unforeseen situation arises, such as detecting another vehicle in the left lane or a sudden change of the vehicle in front, forcing the algorithm to abort the current avoiding maneuver and return to a safe configuration.

A flowchart of the dynamic planning process is depicted in Figure 4.2. First, the scene is discretized through a grid from the information of the obstacles coming from the perception stage. There, a classification is done to determine what type of road user are the obstacles perceived. After adding a security space behind and ahead the obstacle, a virtual lane is computed from the original global path. It is done adding two additional way-points for each lane change, allowing that way to re-plan only the portion of the path where the obstacles are found generating the avoiding path as performing two static curves for each lane change, benefiting from the static local planner. Additionally, a checking process is done to verify if the estimated dynamics of both ego and obstacles are correct for generating a safe path, or if a re-planning process is needed. Finally, the resulting path is transmitted to the control stage for the tracking process.

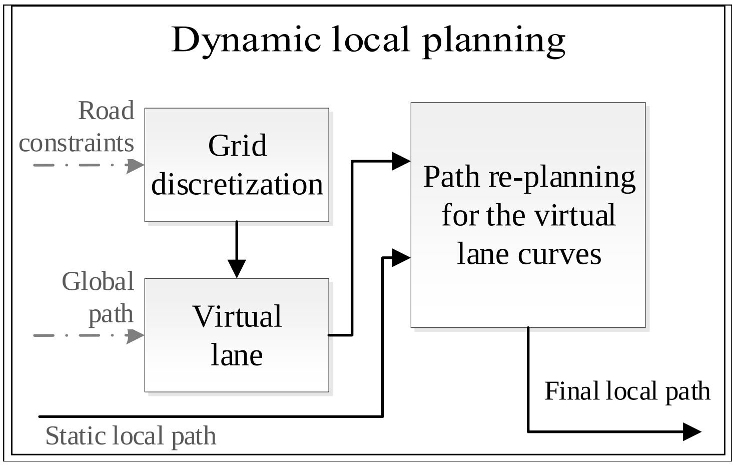


Figure 4.2 – Dynamic local planning system architecture

4.3 Dynamic local planning algorithm based on a virtual lane generation and a grid discretization

This section considers the impact of the dynamic behavior of other road users on the path. Specifically, an obstacle avoidance solution is proposed to adapt the planned path to avoid the possible static obstacles detected in the path and overtaking the dynamic ones, such as other cars, bikes or pedestrians.

Maneuvers to avoid a static obstacle or to overtake a dynamic obstacle have been defined in the literature three-phase maneuvers [Shamir, 2004]: vehicle first changes to the left lane, then it keeps the lane surpassing the obstacle and then returns to the original right lane. With the proposed approach, these phases consist of four curves, as depicted in Fig. 4.3: two for the first lane change and another two for changing back to the original lane.

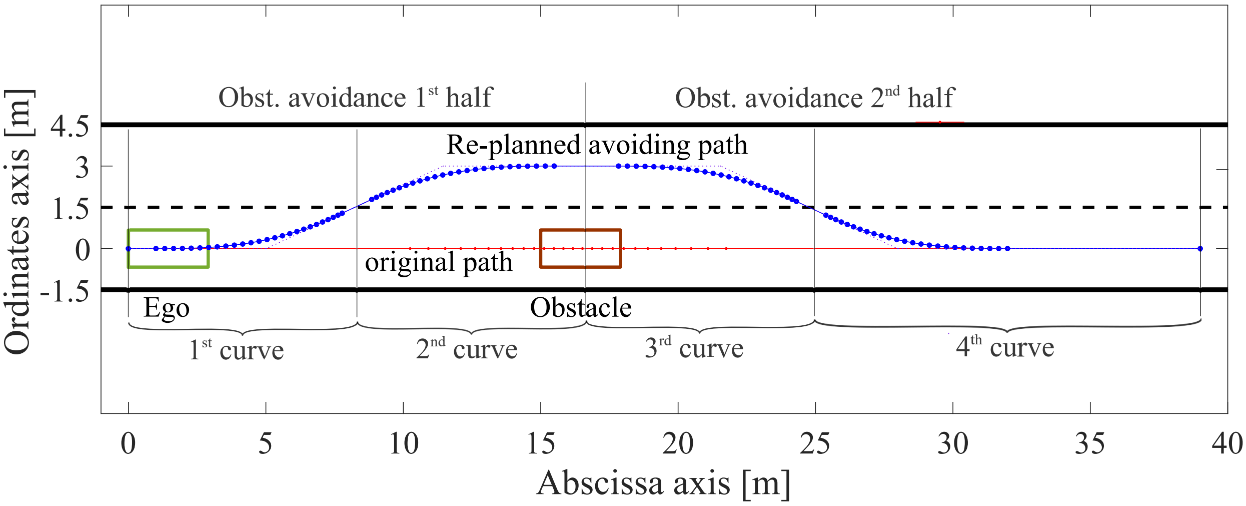


Figure 4.3 – Trajectory planned with the proposed algorithm for obstacle avoidance

The proposed solution for adapting the static path to the dynamic environments avoiding the obstacles is to build a virtual lane overlapping the current lane, which modifies the way-points of the itinerary and makes the local-planner re-plan only the driving area where interaction with other traffic agents occurs. This is based on a grid discretization of the road, where the information about the obstacles is received from the perception and then used to classify them providing a safe clearing distance for the virtual lane generation and therefore, for the obstacle avoidance maneuver. Thus, the proposed algorithm for avoiding static obstacles and overtaking dynamic vehicles combines the following features, shown in the flowchart of Figure 4.2.

4.3.1 Grid-based discretization

The purpose of using grids is to obtain a better description of the scene representing the space as free or occupied by means of cells, making a discretization of the road space allowing the system to consider the uncertainty of the sensors, and as a result, the dynamism of the road [Mouhagir et al., 2017], [Elfes, 1989].

Algorithm 4 details the developed algorithm to generate the grid making this space discretization, representing the space as occupied or free according to the objects detected on it.

Before building the grid, the obstacle detection is done in the perception stage. In case any obstacle is detected in front of the ego-vehicle, the dynamic path planner will receive its Oriented Bounding Box (OBB) (line 2, Algorithm 4). In our case, as the vehicles are equipped with single-layer Lidar sensors, it receives the OBB of the obstacle as the 2D coordinates of the rear part of the vehicle in front, as well as the distance from the automated vehicle position to such obstacle, computed from this information [Merdrignac et al., 2015].

Algorithm 4 Re-planning algorithm for the dynamic path generation

```
Initialization: grid configuration is set
Receive rear coordinates of the obstacles ahead
Classify which type of vehicle is the obstacle ahead
Estimate length of obstacle
Build Oriented Bounding Box
Set the cells occupancy value for the obstacles BBs
```

Table 4.1 – Obstacle types classification based on the width

Road user dimension	**Road users classification according to their dimensions**			
	VRU	Cybercar	Car	Bus/truck
Width [m]	[0.0 - 1.0)	[1.0 - 1.4)	[1.4 - 2.1)	[2.1 - 3)
Length max [m]	3	2.9	5.5	18
Security dist. [m]	1.5	width/2	width/2	width/2

Then, an obstacle classification is done in order to further leave different security distances according to the vehicle type, on a similar way than in [Milanes et al., 2012] (line 3, Algorithm 4). Since the overtaken vehicle is in front, the OBB only provides the information of the obstacle width, but the length of the vehicle is unknown. In consequence, a classification of the obstacle is done by using Table 4.1. There, the different vehicle types are defined based on the European Euro NCAP Structural category [van Ratingen et al., 2016] and the American US EPA Size Class regulations [US Environmental Protection Agency (EPA), 2017]. In addition, the cybercar prototypes have also been included. Thus, the vehicle types considered are VRU (including pedestrian, bikes and two-wheeled vehicles, cybercars (either cycabs or cybuses), passenger cars, and longer vehicles such as trucks or buses. For each vehicle type, both the width-range and its equivalent length have been determined by analyzing the dimensions of some of the reference vehicles on the market. Thus, an estimation of the obstacle length is feasible from the OBB width, and different lateral and longitudinal security distances can be considered respecting the obstacle dimensions (line 4, Algorithm 4). Therefore, for estimating the length of the obstacle, once its width is known and the obstacle has been classified, its length/width relationship is applied.

The space discretization is therefore done by representing the space occupied by the obstacle after the length prediction in the grid matrix. Such area corresponds to the axis aligned Bounding Box, or just Bounding Box (BB), depicted with a red rectangle in upcoming Figures (line 5, algorithm 4). Figure 4.4 represents the different obstacles the ego-vehicle could encounter according to the described vehicles classification with respect to their physical dimensions. There, the Bounding Box of the obstacles, the cells of they occupy and the added distance for safety and comfort reasons are depicted: in garnet for a bike (VRU), in blue for a cybercar, in green for a passengers car and in magenta for a truck.

A discretization of the road space allows an easier recognition of the free space on the road, where the road space occupied by the BB is represented as occupied using red crosses in the Figures.

In order to set the state of the cells where the obstacles lie as occupied (line 6, algorithm 4), a conversion from continuous space to discrete space is done, i.e. from the coordinates of the BB of the obstacles in meters, to the corresponding indexes of the cell. If the borders of the obstacle's BB lie in the middle of the cell, the whole cell will be considered as occupied, setting a one as its

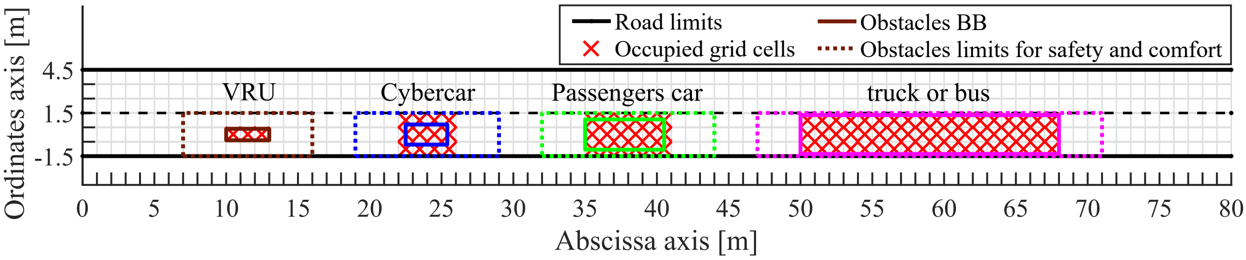


Figure 4.4 – Grid representation of the different vehicle types

value in the matrix. In other case, the cell value will remain as free, represented by a zero value of the cell. It should be noted that a grid centered on the ego-vehicle has been considered.

A parameterizable grid has been considered (line 1, algorithm 4), where the following parameters are established to configure the grid and defining the space discretization: number of road lanes (thanks to the global maps this information can be known), lane width (by default considered as three meters, since lanes on urban roads vary from 3.0 up to 3.7 meters), cell size (set by default at 1 meter), and horizon of the grid, i.e. maximum distance to which the grid is considered (set by default at 40 meters, since the available Lidar sensors operate up to this distance).

The grid is updated then at every planning period in order to consider the changes on the scene. The following sub-sections shows how the virtual lane is generated from the information of the obstacles.

4.3.2 Virtual lane generation

The proposed method to deal with dynamic environments is based on the generation of a virtual lane, i.e. a corridor [Choi et al., 2008] to represent the new road layout overlapping the existing two-lanes road. This subsection details how it is generated to modify the already computed static path to avoid any possible obstacle generating the new dynamic local path, which is further transmitted to the controller.

Once an obstacle is detected blocking the existing static path, the perception system alerts the planning stage, which classifies the obstacle and updates the grid representing the scene, as explained in previous subsection. Thus, it triggers the generation of the virtual road to perform the avoidance maneuver. Figure 4.5 shows in purple the virtual road generated for an automated vehicle performing a path following scenario, where A-B are the following way-points belonging to the itinerary to follow. Although the figure shows just the A-B segment, it is normally part of a bigger path where obstacles can be found, even in turn stretches.

The goal of building the virtual road is to modify the itinerary corresponding to the affected path following road section (A, B) by adding some supplementary way-points (swp_n) to the global path, resulting in $(A, swp_1, swp_2, swp_3, swp_4, B)$, as shown in Figure 4.5. These points define the center of the new virtual lane, from where the limits of the virtual lane are further computed geometrically, letting us to target the overtaking problem with the proposed static planner.

The system works as follows:

1) After an obstacle is detected, classified and its Bounding Box is formed, some lateral and longitudinal security distances will be added, forming the obstacle's safety area for the avoiding maneuver, represented with the external red rectangle in Figure 4.5. On the one hand, a longitudinal distance of the ego-vehicle length is considered both on the front and the rear, based on previous works in the literature [Shamir, 2004, Naranjo et al., 2008]. It ensures

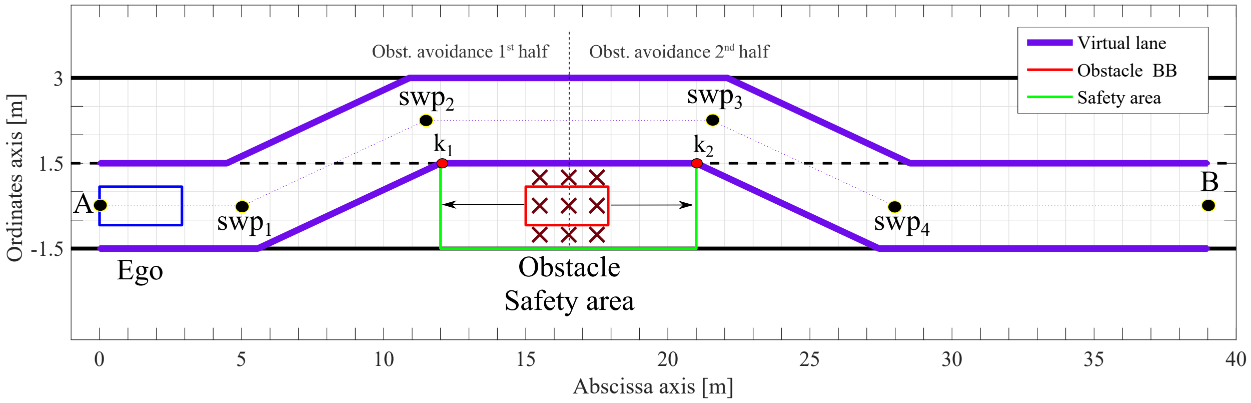


Figure 4.5 – Virtual lane generation with the proposed algorithm for obstacle avoidance

not colliding with the obstacle: neither in the completion of the first lane change (avoiding a frontal collision), nor in the second lane change to the right lane (avoiding a lateral collision). On the other hand, a lateral distance of at least 1.5 meters is considered for the VRU, placing the way-points in the middle of the left lane by default. In any other case, half of the width of the obstacle is considered.

2) The virtual road is built from the obstacle's safety area, whose internal points k_1 and k_2 are obtained after applying the longitudinal and lateral safety distances to the obstacle's BB (see Figure 4.5). These points allow us to place the way-points swp_2 and swp_3 in the left lane and they constitute the last way-point for the first lane change and the first way-point for the second lane change, respectively.

3) Then, the points swp_1 and swp_4 for the starting and finishing phases of the lane changes are computed, placing them perpendicular to the preceding or next way-points, respectively, to avoid abrupt changes on the curvature. For this purpose, an algorithm has been designed to find the minimum slope (φ_1 angle in Fig. 4.6) for the lane change maneuver, as well as the best φ_2 angle that will determine the location of the points swp_2 and swp_3, respectively. That way, it will determine how far the avoiding maneuver begins and the close that the ego-vehicle will pass with respect to the obstacle at the end of the lane change maneuver to be located in the adjacent lane, parallel to the obstacle. The last step moves laterally the way-points of the new global itinerary that form the center of the virtual lane, with a positive and a negative distance of half of the road width. Thus, the algorithm iterates modifying the value of these two angles, getting the position of both way-points. For each configuration, the real-time static local planner will just search the optimal location of the junction point between the two curves for the lane change, i.e. evaluating both curves with the described optimality criteria, optimizing both curves, as explained in the static path planner chapter.

4) Finally, the optimal virtual road is generated (in purple in Figure 4.5) applying a geometric lateral displacement of half of the lane width with respect to the center of the lane, formed by the new global path after adding the way-points to avoid the obstacles. This way, just by modifying the global path we could profit of the developed static local planner providing a dynamic behavior, only evaluating in real-time the minimum feasible slope for the lane change.

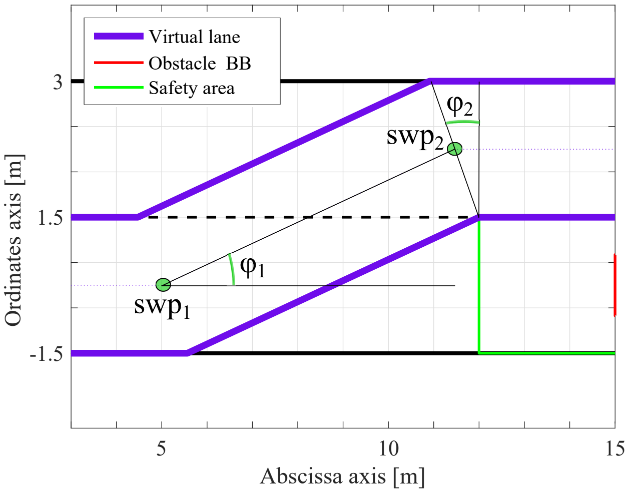


Figure 4.6 – Representation of the slope φ_1 and φ_2 angles for the lane change maneuver

The whole planning architecture proposed in this thesis presents three main advantages: 1) Ability to deal with any road configuration optimizing the trajectory in real-time. 2) Low computational cost, thanks to most of the optimization process is done off-line (in the pre-planning stage), making the computation time almost negligible. 3) Ability to quickly deal with emergency situations if an obstacle avoidance maneuver is feasible with a lateral displacement, or if an emergency braking is needed to minimize the collision risk aborting the avoiding maneuver.

The algorithm described above shows how to adapt the local path to avoid static obstacles. In order to consider dynamic obstacles, the occupancy of the cells on the grid is updated taking into account the obstacle speed. There, a prediction of the space that will be occupied by the obstacle in the worst case is done, considering its maximum speed and its maximum acceleration, as described in Equation 4.1. Thus, all this space is represented as occupied in the grid. Then, the new global path points are calculated in the same way as shown before. Figure 4.7 shows an example where the internal rectangle represents the predicted maximum space occupied by the vehicle, whereas the external rectangle is its safety area, defining the new virtual road. Then, ego-vehicle is localized according to the new global path to generate te local path to overtake it.

$$x_{obst_{predicted}} = x_{obst} + vMax_{obst} * vMax_{obst}/accMax_{obst} \tag{4.1}$$

where $x_{obst_{predicted}}$ represents the estimation of the space occupied by the obstacle (in m), x_{obst} represents the current position (in m), and $vMax_{obst}$ and $accMax_{obst}$ are the maximum speed and maximum acceleration of the obstacle according to the vehicle type (in m/s and m/s^2 respectively).

4.3.3 Re-planning method for dynamic scenarios

Since urban environments are continuously changing, a method that checks the environment and evaluates if the first avoiding trajectory is still feasible or not is needed, i.e. a re-planning approach. This method benefits from the proposed modular planning architecture to adapt the already planned path by displacing the way-points that define the itinerary, if and only if the conditions to perform the planned maneuvers are not feasible any more, either because the dynamics of the ego or the obstacle have changed, or because some unexpected circumstance arises forcing the system either to abort the maneuver and execute an emergency maneuver to return safely to the lane, or to make an emergency brake.

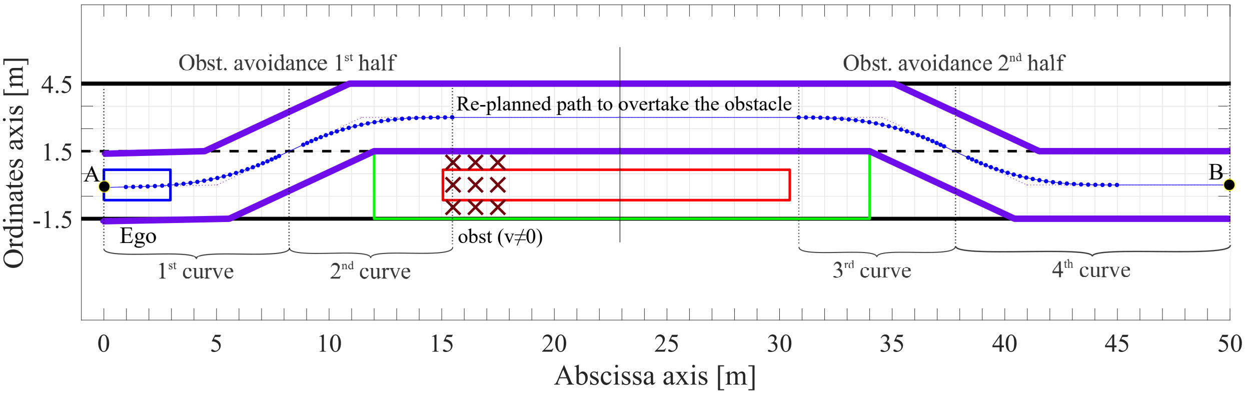


Figure 4.7 – Virtual lane generation for dynamic obstacles considering the obstacle's occupied space prediction

The proposed algorithm is described in Algorithm 5. It takes into account the speeds and accelerations of both ego-vehicle and obstacle to predict their behavior over time. That way, a re-computation of the path has not to be done unless the difference between predicted and traveled distances by both ego and obstacle are substantial, bigger than a parameterized threshold. The default value considered for this threshold is 0.5 meters for low-speeds (up to 20 km/h).

Algorithm 5 Re-planning algorithm for the dynamic path generation

```
1:  Receive global path and static local path
2:  Check if there is any obstacle in the path
3:  Check if ego-vehicle is running faster than obstacle
4:  Compute accelerations of ego and obstacle
5:  Compute distance covered by ego and obstacle during time period
6:  if distance covered by either ego or obstacle on $\Delta T >$ threshold then
7:      Compute Time-to-collision (TTC)
8:      if TTC $\leq 6$ s then
9:          Predict distance to be covered by obstacle on TTC
10:         Predict distance to be covered by obstacle on period T
11:         Project obstacle position
12:         Compute the distance that both ego and obstacle will travel on time period
13:         Compute obstacle Bounding Box from obstacle width
14:         Compute obstacle's safe zone
15:         Compute new global path and new virtual lane
16: Update previous speeds and accelerations
```

Line 1 shows that the algorithm first receives from the static planning stage both the global path containing the way-points of the itinerary and the local path whose points smooth the global itinerary preserving the comfort.

Then, in order to predict their motion, the acceleration of both ego-vehicle and obstacle in front are computed for the current time period, as shown in Equation 4.2 (line 4, Algorithm 5),

$$\begin{aligned} a_{ego_t} &= \frac{v_{ego_t} - v_{ego_{t-1}}}{\Delta T} \\ a_{obst_t} &= \frac{v_{obst_t} - v_{obst_{t-1}}}{\Delta T} \end{aligned} \tag{4.2}$$

where a_{ego_t} and a_{obst_t} are the current acceleration for ego-vehicle and obstacle, respectively; v_{ego_t} and v_{obst_t} are the current speeds. Meanwhile, variables $v_{ego_{t-1}}$ and $v_{obst_{t-1}}$ are the speed of ego-vehicle and obstacle in the previous time period ΔT.

In addition, the algorithm gets the real distance covered by both the ego-vehicle and the obstacle on the last time period. This measurement of the real distance covered will be compared with the predicted values, which will suppose a further trigger to perform the re-planning of the overtaking maneuver if the difference between real and predicted measurements is higher than the specified threshold (lines 5-6, Algorithm 5). Equation 4.3 serves as basis to compute the TTC. This equations represents that the distance covered by ego-vehicle in the current period t have to be the same than the one covered by the obstacle plus the distance between them.

$$s_{ego_t} = s_{obst_t} + dist(ego, obst) \tag{4.3}$$

where s_{ego_t} is the distance covered by the ego-vehcile, s_{obst_t} the distance covered by the obstacle and $dist(ego, obst)$ the distance between them.

Then, if both vehicles present a constant speed, i.e. the accelerations are zero, the time to collision (TTC) is computed as presented in Equation 4.4. On the other hand, if the vehicles present an acceleration different from zero, the TTC is computed as in Equation 4.5. TTC is the time that would pass until reaching the obstacle position, i.e. until colliding with it.

$$TTC = \frac{dist(ego, obst)}{v_{ego} - v_{obst}} \tag{4.4}$$

$$\begin{aligned} s_{ego} &= s_{obst} + dist(ego, obst) \\ TTC &= \frac{-(v_{ego_0} - v_{obst_0}) + \sqrt{-4 \cdot \frac{1}{2} \cdot (a_{ego} - a_{obst}) \cdot (-dist(ego, obst))}}{2 \cdot \frac{1}{2} \cdot (a_{ego} - a_{obst})} \end{aligned} \tag{4.5}$$

where v_{ego_0} and v_{obst_0} are the inital speed of ego-vehicle and obstacle,

A verification has to be done in order to trigger the re-planning process if the TTC is lower than 6 seconds, which is considered in [Milanes et al., 2012] based on the accuracy of the vision-based detection system (line 8, Algorithm 5).

By computing the TTC, a prediction of the distance that both ego and obstacle would travel on this time can be done. This will let the algorithm move the way-points to construct the new virtual lane as explained in previous section, where the location of the way-points added for avoiding the obstacle depend on the projection of the obstacle position in the TTC.

Once the TTC has been computed, the algorithm is able to predict the distances that both the ego-vehicle and the obstacle will cover on this time (line 9, Algorithm 5), i.e. $s_{ego_{TTC}}$ and $s_{obst_{TTC}}$, respectively. Thus, a projection of the obstacle position on the TTC can be done by using Equation 4.6.

$$\begin{aligned} s_{ego_{TTC}} &= s_{ego_0} + v_{ego_0} \cdot TTC + \frac{1}{2} \cdot a_{ego} \cdot TTC^2 \\ s_{obst_{TTC}} &= s_{obst_0} + v_{obst_0} \cdot TTC + \frac{1}{2} \cdot a_{obst} \cdot TTC^2 \end{aligned} \tag{4.6}$$

In addition, on the same way the algorithm predicts the distance that both vehicles will cover on the period time T (Equation 4.7), i.e. s_{ego_T} and s_{obst_T}, respectively. Then, these values are compared in the next period T with the real measurement of the distance actually covered by the vehicle (line 10, Algorithm 5).

$$\begin{aligned} s_{ego_T} &= s_{ego_0} + v_{ego_0} \cdot T + \frac{1}{2} \cdot a_{ego} \cdot T^2 \\ s_{obst_T} &= s_{obst_0} + v_{obst_0} \cdot T + \frac{1}{2} \cdot a_{obst} \cdot T^2 \end{aligned} \tag{4.7}$$

If the algorithm detects in the next time interval T that the re-planning has to be carried out due to a difference between predicted and actual distance covered by overtaking or overtaken vehicles higher than the specified threshold, the process to generate the virtual lane starts as explained above.

First, since the standard perception systems are based on 2D Lidar sensors, as the ones equipped on the RITS team, the planning algorithm only receives the coordinates of the rear track of the obstacle vehile. According to the classification of the different road users in Table 4.1, the Oriented Bounding Box is computed from the width of the obstacle applying the length/width relationship. Once the OBB is computed and the safety zone surrounding the obstacle is built, the avoiding maneuver can be computed as explained in Section 4.3, where the new global path consist of the old way-points and four additional way-points to avoid the obstacle, where their location depends on the safety area points and the longitudinal and lateral safety distances employed. That way, the algorithm only recomputes a new virtual lane (i.e. a new global path that will trigger a new local path for the affected area) if the vehicles do not follow an expected behavior.

Finally, the algorithm has to update the previous speed and acceleration to iterate in the next time-stamp.

Figure 4.8 shows how the virtual-lane is recomputed when an obstacle is continuously accelerating, according to the explained re-planning process. There, a sub-set of virtual lanes is depicted.

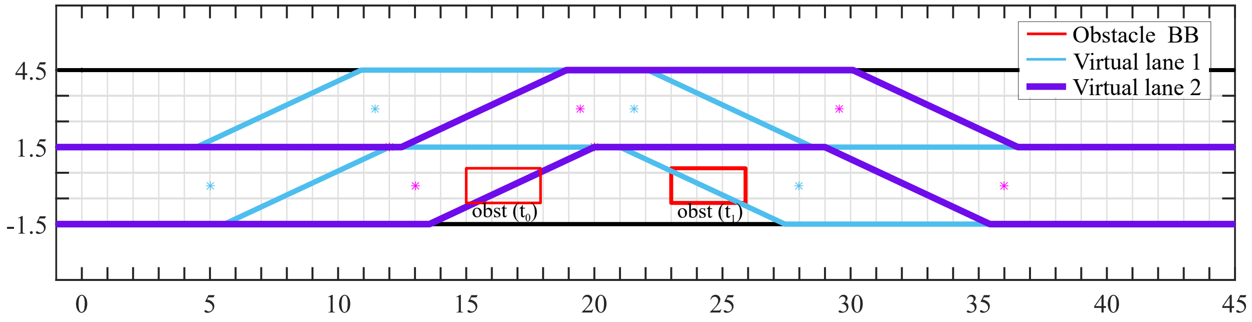


Figure 4.8 – Re-computation of the virtual lane in real-time for re-planning on scenario with changing dynamics of the obstacle

This scenario represents the case where the first prediction done is not correct, because the vehicle is changing its speed over time. In this case, the obstacle is accelerating with a longitudinal acceleration of 2 m/s^2. Thus, the re-planning system has to re-compute the virtual lane in order to generate new avoidance paths.

There, the virtual lane is recomputed over time every 0.5 seconds, but only a sub-set of four virtual lanes during 3 seconds execution has been shown. First virtual lane computed is depicted in light blue, second one in dark blue, third one in purple and last one in green.

4.3.4 Safety avoidance path

During this chapter we have discussed how the proposed local planning approach is able to modify the path avoiding the obstacles that may be encountered in front of the automated vehicle while performing the path following operation. However, sometimes unexpected situations occur in such urban scenarios where they operate, forcing to abort the avoiding maneuver. For instance, if the automated vehicle is performing the avoiding maneuver and another obstacle is detected on the left lane while it is changing its lane, the automated vehicle must abort the avoiding maneuver returning to the original lane, as shown in Figure 4.9. Some other similar situations include: detecting a pedestrian, bike or any other vulnerable road user crossing in front of the car, or detecting that the prediction of the moving obstacle in front is not correct since a huge speed difference is detected from predicted to the real measurement.

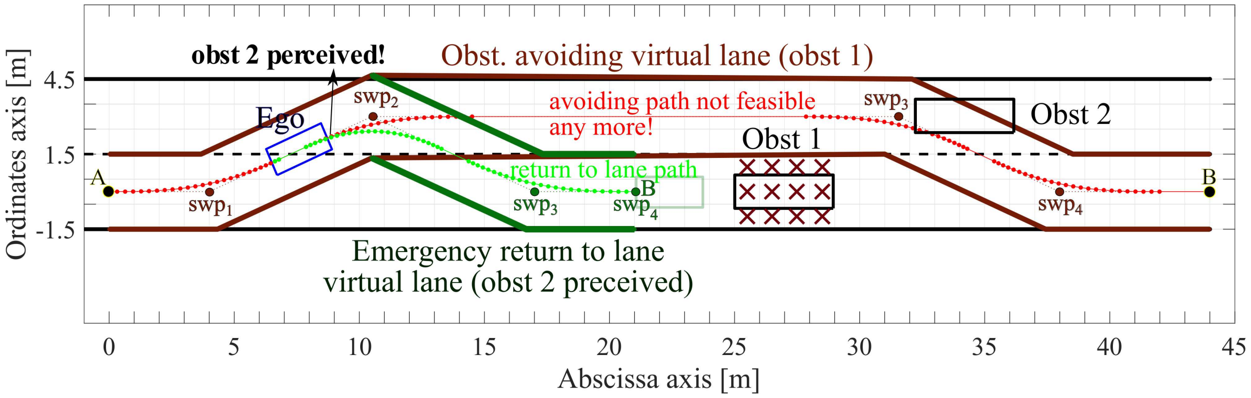

obst 2 perceived!
Obst. avoiding virtual lane (obst 1)
Ego
swp_2
avoiding path not feasible
any more!
swp_3
Obst 2
Obst 1
A
swp_1
return to lane path
B
swp_3
swp_4
B
swp_4
Emergency return to lane
virtual lane (obst 2 preceived)
4.5
1.5
-1.5
Ordinates axis [m]
0
5
10
15
20
25
30
35
40
45
Abscissa axis [m]

Figure 4.9 – Emergency return to the lane maneuver under unexpected vehicle found situation

The algorithm must compute at every planning period a safety emergency path to return to the right lane safely. Thus, a backup global path is generated, modifying the avoiding path removing the way-points on the left lane and adding way-points to make the vehicle drive back to the right lane. Figure 4.10 shows some possible emergency paths to return to the lane for three different time instants of the avoiding maneuver, for the vehicle location A, B and C respectively, assuming the obstacle in front is running at a constant speed.

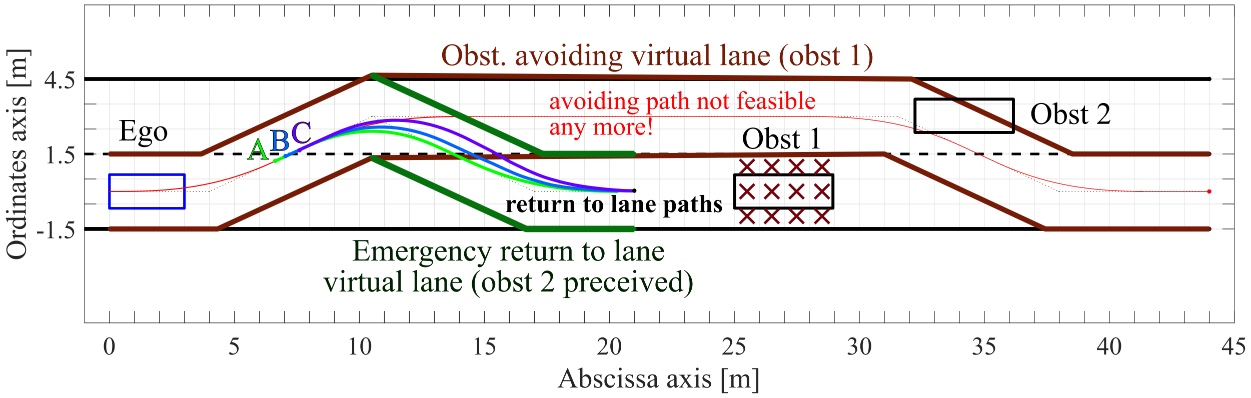

Obst. avoiding virtual lane (obst 1)
avoiding path not feasible
any more!
Obst 2
Ego
ABC
Obst 1
return to lane paths
Emergency return to lane
virtual lane (obst 2 preceived)
4.5
1.5
-1.5
Ordinates axis [m]
0
5
10
15
20
25
30
35
40
45
Abscissa axis [m]

Figure 4.10 – Emergency return to the lane maneuvers under unexpected vehicle found situation

Figure 4.9 shows in dark green the new virtual lane for the maneuver to return safely to the lane, overlapping the previous virtual lane in garnet for the obstacle avoidance maneuver. A new global path consisting on removing the way-points corresponding to the second lane change (swp_4 in garnet and B), replacing them with two additional way-points in the original right lane, which allows to pull the curve to this side of the road and finish the maneuver behind the obstacle. Thus, here the algorithm iterates to search a feasible new path moving this swp_2 way-point from the old position to the vehicle position. In addition, swp_3 is placed in the center of the lane behind the moving obstacle, and sw_4 just behind the obstacle respecting a minimum security distance of one, becoming the new destination point B. Thus, in order to carry out the maneuver a new global path is generated, changing from the old one formed by the way-points in garnet: A, swp_1, swp_2, swp_3, swp_4, B, to the new one in dark green from the current position of the vehicle, formed by: swp_2, swp_3, swp_4, all in green. Then, the virtual lane is computed geometrically and the local planner receives this new itinerary for computing the new local path as an static environment as explained before.

If there is not enough space to perform a safe maneuver to return to the right lane, an emergency braking must be done, avoiding that way any further collision.

4.4 Conclusions

This chapter presented a method that benefit from the proposed planning stage to adapt the planned static path to operate in dynamic environments, where both static and dynamic obstacles are encountered while driving. The proposed approach is based on a grid discretization of the space, where the occupancy of the space is determined, together with a virtual lane generation, which computes a new global path which allows to avoid the static or dynamic obstacles in the same way as a static environment by adapting the global path. This dynamic planning architecture was presented in Figure 4.2, and belongs to the general planning architecture depicted in Figure 3.1.

To determine the occupancy level of the space, a classification of the different road users perceived in front of the vehicle in the scene is done according to the physical relation between the perceived width and the length/width relationship of the different road users considered. There, different lateral and longitudinal security distances have been considered to extend the BB of the obstacle, setting as occupied that space allowing to generate safer maneuvers for each type of obstacle as in [Milanes et al., 2012].

Afterwards, a corridor called virtual lane is built geometrically from the position of the security area surrounding the BB of the obstacle. Additionally, if it is in movement, an initial prediction is done considering the worst case where the obstacle achieves the maximum speed and acceleration.

The re-planning method checks if the predicted motion of both automated vehicle and obstacle are not the same as the real distances covered. In that case, the time-to-collision is re-computed and the space grid is updated setting as occupied the space up to the projected position of the obstacle in the next period of time in the planning process. That way, the dynamic planner only modifies the static path in the segment between the way-points of the itinerary where the obstacles are found, and the avoidance path is only re-computed if there is a big gap between the predicted motion of obstacle and ego vehicle, allowing to alleviate the planning phase to be executed as fast as possible. Additionally, it checks if the avoidance maneuver is still feasible. If some unexpected situation arises, such as another obstacle is detected in the avoidance path while changing lane or a pedestrian crossing in front of the vehicle, the avoidance maneuver is aborted. Then, an emergency path is computed to perform a safe return to the lane maneuver. If this maneuver is not even feasible due to safe criteria (not presenting enough distance) an emergency braking is carried out. That way, the algorithm is able to deal with any dynamic situation a vehicle can encounter when driving in structured urban environments.

Thanks to the adaption of the problem to be solved as a static environment with the proposed real-time static planner, the significant advantage with respect to the obstacle avoidance techniques in the state of the art is that the only parameter to evaluate in real-time is the slope of the lane change maneuver. However, other techniques such as state lattices evaluate in real-time a big set of trajectories to make the full path (lane change, lane keeping and return to the lane), which is more time-consuming, due to the impact of analyzing a big set of curves in real-time. With this information, the global path is updated with the additional way-points to perform the avoiding maneuver, the vehicle is localized with respect to this new global path and the curves are generated by loading the proper ones from the databases and searching the junction point, as described in the static real-time planning stage.

The virtual lane construction for the avoiding maneuver is based on the three-phase method presented in the literature [Shamir, 2004], which consist in two lane changes (a first one to the left lane and a last one to the right lane) and a lane keeping maneuver. Here, each of the two lane changes is carried out by joining two optimal quartic Bézier curves (loading them from the databases), after the evaluation of the optimal junction point is done. Thus, the algorithm ensures generating continuous avoiding paths with a low computational burden.

Chapitre 5

Expériences de validation

Below is a French summary of the following chapter "Validation Tests".

Ce chapitre est structuré comme suit: Tout d'abord, Section 5.1 décrit à la fois le logiciel de simulation (Pro-Sivic et RTMaps) et les véhicules expérimentaux (Cybercars et Citroën C1) utilisés pour tester et valider l'approche de planification proposée dans le cadre de l'architecture automatisée générale du véhicule. Ensuite, les différentes expériences effectuées à la fois sur la simulation et sur des plates-formes réelles sont présentées dans la section 5.2 pour les environnements statiques, puis les résultats des environnements dynamiques dans la section 5.3.

Chapter 5

Valdiation Tests

This chapter presents the results of the tests that have been performed to validate the local path planning approach proposed in this PhD thesis. It has been tested within the modular vehicle architecture for automated vehicles available at INRIA RITS team presented in Section 2.2. Since the scope of this thesis is to make the planning component deal with urban environments while working together with the other modules of the architecture, a fast and real-time response of the approach is pursued. The validation of the proposed planning approach is done using the following scheme: First, simulation of short itineraries combining consecutive curves and straight stretches. They emulate urban environments where the terrain is structured and the road layout is changing along the route. Second, real tests with similar characteristics, where algorithm is validated with different platforms, being able to provide good results for any road layout and vehicle configuration. The proposed solution generates a path from an origin to a destination point, operating without human interaction. Apart from the path following operation on static environments, it also provides an obstacle avoidance operation on simulation under unexpected circumstances where static or dynamic road users are encountered in the path.

This chapter is structured as follows: First, Section 5.1 describes both the simulation software (Pro-Sivic and RTMaps) and the experimental vehicles (Cybercars and Citroën C1) used to test and validate the proposed planning approach operating as part of the general automated vehicle architecture. Then, the different experiments performed both on simulation and on real platforms are presented in Section 5.2 for static environments, and then the results for dynamic environments in Section 5.3.

5.1 Validation platforms: simulator and vehicles

5.1.1 Simulation tools

The simulation environment consists of both Pro-Sivic[1] and RTMaps[2] software tools.

- RTMaps is an asynchronous component-based platform that allows engineers and researchers to develop and test applications for ADAS and autonomous vehicles (among others). It provides a modular toolkit and a framework allowing to use data from vehicle sensors and CAN buses (to time-stamp, record, synchronize and play back data), as well as commanding orders to the vehicle actuators, thanks to the connection with either simulators, such as

---

[1] www.civitec.com

[2] www.intempora.com

Pro-Sivic, or directly to the CAN buses of the vehicles. Since it is based on multi-threading, it benefits on the multi-core architecture of the systems allowing a real-time processing and data fusion.

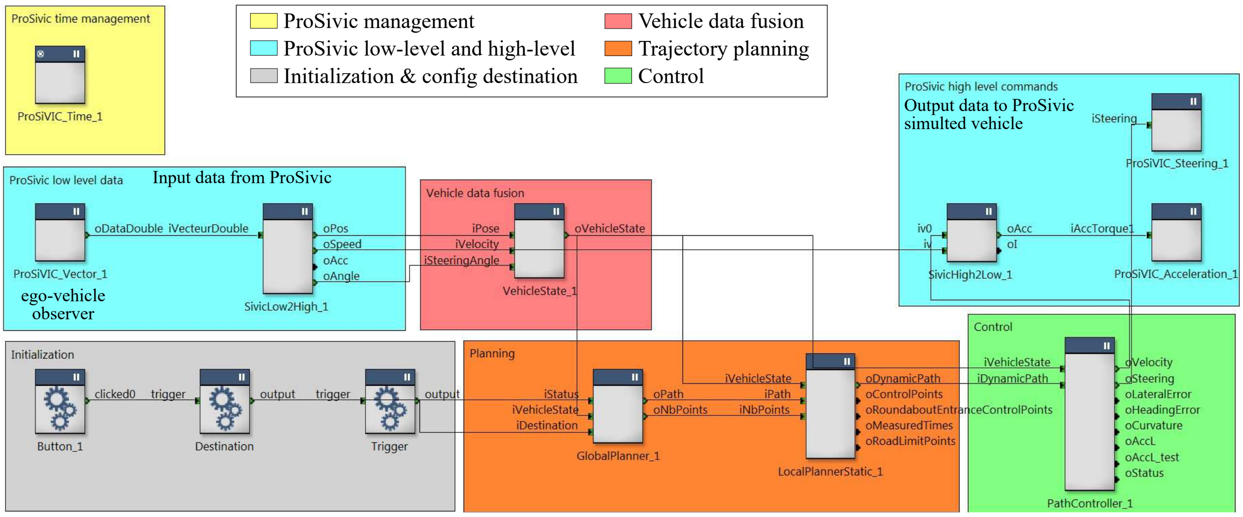
- Pro-Sivic is a virtual prototyping platform for virtual integration of a range of perception sensors, allowing to simulate custom scenarios involving sensors, dynamic actors, such as vehicles and pedestrians, to perform prototyping and testing stages for ADAS, and ITS applications. Pro-SiVIC is used together with RTMaps to perform external processing during a Pro-SiVIC simulation, providing sensor data to RTMaps, whereas RTMaps can send vehicle commands such as steering and acceleration back to Pro-SiVIC, allowing to develop and validate applications for ADAS and automated vehicles. Matlab-Simulink has been used for testing the behavior of the vehicle off-line and validate the pre-planning stage, as described in Chapter 3. In addition, it allows an ease use of the data from the low-level of the vehicle gathered by RTMaps, allowing a fast representation of the results.

The connexion of both RTMaps with either the simulator (Pro-Sivic) or the vehicles (Cycab or Citroën C1) is shown in Figure 5.1, where the main RTMaps blocks for this software interaction are presented. The component named *ProSivic_Vector* in the left part of Figure 5.1a gathers the data of the low-level of the simulated ego-vehicle through the observer in Pro-Sivic, configured to work in RTMaps mode in ProSivic, as shown in Figure 5.1b. This data is received by the low-to-high level controller, and transmitted to the vehicle state component fusing the data to provide the state of the vehicle to the planning stage (position, speed and heading angle). Similarly, the components *ProSivic_Steering* and *ProSivic_Acceleration* in the right part of the figure allow us to transmit the steering and acceleration command to the simulation tool. Previously, the reference velocity and steering angle resulting from the control stage passed through the high-to-low level controller in order to compute the acceleration transmitting it to the actuator of the simulated vehicle. In case of real platforms, the low-to-high and high-to-low level controllers are substituted by that corresponding to the Cycab or the Citroën C1. These components will operate similarly, but receiving the CAN frame to the CAN and building the frame again with the steering and speed commands to send it back again to the low-level actuators of the platforms. As can be seen from the above, the great advantage of RTMaps is the modularity and asynchronous operation it provides, allowing an easy integration of the planning components into the the vehicle architecture to test the planning component both on simulation and real platforms.

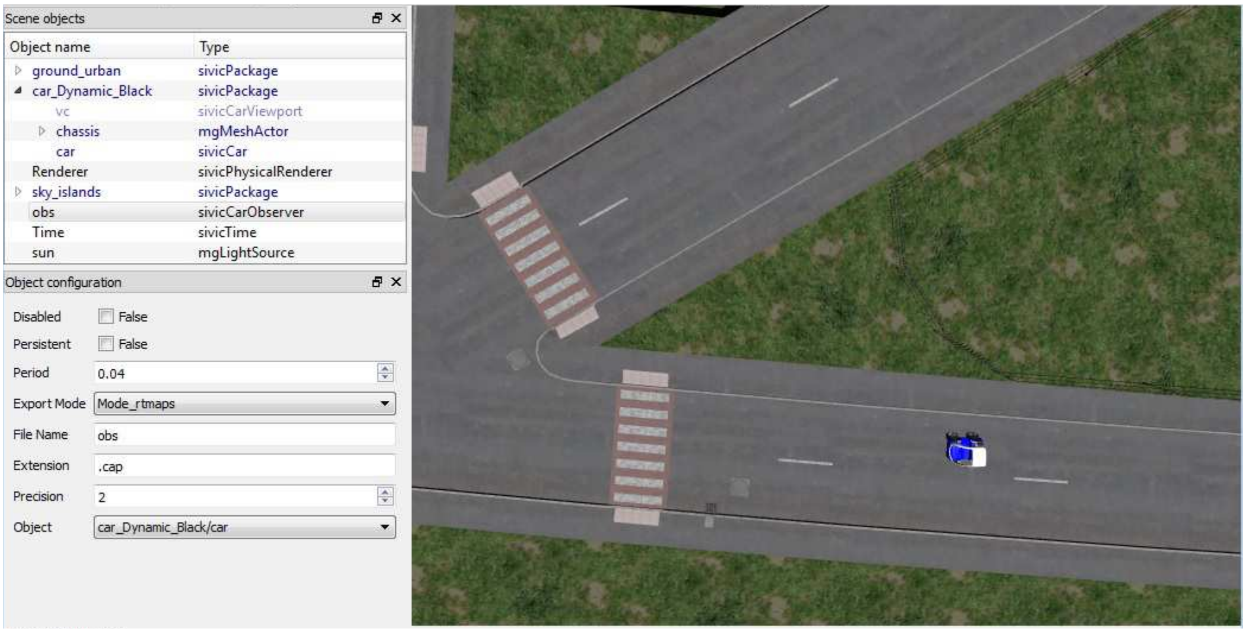
5.1.2 Vehicles

The proposed planning approach has been also tested on the experimental platforms of the INRIA RITS team. The automated vehicles available at the team are the Cybercars (Cycabs and Cybuses) and a recently robotized Citroën C1. Figure 5.2 presents the platforms and their equipment. The aim of research in automated driving is to achieve a higher automation level from the applications related to ADAS already in the third level, up to the fourth one in the medium term. To achieve that automation level, these platforms are equipped with both proprioceptive and exteroceptive sensors, as described below.

- Cybercar platforms, defined as part of the CTS, are low-speed vehicles (up to 5 m/s) designed to operate on urban roads, both for passengers and goods transport [Roldao et al., 2015], without any mechanical actuator, i.e. without pedals and steering wheel. They are golf-like cars with 4 DC motors (one per each wheel). They present a modular architecture based


ProSivic management
ProSivic low-level and high-level
Initialization & config destination
Vehicle data fusion
Trajectory planning
Control
ProSivic time management
ProSiVIC_Time_1
ProSivic low level data
Input data from ProSivic
oDataDouble
iVecteurDouble
ProSiVIC_Vector_1
ego-vehicle observer
oPos
oSpeed
oAcc
oAngle
SivicLow2High_1
Vehicle data fusion
iPose
iVelocity
iSteeringAngle
oVehicleState
VehicleState_1
ProSivic high level commands
Output data to ProSivic simulted vehicle
iSteering
ProSiVIC_Steering_1
iv0
iv
oAcc
oI
iAccTorque1
SivicHigh2Low_1
ProSiVIC_Acceleration_1
Initialization
clicked0
trigger
output
trigger
output
Button_1
Destination
Trigger
Planning
iStatus
iVehicleState
iDestination
oPath
oNbPoints
GlobalPlanner_1
iVehicleState
iPath
iNbPoints
oDynamicPath
oControlPoints
oRoundaboutEntranceControlPoints
oMeasuredTimes
oRoadLimitPoints
LocalPlannerStatic_1
Control
iVehicleState
iDynamicPath
oVelocity
oSteering
oLateralError
oHeadingError
oCurvature
oAccL
oAccL_test
oStatus
PathController_1

(a) RTMaps main components for a simulation test with Pro-Sivic


Scene objects
Object configuration

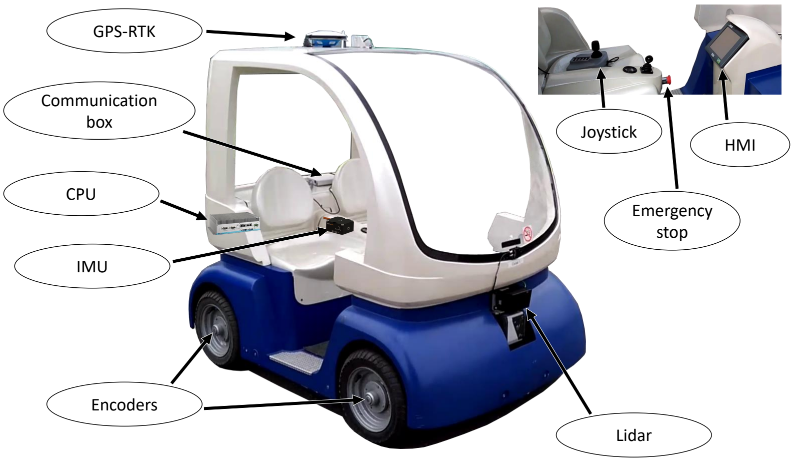
Object name	Type
ground_urban	sivicPackage
car_Dynamic_Black	sivicPackage
vc	sivicCarViewport
chassis	mgMeshActor
car	sivicCar
Renderer	sivicPhysicalRenderer
sky_islands	sivicPackage
obs	sivicCarObserver
Time	sivicTime
sun	mgLightSource

Disabled ☐ False
Persistent ☐ False
Period 0.04
Export Mode Mode_rtmaps
File Name obs
Extension .cap
Precision 2
Object car_Dynamic_Black/car

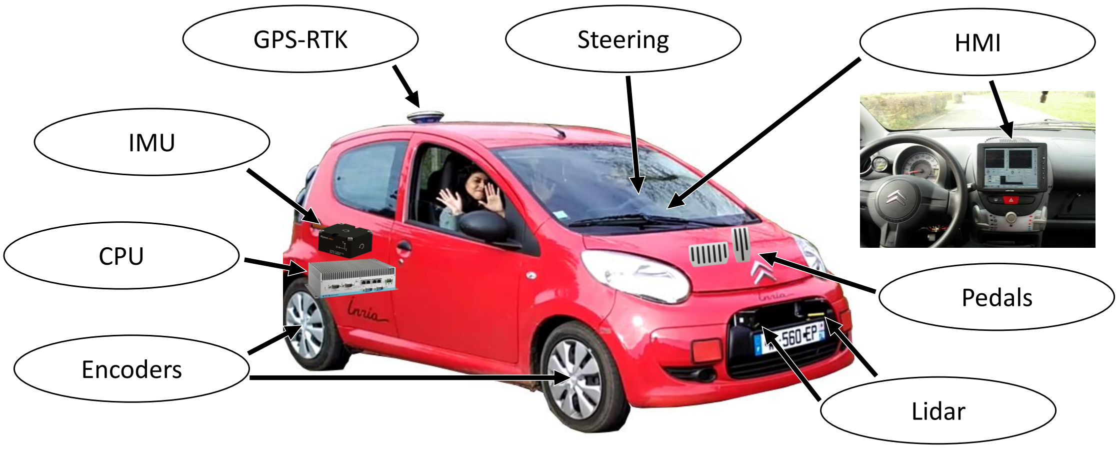
(b) Pro-Sivic configuration of the simulated vehicle's observer

Figure 5.1 – RTMaps and Pro-Sivic interfaces for testing on both simulation and real platforms

on the components shown in Figure 2.15. This constitutes the enhanced and up-to-date version of the architecture proposed in [González and Pérez, 2013, González et al., 2016b]. They are equipped with encoders at the wheels as proprioceptive sensors to get the speed of the vehicle. Apart from the proprioceptive sensors, they are equipped with GPS-RTK modules. It allows the localization of the vehicle in the localization stage, where the data from the GPS-RTK is fused with the information coming from the IMU providing a more accurate positioning. Additionally, 2D LIDAR sensors are equipped in the front of the C1 to perceive and classify the possible obstacles surrounding the vehicle. These sensors also allow to localize the vehicle on the road based on the SLAM technique, which creates a map of the environment while driving.



(a) Cycab electric vehicle and on-board equipment



(b) Citroën C1 robotized electric vehicle and on-board equipment

Figure 5.2 – Experimental platforms of the INRIA RITS team used for on the validation tests

- Citroën C1. This electric vehicle has been robotized for an automated operation of both steering wheel and pedals. It is also equipped with a GPS-RTK and an IMU, both used for the localization of the vehicle. Additionally, it presents two 2D Lidar sensors in the front of the vehicle, which could be used as future work for the obstacles detection and classification on the dynamic operation. These sensors would also allow to localize the vehicle using the SLAM technique, creating a map of the environment and localizing the car on it while running.

The information gathered by these sensors is provided as inputs to the developed local planner module, which will generate the trajectory and will send it through a buffer to the longitudinal and lateral controller, in charge of perform the path tracking. It can be seen in Figure 5.3, where the different modules that form the architecture of the Cycabs are shown. It can be noticed that the vehicle state (magenta modules) is an input of the local planner (orange modules). It consists of the fusion of both GPS (light pink modules) and IMU data (purple modules). The local planner also receives the global path and the number of points it contains as a verification measure. On the other hand, the global planner receives the destination point from the HMI and once it is triggered by the user, it computes the route and is transmitted to the local path. After the local planner computes the path to be followed by the vehicle, it is transmitted to the path controller (green modules). It computes the speed and the steering from the reference path and speed. If the user decides to take control of the car through the joystick, the controller multiplexer will switch from automated to manual operation mode (gray module in the right area). Finally,the high-to-low component (light blue components) receives the desired speed and steering angle from the path controller and builds the CAN frame to be transmitted to the vehicle. The data from the vehicle is received similarly from the low-to-high component and transmitted to the vehicle state component. The RTMaps diagram for the Citroën C1 is similar to the described on for the Cycab, following the same architecture, but with the corresponding components for controlling pedals and steering wheel.

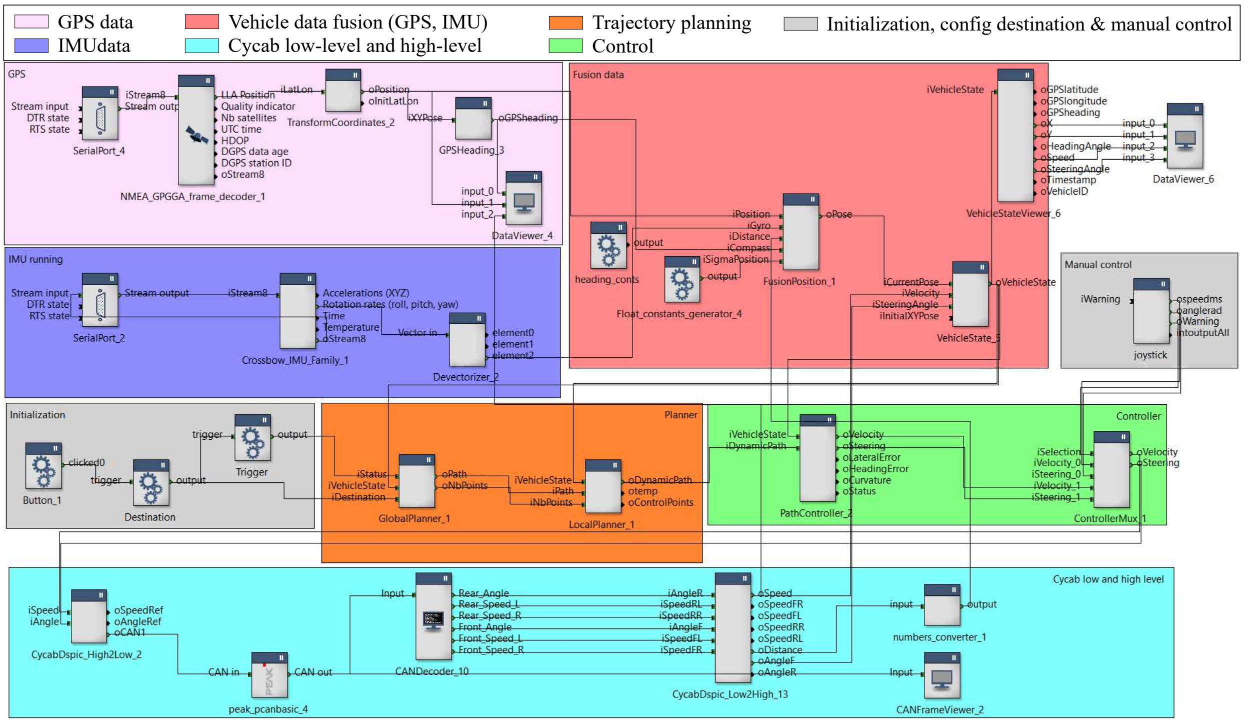

GPS data
IMUdata
Vehicle data fusion (GPS, IMU)
Cycab low-level and high-level
Trajectory planning
Control
Initialization, config destination & manual control
GPS
Stream input
DTR state
RTS state
SerialPort_4
iStream8
Stream outp
LLA Position
Quality indicator
Nb satellites
UTC time
HDOP
DGPS data age
DGPS station ID
oStream8
NMEA_GPGGA_frame_decoder_1
iLatLon
oPosition
oInitLatLon
TransformCoordinates_2
iXYPose
oGPSheading
GPSHeading_3
input_0
input_1
input_2
DataViewer_4
IMU running
Stream input
DTR state
RTS state
Stream output
SerialPort_2
iStream8
Accelerations (XYZ)
Rotation rates (roll, pitch, yaw)
Time
Temperature
oStream8
Crossbow_IMU_Family_1
Vector in
element0
element1
element2
Devectorizer_2
Fusion data
iVehicleState
oGPSlatitude
oGPSlongitude
oGPSheading
oX
oY
oHeadingAngle
oSpeed
oSteeringAngle
oTimestamp
oVehicleID
VehicleStateViewer_6
input_0
input_1
input_2
input_3
DataViewer_6
output
heading_conts
iPosition
iGyro
iDistance
iCompass
iSigmaPosition
oPose
FusionPosition_1
output
Float_constants_generator_4
iCurrentPose
iVelocity
iSteeringAngle
iInitialXYPose
oVehicleState
VehicleState_5
Manual control
iWarning
ospeedms
oanglerad
oWarning
intoutputAll
joystick
Initialization
trigger
output
Trigger
clicked0
Button_1
trigger
output
Destination
Planner
iStatus
iVehicleState
iDestination
oPath
oNbPoints
GlobalPlanner_1
iVehicleState
iPath
iNbPoints
oDynamicPath
otemp
oControlPoints
LocalPlanner_1
iVehicleState
iDynamicPath
oVelocity
oSteering
oLateralError
oHeadingError
oCurvature
oStatus
PathController_2
Controller
iSelection
iVelocity_0
iSteering_0
iVelocity_1
iSteering_1
oVelocity
oSteering
ControllerMux_1
Cycab low and high level
iSpeed
iAngle
oSpeedRef
oAngleRef
oCAN1
CycabDspic_High2Low_2
CAN in
CAN out
peak_pcanbasic_4
Input
Rear_Angle
Rear_Speed_L
Rear_Speed_R
Front_Angle
Front_Speed_L
Front_Speed_R
CANDecoder_10
iAngleR
iSpeedRL
iSpeedRR
iAngleF
iSpeedFL
iSpeedFR
oSpeed
oSpeedFR
oSpeedFL
oSpeedRR
oSpeedRL
oDistance
oAngleF
oAngleR
CycabDspic_Low2High_13
input
output
numbers_converter_1
Input
CANFrameViewer_2

Figure 5.3 – RTMaps architecture for testing on Cycab platforms

Different controllers have been implemented on the team for the platforms: a PD simple gains controller, a LQR controller and a KLSC controller on the Cycabs, whereas a simple gains controller was used on the Citroën C1. These controllers are breifly presented below.

1. First, a feedback proportional derivative (PD) simple gains controller was tested on simulation, on the cycab and later on the C1, where the control low is given in Equation 5.1, as presented in [González and Pérez, 2013]. The steering angle ψ is computed considering the curvature k, the lateral error $error_{lat}$ and the angular error $error_{ang}$, where α_1, α_2 and α_3 represent the controller gains for each of the parameters. This PD controller tries to keep the vehicle on the center of the trajectory, improving that way the tracking.

$$\delta = \alpha_1 \ k + \alpha_2 \ error_{lat} + \alpha_3 \ error_{ang} \tag{5.1}$$

Errors are calculated with respect to the look-ahead point for this and the following controllers. This point is placed in front of the vehicle at a defined look-ahead distance towards the following path point with the current vehicle heading [Lekkas and Fossen, 2012, Kuwata et al., 2009]. This look-ahead distance is defined in the planning stage. It takes a fix value for low-speed, since the errors in the localization do not increase that much the position and angular errors. However, it is speed-dependent for higher speeds, where the faster the vehicle is running the bigger look-ahead distance should be consider to compensate for possible location errors. Angular error is computed with Equation 5.2 and lateral error with Equation 5.3. On the one hand, angular error, also called heading error, is computed as the difference between the reference angle τ_{ref} (computed geometrically from the look-ahead point up to the next path point) and the measured angle τ_{meas} (obtained from the RTK-GPS).

$$error_{ang} = \tau_{ref} - \tau_{meas} = -asin(-cos(\delta_{meas})\ sin(\tau_{ref}) + sin(\tau_{meas})\ cos(\tau_{ref})) \tag{5.2}$$

On the other hand, lateral error is computed as the minimum error between the look-ahead point and each of the path segments.

$$error_{lat} = min(dist(position, pathSegment)) \tag{5.3}$$

2. A proportional Linear-Quadratic Regulation controller (LQR-P) high-level controller was used to enhance the tracking results obtained with the PD controller. This is an optimal feedback controller where the vehicle dynamics are represented by a set of differential linear equations. The reference steering is computed from Equation 5.4. There, the differential lateral error $dLat$ and differential heading error $dAng$ are considered, where α_{dLat} and α_{dAng} are their gains, respectively.

$$\delta = \alpha_1\ k + \alpha_2\ error_{lat} + \alpha_3\ error_{ang} + \alpha_{dLat}\ error_{dLat} + \alpha_{dAng}\ error_{dAng} \tag{5.4}$$

3. A Kinematic Lateral Speed Controller (KLSC) with dynamic gains has been used both in real platforms and in simulation experiments, enhancing the path following performance with respect to the prior controllers on low-speed platforms. Its operation is ruled by Equation 5.5, which dictates the behavior of the steering wheel, computing the reference steering angle of the vehicle.

$$\delta = \arctan(L(-K_\tau \sin(\tau_p) - \frac{K_\tau K_{lat} d_r}{v_u} + \frac{c(s)\cos(\tau_p)}{1 - c(s) d_r})) \tag{5.5}$$

where L is the length of the vehicle, τ_p is the heading error, i.e. the angular deviation of the car with respect to the reference path, and d_r represents the lateral error, i.e. the lateral deviation with respect to the center of the reference path. The term $c(s)$ denotes the curvature of the reference path at the closest point over the path, from the center of the vehicle rear wheels axis and v_u represents the longitudinal speed of the car. The gain K_θ, is the angular gain of the controller, used to moderate the controller sensibility to the heading and lateral errors and K_{lat}, is the lateral gain used to regulate the lateral speed of the car when approaching the path.

4. Finally, a cascade controller by state feedback with transfer function has been used in the Citroën C1. The control parameters are both lateral and heading error, and the curvature. Equation 5.6 presents the computation of the steering angle to correct the deviation on the trajectory tracking, where K_P, K_I and K_D are the proportional, integrative and derivative gains of the high level controller. Figure 5.4 shows the blocks diagram of the high level controller, where a P controller is applied for the curvature, a PI controller for the lateral error, and a PID controller for the angular error.

three different controllers are used for

$$\delta = \left(K_{P_{lat}} + \frac{K_{I_{lat}}}{s}\right) \, error_{lat} + \left(K_{P_{ang}} + \frac{K_{I_{ang}}}{s} + K_{D_{ang}}\, s\right) \, error_{ang} + K_{P_{curv}} \, curvature \tag{5.6}$$

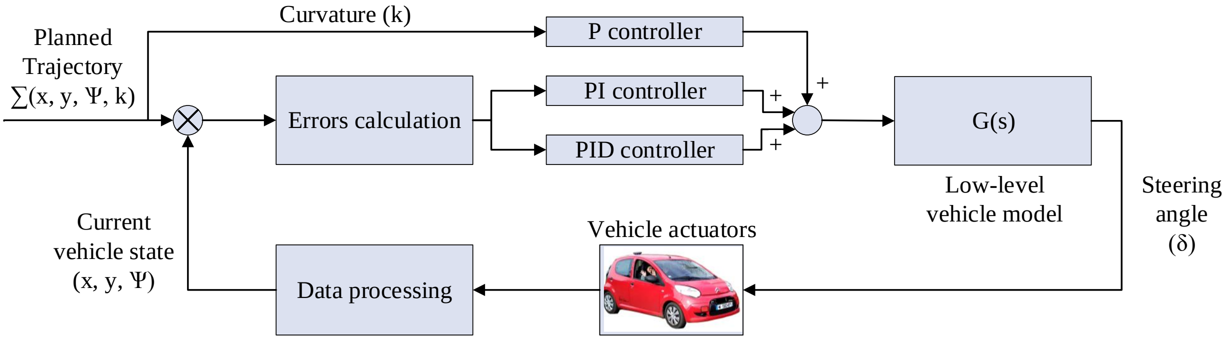


Figure 5.4 – High-level cascade lateral controller for the Citroën C1

5.2 Validation tests for the static path planner

This section presents the different experiments carried out to validate the local path planner for static environments presented in Chapter 3, first on simulation and then on real platforms.

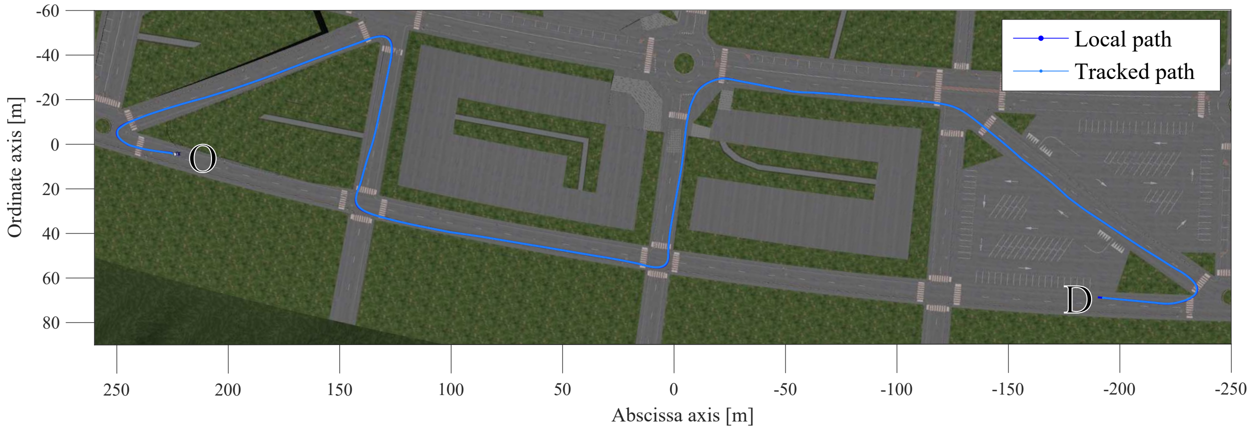
5.2.1 Results on simulation platforms

The two-staged planner has been designed to generate an enhanced G^1 continuous path from a departure to a destination point, dealing with any possible road configuration (such as right turns, narrow turns, slight turns, intersections, roundabouts, etc) on a structured urban environment as if they were turns.

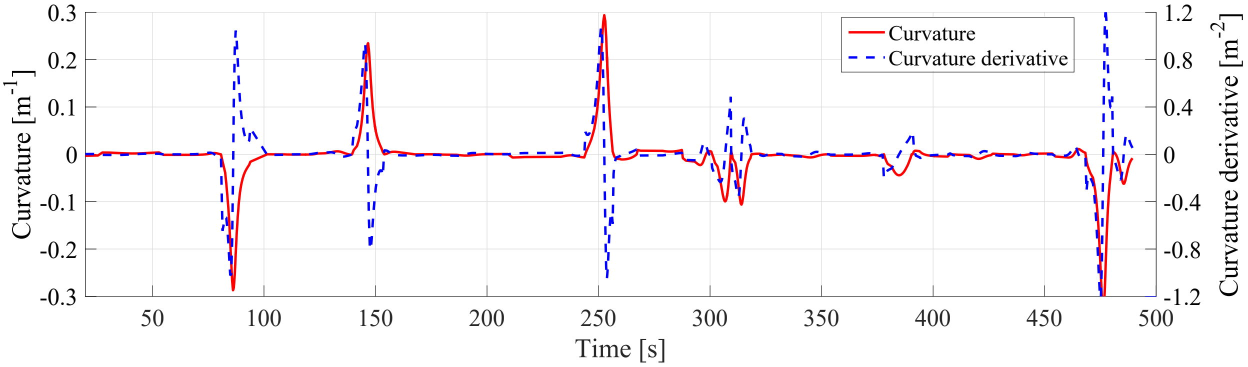
Figure 5.5 presents a simulated urban scenario on Pro-SIVIC, where the automated car running with the developed planning approach performs a path following scenario from origin point O to destination point D. Figure 5.5a shows both the path planned by the component developed in RTMaps (i.e. the local path, in dark blue) and the path tracked by the controller of the vehicle (in light blue). There, the itinerary has been chosen because of the following reasons:

- It presents several curves with different sharpness: where there are two narrow-turns, three right-turns and one long-turn less sharp.
- These curves present different concavity changes, since there are consecutive right turns, consecutive left turns and different turns with a concavitiy change, i.e. with different direction of rotation (from left-turn to a right-turn or vice versa).

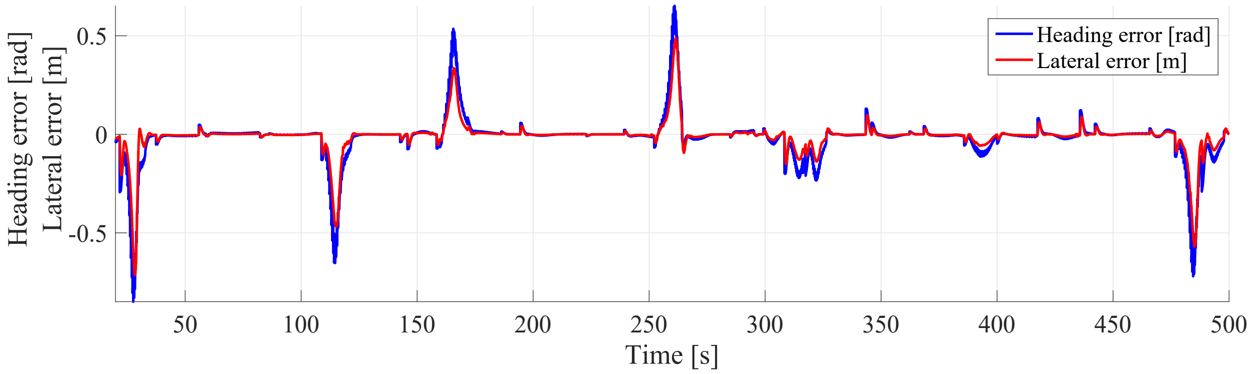
These characteristics allow the simulation to provide a realistic scenario to validate the planner, which considers the road layout in terms of sharpness, analyzing how difficult is to perform the curves and thus, offering a way to trade-off the available space between curves with the sharpness of the turns. Additionally, the changes on the direction of rotation were considered to provide a human-like driving style where the driver approaches to the external side of the lane when several consecutive turns are presented in the same sense.


Local path
Tracked path
O
D
Ordinate axis [m]
-60
-40
-20
0
20
40
60
80
250
200
150
100
50
0
-50
-100
-150
-200
-250
Abscissa axis [m]

(a) Planned and tracked paths



(b) Curvature and curvature derivative



(c) Control variables: lateral and heading error

Figure 5.5 – Comparison of planned and tracked paths on the simulation scenario

The variables considered on the optimality criteria are illustrated in Figure 5.5b, where curvature profile is drawn in red line and its derivative in dashed blue lines. As can be noticed, the maximum curvature in the whole path is of 0.3 m^{-1}, and the maximum curvature change is slightly higher than 1 m^{-1}. It means the path generated is continuous and leads to a smoother path tracking. Furthermore, the variables considered in the control algorithm are illustrated in Figure 5.5c, where blue line represents the heading error and red line the lateral error. There, lateral error is below 0.2 meters both for the slight turns and the right-angle turns, whereas a maximum lateral error of around 0.6 meters can be appreciated for the narrowest turn. These results confirm that the generated path meets the continuity and comfort requirements for vehicles running at low-speed.

Figure 5.6 presents a simulated urban scenario where consecutive curves are presented in a short period of time. The itinerary has been defined to show the performance of the approach on urban environments where multiple changes on the road configuration are found in a short time horizon. This scenario represents one lane road formed by four turns, where two of them are right turns and the other left turns, combining right-angle turns, sharper turns and a more opened turn.

Results with the proposed approach have been depicted together with results with a prior algorithm in the team, shown in [González et al., 2014], where the optimality criteria was considering only to reduce the curvature in the three most significant points of the curves, in contrast with the proposed approach which considers both curvature and curvature derivative in all the curve points, both for the upcoming curve as for the next one planned in advance.

A comparison of the proposed two-staged planning module have been carried out with respect to a fully real-time approach, as the one presented in [González et al., 2014] previously developed in the team, in order to validate the approach. There, the optimality analysis considers minimizing the curvature at the three most significant points, and the different paths are analyzed in real-time by changing the location of the control point in a established interval, considering a planning horizon of one curve, i.e. the optimality function evaluates only the upcoming curve.

Figure 5.6a shows in blue the paths for the proposed approach, whereas the paths for the approach in [González et al., 2014] are depicted in red. Besides, solid lines represent the planned local path, whereas the dotted lines represent the paths tracked by the controller of the simulated vehicle.

The variables of the planning optimality criteria are shown in Figure 5.6b, depicting both curvature profiles with solid lines, and curvature derivative profiles with dotted lines. Additionally, the control variables are shown in Figure 5.6c. There, lateral and angular errors for the proposed approach are depicted in blue and light blue respectively, whereas those for the entirely real-time approach are depicted in red and light red.

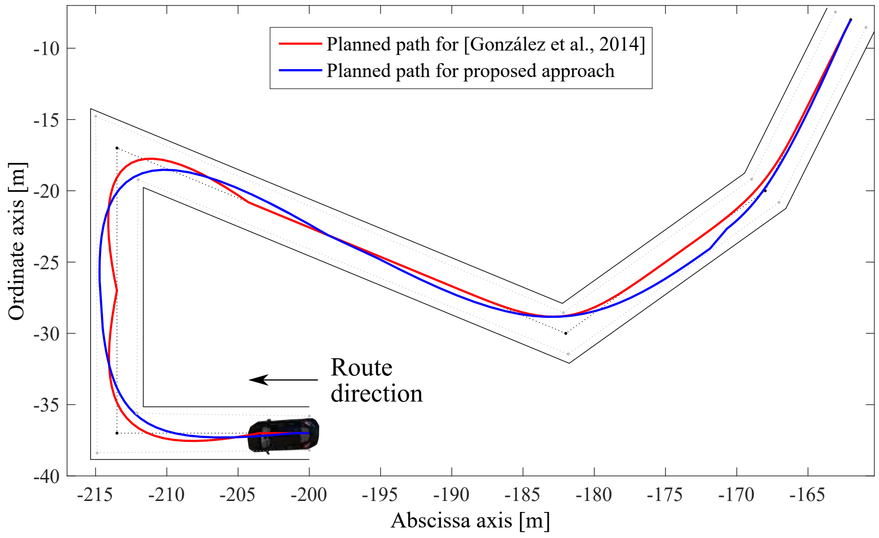
Table 5.1 summarizes the prior results, where an improvement of the curvature is appreciated. It is clearly manifested in lower peak curvature values: A reduction of around 0.2 $m^{-}1$ in the maximum curvature peak is observed with respect to the approach shown in [González et al., 2014], meaning an improvement of 32%. An improvement in the curvature derivative can be also appreciated in both peak and mean values. These improvements rely on the generation of smoother paths, which contribute to the comfort of the passengers, allowing to track smoother paths.

The results shown the human-like driving style in the performance for the two first curves. Since they are both right turns, the algorithm consider using the whole lane width without passing through the center of the lane, opening that way the trajectory. This is directly translated in a reduction of the maximum curvature, since the second turn is the sharpest one in the scenario. Same behavior can be noticed for the last two curves, but in the opposite direction. This improved performance is obtained thanks to the described optimality criteria, where the planning horizon was extended to two curves, not only optimizing the closest curve, but also the next upcoming one, considering to that end the information of three turns as explained in Section 3.3.2.

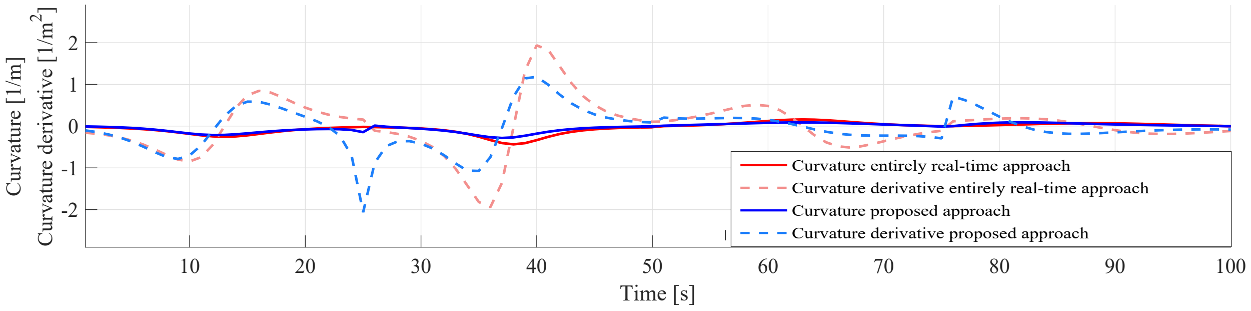
5.2.2 Results on real platforms

The developed local planner has also been tested on the experimental vehicles of the RITS team: both in the Cycabs and in the robotized electric Citroën C1, presented in Figure 5.7.

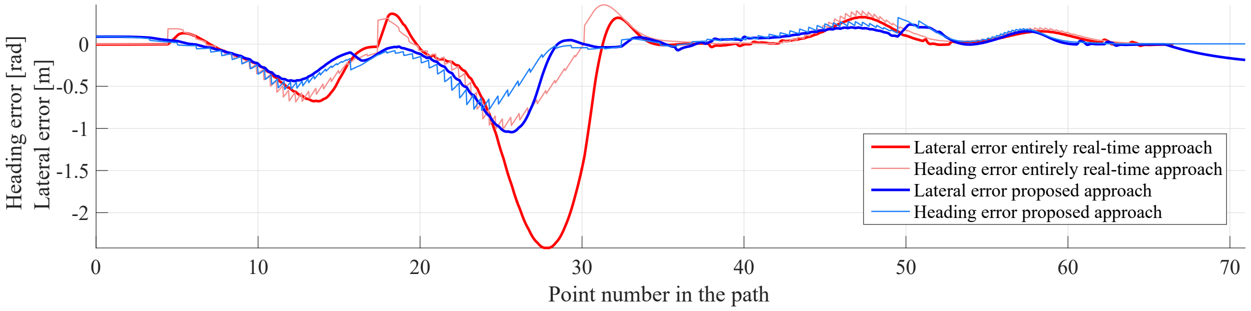
Thus, this section presents the experiments carried out with these platforms at the Inria-Rocquencourt facilities. Several itineraries were tested with both Cycab and Citroën C1. Among



(a) Planned and tracked paths



(b) Curvature and curvature derivative



(c) Control variables: lateral and heading error

Figure 5.6 – Comparison of planned and tracked paths on the simulation scenario

Table 5.1 – Comparison of comfort variables: curvature and curvature derivative

Algorithm	**Measurements**			
	$\boldsymbol{\lvert\mu_k\rvert\ (1/m)}$	$\boldsymbol{\lvert k_{max}\rvert\ (1/m)}$	$\boldsymbol{\lvert\mu'_k\rvert\ (1/m^2)}$	$\boldsymbol{\lvert k'_{max}\rvert\ (1/m^2)}$
González et al., 2014	0.0929	0.4295	0.4286	1.9522
Proposed approach	0.0657	0.2891	0.2916	1.2554

(a) Cybercar performing a curve during a path following test

(b) Citroën C1 performing a curve during a path following test

Figure 5.7 – INRIA RITS platforms running on automated way on the test track

them, the following can be highlighted.

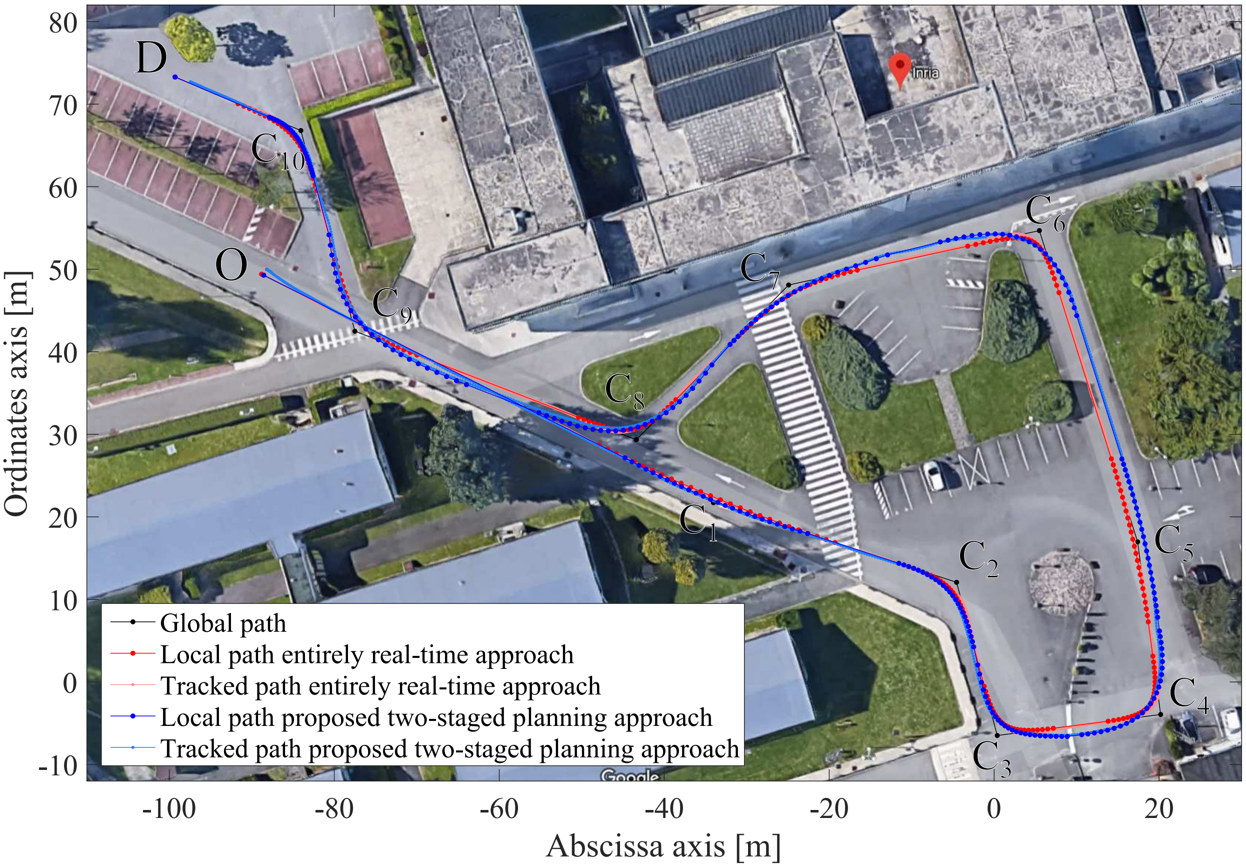
5.2.2.a Cycab experiments

Figure 5.8a shows a path following itinerary performed by the Cycab from the origin point O to the destination point D. The experiment has been performed at low-speed, 3 m/s, which is close to the maximum speed of these vehicles for a safe operation.

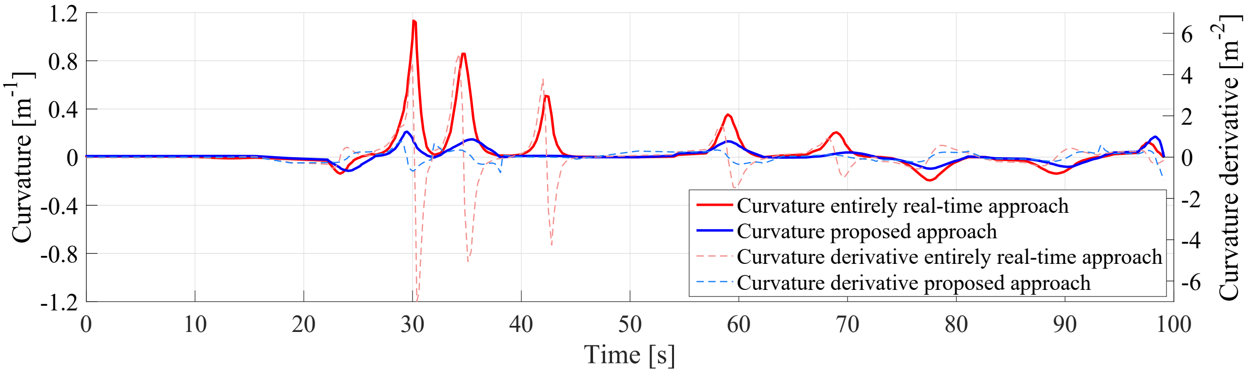
In the same way as in the simulation tests, this itinerary has been defined to present several intersections or some other road configuration on the road layout in a short time horizon, making it ideal for the kind of urban scenarios where these automated vehicles may operate in the short term without any human intervention.

Specifically, it presents first a straight stretch, then the vehicle performs a very slight left displacement in order to reduce the curvature of the following curve (C_1 curve), which is sharper and there is much less space to perform it. Afterwards, the vehicle performs the U-turn formed by curves from C_2 to C_4. Later, it takes the right exit of the roundabout performing the curve C_5 and leads to the right-angle C_6 facing the INRIA building. Then, the vehicle continues during a short period on the lane and then takes the left exit on the intersection, performing curve C_7, and returns to the main road with curve C_8. Finally, the vehicle goes straight ahead a few meters and then performs curve C_9 to enter in the parking slot, and curve C_{10} to arrive to the destination D.

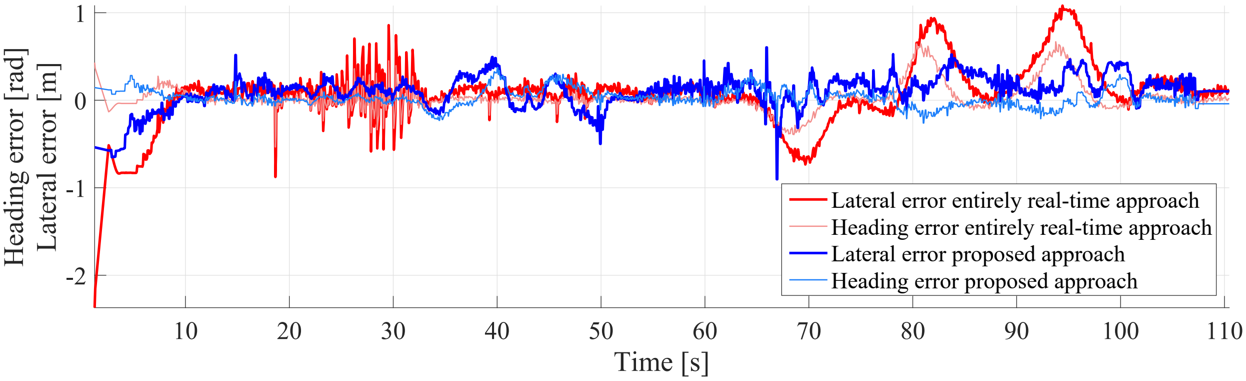
In the same way as in the simulation experiments, the proposed approach has been compared with entirely real-time planning approaches such as the one in [González et al., 2014]. Figure 5.8b


D
O
C_{10}
C_9
C_8
C_7
C_6
C_5
C_4
C_3
C_2
C_1
Inria
Google
Global path
Local path entirely real-time approach
Tracked path entirely real-time approach
Local path proposed two-staged planning approach
Tracked path proposed two-staged planning approach
Ordinates axis [m]
80
70
60
50
40
30
20
10
0
-10
-100
-80
-60
-40
-20
0
20
Abscissa axis [m]

(a) Paths planned and tracked by the Cycab on the defined itinerary



(b) Curvature and curvature derivative of the path



(c) Control variables: lateral and heading error

Figure 5.8 – Path following itinerary performed by the Cycab platform at Inria-Rocquencourt

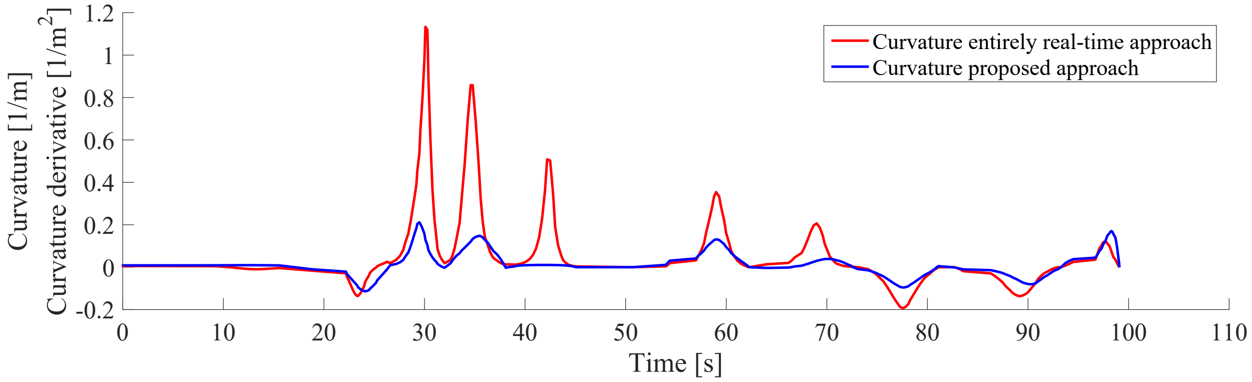


Figure 5.9 – Curvature details of the path in Figure 5.8a

shows the value of the variables considered in the planning optimization process, i.e. curvature and curvature derivative, depicted in blue and light blue for the proposed approach, and in red and light red for the other approach, respectively. Additionally, the control variables, i.e. lateral and angular errors are depicted using same colors criteria in Figure 5.8c.

An analysis of the results obtained from both continuity of the path and comfortability points of view is presented below.

First of all, it is worth noting a big difference between both curvature derivative profiles, where the maximum value for the proposed approach is of 0.75 m^{-2} whereas the maximum one for the other approach is of 5 m^{-2}. This reduction in the curvature changes means the generation of trajectories with much less abrupt movements of the steering, which is translated into a smoother path tracking.

Since there is such a gap between curvature derivative profiles, Figure 5.9 presents the curvature profiles for both systems separately. There, the curves with bigger difference in the curvature profiles are the ones corresponding to the U-turn, i.e. curves C_3, C_4 and C_5, being that difference of 0.9074 m^{-1}, 0.7102 m^{-1}, and 0.49 m^{-1}, respectively. This improvement in the planning variables directly comes from the human-like driving behavior provided to the system. For instance, since those three are left turns, the planning strategy decides to place the joint point between curves using the lane width up to the limits defining the safe area. Furthermore, a better distribution of the space between curves is a key element. Analyzing the junction between curves C_3 and C_4, it can be noticed that the proposed approach starts the curve C_4 much before, and closer to the end of C_3 since both are sharp turns. Same behavior can be noticed but in a smaller scale for curves C_6, C_7, C_8 and C_9. The generation of smoother curves makes possible to reduce the errors in the path controller. Figure 5.8c shows both lateral and longitudinal errors for the whole path. There, a reduction on both the lateral and heading error is observed, mostly in turns with less space available among them, such as curves C_8, C_9 and C_{10}.

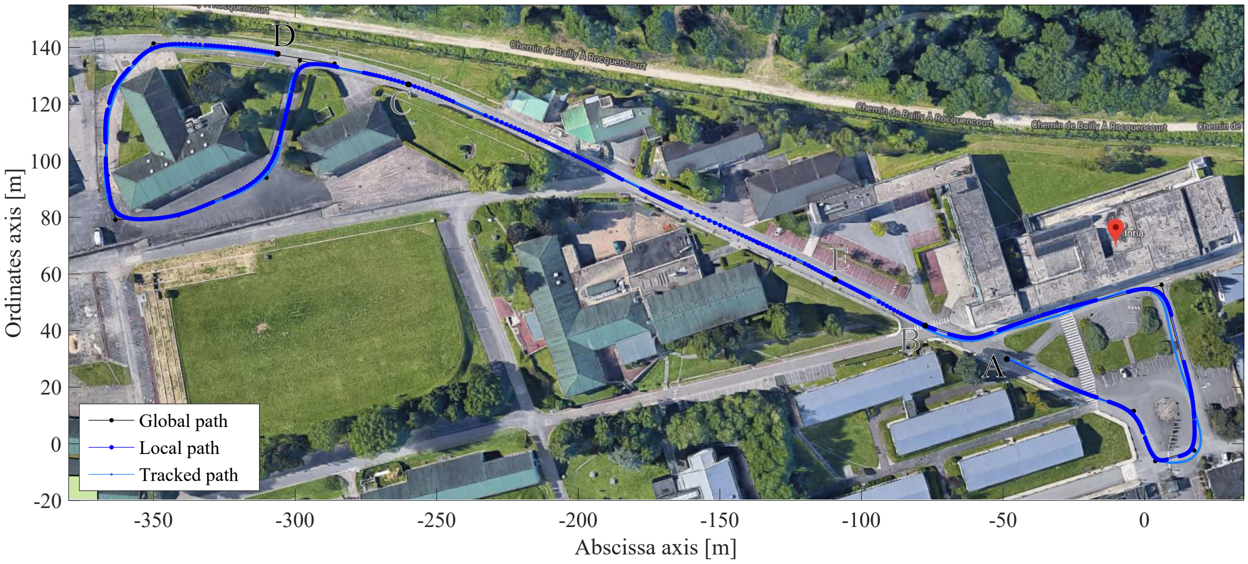
In addition to the improvement in terms of physical path, a faster planning is also achieved thanks to the use of optimal curves loaded from the databases generated in the pre-planning stage. The execution time of both algorithms has been compared, showing a reduction of the mean computation burden with the proposed approach, being 0.0341 s the the real-time planning time for the proposed approach, whereas 0.1122 s is the one for the fully real-time algorithm.

Thus, the analysis of the results has confirmed that the static planning strategy offers good results at low-speeds in terms of path continuity, generating smooth paths making possible an easier path tracking in the control stage.

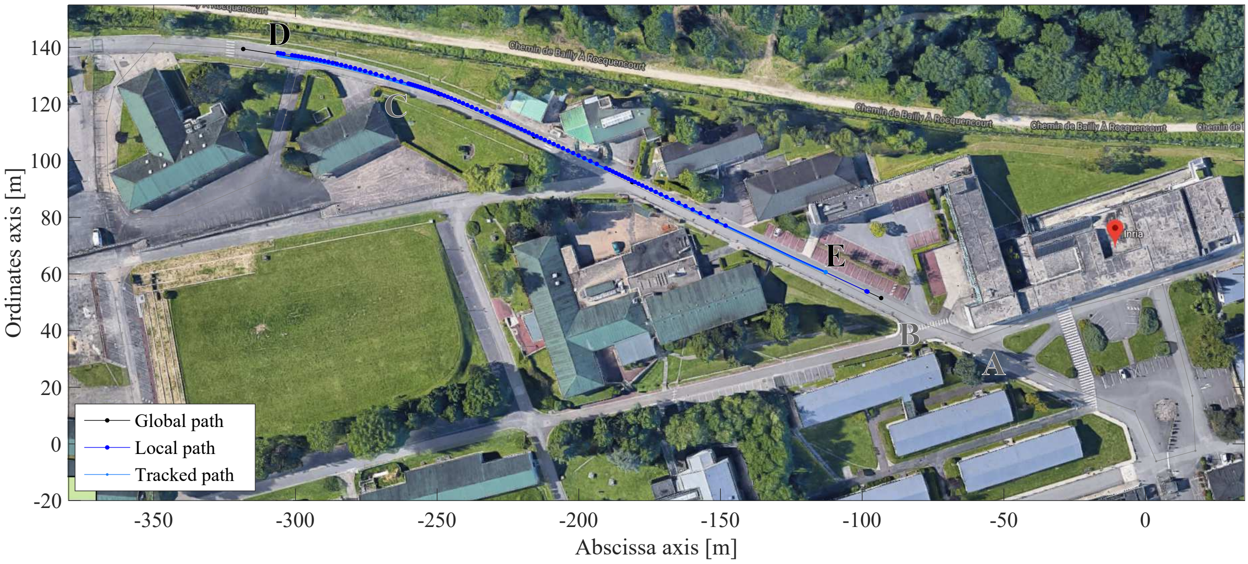
5.2.2.b Citroën C1 experiments

Previous subsection validated the proposed planning approach in static environments for low-speed vehicles. This subsection shows how this approach can be used for passenger vehicles such as the Citroën C1 car of the INRIA RITS team, running at medium-speed.

Figure 5.10 shows the path following itinerary performed at the INRIA-Rocquencourt test track, from point A to point E. performed to validate the approach. There, two areas can be distinguished: First, a *one-way* path from point A to D (Figure 5.10a, and then a *return* path from point D to point E (Figure 5.10b). There, the vehicle performs the path at a speed of 25 km/h, since the maximum speed of the track is 30 km/h and due to limitations on the available controller of the car.


D
C
E
B
A
Inria
Chemin de Bailly A Rocquencourt
140
120
100
80
60
40
20
0
-20
Ordinates axis [m]
-350
-300
-250
-200
-150
-100
-50
0
Abscissa axis [m]
Global path
Local path
Tracked path

(a) Paths planned and tracked on the defined itinerary: A-D area


D
C
E
B
A
Inria
Chemin de Bailly A Rocquencourt
140
120
100
80
60
40
20
0
-20
Ordinates axis [m]
-350
-300
-250
-200
-150
-100
-50
0
Abscissa axis [m]
Global path
Local path
Tracked path

(b) Paths planned and tracked on the defined itinerary: D-E area

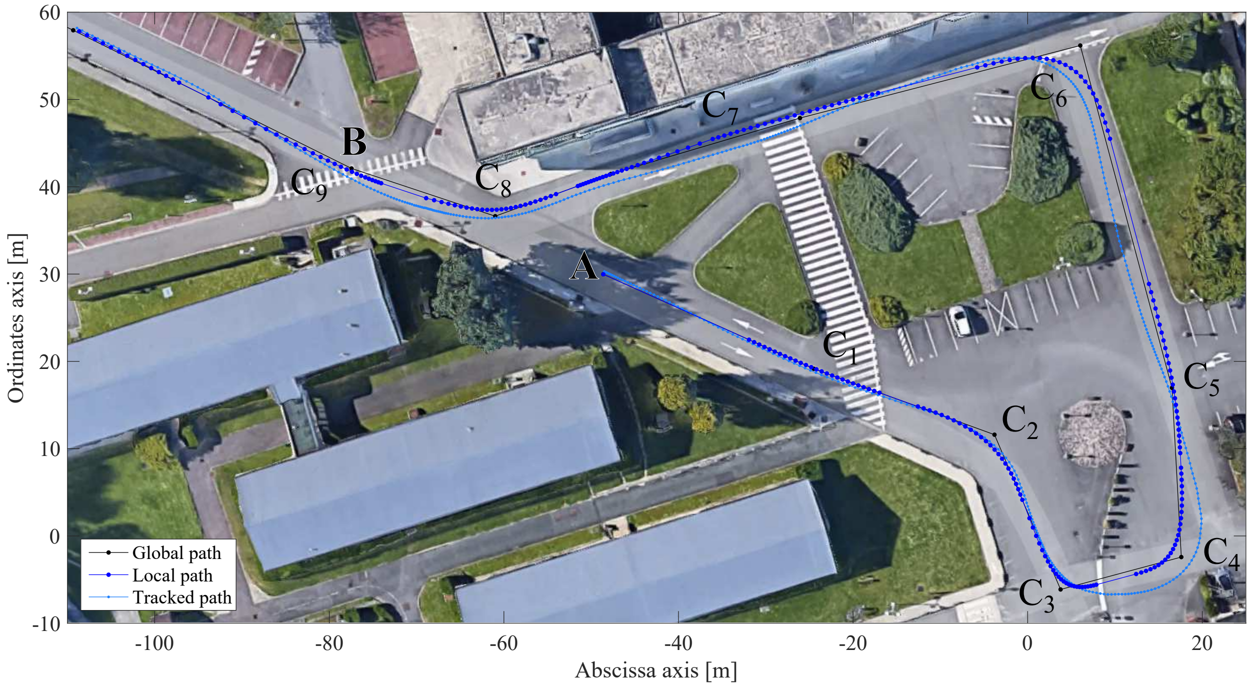
Figure 5.10 – Path following itinerary performed by the Citroën C1 platform at Inria-Rocquencourt

The designed itinerary was developed to test not only the planning approach proposed on this work, but to validate the high-level controllers for the actuators as well (both for the steering wheel and the accelerator and brake pedal) as part of the work of other team members. For that reason, it combines curved areas with straight stretches. Since the strength of the planning algorithm is to deal with changing scenarios, the curved areas of the itinerary are studied in detail: First area, shown in Figure 5.11 considers the points from the departure point A to the point B. Second area, shown in Figure 5.10b, consists of the itinerary between points C and D.

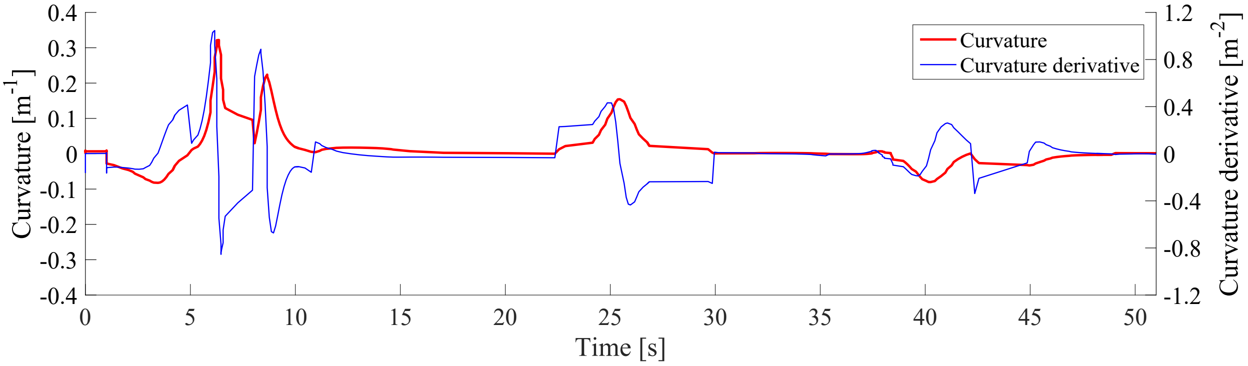
An analysis of the paths for the first area (Figure 5.11a) is extracted below. For that purpose, Figure 5.11b shows the curvature (in red) and curvature derivative (in blue) profiles for the planned path. Meanwhile, Figure 5.11c shows the corresponding lateral (dark blue) and angular (light blue) errors.

The planned path for the part of the itinerary common to the experiment with the Cycab platforms is not the same. This is due to the consideration of the kinematics of the vehicle in the planning stage. For instance, the resulting path for the curves C_3 and C_4 uses less lane space, that is, is closer to the center of the lane. This is because of the higher width of the C1 with respect to the Cycab, which reduces the safe space to place the control points (Convex Hull polygon) and then the junction points between curves. It also helps to perform the curve when the vehicle is traveling at higher speeds, like in this experiment where the car is running at 25 km/h. Although the generated path considering the kinematic and dynamic characteristics of the vehicle is less flexible, it still maintain a smooth behavior. Curvature profile is continuous and its maximum value is 0.32 m^{-1} for the first turn and 0.2 m^{-1} as much for the other sharp turns. In addition, the changes on the curvature are also reduced, being always below 1 m^{-2}. Although the lateral error in the sharp turns is important, it is based on the available simple-gains controller on the C1 platform, which leads to higher errors for medium-speeds. As has been shown before, the smoother the path planned, the better the path tracking on the controller. Thus, the application of the planning approach with a more precise high-level lateral controller is envisaged for future works.

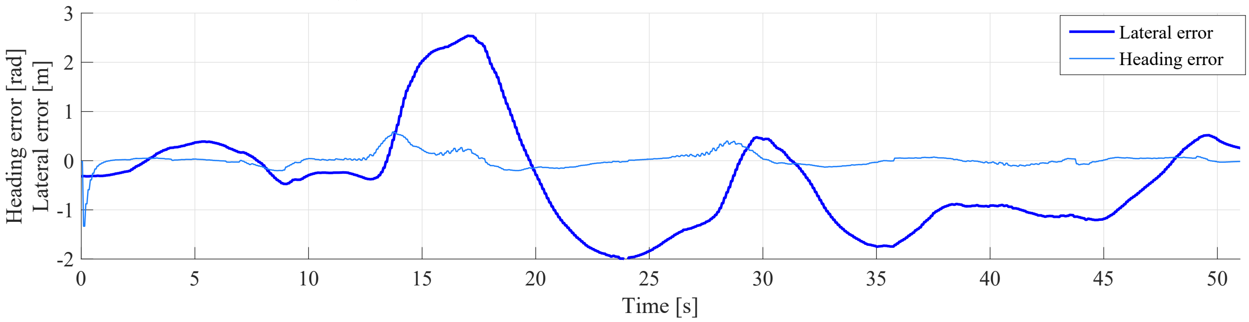
Figure 5.12a shows the path for the area C-D, where its curvature and curvature derivative profiles are depicted in Figure 5.12b, and the lateral and angular errors of the controller in Figure 5.12c. Similar results are obtained in terms of curvature continuity and smoothness. As can be noticed in the curvature profile, even for the narrowest turn performed with curve C_{10}, the curvature value is low (0.209 m^{-1}). It is worthy to mention that the curvature of curve C_{10} is that low thanks to the use of the road width up to the feasible limits to plan a more opened path. Additionally, it has been shown how the curvature change between curves C_{10} and C_{11} make the vehicle to pass through the center of the lane between both curves, but not abruptly but using the whole space between way-points searching the best junction point for both curves.


A
B
C1
C2
C3
C4
C5
C6
C7
C8
C9
Global path
Local path
Tracked path
Ordinates axis [m]
Abscissa axis [m]
60
50
40
30
20
10
0
-10
-100
-80
-60
-40
-20
0
20

(a) Area A-B from paths of Figure 5.10a

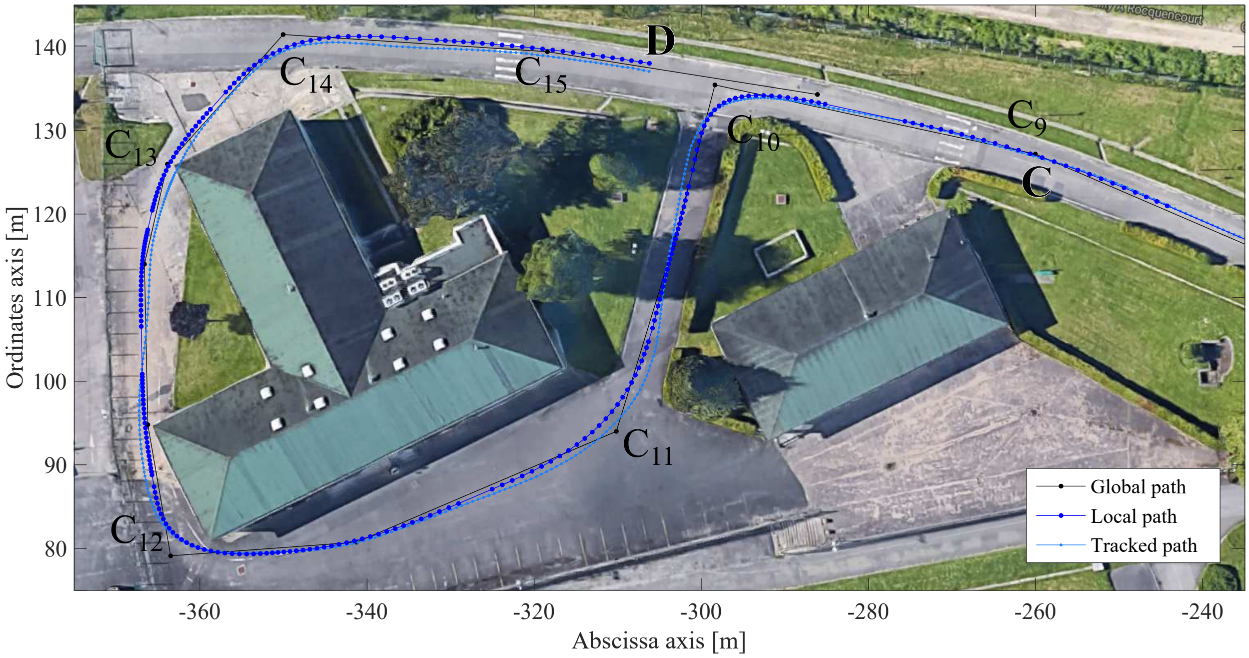


(b) Curvature and curvature derivative of the path

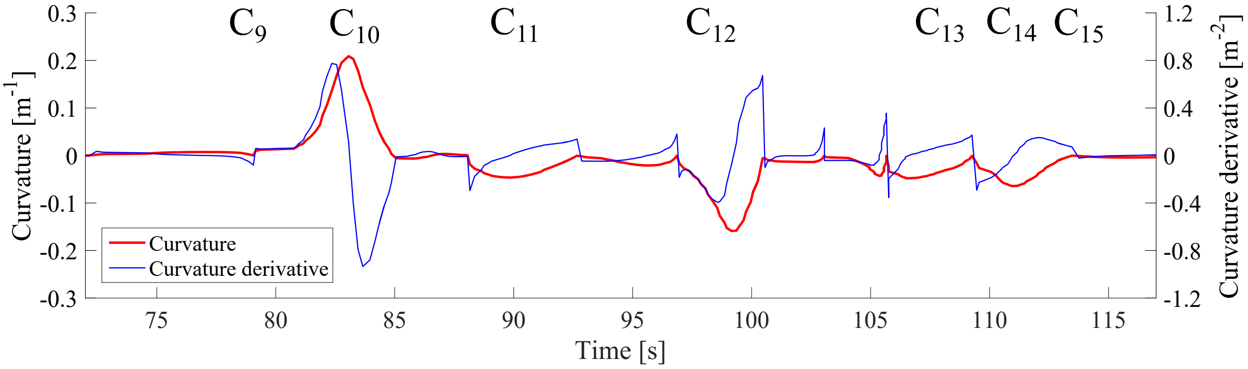


(c) Control variables: lateral and heading error

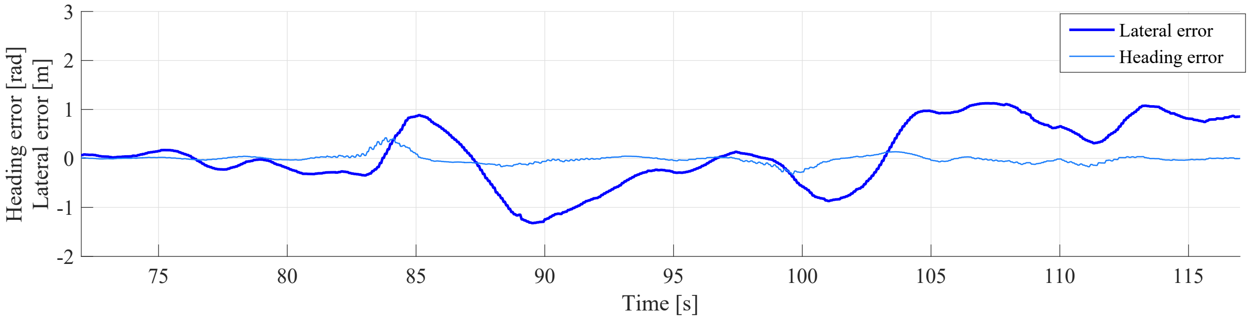
Figure 5.11 – Path following itinerary from Figure 5.10: A-B area


D
C14
C15
C13
C10
C9
C
C11
C12
Global path
Local path
Tracked path
Ordinates axis [m]
Abscissa axis [m]
140
130
120
110
100
90
80
-360
-340
-320
-300
-280
-260
-240

(a) Area C-D from paths of Figure 5.10a



(b) Curvature and curvature derivative of the path



(c) Control variables: lateral and heading error

Figure 5.12 – Path following itinerary from Figure 5.10: C-D area

5.3 Validation tests for the dynamic path planner

Previous section shown the experiments carried out to validate the planning approach on static urban environments, performing a path following from an origin to a destination point. This section introduces the experiments carried out to validate the planning approach for dynamic environments presented in Chapter 4, where the vehicle has to deal with obstacles found in the path, modifying the already planned static path to avoid them in a safe and smooth way.

5.3.1 Results on simulation platforms

For this purpose, the algorithm has been tested on simulation under the following use cases: First, the dynamic local planner is validated for scenarios where static obstacles are found in the path, in the same lane as the automated vehicle is running. Second, the performance of the system in the presence of dynamic obstacles in the path with different speed profiles (both constant and non-constant speeds) is presented, where the adaptability of the path is achieved considering dynamics of both ego and avoided obstacle. Third, the response of the system when unexpected obstacles are detected in the other lane is presented, forcing the vehicle to make a safe maneuver to return to the original lane.

Since the static planning has already been validated, as a matter of simplicity, the scenario created on Pro-Sivic to validate the dynamic planner on the defined use cases considers an itinerary representing only a straight stretch being part of a whole path following itinerary combining cured-areas and straight stretches. That way, we focus on avoiding the obstacle on the straight stretches where there is enough space to perform the avoidance maneuver.

The information of the obstacles usually comes from the perception stage, where the lasers on the vehicle provide the distance at which they are separated from the ego-vehicle position, as well as the Oriented Bounding Box representing the width of the obstacle (in case of 2D lasers) and the object type. To facilitate the validation experiments, the position of the obstacles is well-known during the simulation. Thus, the distance to the obstacles is computed at all the time.

5.3.1.a Static obstacles scenario

Figure 5.13a shows the experiment carried out to validate the approach on scenarios where static obstacles are found in the path. The ego-vehicle is the white passenger car and is located in the center of the right lane, whereas the obstacle is a Cybus platform located fifteen meters ahead of the ego-vehicle, represented in black color. First of all, the grid is generated centered on the vehicle's front position, where the lasers are located, allowing it to perceive the different objects on the road, and classify them later. It is represented with grey lines in the Figure. This grid is setted-up as explained in Section 4.3.1. In this example, the chosen resolution has been cells of one meter, that way the lane width is discretized in three cells. In spite of the position of the vehicle is well-known, the construction of the virtual lane is done like if this information had come from the perception stage. Then, a classification of the obstacle is done according to the width of the obstacle, computed from the obstacle Oriented Bounding Box (OBB). If the lasers equipped are 2D, like in our case, the information coming from the OBB only give us the information about the width of the rear part of the obstacles. From this information, the classification is done according to the relationship between length and width of the different road user types, being in our case: vulnerable, cybercar, car and bus or truck. After being classified as a cybercar, the length of the vehicle is estimated, and the Bounding Box of the obstacle is updated with that information, as shown in the Figure 5.13a with a red rectangle surrounding the obstacle. The grid is then updated, setting the cells where the obstacles lie as occupied, represented with garnet crosses in the Figure. The safety area for the obstacle avoidance maneuver is computed, maintaining the longitudinal and lateral safety distances, which are in this case the length of the ego-vehicle and the limit of the lane, respectively. The points of the safety area lying in the line separating right and left lanes are points of the virtual lane, k_1 and k_2 respectively, as explained in Chapter 4. Once this area is computed, from the internal vertices are computed the points swp_2 and swp_3, which are the way-points located in the center of the left lane allowing the static planner to compute the curves to perform both lane changes. These two points are computed applying the angle φ_2, which longitudinal separation of the way-points with respect to the obstacle position, making the lane change maneuver end further or closer to the obstacle. For natural criteria, this angle φ_2 is prioritized to zero leading to more natural lane changes. On the other hand, the angle φ_1 is the slope for the lane change maneuvers. Thus, the algorithm evaluates the different curves changing these angles in the available space between both vehicles. There, two curves are evaluated for the first lane change and another two for the second lane change, and the only parameter that the static planner searches in real-time is the junction point between curves. It starts to search from the least values of the slope φ_1, prioritizing that way smoother lane changes. The optimality criteria for the curves evaluation is the one described in Chapter 3.

After applying the angles φ_1 corresponding to the optimal curves found for both maneuvers, the way-points defining the beginning and the end of the two lane changes are obtained (swp_1 and swp_4 respectively, being placed in the center of the lane. The virtual lane is therefore generated geometrically, applying half of the lane width to both sides. This virtual lane defines the new road limits, considered in the static planner to ensure maintaining the ego-vehicle in the road without invading the sidewalk.

Figure 5.12a shows in blue the resulting planned path for the avoiding maneuver, whereas the path performed by the vehicle is depicted in light blue. This represents the optimal solution found after the curves evaluation, where the corresponding parameters are $\varphi_1 = 65^{\text{o}}$ and $\varphi_2 = 20^{\text{o}}$.

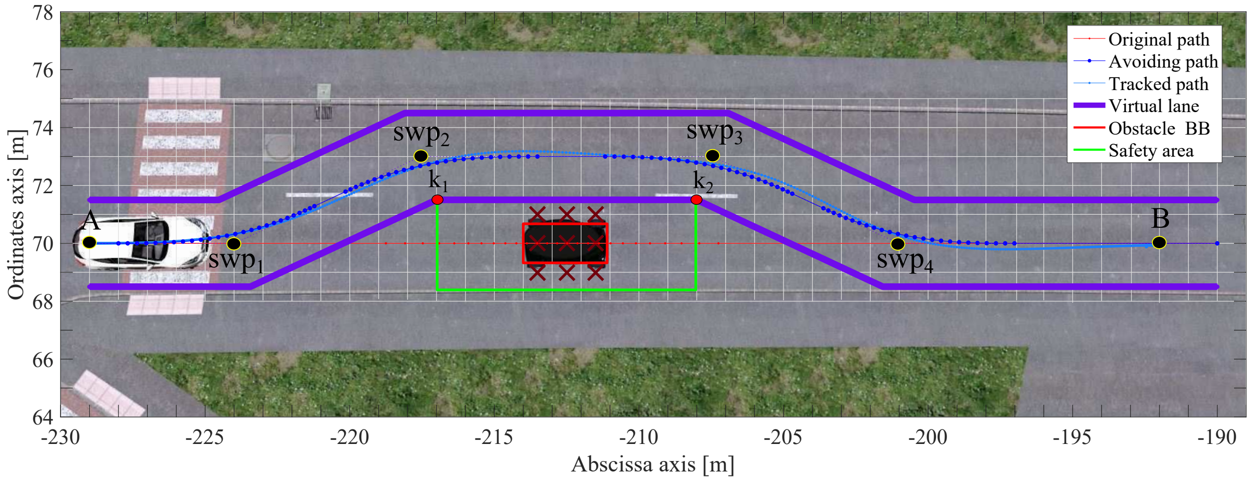
Figure 5.13b shows the curvature and the curvature derivative profiles for the avoiding maneuver. It can be noticed that the maximum curvature is lower than 0.1 m^{-1}, and the maximum curvature change is of about 1 m^{-2}. These values confirm the smoothness of the generated avoiding path. Additionally, Figure 5.13c shows that the lateral error while tracking the avoiding path keeps below 0.4 meters, thanks to the contribution in terms of smoothness of the path planned.

5.3.1.b Dynamic obstacles scenario

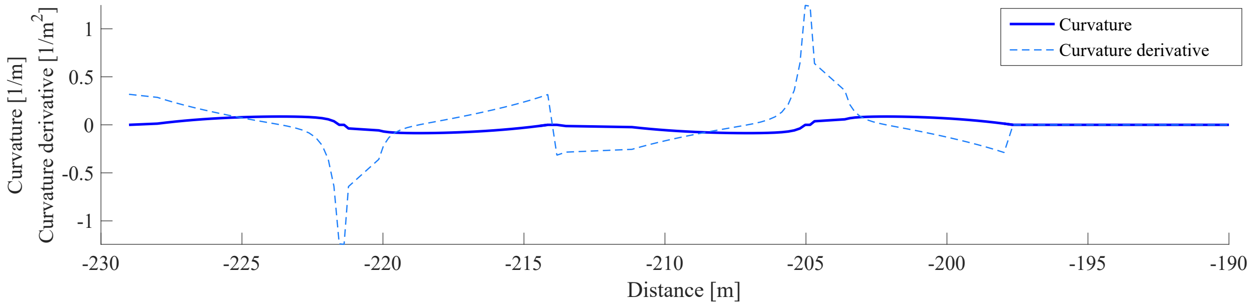
In addition to adapting the path to static obstacles found in the path, the proposed dynamic planning approach also deals with dynamic obstacles found in the path. Figure 5.14 shows a scenario where the ego-vehicle finds a moving obstacle blocking the path. First, as described in the chapter 4, the algorithm predicts the worst case possible, where obstacle presents maximum speed and acceleration. That way, the Bounding Box of the obstacle is updated considering this information about the obstacle dynamics. The prediction will make that the virtual lane is recomputed by projecting the way-points conforming the new global path for the two lane change maneuvers.

Thus, 5.14a shows in blue the new planned avoiding path, where the performance of the ego-vehicle is depicted in light blue. As can be seen, by considering the vehicle dynamics, what it is done is to extend the lane keeping phase of the overtaking maneuver. The associated curvature and curvature derivative profiles are presented in Figure 5.14b. As for the static obstacles scenario, the values of both curvature and curvature derivative keep low, contributing to the comfort of the path. In addition, the errors from the simulated vehicle performance are as low as for the static obstacles use case, confirming that the approach adapts to dynamic environments without being affected the tracking of the path.

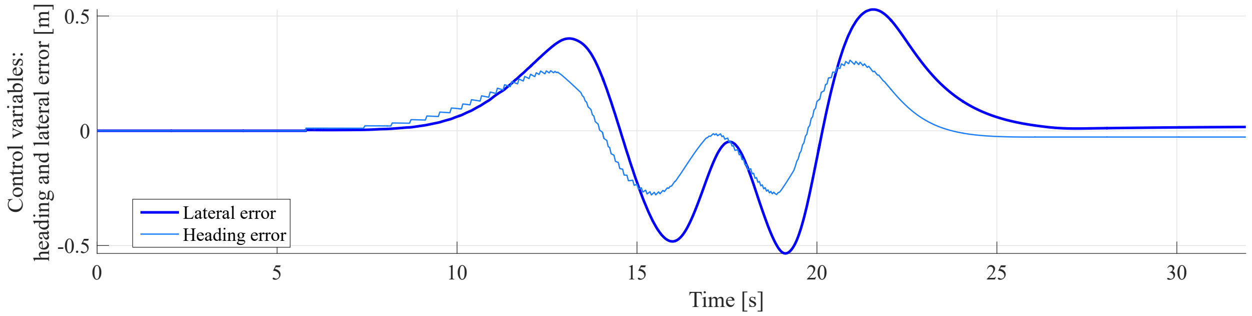
One of the great advantages of this method is that the path does not have to be recomputed all the time, with difference to other approaches based on lattices. Only when either the difference



(a) Original path, avoiding planned path and tracked path by the simulated vehicle



(b) Curvature and curvature derivative of the path

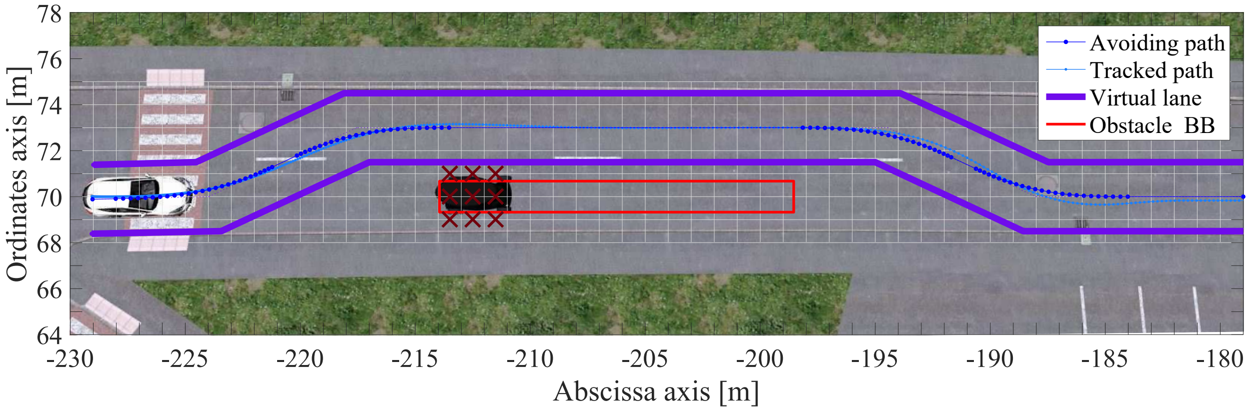


(c) Control variables: lateral and heading error

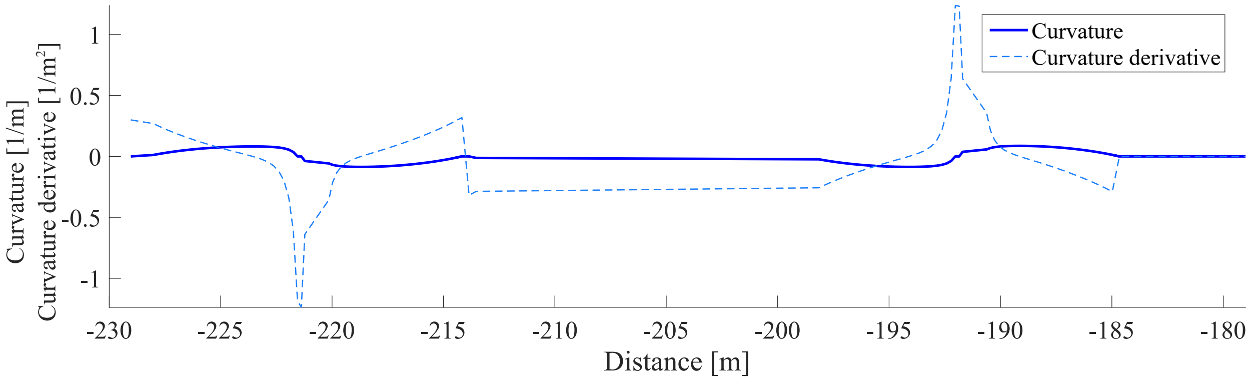
Figure 5.13 – Dynamic path on a straight stretch scenario with static obstacles

between expected distance traveled by obstacle or ego-vehicle is higher than the specified threshold, or the time to collision is below six seconds, the re-planning process is started.

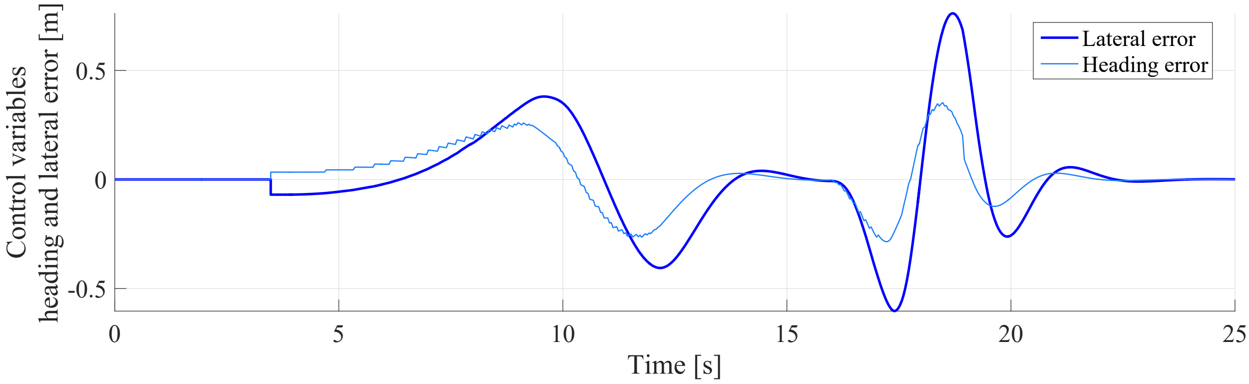
Figure 5.15 shows an application of the algorithm on a changing scene where there is a wrong classification of the type of obstacle in front of the ego-vehicle. First, after receiving the coordinates of the obstacle in front, it is classified as a cybercar, according to the vehciles classification already shown in Chapter 4. Then, their dimensions are predicted, obtaining its predicted Bounding Box represented in red in Figure 5.15a. A first virtual lane is computed (depicted in purple), and the planning algorithm plans a path on it to arrive to the destination point. However, when the ego-vehicle is in the point A, the perception system detects that the obstacle dimensions have changed, being represented in garnet. In that moment, a re-computation of the virtual lane is



(a) Predicted obstacle position, avoiding planned path through the virtual lane construction and tracked path by the simulated vehicle



(b) Curvature and curvature derivative of the path

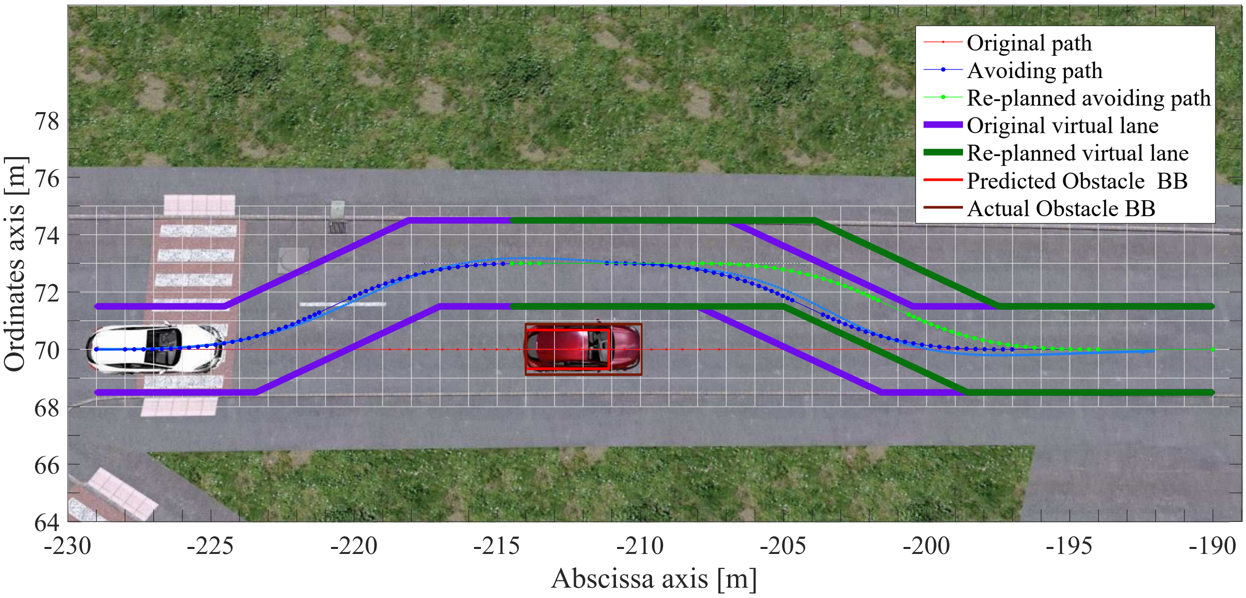


(c) Control variables: lateral and heading error

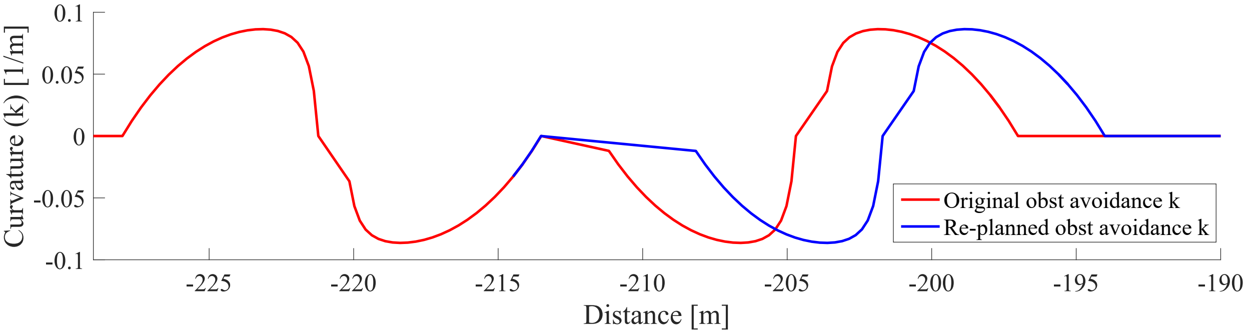
Figure 5.14 – Dynamic path on a straight stretch scenario with dynamic obstacles

done according to the actual dimensions of the vehicle, as depicted in dark green, and a new path is planned on this new virtual lane, allowing the vehicle to adapt its trajectory to the destination point in a fast way. Additionally, a continuous curvature profile is kept, as shown in Figure 5.15b, where the curvature is always below 0.2 m^{-1} and the curvature derivative is bounded to X m^{-2}. It eases the tracking of smooth lane changes curves for avoiding the obstacle when some unexpected situations arise.

This way, the problem of finding a smooth path for avoiding both static and obstacles is reduced



(a) Original and recomputed virtual lane and re-planned path

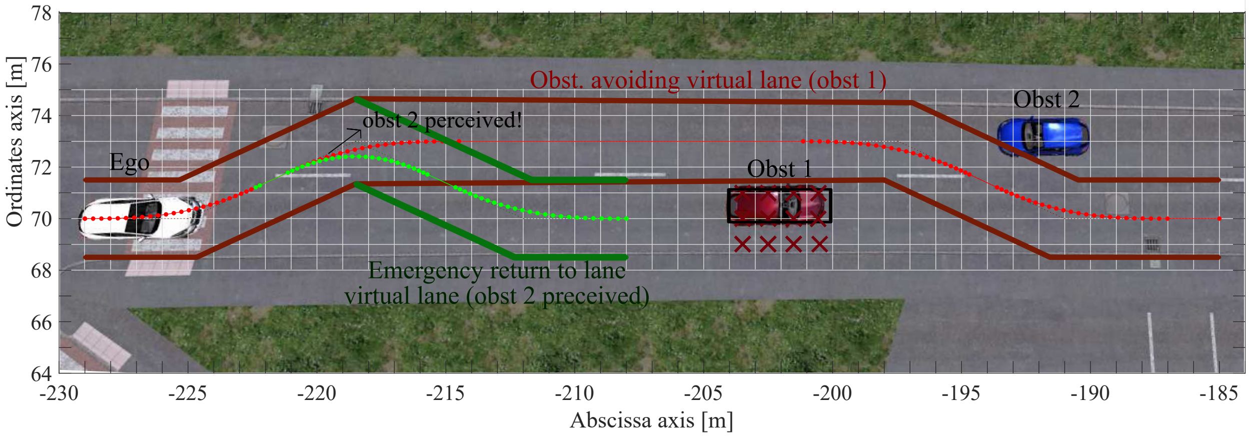


(b) Curvature of the path

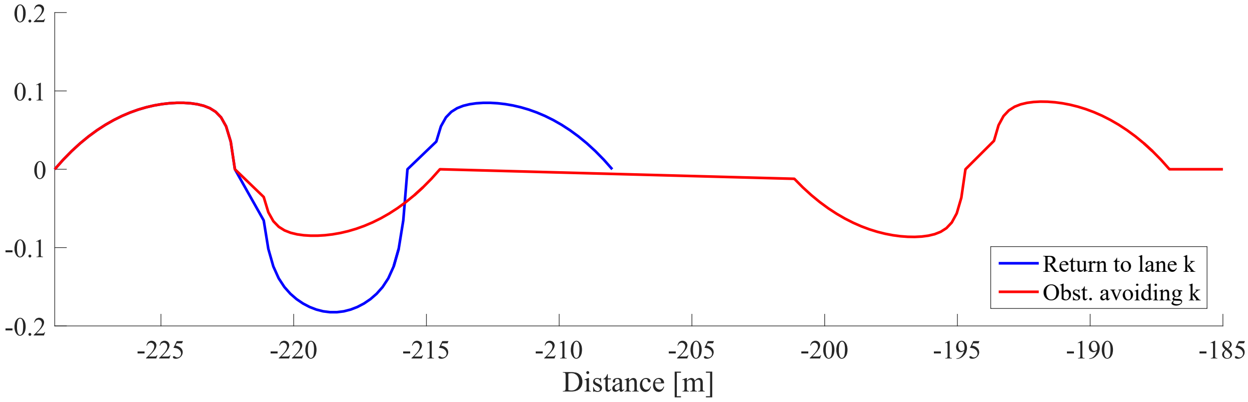
Figure 5.15 – Virtual lane re-computation and path replanning applied to a wrong prediction of the obstacle dimensions

to the modification of the global path in the straight stretch, adding additional way-points which allow the dynamic planning algorithm to make use of the static planner to evaluate the pair of curves for each lane change, by changing only the slope of the maneuvers and the desired separation with respect to the obstacle.

Finally, the algorithm is able also to react to unexpected circumstances, like for example when an obstacle is found in the left lane once the avoiding maneuver has started, forcing us to make a safe return to the lane. The algorithm computes in these cases an emergency path to perform this maneuver aborting the original lane change, as shown in Figure 5.16a. First, ego vehicle starts the maneuver for avoiding the obstacle Obst1, whose planned path and corresponding virtual are depicted in red and dark red respectively. When the ego-vehicle perceives that another obstacle (Obst2) is in the left lane, blocking the planned path, it switches from the avoiding path to the safe return to the original lane one, whose path and virtual lane are depicted in light green and dark green respectively. As can be seen in Figure 5.16b, the algorithm ensures performing a smooth path even in emergency situations such the described. The maximum curvature reached for the emergency maneuver (depicted in blue) is bounded below to 0.2 m^{-1}, keeping a continuous curvature profile, where the maximum curvature change is produced in the switching process between paths. It means an increment of only 0.1 m^{-1} on the maximum curvature with regard to the original obstacle avoidance path in the same point.


Ordinates axis [m]
78
76
74
72
70
68
66
64
Obst. avoiding virtual lane (obst 1)
Obst 2
obst 2 perceived!
Ego
Obst 1
Emergency return to lane
virtual lane (obst 2 preceived)
-230
-225
-220
-215
-210
-205
-200
-195
-190
-185
Abscissa axis [m]

(a) Obstacle avoiding and re-computed emergency return to the lane paths and virtual lanes



(b) Curvature and curvature derivative of the path

Figure 5.16 – Emergency return to the lane re-computation of the virtual lane, aborting the original avoiding maneuver

The described dynamic planning strategy improves the previous works on the team [González et al., 2014] in terms of path smoothness and computation cost, thanks to the use of the described static planner which generates the optimal path for every virtual lane for the obstacle avoidance maneuvers. A re-planning strategy to avoid re-computing the path at every planning period is presented as well. The adaptability of the algorithm to any circumstances let us consider emergency situations forcing to abort lane change maneuvers and generating a safe trajectory to return to the original lane. These situations are handled just by computing a new virtual lane modifying the location of the way-points according to both ego-vehicle and obstacle state.

Chapitre 6

Conclusion

Below is a French summary of the following chapter "Introduction".

Ce chapitre présente d'abord les conclusions tirées après l'achèvement de ce travail de trois ans. Ensuite, les contributions relatives à l'état de l'art sont extraites. Enfin, quelques directives concernant les orientations futures du sujet de planification des trajectoires dans la conduite automatisée sont données.

Chapter 6

Conclusion

This chapter presents first the conclusions extracted after the completion of this three-years work. Then, the contributions with respect to the state of the art are extracted. Finally, some guidelines regarding the future directions of the path planning topic in automated driving are given.

6.1 Conclusions and remarks

Development and commercialization of ADAS have risen during last years. It has a big impact from a safety point of view, allowing to reduce the gap between automated levels, aiming to achieve a fourth level of automation in a short term. However, recent accidents, such the first death where an automated vehicle was directly responsible, show that there is still work to do before having the whole architecture working together robustly. Additionally, providing a natural driving behavior to these systems is still a huge challenge. Crashes produced last year in automated urban navigation have shown that the other drivers over reacted intending to compensate the performance of the automated vehicles, since they were not navigating in the same way as a human driver would do. Providing a real-time human-like driving style would increase the acceptance of automated vehicles in society, since these vehicles would navigate in a safer way, performing more comfortable trajectories, and more predictable by the rest of road users.

This PhD thesis presents a two-staged path planning architecture to address this problem on urban environments, which supposes a bigger challenge due to the dynamism they present. It comprises pre-planning and real-time planning stages. First, the physical characteristics of both road and ego-vehcile are considered to pre-compute the optimal curves for adjusting the best to every single turn the vehicle can encounter. Then, the actual information of the map is considered to perform the desired itinerary by joining the optimal pre-computed curves generating an enhanced G_1 continuous path. The proposed approach is able to deal with any road configuration, treating them as turns, under the assumption of an accurate global map providing the information of the road. The proposed planner not only provides a solution to the navigation problem on static environments, but also adapts the path to dynamic environments, where both static and dynamic obstacles could be found in the path. Thanks to a grid-based discretization of the road, the position of the obstacles is considered to build a virtual lane for generating the avoidance path. That way, the original path is adapted considering the dynamics of the obstacle. Finding an avoidance path is understood as evaluating the optimal slopes for performing the two lane changes and finding the optimal junction point between curves. This way, the dynamic planner transforms the avoiding problem into a new path following itinerary that is solved by the static planner.

The proposed planning strategy for automated vehicles has been validated both in simulation and in real platforms. Pro-Sivic, together with RTMaps, were used to create the static and dynamic urban scenarios to validate the two stages of the approach. Additionally, Matlab-Simulink was used to validate the pre-planning stage. Both Cybercars and the robotized Citroën C1 were employed in the validation experiments at INRIA-Rocquencourt facilities. Results showed a good performance of the system, providing a human-like path able to adapt to the road layout and to the dynamism of urban environments. Reducing curvature and curvature derivative profiles as possible make vehicles perform smoother and more comfortable paths, contributing to the development of safer ADAS.

6.2 Contributions to the state of the art

The contributions to the current thesis can be summarized to the following, according to the field of application.

6.2.1 Static environments

- Pre-planning stage generates databases containing the optimal curves for every single turn the vehicle can find in the road, considering both vehicle kinematics and urban road characteristics.

- Different databases are generated with respect to the vehicle position at the beginning and end of the curve, i.e. depending on the use of the lane width.

- Real-time planning stage generates an enhanced G_1 continuous path by joining the optimal quartic Bézier pre-computed curves, allowing a good adaptation to the road layout, performing any road configuration as turns.

- Path continuity and smoothness thanks to an optimality function searching to minimize both the peaks on the curvature and the abrupt changes on it, for both the upcoming curve and the next one, providing an extended planning horizon that optimizes that way two curves in parallel.

- Intelligent algorithm to find in real-time the optimal location of the junction point between curves, according to the available distance and the sharpness relation between curves, deciding to use the available width of the lane or not and loading the proper database. Furthermore, this junction between curves is continuous thanks to the considered curvature zero constraint at the beginning and at the end of the optimal curves generated.

- Fast real-time planning: thanks to the pre-planning stage, this time can be below 35 ms.

6.2.2 Dynamic environments

- Dynamic planning algorithm which makes use of the static planning algorithm to modify the planned static path generating the avoiding path in a fast way.

- Path continuity and smoothness criteria seeked while keeping simplicity: only the slope of the lane changes and the distance to the obstacle are the parameters to evaluate in real-time.

- Each lane change path consists of two curves interpolated by the static local planner, keeping that way curvature continuity and smoothness in the obstacle avoidance maneuver.

- Adaptation of the path thanks to the generation of a virtual lane, which modifies the global path adding two extra way-points for each of the two lane changes.

- Re-planning process is not computed every time period of the planning time: it is only done either if the first prediction of the obstacle position is not correct, or if the distance traveled by both ego-vehicle and obstacle changes substantially, or if the time to collision is below the acceptable limits. It leads to the generation of faster dynamic paths.

6.3 Future work

This section presents the research perspective on motion planning systems for automated vehicles. Based on the review of the state of the art, three different research lines can be distinguished in this domain:

- With the technological evolution and economic impact shown in the late 90s and beginning of the 00s, graph-search planning approaches supposed a big revolution in motion planning, in robotics field first and later in automated vehicles.

- Since the generated paths are not continuous in base, sampling based techniques where addressed to provide constrained and faster trajectories. Parametric-based techniques appeared to generate curvature continuous paths dealing with intersections, narrow U-turns or similar urban configurations. These techniques have been used in combination with search based techniques such as state lattices in order to consider dynamic changes on the scene.

- Last years, since the decision making is still an unsolved problem that prevents to reaching the desired fourth automation level, both industry and research have put their efforts on improving the decisional stage by studying the behavior of drivers and making the system to learn. Thus, before having the urban autopilot available, robust algorithms have to be deployed to predict the behavior of the other road users, mostly on urban environments such as on road intersections, pedestrian crossing or parking lots.

Machine-learning algorithms may be considered on the path planning strategies in the short term. Research on this line is trying to study traffic scenarios, such as in [Fridman et al., 2018], where the perception, planning and control systems are handled by a single neural network in the reinforcement learning process in a micro-traffic simulation. As an open challenge, planning methods will have to provide safe and system compliant performance in complex, cluttered environments while modeling the uncertain motion of other traffic participants [Schwarting et al., 2018]. So far, most approaches are rule-based, i.e. use a state machine to switch between predefined behaviors. It supposes a lack of generalization to unknown situations and to deal with uncertainties.

An integrated perception and planning solution is expected, where the control input for the vehicle is generated directly from sensory information relying on machine learning. There, deep-learning based algorithms have a great potential to improve planning algorithms by learning how the other vehicles react according to the traffic situation and thus letting us to achieve the fourth automation level.

Creating fault-tolerant planning systems is a must in order to react when failures occur in the perception and vision algorithms, which is an unsolved problem so far. This would probably avoid accidents where a wrong recognition or classification of the road actors in front of the vehicle could lead to fatalities. Eluding these accidents could promote policies and investment for the research and development of automated vehicles on urban roads, as well as an increase of the citizens' confidence of the automated vehicle.

Appendix A

Appendix Title

The following XML code (Listing A.1) shows a subset of one of the four generated databases containing the configuration parameters to generate the optimal pre-computed curves found in the pre-planning process. This subset corresponds to the optimal curves for 90° turns, where the arriving distance to the center of the turn is four meters, and the exiting distance ranges from 4 meters up to 40 meters.

```
<turn>
<angle alpha="30">
...
<angle alpha="90">
<segment1 length="4">
<segment2 Bezier_order="4" length="6" d1_seg1="4" d1_seg2="6" d2_seg1="
   2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="8" d1_seg1="4" d1_seg2="6" d2_seg1="
   2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="10" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="12" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="14" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="16" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="18" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
<segment2 Bezier_order="4" length="20" d1_seg1="4" d1_seg2="6" d2_seg1=
   "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
   26.682454851472151" />
```

```
<segment2 Bezier_order="4" length="22" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="24" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="26" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="28" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="30" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="32" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="34" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="36" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="38" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
<segment2 Bezier_order="4" length="40" d1_seg1="4" d1_seg2="6" d2_seg1=
    "2" d2_seg2="4" dLat_P1="0" dLat_P2="0" dLat_PMiddle1="0" cost="
    26.682454851472151" />
</segment1>
<segment1 length="6">
 ...
</angle>
 ...
</turn>
```

Listing A.1 – Database XML subset code

Bibliography

[Ackermann, 1999] Ackermann, J., 1999. Robust Control for Car Steering. Springer London, London, pp. 1–15.

[Al-Dweik et al., 2017] Al-Dweik, A. J., Mayhew, M., Muresan, R., Ali, S. M. and Shami, A., 2017. Using technology to make roads safer: Adaptive speed limits for an intelligent transportation system. IEEE Vehicular Technology Magazine 12(1), pp. 39–47.

[Alia et al., 2015] Alia, C., Gilles, T., Reine, T. and Ali, C., 2015. Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method. In: 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 674–679.

[Andreasson et al., 2015] Andreasson, H., Bouguerra, A., Cirillo, M., Dimitrov, D. N., Driankov, D., Karlsson, L., Lilienthal, A. J., Pecora, F., Saarinen, J. P., Sherikov, A. and Stoyanov, T., 2015. Autonomous transport vehicles: Where we are and what is missing. IEEE Robotics Automation Magazine 22(1), pp. 64–75.

[Aoude et al., 2010] Aoude, G. S., Luders, B. D., Levine, D. S. and How, J. P., 2010. Threat-aware path planning in uncertain urban environments. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 6058–6063.

[Auer et al., 2016] Auer, A., Feese, S. and Lockwood, S., 2016. History of intelligent transportation systems. Technical Report FHWA-JPO-16-329, U.S. Department of Transportation.

[Bacha et al., 2008] Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., Cacciola, S., Currier, P., Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Covern, D. V. and Webster, M., 2008. Odin: Team victortango's entry in the darpa urban challenge. Journal of Field Robotics 25(8), pp. 467–492.

[Baran et al., 2010] Baran, I., Lehtinen, J. and Popovic, J., 2010. Sketching clothoid splines using shortest paths. Comput. Graph. Forum 29, pp. 655–664.

[Barbaresso et al., 2015] Barbaresso, J., Cordahi, G., Garcia, D., Hill, C., Jendzejec, A. and Wright, K., 2015. Usdot's intelligent transportation systems (its) its strategic plan 2015-2019. Technical report, US Department of Transportation (USDOT).

[Bauda et al., 2017] Bauda, M. A., Bazot, C. and Larnier, S., 2017. Real-time ground marking analysis for safe trajectories of autonomous mobile robots. In: 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), pp. 1–6.

[Berglund et al., 2010] Berglund, T., Brodnik, A., Jonsson, H., Staffanson, M. and Soderkvist, I., 2010. Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles. IEEE Transactions on Automation Science and Engineering 7(1), pp. 167–172.

[Bertails-Descoubes, 2012] Bertails-Descoubes, F., 2012. Super-Clothoids. Computer Graphics Forum 31(2), pp. 509–518.

[Bertolazzi and Frego, 2012] Bertolazzi, E. and Frego, M., 2012. Fast and accurate clothoid fitting. ArXiv e-prints.

[Bertolazzi and Frego, 2015] Bertolazzi, E. and Frego, M., 2015. G1 fitting with clothoids. Mathematical Methods in the Applied Sciences 38(5), pp. 881–897.

[Bestaoui Sebbane, 2014] Bestaoui Sebbane, Y., 2014. Motion Planning. Springer International Publishing, Cham, pp. 59–170.

[Bohren et al., 2008] Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J. and Satterfield, B., 2008. Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. Journal of Field Robotics 25(9), pp. 598–614.

[Breemersch, n.d.] Breemersch, T., n.d. Integrated approach reducing car co2 emissions - measures associated with the use of the vehicle. Technical report, Transport & Mobility Leuven.

[Brezak and Petrović, 2014] Brezak, M. and Petrović, I., 2014. Real-time approximation of clothoids with bounded error for path planning applications. IEEE Transactions on Robotics 30(2), pp. 507–515.

[Broggi et al., 2014] Broggi, A., Cerri, P., Debattisti, S., Laghi, M. C., Medici, P., Panciroli, M. and Prioletti, A., 2014. Proud-public road urban driverless test: Architecture and results. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 648–654.

[Broggi et al., 2012] Broggi, A., Medici, P., Zani, P., Coati, A. and Panciroli, M., 2012. Autonomous vehicles control in the vislab intercontinental autonomous challenge. Annual Reviews in Control 36(1), pp. 161 – 171.

[Chen et al., 2014] Chen, C., He, Y., Bu, C., Han, J. and Zhang, X., 2014. Quartic bezier curve based trajectory generation for autonomous vehicles with curvature and velocity constraints. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 6108–6113.

[Chen et al., 2013] Chen, J., Zhao, P., Mei, T. and Liang, H., 2013. Lane change path planning based on piecewise bezier curve for autonomous vehicle. In: Proceedings of 2013 IEEE International Conference on Vehicular Electronics and Safety, pp. 17–22.

[Choi et al., 2008] Choi, J., Curry, R. and Elkaim, G., 2008. Path planning based on bézier curve for autonomous ground vehicles. In: Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, pp. 158–166.

[Choi et al., 2010] Choi, J.-W., Curry, R. and Elkaim, G., 2010. Real-time obstacle-avoiding path planning for mobile robots. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics.

[Cicchino, 2016] Cicchino, J. B., 2016. Effectiveness of forward collision warning. systems with and without autonomous emergency braking in reducing police-reported crash rates. highway safety. Technical report, Insurance Institute for Highway Safety.

[Connell and La, 2017] Connell, D. and La, H. M., 2017. Dynamic path planning and replanning for mobile robots using RRT*. CoRR.

[Daniel et al., 2014] Daniel, K., Nash, A., Koenig, S. and Felner, A., 2014. Theta*: Any-angle path planning on grids. CoRR.

[Delling et al., 2009] Delling, D., Sanders, P., Schultes, D. and Wagner, D., 2009. Engineering Route Planning Algorithms. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 117–139.

[Dingus et al., 1996] Dingus, T. A., Hulse, M. C., Jahns, S. K., Alves-Foss, J., Confer, S., Rice, A., Roberts, I., Hanowski, R. J. and & Sorenson, D., 1996. Development of human factors guidelines for advanced traveler information systems and commercial vehicle operations: Literature review. Technical report, Federal Highway Administration, United States.

[Dubins, 1957] Dubins, L. E., 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics 79(3), pp. 497–516.

[Elfes, 1989] Elfes, A., 1989. Using occupancy grids for mobile robot perception and navigation. Computer 22(6), pp. 46–57.

[Elvik, 2009] Elvik, R., 2009. The power model of the relationship between speed and road safety. update and new analyses. Technical report, Institute of Transport Economics (TOI), Norwegian Centre for Transport Research.

[Englund et al., 2016] Englund, C., Chen, L., Ploeg, J., Semsar-Kazerooni, E., Voronov, A., Bengtsson, H. H. and Didoff, J., 2016. The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles. IEEE Wireless Communications 23(4), pp. 146–152.

[ERTRAC, 2017] ERTRAC, 2017. Automated driving roadmap. Technical report, ERTRAC Working Group.

[Eureka, 1987-1995] Eureka, 1987-1995. Prometheus project.

[European Commission, 1988-1991] European Commission, 1988-1991. Drive project.

[European Commission, 2010] European Commission, 2010. Towards a european road safety areas - policy orientations on road safety 2011-2020. Technical report.

[European Commission, 2016a] European Commission, 2016a. Advanced Driver Assistance Systems.

[European Commission, 2016b] European Commission, 2016b. Road safety in the european union. Technical report, European Commission.

[European Commission, 2017] European Commission, 2017. EU transport in figures. Statistical pocketbook 2017.

[European Parliament and Council of the European Union, 2015] European Parliament and Council of the European Union, 2015. Regulation (eu) 2015/758 of the european parliament and of the council of 29 april 2015 concerning type-approval requirements for the deployment of the ecall in-vehicle system based on the 112 service and amending directive 2007/46/ec.

[Ferguson and Stentz, 2006a] Ferguson, D. and Stentz, A., 2006a. Anytime rrts. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5369–5375.

[Ferguson and Stentz, 2006b] Ferguson, D. and Stentz, A., 2006b. Using interpolation to improve path planning: The field d* algorithm. Journal of Field Robotics 23(2), pp. 79–101.

[Fildes et al., 2015] Fildes, B., Keall, M., Bos, N., Lie, A., Page, Y., Pastor, C., Pennisi, L., Rizzi, M., Thomas, P. and Tingvall, C., 2015. Effectiveness of low speed autonomous emergency braking in real-world rear-end crashes. Accident Analysis & Prevention 81(Supplement C), pp. 24 – 29.

[Fiore, 2008] Fiore, G. A., 2008. A robust motion planning approach for autonomous driving in urban areas. Master's thesis.

[Fraichard and Scheuer, 2004] Fraichard, T. and Scheuer, A., 2004. From reeds and shepp's to continuous-curvature paths. IEEE Transactions on Robotics 20(6), pp. 1025–1035.

[Fridman et al., 2018] Fridman, L., Jenik, B. and Terwilliger, J., 2018. Deeptraffic: Driving fast through dense traffic with deep reinforcement learning. CoRR.

[Fu et al., 2015] Fu, M., Zhang, K., Yang, Y., Zhu, H. and Wang, M., 2015. Collision-free and kinematically feasible path planning along a reference path for autonomous vehicle. In: 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 907–912.

[Fulgenzi et al., 2009] Fulgenzi, C., Spalanzani, A. and Laugier, C., 2009. Probabilistic motion planning among moving obstacles following typical motion patterns. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4027–4033.

[Funke et al., 2016] Funke, J., Brown, M., Erlien, S. M. and Gerdes, J. C., 2016. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. IEEE Transactions on Control Systems Technology PP(99), pp. 1–13.

[Funke et al., 2012] Funke, J., Theodosis, P., Hindiyeh, R., Stanek, G., Kritatakirana, K., Gerdes, C., Langer, D., Hernandez, M., Müller-Bessler, B. and Huhnke, B., 2012. Up to the limits: Autonomous audi tts. In: 2012 IEEE Intelligent Vehicles Symposium, pp. 541–547.

[Gackstatter et al., 2010] Gackstatter, C., Heinemann, P., Thomas, S. and Klinker, G., 2010. Stable road lane model based on clothoids. In: G. Meyer and J. Valldorf (eds), Advanced Microsystems for Automotive Applications 2010, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 133–143.

[Garrido et al., 2016a] Garrido, F., González, D., Milanés, V., Pérez, J. and Nashashibi, F., 2016a. Optimized trajectory planning for Cybernetic Transportation Systems. In: 9th IFAC Symposium on Intelligent Autonomous Vehicles (IAV), pp. 1–6.

[Garrido et al., 2016b] Garrido, F., González, D., Milanés, V., Pérez, J. and Nashashibi, F., 2016b. Real-time planning for adjacent consecutive intersections. In: 19th International IEEE Conference on Intelligent Transportation Systems - ITSC 2016.

[Gasser et al., 2013] Gasser, T. M., Arzt, C., Ayoubi, M., Bartels, A., Bürkle, L., Eier, J., Flemisch, F., Häcker, D., Hesse, T., Huber, W., Lotz, C., Maurer, M., Ruth-Schumacher, S., Schwarz, J. and Vogt, W., 2013. Legal consequences of an increase in vehicle automation. Technical report, Federal Highway Research Institute (Bundesanstalt für Strassenwesen, BASt).

[Geiger et al., 2012] Geiger, A., Lauer, M., Moosmann, F., Ranft, B., Rapp, H., Stiller, C. and Ziegler, J., 2012. Team annieway's entry to the 2011 grand cooperative driving challenge. IEEE Transactions on Intelligent Transportation Systems 13(3), pp. 1008–1017.

[Gerdes and Rossetter, 1999] Gerdes, J. C. and Rossetter, E. J., 1999. A unified approach to driver assistance systems based on artificial potential fields. Journal of Dynamic Systems, Measurement, and Control 123(3), pp. 431–438.

[Ghilardelli et al., 2014] Ghilardelli, F., Lini, G. and Piazzi, A., 2014. Path generation using $mbieta^4$-splines for a truck and trailer vehicle. IEEE Transactions on Automation Science and Engineering 11(1), pp. 187–203.

[Gindele et al., 2009] Gindele, T., Brechtel, S., Schroder, J. and Dillmann, R., 2009. Bayesian occupancy grid filter for dynamic environments using prior map knowledge. In: 2009 IEEE Intelligent Vehicles Symposium, pp. 669–676.

[Girbés et al., 2011] Girbés, V., Armesto, L. and Tornero, J., 2011. Continuous-curvature control of mobile robots with constrained kinematics*. IFAC Proceedings Volumes 44(1), pp. 3503 – 3508. 18th IFAC World Congress.

[Glaser et al., 2010] Glaser, S., Vanholme, B., Mammar, S., Gruyer, D. and Nouveliere, L., 2010. Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. IEEE Transactions on Intelligent Transportation Systems 11(3), pp. 589–606.

[González Bautista, 2017] González Bautista, D., 2017. Functional architecture for automated vehicles trajectory planning in complex environments. Theses, PSL Research University.

[González and Pérez, 2013] González, D. and Pérez, J., 2013. Control architecture for cybernetic transportation systems in urban environments. In: 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 1119–1124.

[González et al., 2016a] González, D., Milanés, V., Pérez, J. and Nashashibi, F., 2016a. Speed profile generation based on quintic bézier curves for enhanced passenger comfort. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 814–819.

[González et al., 2016b] González, D., Pérez, J., Milanés, V. and Nashashibi, F., 2016b. A review of motion planning techniques for automated vehicles. IEEE Transactions on Intelligent Transportation Systems 17(4), pp. 1135–1145.

[González et al., 2014] González, D., Pérez Rastelli, J., Lattarulo, R., Milanés, V. and Nashashibi, F., 2014. Continuous curvature planning with obstacle avoidance capabilities in urban scenarios. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1430–1435.

[Google, 2015] Google, 2015. Google self-driving car project monthly report. Technical report, Google.

[Gu and Dolan, 2012] Gu, T. and Dolan, J. M., 2012. On-road motion planning for autonomous vehicles. In: C.-Y. Su, S. Rakheja and H. Liu (eds), Intelligent Robotics and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 588–597.

[Gu et al., 2013] Gu, T., Snider, J., Dolan, J. M. and w. Lee, J., 2013. Focused trajectory planning for autonomous on-road driving. In: 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 547–552.

[Han et al., 2010] Han, L., Yashiro, H., Nejad, H. T. N., Do, Q. H. and Mita, S., 2010. Bézier curve based path planning for autonomous vehicle in urban environment. In: 2010 IEEE Intelligent Vehicles Symposium, pp. 1036–1042.

[Heald, 1985] Heald, M. A., 1985. Rational approximations for the fresnel integrals. Math. Comp. 44 pp. 459–461.

[Horst and Barbera, 2006] Horst, J. and Barbera, A., 2006. Trajectory generation for an on-road autonomous vehicle.

[Howard and Kelly, 2007] Howard, T. M. and Kelly, A., 2007. Optimal rough terrain trajectory generation for wheeled mobile robots. The International Journal of Robotics Research 26(2), pp. 141–166.

[Hsieh and Ozguner, 2008] Hsieh, M. F. and Ozguner, U., 2008. A parking algorithm for an autonomous vehicle. In: 2008 IEEE Intelligent Vehicles Symposium, pp. 1155–1160.

[Hundelshausen et al., 2009] Hundelshausen, F. v., Himmelsbach, M., Hecker, F., Mueller, A. and Wuensche, H.-J., 2009. Driving with Tentacles - Integral Structures for Sensing and Motion. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 393–440.

[hwan Jeon et al., 2013] hwan Jeon, J., Cowlagi, R. V., Peters, S. C., Karaman, S., Frazzoli, E., Tsiotras, P. and Iagnemma, K., 2013. Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In: 2013 American Control Conference, pp. 188–193.

[Hwang and Ahuja, 1992] Hwang, Y. K. and Ahuja, N., 1992. Gross motion planning&mdash;a survey. ACM Comput. Surv. 24(3), pp. 219–291.

[International Council on Clean Transportation (ICCT), 2017] International Council on Clean Transportation (ICCT), 2017. European vehicle market statistics pocketbook 2017/18. Technical report, The Internatioanl Council on Clean Transportation (ICCT).

[International Energy Agency (IEA), 2017] International Energy Agency (IEA), 2017. Global ev outlook 2017 - two million and counting. Technical report, International Energy Agency (IEA).

[ITF, 2016] ITF, 2016. Road safety annual report 2016.

[Ji et al., 2017] Ji, J., Khajepour, A., Melek, W. W. and Huang, Y., 2017. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. IEEE Transactions on Vehicular Technology 66(2), pp. 952–964.

[Kala and Warwick, 2011] Kala, R. and Warwick, K., 2011. Planning of multiple autonomous vehicles using rrt. In: 2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS), pp. 20–25.

[Kammel et al., 2008] Kammel, S., Ziegler, J., Pitzer, B., Werling, M., Gindele, T., Jagzent, D., Schröder, J., Thuy, M., Goebl, M., Hundelshausen, F. v., Pink, O., Frese, C. and Stiller, C., 2008. Team annieway's autonomous system for the 2007 darpa urban challenge. Journal of Field Robotics 25(9), pp. 615–639.

[Kanayama and Hartman, 1989] Kanayama, Y. and Hartman, B. I., 1989. Smooth local path planning for autonomous vehicles. In: Proceedings, 1989 International Conference on Robotics and Automation, pp. 1265–1270 vol.3.

[Karaman and Frazzoli, 2010] Karaman, S. and Frazzoli, E., 2010. Incremental sampling-based algorithms for optimal motion planning. CoRR.

[Karaman and Frazzoli, 2011] Karaman, S. and Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. CoRR.

[Karaman et al., 2011] Karaman, S., Walter, M. R., Perez, A., Frazzoli, E. and Teller, S., 2011. Anytime motion planning using the rrt*. In: 2011 IEEE International Conference on Robotics and Automation, pp. 1478–1483.

[Katrakazas et al., 2015] Katrakazas, C., Quddus, M., Chen, W.-H. and Deka, L., 2015. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. Transportation Research Part C: Emerging Technologies 60, pp. 416 – 442.

[Kavraki et al., 1996] Kavraki, L. E., Svestka, P., Latombe, J. C. and Overmars, M. H., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), pp. 566–580.

[Kutila et al., 2014] Kutila, M., Pyykönen, P., van Koningsbruggen, P., Pallaro, N. and Pérez-Rastelli, J., 2014. The deserve project: Towards future adas functions. In: 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), pp. 308–313.

[Kuwata et al., 2008] Kuwata, Y., Fiore, G. A., Teo, J., Frazzoli, E. and How, J. P., 2008. Motion planning for urban driving using rrt. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1681–1686.

[Kuwata et al., 2009] Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E. and How, J. P., 2009. Real-time motion planning with applications to autonomous urban driving. IEEE Transactions on Control Systems Technology 17(5), pp. 1105–1118.

[Labakhua et al., 2008] Labakhua, L., Nunes, U., Rodrigues, R. and Leite, F. S., 2008. Smooth Trajectory Planning for Fully Automated Passengers Vehicles: Spline and Clothoid Based Methods and Its Simulation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 169–182.

[Latombe, 1991] Latombe, J.-C., 1991. Robot Motion Planning. Kluwer Academic Publishers, Norwell, MA, USA.

[Lauer, 2011] Lauer, M., 2011. Grand cooperative driving challenge 2011 [its events]. IEEE Intelligent Transportation Systems Magazine 3(3), pp. 38–40.

[Lavalle, 1998] Lavalle, S. M., 1998. Rapidly-exploring random trees: A new tool for path planning. Technical report.

[LaValle, 2006] LaValle, S. M., 2006. Planning Algorithms. Cambridge University Press.

[LaValle and Kuffner, 1999] LaValle, S. M. and Kuffner, J. J., 1999. Randomized kinodynamic planning. In: Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Vol. 1, pp. 473–479 vol.1.

[Lee and Moon, 2018] Lee, C. and Moon, J. H., 2018. Robust lane detection and tracking for real-time applications. IEEE Transactions on Intelligent Transportation Systems pp. 1–6.

[Lekkas, 2014] Lekkas, A. M., 2014. Guidance and Path-Planning Systems for Autonomous Vehicles. PhD thesis.

[Lekkas and Fossen, 2012] Lekkas, A. M. and Fossen, T. I., 2012. A time-varying lookahead distance guidance law for path following. IFAC Proceedings Volumes 45(27), pp. 398 – 403. 9th IFAC Conference on Manoeuvring and Control of Marine Craft.

[Li et al., 2009] Li, Q., Zeng, Z., Yang, B. and Zhang, T., 2009. Hierarchical route planning based on taxi gps-trajectories. In: 2009 17th International Conference on Geoinformatics, pp. 1–5.

[Li and Ruichek, 2014] Li, Y. and Ruichek, Y., 2014. Occupancy grid mapping in urban environments from a moving on-board stereo-vision system. Sensors 14(6), pp. 10454–10478.

[Likhachev and Ferguson, 2008] Likhachev, M. and Ferguson, D., 2008. Planning long dynamically-feasible maneuvers for autonomous vehicles. In: Robotics:Science and Systems IV.

[Likhachev et al., 2008] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A. and Thrun, S., 2008. Anytime search in dynamic graphs. Artificial Intelligence 172(14), pp. 1613 – 1643.

[Lima et al., 2015] Lima, P. F., Trincavelli, M., Mårtensson, J. and Wahlberg, B., 2015. Clothoid-based model predictive control for autonomous driving. In: 2015 European Control Conference (ECC), pp. 2983–2990.

[Liu et al., 2008] Liu, W., Zhang, H., Duan, B., Yuan, H. and Zhao, H., 2008. Vision-based real-time lane marking detection and tracking. In: 2008 11th International IEEE Conference on Intelligent Transportation Systems, pp. 49–54.

[Marouf et al., 2014] Marouf, M., Pollard, E. and Nashashibi, F., 2014. Automatic parallel parking and platooning to redistribute electric vehicles in a car-sharing application. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 486–491.

[McNaughton et al., 2011] McNaughton, M., Urmson, C., Dolan, J. M. and Lee, J. W., 2011. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In: 2011 IEEE International Conference on Robotics and Automation, pp. 4889–4895.

[Meek and Walton, 2004] Meek, D. and Walton, D., 2004. An arc spline approximation to a clothoid. Journal of Computational and Applied Mathematics 170(1), pp. 59 – 77.

[Meidenbauer, 2007] Meidenbauer, K. R., 2007. An investigation of the clothoid steering model for autonomous vehicles. Master's thesis.

[Merdrignac et al., 2015] Merdrignac, P., Pollard, E. and Nashashibi, F., 2015. 2d laser based road obstacle classification for road safety improvement. In: 2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO), pp. 1–6.

[Mielenz, 2000] Mielenz, K. D., 2000. Computation of fresnel integrals ii. Journal of Research of the National Institute of Standards and Technology.

[Milanes et al., 2012] Milanes, V., Llorca, D. F., Villagra, J., Perez, J., Fernandez, C., Parra, I., Gonzalez, C. and Sotelo, M. A., 2012. Intelligent automatic overtaking system using vision for vehicle detection. Expert Systems with Applications 39(3), pp. 3362 – 3373.

[Miller, 1968] Miller, R. B., 1968. Response time in man-computer conversational transactions. In: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, AFIPS '68 (Fall, part I), ACM, New York, NY, USA, pp. 267–277.

[Montemerlo et al., 2008] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A. and Thrun, S., 2008. Junior: The stanford entry in the urban challenge. Journal of Field Robotics 25(9), pp. 569–597.

[Mosquet et al., 2015] Mosquet, X., Andersen, M. and Arora, A., 2015. A roadmap to safer driving through advanced driver assistance systems. Technical report, Prepared for MEMA by The Boston Consulting Group.

[Mouhagir et al., 2017] Mouhagir, H., Cherfaoui, V., Talj, R., Aioun, F. and Guillemard, F., 2017. Using Evidential Occupancy Grid for Vehicle Trajectory Planning Under Uncertainty with Tentacles. In: IEEE 20th International Conference on Intelligent Transportation , Yokohama, Japan.

[Murgovski and Sjöberg, 2015] Murgovski, N. and Sjöberg, J., 2015. Predictive cruise control with autonomous overtaking. In: 2015 54th IEEE Conference on Decision and Control (CDC), pp. 644–649.

[Nagy et al., 2015] Nagy, ., Csorvási, G. and Kiss, D., 2015. Path planning and control of differential and car-like robots in narrow environments. In: 2015 IEEE 13th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 103–108.

[Naranjo et al., 2008] Naranjo, J. E., Gonzalez, C., Garcia, R. and de Pedro, T., 2008. Lane-change fuzzy control in autonomous vehicles for the overtaking maneuver. IEEE Transactions on Intelligent Transportation Systems 9(3), pp. 438–450.

[National Highway Traffic Safety Administration (NHTSA), 2013] National Highway Traffic Safety Administration (NHTSA), 2013. Preliminary statement of policy concerning automated vehicles. Technical report.

[Neto et al., 2010] Neto, A. A., Macharet, D. G. and Campos, M. F. M., 2010. Feasible rrt-based path planning using seventh order bezier curves. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1445–1450.

[NHTSA National Center for Statistics and Analysis, 2017a] NHTSA National Center for Statistics and Analysis, 2017a. Rural/urban comparison of traffic fatalities: 2015 data. (Traffic Safety Facts. Report No. DOT HS 812 393).

[NHTSA National Center for Statistics and Analysis, 2017b] NHTSA National Center for Statistics and Analysis, 2017b. Summary of motor vehicle crashes (Early edition): 2015 data. (Traffic Safety Facts. Report No. DOT HS 812 376).

[Nilsson et al., 2017] Nilsson, J., Brännström, M., Coelingh, E. and Fredriksson, J., 2017. Lane change maneuvers for automated vehicles. IEEE Transactions on Intelligent Transportation Systems 18(5), pp. 1087–1096.

[Nodine et al., 2011] Nodine, E. E., Lam, A. H. and Najm, W. G., 2011. Safety impact of an integrated crash warning system based on field test data. Technical report, National Highway Traffic Safety Administration.

[Nowacki, 2012] Nowacki, G., 2012. Development and standardization of intelligent transport systems. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation 6(3), pp. 403–411.

[Paden et al., 2016] Paden, B., Cáp, M., Yong, S. Z., Yershov, D. S. and Frazzoli, E., 2016. A survey of motion planning and control techniques for self-driving urban vehicles. CoRR.

[Panchea et al., 2017] Panchea, A. M., Chapoutot, A. and Filliat, D., 2017. Extended Reliable Robust Motion Planners. In: 56th IEEE Conference on Decision and Control, IEEE, Melbourne, Australia.

[Parent and de La Fortelle, 2005] Parent, M. and de La Fortelle, A., 2005. Cybercars : Past, present and future of the technology. CoRR.

[Pendleton et al., 2017] Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., Rus, D. and Ang, M. H., 2017. Perception, planning, control, and coordination for autonomous vehicles. Machines.

[Petrov and Nashashibi, 2014a] Petrov, P. and Nashashibi, F., 2014a. Modeling and nonlinear adaptive control for autonomous vehicle overtaking. IEEE Transactions on Intelligent Transportation Systems 15(4), pp. 1643–1656.

[Petrov and Nashashibi, 2014b] Petrov, P. and Nashashibi, F., 2014b. Saturated feedback control for an automated parallel parking assist system. In: 2014 13th International Conference on Control Automation Robotics Vision (ICARCV), pp. 577–582.

[Piazzi et al., 2002] Piazzi, A., Bianco, C. G. L., Bertozzi, M., Fascioli, A. and Broggi, A., 2002. Quintic g2-splines for the iterative steering of vision-based autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems 3(1), pp. 27–36.

[Piegl and Tiller, 1996] Piegl, L. and Tiller, W., 1996. The NURBS Book. second edn, Springer-Verlag, New York, NY, USA.

[Pivtoraiko and Kelly, 2005] Pivtoraiko, M. and Kelly, A., 2005. Efficient constrained path planning via search in state lattices. In: The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space, Pittsburgh, PA.

[Pivtoraiko and Kelly, 2008] Pivtoraiko, M. and Kelly, A., 2008. Differentially constrained motion replanning using state lattices with graduated fidelity. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2611–2616.

[Pivtoraiko and Kelly, 2009] Pivtoraiko, M. and Kelly, A., 2009. Fast and feasible deliberative motion planner for dynamic environments. In: International Conference on Robotics and Automation.

[Pivtoraiko et al., 2009] Pivtoraiko, M., Knepper, R. A. and Kelly, A., 2009. Differentially constrained mobile robot motion planning in state lattices. Journal of Field Robotics 26(3), pp. 308–333.

[Prautzsch et al., 2002] Prautzsch, H., Boehm, W. and Paluszny, M., 2002. Bézier and B-Spline Techniques. Springer-Verlag Berlin Heidelberg.

[Press et al., 1992] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., 1992. Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing. Cambridge University Press, New York, NY, USA.

[Pérez et al., 2011] Pérez, J., Milanés, V., Onieva, E., Godoy, J. and Alonso, J., 2011. Longitudinal fuzzy control for autonomous overtaking. In: 2011 IEEE International Conference on Mechatronics, pp. 188–193.

[Pérez Rastelli et al., 2014] Pérez Rastelli, J., Lattarulo, R. and Nashashibi, F., 2014. Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings, pp. 510–515.

[Qian et al., 2016] Qian, X., Navarro, I., de La Fortelle, A. and Moutarde, F., 2016. Motion planning for urban autonomous driving using bézier curves and mpc. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 826–833.

[Rajamani, 2011] Rajamani, R., 2011. Vehicle Dynamics and Control. Mechanical Engineering Series, Springer US.

[Reeds and Shepp, 1990] Reeds, J. A. and Shepp, L. A., 1990. Optimal paths for a car that goes both forwards and backwards. PACIFIC JOURNAL OF MATHEMATICS.

[Resende and Nashashibi, 2010] Resende, P. and Nashashibi, F., 2010. Real-time dynamic trajectory planning for highly automated driving in highways. In: 13th International IEEE Conference on Intelligent Transportation Systems, pp. 653–658.

[Richter et al., 2016] Richter, T., Ruhl, S., Ortlepp, J. and Bakaba, E., 2016. Prevention of overtaking accidents on two-lane rural roads. Transportation Research Procedia 14, pp. 4140 – 4149. Transport Research Arena TRA2016.

[Roldao et al., 2015] Roldao, L., Perez, J., Gonzalez, D. and Milanes, V., 2015. Description and technical specifications of cybernetic transportation systems: an urban transportation concept. In: 2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES), pp. 176–181.

[Romani and Sabin, 2004] Romani, L. and Sabin, M., 2004. The conversion matrix between uniform b-spline and bézier representations. Computer Aided Geometric Design 21(6), pp. 549 – 560.

[Ruhai et al., 2010] Ruhai, G., Weiwei, Z. and Zhong, W., 2010. Research on the driver reaction time of safety distance model on highway based on fuzzy mathematics. In: 2010 International Conference on Optoelectronics and Image Processing, Vol. 2, pp. 293–296.

[Ryu et al., 2013] Ryu, J.-H., Ogay, D., Bulavintsev, S., Kim, H. and Park, J.-S., 2013. Development and Experiences of an Autonomous Vehicle for High-Speed Navigation and Obstacle Avoidance. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 105–116.

[SAE International, 2014] SAE International, 2014. J3016: Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems.

[SAE International, 2016] SAE International, 2016. J3016: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles.

[Saha, 2006] Saha, M., 2006. Motion Planning with Probabilistic Roadmaps. Stanford University.

[Schwarting et al., 2018] Schwarting, W., Alonso-Mora, J. and Rus, D., 2018. Planning and decision-making for autonomous vehicles. Annual Review of Control, Robotics, and Autonomous Systems 1(1), pp. null.

[Sederberg, 2007] Sederberg, T. W., 2007. Computer Aided Geometric Design.

[Shamir, 2004] Shamir, T., 2004. How should an autonomous vehicle overtake a slower moving vehicle: design and analysis of an optimal trajectory. IEEE Transactions on Automatic Control 49(4), pp. 607–610.

[Shiller and Sundar, 1998] Shiller, Z. and Sundar, S., 1998. Emergency lane-change maneuvers of autonomous vehicles. Journal of Dynamic Systems, Measurement, and Control.

[Shladover, 1997] Shladover, S., 1997. Ahs demo '97 issue. Research Updates in Intelligent Transportation Systems.

[Simon and Becker, 1999] Simon, A. and Becker, J. C., 1999. Vehicle guidance for an autonomous vehicle. In: Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383), pp. 429–434.

[Sánchez-Reyes and Chacón, 2003] Sánchez-Reyes, J. and Chacón, J., 2003. Polynomial approximation to clothoids via s-power series. Computer-Aided Design 35(14), pp. 1305 – 1313.

[Society of Automotive Engineers. Vehicle Dynamics Committee, 1978] Society of Automotive Engineers. Vehicle Dynamics Committee, 1978. Vehicle Dynamics Terminology: SAE J670e : Report of Vehicle Dynamics Committee Approved July 1952 and Last Revised July 1976. Handbook supplement, Society of Automotive Engineers.

[Stentz, 1994] Stentz, A., 1994. Optimal and efficient path planning for partially-known environments. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 3310–3317 vol.4.

[Summala, 2000] Summala, H., 2000. Brake reaction times and driver behavior analysis. Transportation Human Factors 2(3), pp. 217–226.

[Sussman, 2008] Sussman, J., 2008. Perspectives on Intelligent Transportation Systems (ITS). Springer US.

[Takada et al., 1989] Takada, K., Tanaka, Y., Igarashi, A. and Fujita, D., 1989. Road/automobile communication system (racs) and its economic effect. In: Vehicle Navigation and Information Systems Conference, 1989. Conference Record, pp. A15–A21.

[Tanzmeister et al., 2016] Tanzmeister, G., Wollherr, D. and Buss, M., 2016. Grid-based multi-road-course estimation using motion planning. IEEE Transactions on Vehicular Technology 65(4), pp. 1924–1935.

[Trehard et al., 2014] Trehard, G., Alsayed, Z., Pollard, E., Bradai, B. and Nashashibi, F., 2014. Credibilist simultaneous localization and mapping with a lidar. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2699–2706.

[Trepagnier et al., 2007] Trepagnier, P. G., Nagel, J., Kinney, P. M., Koutsougeras, C. and Dooner, M., 2007. KAT-5: Robust Systems for Autonomous Vehicle Navigation in Challenging and Unknown Terrain. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 103–128.

[Tsourdos et al., 2010] Tsourdos, A., White, B. and Shanmugavel, M., 2010. John Wiley & Sons, Ltd, pp. 29–63.

[Urmson and Simmons, 2003] Urmson, C. and Simmons, R., 2003. Approaches for heuristically biasing rrt growth. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), Vol. 2, pp. 1178–1183 vol.2.

[US Environmental Protection Agency (EPA), 2017] US Environmental Protection Agency (EPA), 2017. Code of federal regulations. title 40 - protection of environment, section 600.315-08 - classes of comparable automobiles.

[Vaca et al., 2016] Vaca, M., Traver, J. E., Milanes, V., Perez, J., Gonzalez, D. and Nashashibi, F., 2016. Automated global planner for cybernetic transportation systems. In: 2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 1–6.

[van Ratingen et al., 2016] van Ratingen, M., Williams, A., Lie, A., Seeck, A., Castaing, P., Kolke, R., Adriaenssens, G. and Miller, A., 2016. The european new car assessment programme: A historical review. Chinese Journal of Traumatology 19(2), pp. 63 – 69.

[Verband der Automobilindustrie e. V. (VDA), 2015] Verband der Automobilindustrie e. V. (VDA), 2015. Autmoation: From driver assistance systems to automated driving. Technical report.

[w. Choi et al., 2010] w. Choi, J., Curry, R. E. and Elkaim, G. H., 2010. Curvature-continuous trajectory generation with corridor constraint for autonomous ground vehicles. In: 49th IEEE Conference on Decision and Control (CDC), pp. 7166–7171.

[Walton and Meek, 2005] Walton, D. and Meek, D., 2005. A controlled clothoid spline. Computers & Graphics 29(3), pp. 353 – 363.

[Walton et al., 2003] Walton, D., Meek, D. and Ali, J., 2003. Planar G^2 transition curves composed of cubic Bézier spiral segments. Journal of Computational and Applied Mathematics 157(2), pp. 453 – 476.

[Walton et al., 2000] Walton, J. R., Crabtree, J. D., Osborne, M. L., Pigman, J. G., Weber, J. M.and Grossardt, T. H. and Hartman, D. G., 2000. Intelligent transportation systems strategic plan (final report). Technical Report 299, Kentucky Transportation Center.

[Wang et al., 2001] Wang, L., Miura, K., Nakamae, E., Yamamoto, T. and Wang, T., 2001. An approximation approach of the clothoid curve defined in the interval [0, $\pi/2$] and its offset by free-form curves. Computer-Aided Design 33(14), pp. 1049 – 1058.

[Warren, 1989] Warren, C. W., 1989. Global path planning using artificial potential fields. In: Proceedings, 1989 International Conference on Robotics and Automation, pp. 316–321 vol.1.

[Willms and Yang, 2006] Willms, A. R. and Yang, S. X., 2006. An efficient dynamic system for real-time robot-path planning. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 36(4), pp. 755–766.

[Wolf and Burdick, 2008] Wolf, M. T. and Burdick, J. W., 2008. Artificial potential functions for highway driving with collision avoidance. In: 2008 IEEE International Conference on Robotics and Automation, pp. 3731–3736.

[Woodside Capital Partners (WCP), 2016] Woodside Capital Partners (WCP), 2016. Beyond the headlights: Adas and autonomous sensing.

[Xu et al., 2012] Xu, W., Wei, J., Dolan, J. M., Zhao, H. and Zha, H., 2012. A real-time motion planner with trajectory optimization for autonomous vehicles. In: 2012 IEEE International Conference on Robotics and Automation, pp. 2061–2067.

[Yang and Sukkarieh, 2008] Yang, K. and Sukkarieh, S., 2008. 3d smooth path planning for a uav in cluttered natural environments. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 794–800.

[You et al., 2015] You, F., Zhang, R., Lie, G., Wang, H., Wen, H. and Xu, J., 2015. Trajectory planning and tracking control for autonomous lane change maneuver based on the cooperative vehicle infrastructure system. Expert Systems with Applications 42(14), pp. 5932 – 5946.

[Zhang et al., 2011] Zhang, J., Wang, F. Y., Wang, K., Lin, W. H., Xu, X. and Chen, C., 2011. Data-driven intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems 12(4), pp. 1624–1639.

[Zhang et al., 2013] Zhang, S., Deng, W., Zhao, Q., Sun, H. and Litkouhi, B., 2013. Dynamic trajectory planning for vehicle autonomous driving. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp. 4161–4166.

[Ziegler and Stiller, 2009] Ziegler, J. and Stiller, C., 2009. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1879–1884.

[Ziegler and Werling, 2008] Ziegler, J. and Werling, M., 2008. Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In: 2008 IEEE Intelligent Vehicles Symposium, pp. 787–791.

Resumé

Les systèmes de transport intelligents (STI) sont conçus pour améliorer les transports, réduire les accidents, le temps de transport et la consommation de carburant, tout en augmentant la sécurité, le confort et l’efficacité de conduite. L’objectif final de ITS est de développer ADAS pour faciliter les tâches de conduite, jusqu’au développement du véhicule entièrement automatisé. Les systèmes actuels ne sont pas assez robustes pour atteindre un niveau entièrement automatisé à court terme. Les environnements urbains posent un défi particulier, car le dynamisme de la scène oblige les algorithmes de navigation à réagir en temps réel aux éventuels changements, tout en respectant les règles de circulation et en évitant les collisions avec les autres usagers de la route. Sur cette base, cette thèse propose une approche de planification locale en deux étapes pour apporter une solution au problème de la navigation en milieu urbain. Premièrement, les informations statiques des contraintes de la route et du véhicule sont considérées comme générant la courbe optimale pour chaque configuration de virage réalisable, où plusieurs bases de données sont générées en tenant compte de la position différente du véhicule aux points de début et de fin des courbes, permettant ainsi une analyse réaliste. planificateur de temps pour analyser les changements de concavité en utilisant toute la largeur de la voie. Ensuite, la configuration réelle de la route est envisagée dans le processus en temps réel, où la distance disponible et la netteté des virages à venir et consécutifs sont étudiées pour fournir un style de conduite à la manière humaine optimisant deux courbes simultanément, offrant ainsi un horizon de planification étendu. Par conséquent, le processus de planification en temps réel recherche le point de jonction optimal entre les courbes. Les critères d’optimalité minimisent à la fois les pics de courbure et les changements abrupts, en recherchant la génération de chemins continus et lisses. Quartic Béziers est l’algorithme d’interpolation utilisé en raison de ses propriétés, permettant de respecter les limites de la route et les restrictions cinématiques, tout en permettant une manipulation facile des courbes. Ce planificateur fonctionne à la fois pour les environnements statiques et dynamiques. Les fonctions d’évitement d’obstacles sont présentées en fonction de la génération d’une voie virtuelle qui modifie le chemin statique pour effectuer chacune des deux manoeuvres de changement de voie sous la forme de deux courbes, convertissant le problème en un chemin statique. Ainsi, une solution rapide peut être trouvée en bénéficiant du planificateur local statique. Il utilise une discrétisation en grille de la scène pour identifier l’espace libre nécessaire à la construction de la route virtuelle, où le critère de planification dynamique consiste à réduire la pente pour les changements de voie. Des essais de simulation et des tests expérimentaux ont été réalisés pour valider l’approche dans des environnements statiques et dynamiques, adaptant la trajectoire en fonction du scénario et du véhicule, montrant la modularité du système.

Mots Clés

Planification des trajectoires, véhicules autonomes, systèmes de transport intelligents

Abstract

Intelligent Transportation Systems (ITS) developments are conceived to improve transportation reducing accidents, transport time and fuel consumption, while increasing driving security, comfort and efficiency. The final goal of ITS is the development of ADAS for assisting in the driving tasks, up to the development of the fully automated vehicle. Despite last ADAS developments achieved a partial-automation level, current systems are not robust enough to achieve fully-automated level in short term. Urban environments pose a special challenge, since the dynamism of the scene forces the navigation algorithms to react in real-time to the eventual changes, respecting at the same time traffic regulation and avoiding collisions with other road users. On this basis, this PhD thesis proposes a two-staged local planning approach to provide a solution to the navigation problem on urban environments. First, static information of both road and vehicle constraints is considered to generate the optimal curve for each feasible turn configuration, where several databases are generated taking into account different position of the vehicle at the beginning and ending points of the curves, allowing the real-time planner to analyze concavity changes making use of the full lane width. Then, actual road layout is contemplated in the real-time process, where both the available distance and the sharpness of upcoming and consecutive turns are studied to provide a human-like driving style optimizing two curves concurrently, offering that way an extended planning horizon. Therefore, the real-time planning process searches the optimal junction point between curves. Optimality criteria minimizes both curvature peaks and abrupt changes on it, seeking the generation of continuous and smooth paths. Quartic Béziers are the interpolating-based curve algorithm used due to their properties, allowing compliance with road limits and kinematic restrictions, while allowing an easy manipulation of curves. This planner works both for static and dynamic environments. Obstacle avoidance features are presented based on the generation of a virtual lane which modifies the static path to perform each of the two lane change maneuvers as two curves, converting the problem into a static-path following. Thus, a fast solution can be found benefiting from the static local planner. It uses a grid discretization of the scene to identify the free space to build the virtual road, where the dynamic planning criteria is to reduce the slope for the lane changes. Both simulation and experimental test have been carried out to validate the approach, where vehicles performs path following on static and dynamic environments adapting the path in function of the scenario and the vehicle, testing both with low-speed cybercars and medium-speed electic platforms, showing the modularity of the system.

Keywords

Path planning, automated vehicles, intelligent transportation systems