Decision-based motion planning for cooperative and autonomous vehicles

Florent Altché

THÈSE DE DOCTORAT

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

Préparée à MINES ParisTech

Decision-based motion planning for cooperative and autonomous vehicles

Prise de décision et planification de trajectoire pour les véhicules coopératifs et autonomes

École doctorale n°432

SCIENCE DES MÉTIERS DE L'INGÉNIEUR

Spécialité MATHÉMATIQUES, INFORMATIQUE TEMPS-RÉEL, ROBOTIQUE

COMPOSITION DU JURY :

Mme Françoise PRÊTEUX
École des Ponts ParisTech,
Président du jury

M. Ching-Yao CHAN
BDD - University of California, Berkeley,
Rapporteur

M. Thierry FRAÎCHARD
INRIA Rhône-Alpes,
Rapporteur

M. Jean-Paul LAUMOND
LAAS - CNRS,
Membre du jury

M. Jorge VILLAGRA
CSIC - Universidad Politécnica de Madrid,
Membre du jury

M. Arnaud DE LA FORTELLE
CAOR - Mines ParisTech,
Membre du jury

Soutenue par **Florent ALTCHÉ**
le 30 août 2018

Dirigée par **Arnaud DE LA FORTELLE**

MINES ParisTech | PSL RESEARCH UNIVERSITY PARIS

Abstract

The deployment of future self-driving vehicles is expected to have a major socioeconomic impact due to their promise to be both safer and more traffic-efficient than human-driven vehicles. In order to live up to these expectations, the ability of autonomous vehicles to plan safe trajectories and maneuver efficiently around obstacles will be paramount. However, motion planning among static or moving objects such as other vehicles is known to be a highly combinatorial problem, that remains challenging even for state-of-the-art algorithms. Indeed, the presence of obstacles creates exponentially many discrete maneuver choices, which are difficult even to characterize in the context of autonomous driving.

This thesis explores a new approach to motion planning, based on using this notion of *driving decisions* as a guide to give structure to the planning problem, ultimately allowing easier resolution. This decision-based motion planning approach can find applications in cooperative driving, for instance to coordinate multiple vehicles through an unsignalized intersection, as well as in autonomous driving where a single vehicle plans its own trajectory.

In the case of cooperative driving, decisions are known to correspond to the choice of a relative ordering for conflicting vehicles, which can be conveniently encoded as a graph. This thesis introduces a similar graph representation in the case of autonomous driving, where possible decisions – such as overtaking the vehicle at a specific time – are much more complex. Once a decision is made, planning the best possible trajectory corresponding to this decision is a much simpler problem, both in cooperative and autonomous driving. This decision-aware approach may lead to more robust and efficient motion planning, and opens exciting perspectives for combining classical mathematic programming algorithms with more modern machine learning techniques.

Contents

Introduction 3

1 A primer on automated vehicles 3
- 1.1 Context, motivations and challenges . . . 3
- 1.2 Motion planning in automated vehicles . . . 6

2 Motion planning 9
- 2.1 Motion planning problems . . . 9
- 2.2 Motion planning algorithms . . . 12
- 2.3 Motion planning, homotopy and decision . . . 13

3 Contributions and thesis outline 17
- 3.1 Cooperative driving . . . 17
- 3.2 Autonomous driving . . . 18
- 3.3 Towards practical implementation . . . 18
- 3.4 Publications . . . 18

I Cooperative motion planning 21

4 Priority-based cooperative decision-making 25
- 4.1 Multi-robot coordination and motion planning . . . 25
- 4.2 Priority and decision-making . . . 26
- 4.3 Mixed-integer programming . . . 28
- 4.4 Modeling priorities . . . 28
- 4.5 Dynamic constraints . . . 34
- 4.6 Chapter conclusion . . . 35

5 Optimal coordination of robots along fixed paths 37
- 5.1 Time-optimal coordination . . . 37
- 5.2 Problem modeling . . . 39
- 5.3 MILP formulation . . . 40
- 5.4 Simulation results . . . 42
- 5.5 Some results for traffic management . . . 44
- 5.6 Chapter conclusion . . . 47

6 Supervised semi-autonomy 49
- 6.1 Supervised driving . . . 49
- 6.2 Supervision problem . . . 51
- 6.3 Infinite horizon formulation . . . 55
- 6.4 An equivalent finite horizon problem . . . 58
- 6.5 Simulation Results . . . 61

6.6 Chapter conclusion . . . 65

II Motion planning for autonomous driving 69

7 Decision-free, near-limits motion planning 73
7.1 Aggressive motion planning . . . 73
7.2 A simple second-order integrator model . . . 74
7.3 MPC formulation for trajectory planning . . . 76
7.4 Simulation results . . . 79
7.5 Chapter conclusion . . . 83

8 Classes of trajectories in autonomous driving 85
8.1 Motion planning for autonomous driving . . . 85
8.2 State representation . . . 86
8.3 Free space-time . . . 87
8.4 Maneuver variants and homotopy classes . . . 88
8.5 Chapter conclusion . . . 90

9 Navigation graph 91
9.1 Free space partitioning . . . 91
9.2 A guiding example . . . 92
9.3 Mathematical results . . . 97
9.4 Navigation graph and homotopy classes . . . 103
9.5 Chapter conclusion . . . 103

10 Graph-based decision-making 105
10.1 Decision-aware motion planning . . . 105
10.2 Motion planning on the navigation graph . . . 106
10.3 Global optimum search . . . 107
10.4 Heuristics . . . 111
10.5 Chapter conclusion . . . 112

III Beyond simulations: bridging the gaps 115

11 Trajectory prediction 119
11.1 Trajectory and behavior prediction . . . 119
11.2 Physics-based Monte-Carlo estimation . . . 121
11.3 A neural network approach . . . 128
11.4 Chapter conclusion . . . 136

12 A simplified implementation: velocity planning in the real world 139
12.1 Autonomous roundabout entry . . . 139
12.2 Decision-making for velocity planning . . . 140
12.3 Trajectory generation . . . 142
12.4 Experimental results . . . 143
12.5 Chapter conclusion . . . 146

IV Appendices **169**

A Complements on Chapter 4 **A1**
A.1 Computation of minimum bounding hexagons . . . A1
A.2 Sub-timestep collision avoidance . . . A2

B Complements on Chapter 5 **B1**
B.1 Helper constraints . . . B1
B.2 Influence of time step duration on optimality . . . B2

C Complements on Chapter 6 **C1**
C.1 Detailed demonstrations . . . C1
C.2 Discussion on implementation . . . C5

D Complements on Chapter 7 **D1**
D.1 Semi-infinite obstacles and local optima . . . D1
D.2 Simulation model . . . D2
D.3 Deriving a simpler dynamic model . . . D4

E Complements on Chapter 8 **E1**
E.1 Unicity of the Frenet coordinates . . . E1

F Complements on Chapter 9 **F1**
F.1 Demonstration of Theorem 6 . . . F1

G Résumés en français **G1**
G.1 Introduction . . . G1
G.2 Partie I . . . G2
G.3 Partie II . . . G4
G.4 Partie III . . . G5

List of Figures

2.1 Sources of complexity in motion planning . . . 10
2.2 Geometric and configuration space . . . 11
2.3 Examples of motion planning problems . . . 11
2.4 Homotopy classes for paths around an obstacle . . . 14
2.5 Convex cell decomposition of the path planning problem . . . 14
2.6 Homotopy classes in a 3-robot coordination problem . . . 15

4.1 Example of a two-robot coordination problem . . . 26
4.2 Correspondence between physical and coordination spaces . . . 27
4.3 Examples of paths and corresponding collision regions . . . 30
4.4 Bounding hexagon approximation of a collision region . . . 31
4.5 Segmentation of the collision-free portion of the coordination space . . . 32
4.6 Priority variables and coordination space . . . 33
4.7 Partition of the coordination space using priorities . . . 33

5.1 Paths inside and outside the coordination region . . . 39
5.2 Example of individual suboptimality . . . 41
5.3 Optimal trajectories within given priority classes . . . 43
5.4 Average computation time . . . 43
5.5 Comparison of FCFS and optimal ordering . . . 45
5.6 Average delays for optimal and FCFS orderings . . . 46
5.7 Computation time for various gap tolerance levels . . . 46

6.1 Road configurations and corresponding supervision area . . . 51
6.2 No stop and acceleration regions . . . 53
6.3 Merging scenario . . . 62
6.4 Intersection crossing scenario . . . 63
6.5 Roundabout scenario . . . 64
6.6 Computation time . . . 65

7.1 Feasible acceleration envelopes . . . 75
7.2 Approximation of feasible accelerations . . . 76
7.3 Modeling of an obstacle as a parabola . . . 78
7.4 Reference path . . . 80
7.5 Comparison of achieved speed . . . 81
7.6 Lateral positioning error . . . 82
7.7 Achieved vehicle speed with obstacles . . . 82
7.8 Lateral deviation . . . 82
7.9 MPC computation time . . . 83

8.1 Example of reference path in an on-road driving situation . . . 86

8.2 Frenet representation of a point on the road . . . 87
8.3 Physical and configuration space representation of obstacles . . . 87
8.4 Configuration space-time for a 3 vehicle scenario . . . 88
8.5 Configuration space-time for a single obstacle . . . 89
8.6 Homotopy classes relative to a region . . . 90

9.2 Guiding example situation . . . 92
9.3 Free space-time in the example situation . . . 93
9.4 Partitioning of the 2D free space . . . 94
9.5 Continuous partitioning of the free space-time . . . 95
9.6 Navigation graph . . . 95
9.7 Discrete partitioning of the free space-time . . . 96
9.8 Discrete-time navigation graph . . . 96
9.9 Trapeze decomposition . . . 98
9.10 Semantic partition around a single obstacle . . . 98
9.11 A more complex example with 3 obstacles. . . . 100

10.1 Another partition around a single obstacle . . . 108
10.2 Optimal trajectories with and without time margin . . . 110

11.1 Raw data for the estimation algorithm . . . 122
11.2 Overlay of the LiDAR point cloud on the navigable region . . . 123
11.3 Occupancy predictions . . . 125
11.4 Smoothing of the lateral position and speed. . . . 129
11.5 Vehicles of interest around the target vehicle . . . 129
11.6 Internal structure of an LSTM cell . . . 131
11.7 Network architecture used as reference design . . . 132
11.8 Predicted trajectories . . . 133
11.9 Distribution of error on the test set for the bagged predictor. . . . 135
11.10 Prediction delay . . . 136

12.1 Representations of occupancy predictions . . . 141
12.2 Decision tree . . . 142
12.3 Simulation results . . . 144
12.4 Reference path . . . 145
12.5 Experimental results . . . 145

A.2 Example of corner-cutting phenomenon . . . A3
A.3 Possible sub-timestep collision between following vehicles . . . A3

B.1 Example of helper constraints . . . B1
B.2 Average relative optimality loss from time step duration . . . B2
B.3 Average relative optimality loss for varying parameters . . . B3

C.1 Overshoot phenomenon . . . C3

D.1 Multiple local optima with a single semi-infinite obstacle . . . D1
D.2 Obstacle modeling as bounding parabola . . . D1
D.3 Simulation model of the vehicle in the (x, y) plane . . . D3
D.4 Feasible accelerations for varying v_x . . . D6
D.5 Feasible accelerations for varying v_y . . . D7
D.6 Feasible accelerations for varying μ . . . D8

D.7 Accumulation of samples for low friction coefficient D9
D.8 Variations of the a_X^{min} and a_X^{max} D10

E.1 Non-unicity of X_γ E2

List of Tables

1.1 US Crash statistics for 2015 4
1.2 Levels of vehicle automation 5

2.1 Overview of sampling-based algorithms for motion planning 12

7.1 Absolute lateral positioning error for both planners. 81

9.1 Validity sets Adj(A, B) 93
9.2 Time margins of example paths 97

10.1 Computation time for 10 time steps 110
10.2 Computation time for 20 time steps 111

11.1 RMS error for the tested models 134

B.1 Parameters used in the studied scenarios B3

D.1 Notations for the 9DoF model D2

Introduction

Chapter 1

A primer on automated vehicles

"If I had asked people what they wanted, they would have said faster horses."

Henry Ford

Contents

1.1 Context, motivations and challenges 3
1.1.1 Road safety 4
1.1.2 Traffic efficiency 4
1.1.3 Societal and economic impacts 4
1.1.4 Taxonomy 5
1.2 Motion planning in automated vehicles 6
1.2.1 A classical architecture: the perceive, plan, act paradigm 6
1.2.2 An alternative approach: end-to-end learning 6

1.1 Context, motivations and challenges

The advent of automobiles has revolutionized personal transportation by giving billions of people to travel almost anywhere on land at a reasonable cost. In 2002, car ownership levels in western countries were estimated at 0.55 car per capita (OECD average), even raising above 0.8 cars per capita in the United States [1].

Although it certainly is empowering, the actual driving task remains cumbersome, especially in dense traffic or on monotonous highways, prompting car manufacturers to provide increasing amounts of assistance to the driver, with the goal of ultimately providing cars capable of driving without human intervention. However, the disruptive potential of autonomous driving goes well beyond increased comfort.

In Sections 1.1.1 and 1.1.2, we discuss some of these expected benefits and related challenges, motivating the intense research and engineering effort currently focused on self-driving vehicles and, in particular, motion planning which is at the center of this thesis. In Section 1.1.3, we briefly discuss broader-picture socioeconomic implications of autonomous driving. Finally, Section 1.1.4, provides a simplified taxonomy of "intelligent" vehicles, in order to help clarify the rest of the thesis.

1.1.1 Road safety

First, removing the – fallible – human driver by a near-infallible automatic pilot is expected to drastically reduce the occurrence of accidents; in the US alone, roughly 6 million crashes are reported to the police each year [2], and more than 90% of those can be attributed to human error [3].

Table 1.1 breaks down the consequence of reported road accidents in the US in 2015, compared to the number of vehicle-kilometers driven on the same year. This table highlights the remarkably low rate of accidents for human drivers: on average, roughly 10 million kilometers are driven before the occurrence of an accident severe enough to be reported, and almost 200 million before recording a fatality. Due to these extremely low figures, of the major challenges for autonomous driving – although not in the scope of this thesis – lies in the ability to achieve comparable levels of safety, and to demonstrate this performance with a fleet of only dozens of vehicles [4].

Table 1.1 – US Crash statistics for 2015 (computed from data in [2])

Type of accident	Number	Rate per 10^8 veh. · km
Fatal	32 166	0.646
Injury	1 715 000	34.4
Property damage only	4 548 000	91.3

1.1.2 Traffic efficiency

A second wildly anticipated benefit of autonomous driving is its potential to increase the throughput of existing road infrastructure [5], in hope to reduce congestion. This improvement is thought to be possible due to several complementary aspects.

First, autonomous vehicles are expected to have a much shorter reaction time than human drivers, thus allowing to reduce safety distance or start faster when a traffic light turns green. Moreover, they can also help smooth traffic when nearing saturation, thus delaying the transition towards congestion [6].

Second, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication can help further improve this efficiency, for instance by allowing vehicles to drive in compact platoons [7].

Third, as the penetration rate of autonomous vehicles increases, the infrastructure can be adapted to their improved performance, for instance using narrower lanes or higher speed limits. Finally, in an even longer time frame, some authors have proposed to overhaul traffic management methods such as stop signs or traffic lights, and replace those with so-called Autonomous Intersection Management schemes [8], which allows optimizing the crossing order of individual vehicles.

1.1.3 Societal and economic impacts

In a broader picture, autonomous driving is expected to have ripple effects across a wide part of society and of the economy [5]. Indeed, self-driving cars promise to grant the same freedom of movement to almost anyone, by allowing elderly or disabled people to use a vehicle by themselves. Going another step further, some authors have proposed that autonomous driving would trigger a shift in vehicle ownership patterns, notably through

Table 1.2 – Levels of vehicle automation (as defined in [10])

Level	Name	Example
0	No automation	Legacy vehicles
1	Driver assistance	Adaptive cruise control
2	Partial driving automation	Traffic jam assist
3	Conditional driving automation	Highway pilot
4	High driving automation	City pilot on clear daytime
5	Full driving automation	Robot-taxi

an increase of car-sharing [9] which could, in turn, help decrease the amount of driving and parked vehicles in urban centers.

Although autonomous driving is largely praised for its potential benefits, some possible downsides – most of which are discussed in [6] – need also be mentioned. For instance, whether self-driving vehicles will effectively decrease traffic congestion is still debated, as temporarily reduced congestion could also prompt more people to use their own car. Since they can work or rest during their commute, people could also choose to live further away from their workplace, potentially leading to higher urban sprawl. Due to this induced traffic, the impact on public transportation is more uncertain: although removing the need for a driver may help decrease costs, a reduction in the number of passengers may require higher subsidizing as the poorest population will likely remain excluded from this progress. Similarly, although the ability to operate a vehicle without a driver can be highly appealing to the freight and taxi industry, the loss of employment for professional drivers should also be kept in mind, as well as potential effects in businesses such as insurance companies.

1.1.4 Taxonomy

Before moving on to the technical aspects of this thesis, we provide a brief overview of the taxonomy of automated or connected vehicles.

Dynamic driving task As defined in SAE International's J3016 standard [10], the *dynamic driving task* comprises short- and medium-term actions required to operate a vehicle on-road safely. This task includes longitudinal and lateral control, monitoring and interpreting the vehicle's environment and plan responses to events, as well as planning tactical maneuvers such as lane changes.

Automated vehicles *Automated vehicles* are vehicles equipped with systems capable of performing parts or all of the dynamic driving task [10]; depending on these capacities, automated driving systems are categorized into six levels as summarized in Table 1.2. Note that in order to be considered automated, a vehicle should be able to perform these driving tasks only using its own hardware, and without requiring communication from outside entities.

Legacy vehicles In this thesis, we describe as *legacy* vehicles with low to no driving automation in a particular situation, mostly corresponding to levels 2 and below. However, this term could also describe vehicles of level 3 or 4 in contexts where their automated driving systems cannot function.

Autonomous or self-driving vehicles As opposed to legacy vehicles, we use the terms *autonomous* or *self-driving* to designate fully automated vehicles which can function by themselves without requiring communication. Therefore, this term mostly relates to level 5 vehicles, although it could apply to level 4 systems in contexts where automated driving without human intervention is possible.

Connected vehicles Regardless of their level of automation, vehicles can also be *connected – i.e.*, able to communicate wirelessly either with other vehicles (known as vehicle-to-vehicle, or V2V communications) or with equipment attached to the road infrastructure (vehicle-to-infrastructure, or V2I). These vehicular communication capacities, commonly referred to as V2X, can rely on several technologies with various performance [11].

Cooperative vehicles Although vehicular communications can be used in legacy vehicles (*e.g.*, to provide traffic information), connectivity is also a means to improve safety and traffic efficiency further than what could be achieved using only automated vehicles. Indeed, communications could allow vehicles to cooperatively – the infrastructure acting as an authority – make and follow decisions such as traveling in platoons [12] or passing intersections autonomously [13]. In this thesis, we use the term *cooperative* to describe partially or fully automated vehicles capable of using communication to coordinate with others.

1.2 Motion planning in automated vehicles

To close this contextual introduction, we propose to briefly discuss the role of motion planning in automated vehicles. Throughout the rest of this thesis, we use the term *ego-vehicle* to designate a particular vehicle, from the viewpoint of which we consider the driving task.

1.2.1 A classical architecture: the perceive, plan, act paradigm

Automated vehicles are a particular form of wheeled robots; as such, usual approaches to automated driving systems rely on the well-established *perceive-plan-act* paradigm (see, *e.g.*, [14]). In this framework, a first perception layer is in charge of combining sensor data and potential prior knowledge such as mapping information into a suitable representation of the state of the ego-vehicle and its environment. The motion planning layer then computes a feasible and efficient trajectory for the ego-vehicle; finally, a control layer activates the vehicle's actuators to follow this planned trajectory.

1.2.2 An alternative approach: end-to-end learning

With the dramatic increase of available computational power and the undeniable success of deep learning approaches in applications ranging from computer vision [15] to playing Go [16], a new approach called *end-to-end learning* has emerged for driving automation. This technique leverages machine learning to design a policy directly computing a sequence of actions (*e.g.*, steering angle and acceleration) from raw sensor inputs (*e.g.*, camera images), without requiring human-designed algorithms.

Currently, two major techniques are being investigated for end-to-end driving. In a classical *supervised* approach (see, *e.g.*, [17]), the algorithm is taught to imitate the behavior of actual human drivers in given situations, thus requiring a significant amount

of training data recorded on real vehicles. More recently, the success of *(deep) reinforcement learning* algorithms in outreaching human [18] or even AI [19] experts in a variety of tasks without explicitly requiring training data has sparked a vast amount of interest in the artificial intelligence community. In this setting, an *agent* (the autonomous vehicle) learns through trial-and-error to choose actions maximizing a reward function defined by a human operator. For obvious reasons, reinforcement learning techniques are generally applied in simulations only, and it is unclear how training results can be transferred to the real world.

These methods have the common advantage of considerably simplifying the development of driving algorithms, and mimic the way humans learn to perform complex tasks. However, they suffer from several downsides that need to be addressed by future research. First, current learning algorithms behave as black boxes with limited interpretability of failure scenarios and no method currently exist to provide behavior guarantees. Second, guaranteeing that trained models will respond well to unseen situations is still an active research topic. Finally, deep learning approaches have been shown to be vulnerable to *(adversarial) attacks*, where a tiny modification of the inputs leads to arbitrary changes in the outputs [20], which can result in security threats. For these reasons, we argue that using intrinsic structural properties of the underlying problem to guide or constrain learning algorithms is important for safety-critical applications; the results provided in this thesis may therefore constitute a first step towards this goal.

Chapter 2

Motion planning

"A good plan implemented today is better than a perfect plan implemented tomorrow."

George Patton (Military)

Contents

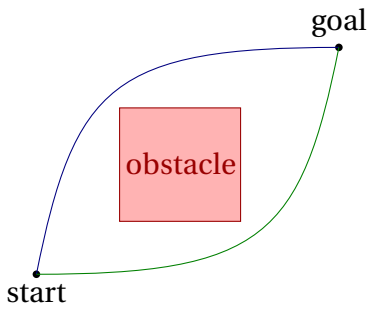
2.1 Motion planning problems 9
2.1.1 State representations 10
2.1.2 Path planning 11
2.1.3 Trajectory planning 11
2.2 Motion planning algorithms 12
2.2.1 Sampling-based 12
2.2.2 Optimization-based 12
2.3 Motion planning, homotopy and decision 13
2.3.1 Homotopy classes of paths 13
2.3.2 Divide-and-conquer approaches 14
2.3.3 Motion planning and decision-making 15
2.3.4 Algorithm evaluation 15

2.1 Motion planning problems

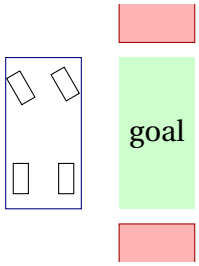
The general notion of *motion planning* refers to the computation of a means for a *system* to reach a desired *goal* starting from a given initial state, and is particularly important in the field of robotics [21]. As described in the next subsections, motion planning actually comprises a range of slightly distinct subproblems.

Sources of complexity in general motion planning arise from multiple factors. First, the system generally has to evolve around forbidden *regions* such as obstacles or off-limits areas, which usually results in non-convex search spaces and multiple local optima can exist [22], as illustrated in Figure 2.1a. Second, kinematic constraints on the system such as nonholonomicity[1] adds another layer of complexity [23] when trying to find feasible

[1] A system is non-holonomic when it has fewer controllable degrees of freedom (DoF) than its total number of DoF.



(a) Obstacles and non-convexity: multiple locally optimal solutions exist



(b) Nonholonomicity: the car cannot directly park in the goal region

Figure 2.1 – Sources of complexity in motion planning

solutions, as illustrated in Figure 2.1b in the case of parallel parking. Third, dynamic effects such as slip may have to be taken into account, for instance when dealing with fast or highly precise maneuvers.

2.1.1 State representations

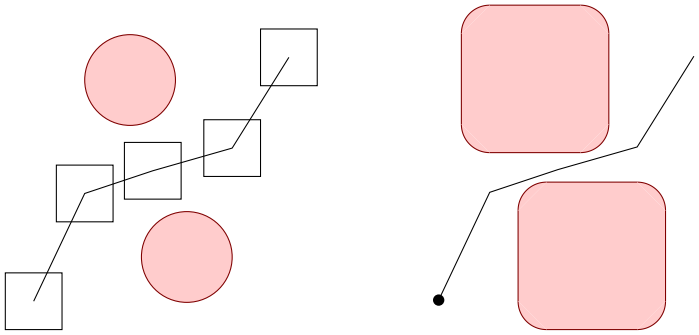
Before presenting different subproblems that fall within the scope of *motion planning*, we briefly present a terminology regarding the representations of a system's "state". For the sake of simplicity, we restrict the discussion to systems comprised of a single or multiple mobile robots.

Pose In this thesis, we call *pose* of a single robot having n degrees of freedom the given of $\mathbf{x} \in \mathbb{R}^n$ fully describing the current position and orientation of the robot's frame and its joints. For instance, in the case of a simplified 2D vehicle representation, a pose could be given by the vehicle's position, orientation, and steering angle.

Configuration As introduced in [24], we call *configuration* of a system of robots the collection of the poses of each individual robot in the system. The *configuration space* $\mathscr{C}$ is the set of all configurations. For a single robot, pose and configuration are equivalent. We note $\mathscr{C}^f$ the (collision-)free part of the configuration space, *i.e.* where no collision between robots happens, and $\mathscr{C}^o$ its complement so that $\mathscr{C} = \mathscr{C}^f \uplus \mathscr{C}^o$ where $\uplus$ denotes a union of disjoint sets.

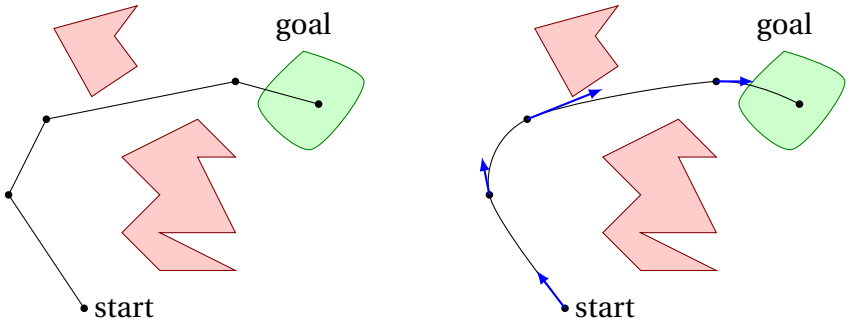
The major advantage of using the configuration space is that this formulation allows a reformulation of multi-body collision avoidance constraints as simpler point-outside-polygon problem (assuming polygonal obstacles) or point-outside-region in the general case, as illustrated in Figure 2.2. A standard approach in motion planning is to use the Minkowski difference to compute the collision-free space [25].

State We call *state* $\mathbf{X}$ of a system (either a single or multiple robots) the given of its configuration $\mathbf{x}$ and its time derivatives up to a sufficient order, such that the dynamics of the system (without any form of control) can be expressed as a differential equation $\dot{\mathbf{X}} = f_0(\mathbf{X})$. The *state space* $\mathscr{X}$ is the set of all states; we denote by $\mathscr{X}^f$ the free portion of the state space, *i.e.* the set $\left\{ \mathrm{X} \in \mathscr{X}^f \,\middle|\, \Pi_{\mathscr{C}}(\mathrm{X}) \in \mathscr{C}^f \right\}$ where $\Pi_{\mathscr{C}}(\mathrm{X})$ denotes the projection of X in the configuration space.



(a) Geometric space (b) Configuration space

Figure 2.2 – Comparison of planning in the geometric space and in the configuration space



(a) Path planning (b) Trajectory planning

Figure 2.3 – Examples of motion planning problems; red polygons represent obstacles.

Control We call *control* $\mathbf{u}$ an input to the system that can change the evolution of its state, such that under this control $\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{u})$. We note $\mathcal{U}$ the set of all *admissible* controls, *i.e.* which are actually feasible by the system.

2.1.2 Path planning

The *path planning* problem corresponds to finding a (collision-free) path between an initial configuration $x_0 \in \mathcal{C}^f$ and a target region $\mathrm{X}^{goal} \neq \emptyset$ (possibly a single point) with $\mathrm{X}^{goal} \subset \mathcal{C}^f$ is that of finding a continuous function $f : [0,1] \to \mathcal{C}^f$ such that $f(0) = x_0$ and $f(1) \in \mathrm{X}^{goal}$. Figure 2.3a presents a possible path for a point-mass system among obstacles in $\mathbb{R}^2$.

Note that, although it is the simplest form of motion planning problems, path planning amidst obstacles is known to be NP-hard [26], with a complexity growing exponentially in the number of obstacles.

2.1.3 Trajectory planning

The *trajectory planning* problem is the time-based equivalent of path planning, applied in the state space. For an initial state $\mathrm{X}_0 \in \mathcal{X}^f$ and a non-empty goal region $\mathrm{X}^{goal} \subset \mathcal{X}^f$, the trajectory planning problem[2] is that of finding a function $g : [0, +\infty] \to \mathcal{X}^f$ with $f(0) = \mathrm{X}_0$ and $\exists \mathrm{T} > 0$ such that $\forall t \geq \mathrm{T}, f(t) \in \mathrm{X}^{goal}$. A trajectory is said to be (dynamically) *feasible* if there exists an admissible control $u : [0, \mathrm{T}] \to \mathcal{U}$ realizing this trajectory.

[2] In the case of static obstacles

Note that the trajectory planning problem can conveniently be cast as a path planning one in the "state-time space" $\mathbb{R}_+ \times \mathcal{X}^f$, with the first dimension representing time, and using the added constraint that the first component of the path should be strictly increasing. This approach also allows handling the case of moving obstacles by considering the free portion of the state-time space $\left\{(t, \mathrm{X}) \in \mathbb{R}_+ \times \mathcal{X} \,\middle|\, \mathrm{X} \in \mathcal{X}^f(t)\right\}$, where $\mathcal{X}^f(t)$ denotes the free portion of the state space at time t [27, 28].

2.2 Motion planning algorithms

Due to the existence of obstacles which effectively cause the search space to be non-convex, generic motion planning in an arbitrary space is an NP-hard problem. basically corresponding to the choice of a maneuver variant [29]. In this section, we present an overview of the existing literature on motion planning algorithms; these techniques can be grouped as either *sampling-based* or *optimization-based* [30, 31].

2.2.1 Sampling-based

Sampling-based algorithms are often used to deal with NP-hard problems, as they guarantee a fixed runtime and can have interesting properties such as asymptotic completeness (ensuring that a solution can eventually be found) or asymptotic optimality (ensuring that the best solution found converges to the optimum as the number of samples increases). In the motion planning literature, sampling-based algorithms can be further distinguished based on whether samples are chosen deterministically or randomly, and whether samples correspond to system states or controls. Table 2.1 presents an overview of different sampling-based algorithms used in motion planning. A more detailed review of existing algorithms can be found, *e.g.*, in [30].

Apart from these differences, sampling-based algorithms all operate on the same principle: a predetermined number of samples is chosen in the sampled space, and the corresponding trajectories are then evaluated with respect to a chosen cost function – possibly using forward integration in the case of samples in the control space. The best trajectory for this cost function is then selected.

One of the main downsides of sampling-based approaches is that they ignore the existence of maneuver variants, which are never considered explicitly. Therefore, these algorithms have a high risk of being trapped around local optima without even exploring certain configurations.

2.2.2 Optimization-based

In this review, we use the term *optimization-based* to describe techniques based on directly using mathematical programming (*e.g.*, constrained optimization) for the motion planning problem.

Table 2.1 – Overview and classification of sampling-based algorithms for motion planning

	Random sampling	Deterministic sampling
State space	PRM [32], RRT [33]	State lattices [34], motion primitives [35]
Control space	MPPI [36]	Mobile automaton [37]

In particular, Model Predictive Control (MPC) techniques – although originally designed for control systems – has been extensively used in motion planning (see, *e.g.*, [38]). In this framework, motion planning is formulated as an optimization problem where the system's dynamics are used as constraints alongside with obstacle avoidance requirements. For real-time tractability, this optimization problem is usually solved over a finite time horizon, and replanning is frequently used to take uncertainty and new information into account. Although it has been demonstrated to work well in practice, a particular problem of model predictive control lies in guaranteeing infinite horizon properties (*e.g.*, recursive feasibility) whereas the formulation only considers a finite planning horizon [39]. A possible approach to provide such guarantees lies in the use of invariant sets theory (see, *e.g.*, [40]); however, finding such invariant sets may be difficult for complex systems or dynamic environments, and this verification is rarely performed in practice.

Another difficulty when using MPC methods for motion planning is linked to the intrinsic NP-hardness of the problem, and the modeling of obstacle-related constraints is paramount to the actual computability of the problem. A possible approach is to model obstacles as holes (usually rectangles or ellipses) in the search space, and use gradient descent to perform the actual optimization (see, *e.g.*, [14]). However, as with sampling-based approach, the non-convexity of the search space implies a high probability of these approaches resulting in a local optimum [29].

Note that, although this section mostly focused on MPC, other optimization-based approaches, such as potential-field methods [41], have also been used for path planning, although their generalization to time-dependent problems may be difficult.

2.3 Motion planning, homotopy and decision

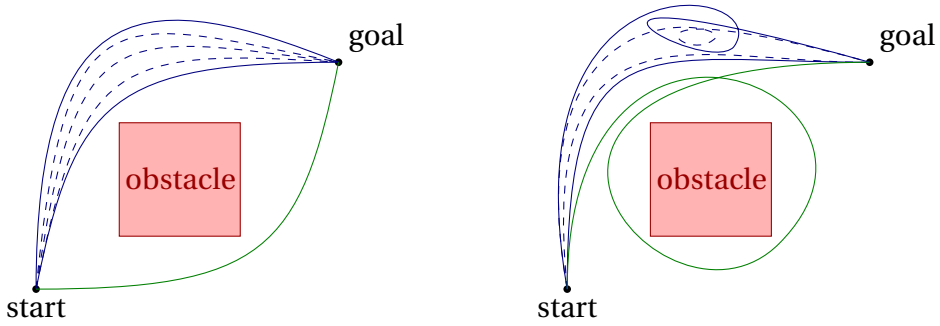
As mentioned earlier, one of the major challenges of motion planning lies in the non-convexity of the search space in the presence of obstacles. A particularly powerful tool to help describe this complexity is linked to the notion of *homotopy classes*, which has recently been leveraged in divide-and-conquer approaches to motion planning for a single- or multi-robots systems [42, 43, 29].

2.3.1 Homotopy classes of paths

In the case of a single robot navigating among obstacles, it has long been known that the uncountable set of collision-free paths between a given start and end point can be partitioned into a countable[3] number of *homotopy classes* [44]. Mathematically, two paths having the same start and end points are said to be *homotopic* if they can be continuously deformed into one another while remaining in $\mathscr{C}^f$, as illustrated in Figure 2.4.

The notion of homotopy allows defining an equivalence relation between collision-free paths [45]: two such paths – sharing the same start and end points – are said to be equivalent if they are homotopic. This equivalence relation in turn defines a partition of the set of collision-free paths into *homotopy classes*. In the example of Figure 2.4, and if requiring the paths to have an increasing horizontal component, only two classes of paths can be distinguished as avoiding the obstacle "from above" of "from below" (Figure 2.4a). If monotonicity is not required, there exist a (countable) infinity of classes corresponding to an arbitrary number of loops around the obstacle (Figure 2.4b).

[3] Or finite if at least one coordinate is required to vary monotonously



(a) Monotonous case (no loop)

(b) Non-monotonous case (loops)

Figure 2.4 – Homotopy classes for paths around an obstacle. The blue paths are homotopic since they can be continuously deformed into one another (dashed paths), but cannot be deformed into the green path without entering the obstacle region.

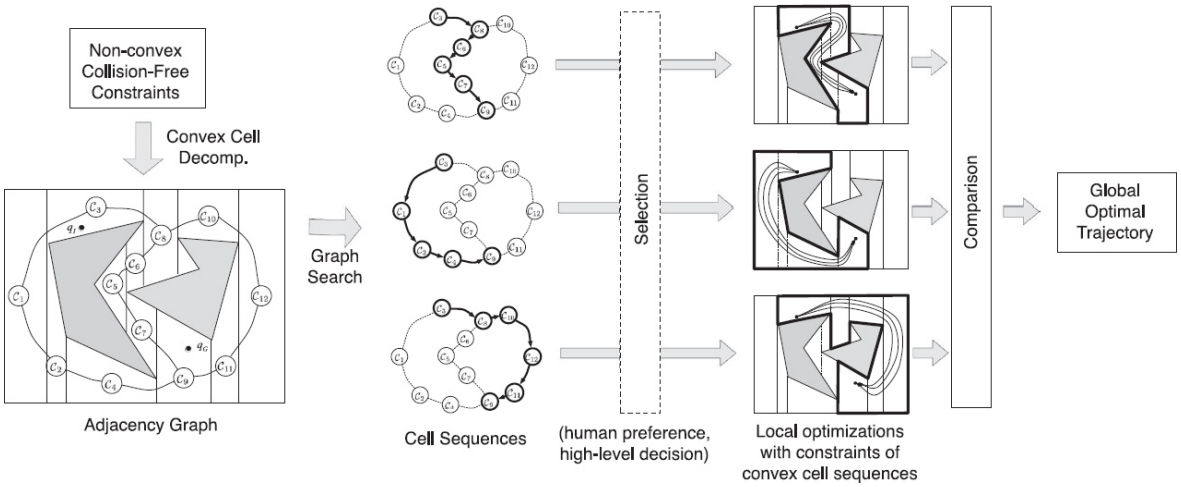


Figure 2.5 – Decomposition of the path planning problem in sequences of convex cells (from [43]).

However, the use of homotopy classes is shown in Chapter 8 to be poorly suited to encode decisions in the case of autonomous driving, where lateral vehicle movement is not monotonous. For this reason, other approaches are required to properly describe the discrete component of motion planning.

2.3.2 Divide-and-conquer approaches

Divide-and-conquer strategies have been applied very early in order to decompose motion planning into simpler subproblems, and show promising potential to generalize the notion of homotopy.

The so-called *path-velocity decomposition* [46] – consisting in first choosing a collision-free path, then planning the velocity of the system along this path – is a form of divide-and-conquer approach, which has been highly successful in motion planning. However, this approach is difficult to generalize in the case of moving obstacles.

More recently, some authors have proposed a geometric decomposition of the path planning problem with fixed obstacles with interesting results [43], as illustrated in Figure 2.5. In doing so, they create a graph-based representation of the free space; paths in this graph are shown to bijectively correspond to homotopy classes of collision-free paths.

Decomposition-based approaches have also been used in the field of multi-robot co-

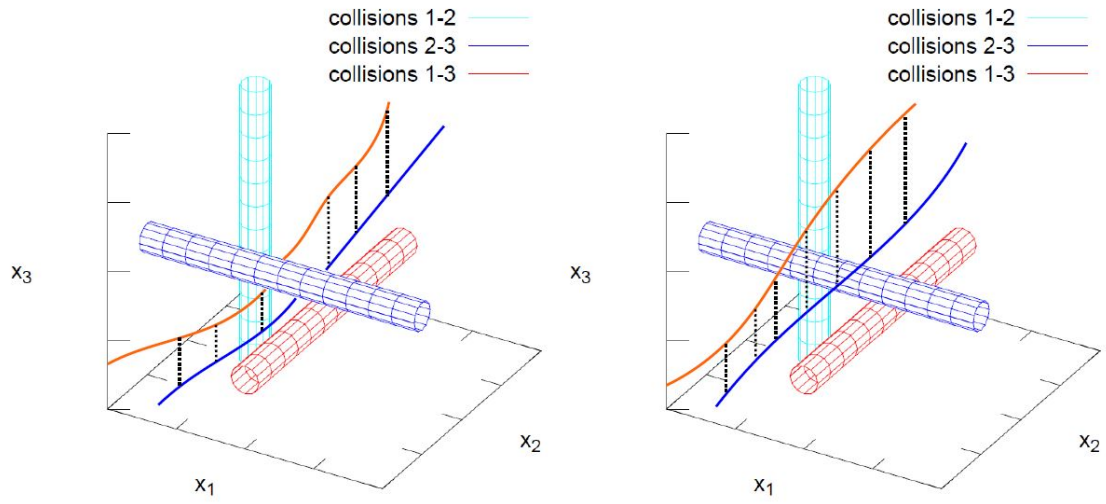


Figure 2.6 – Homotopy classes in a 3-robot coordination problem. Axes correspond to the curvilinear x_i position of robot i along a predetermined path; cylinders represent the obstacle region where two robots would collide (from [42]).

ordination using the notion of "priorities", which exactly correspond to specifying a homotopy class in an N-robots coordination space, as illustrated in Figure 2.6 [42].

2.3.3 Motion planning and decision-making

In this thesis, we argue that the reason for the success of these divide-and-conquer approaches in the field of motion planning lies in the fact that homotopy classes are intrinsically linked to discrete *decisions* relative to how obstacles should be avoided in certain particular cases. Making such a decision essentially corresponds to solving the discrete portion of the planning problem; once done, usual continuous optimization techniques can be effectively used to find an optimal trajectory within the corresponding class.

Therefore, there are undeniable advantages to using an explicit or implicit representation of the possible decisions to solve motion planning problems in complex scenarios such as the navigation of a self-driving vehicle in an urban setting, or the coordination of cooperative vehicles in critical parts of the road infrastructure such as intersections or roundabouts.

However, we show in Chapter 8 that the notion of homotopy does not always overlap with driving decisions, especially in the case of motion planning for autonomous vehicles. To the best of our knowledge, no framework exists to enumerate and represent driving decisions, let alone use them efficiently for motion planning. One of the major contributions of this thesis is the introduction of a mathematical framework allowing systematic enumeration of driving decisions in generic situations, as well as algorithms building upon this framework for autonomous motion planning.

2.3.4 Algorithm evaluation

As a final consideration before closing this introductory chapter, let us point out that motion planning has an extremely wide range of applications, and the idea of a "one-size-fits-all" motion planning algorithm seems illusory with respect to the current state of the art.

In the case of automated driving, we argue that real-time performance is more important than the optimality of the output. As a result, the ability of an algorithm to rapidly find a solution of acceptable quality is paramount. A second performance criterion is the ability of the algorithm to progressively improve the quality of the solution, ultimately converging towards the actual optimum given enough time; third, the speed of this convergence is also important.

Another aspect to be taken into account is that the "optimality" of the solution obviously depends on the objective function used to evaluate it. Although determining what makes a trajectory desirable or not is out of the scope of this thesis, we wish to point out that different kinds of algorithms or problem representations may allow different levels of expressivity in the objective function. In particular, we show in Chapter 9 that carefully crafted problem formulations can provide access to metrics such as margin for error – which is arguably an important evaluation criterion, but is not easily taken into account in other planning frameworks.

Chapter 3

Contributions and thesis outline

" Basic research is like shooting an arrow in the air and, where it lands, painting a target. "

Homer B. Adkins (Chemist)

Contents

3.1 Cooperative driving 17
3.2 Autonomous driving 18
3.3 Towards practical implementation 18
3.4 Publications 18

As introduced in the previous chapter, one of the main challenges of motion planning originates from the discrete choices regarding how each obstacle should be avoided, effectively resulting in a combinatorial problem. In this thesis, we advocate that a good mathematical representation of these choices is key to improve the performance and robustness of planning algorithms, thus stressing the importance of *decision-making* in this context; the thesis provides three main contributions to existing knowledge, both in the field of cooperative and autonomous driving.

3.1 Cooperative driving

In the case of cooperative driving of fully autonomous vehicles, a previous thesis studied *Priority-based coordination of mobile robots* [42] and evidenced that the given of pairwise precedence relations between robots were necessary and sufficient to fully define a single homotopy class of collision-free (system) trajectory. In reference [42], priority relations are described as *a plan to be executed* in order to properly coordinate the robots, and a simple priority-preserving control was proposed to effectively allow the robots to comply with this ordering. However, efficient priority assignment policies have not been yet been proposed.

A first contribution of the present thesis is the proposal of an *optimal* decision-making algorithm to coordinate autonomous or semi-autonomous robots, that builds upon and extends the priority framework of [42]. In Chapter 4, we establish the link between this priority framework and decision-making in the case of multi-robots coordination, and propose a mixed-integer programming approach to decision-making. This technique is

then adapted to the case of optimal coordination of autonomous robots in Chapter 5, or to the *supervision* of semi-autonomous vehicles in Chapter 6.

By allowing the use of powerful optimization methods such as mixed-integer programming, this decision-making framework is shown to be capable to output real-time optimal coordinated trajectories for up to a dozen of robots. Besides cooperative driving, we forecast potential applications to a much wider range of scenarios, including automated warehouses or guided transportation.

3.2 Autonomous driving

The second major contribution of this thesis is a generic framework to describe the discrete choices involved in motion planning in the case of autonomous driving, *i.e.* with numerous mobile obstacles in a relatively structured environment. This framework, generalizing that of [43], allows encoding the whole decision-making problem as a "shortest-path search" in a *navigation graph*, as presented in Chapter 9; Chapter 10 presents exact and heuristic approaches to use this mathematical framework for actual motion planning.

To the best of our knowledge, this framework is the first to allow systematic enumeration and evaluation of possible driving decisions in generic situations. Previous studies usually assimilate these decisions with homotopy classes of trajectories [43, 29]; however, this notion is shown in Chapter 8 to be insufficient in the context of autonomous motion planning. By contrast, this thesis proposes to generalize this mathematical notion into a decision-making framework explicitly designed for the needs of autonomous driving, opening exciting perspectives to incorporate recent machine learning approaches while guaranteeing vehicle safety.

3.3 Towards practical implementation

Finally, the third part of this thesis focuses on feeding the decision-making algorithms in order to bridge the gap between theory and practice. Indeed, a large part of the motion planning literature has focused on simulation results with perfect or near-perfect sensing capacities, which is far beyond the current state-of-the-art of perception systems for automated vehicles. In Chapter 11, we present early approaches to estimate probable, short-term future trajectories for surrounding vehicles, which we argue is a cornerstone requirement for fully automated driving. Finally, Chapter 12 presents a simplified real-world implementation of our decision-making algorithms to a peri-urban driving scenario serving as a proof-of-concept for the ideas developed throughout the thesis.

3.4 Publications

Most of the results presented in this thesis have been published as conference or journal articles; the list below presents these publications and indicates, when applicable, the corresponding chapter in the thesis.

Articles used as basis for thesis chapters

- F. Altche, X. Qian, and A. de La Fortelle, "Time-optimal coordination of mobile robots along specified paths," in *2016 IEEE/RSJ International Conference on Intelligent Robots*

and Systems (IROS), pp. 5020–5026, IEEE, oct 2016 (used in Chapter 5)

- F. Altché and A. de La Fortelle, "Analysis of optimal solutions to robot coordination problems to improve autonomous intersection management policies," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2016-August, pp. 86–91, IEEE, jun 2016 (used in Chapter 5)

- F. Altché, X. Qian, and A. de La Fortelle, "Least restrictive and minimally deviating supervisor for Safe semi-autonomous driving at an intersection: An MIQP approach," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2520–2526, IEEE, nov 2016 (used in Chapter 6)

- F. Altche, X. Qian, and A. de La Fortelle, "An Algorithm for Supervised Driving of Cooperative Semi-Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3527–3539, dec 2017 (used in Chapters 4 and 6)

- F. Altché, P. Polack, and A. de La Fortelle, "A Simple Dynamic Model for Aggressive, Near-Limits Trajectory Planning," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1–6, 2017 (used in Chapter 7)

- F. Altche, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, IEEE, oct 2017 (used in Chapter 7)

- F. Altche and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2126–2133, IEEE, dec 2017 (used in Chapters 9 and 10)

- F. Altche and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359, IEEE, oct 2017 (used in Chapter 11)

Collaborations not presented in this thesis

- X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 205–210, IEEE, nov 2016

- X. Qian, F. Altché, J. Gregoire, and A. de La Fortelle, "Autonomous Intersection Management Systems: Criteria, Implementation and Evaluation," *IET Intelligent Transport Systems*, feb 2017

- P. Polack, F. Altche, B. D'Andrea-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 812–818, IEEE, jun 2017

- P. Polack, F. Altché, B. D'Andréa-Novel, and A. de La Fortelle, "Guaranteeing Consistency in a Motion Planning and Control Architecture Using a Kinematic Bicycle Model," in *Proceedings of the American Control Conference*, jun 2018

Part I

Cooperative motion planning

Introduction

The first part of this thesis expands on the results on priority-based coordination of multiple mobile robots [42] to illustrate the role of decision-making in cooperative motion planning for multiple vehicles. In a multi-robot coordination problem, a set of cooperative mobile robots having individual goals (*e.g.*, a destination) have to share a finite resource (namely, ground space) as efficiently as possible. This problem is of particular interest for *autonomous intersection management*, where vehicles cooperate with one another or with the road infrastructure in order to improve safety and efficiency at road intersections over more classical techniques such as traffic lights. Although we use the generic term of *robot* for the sake of generality, the discussion can of course be applied to fully (and, in some instances, partially) automated vehicles as well.

Multi-robot coordination can be considered from two complementary viewpoints, which lead to widely different solving algorithms. A first approach is to consider coordination as a resource allocation problem to be solved using job scheduling algorithms [59] or queuing theory (see, *e.g.*, [60]). A second method (see, *e.g.*, [61]) is to model the multi-robots coordination problem as that of planning a collision-free trajectory for a system of mobile robots, and hence falls into the class of motion planning problems. In the following chapters, we adopt this second point of view in the following chapters; the compared advantages and downsides of both formulations are compared in Chapter 5.

As presented in Chapter 2, *priorities* are a convenient way to encode homotopy classes of collision-free trajectories for the system of vehicles, which effectively correspond to discrete choices between relative vehicle orderings. The first contribution of this thesis is the use of priority relations within the framework of mathematical (or, more precisely, mixed-integer) programming to achieve *optimal* coordination of the vehicles. The resulting framework is highly versatile, and the use of well-chosen objective functions allows achieving widely different goals.

In this thesis, we argue that the performance and versatility of this framework are made possible by the choice of a proper representation of the intrinsic decisions involved in the motion planning problem. In this case, decisions are encoded as priority relations or, equivalently, as homotopy classes.

Sketch of Part I This part is divided into three chapters. Chapter 4 lays the foundations of our priority-based framework for cooperative decision-making of multiple robots, which we then use throughout the rest of Part I. A first application of this approach to optimal coordination of automated vehicles is then presented in Chapter 5; in this case, vehicles act as a slave to an authority which prescribes their driving speed through an intersection in order to optimize its throughput. In Chapter 6, we choose a slightly different cost function which results in a widely different behavior known as *supervised driving*: in this case, the authority only acts to correct driver errors which would otherwise inevitably lead to a collision.

Chapter 4

Priority-based cooperative decision-making

" In any moment of decision, the best thing you can do is the right thing. The worst thing you can do is nothing. "

Theodore Roosevelt

Contents

4.1 Multi-robot coordination and motion planning 25
4.2 Priority and decision-making 26
4.3 Mixed-integer programming 28
4.4 Modeling priorities 28
4.4.1 Conflict regions 29
4.4.2 Subregion indicators 30
4.4.3 Priority variables 32
4.5 Dynamic constraints 34
4.6 Chapter conclusion 35

4.1 Multi-robot coordination and motion planning

The problem of multi-robot coordination is that of best sharing a finite resource – in this case, space – to allow multiple mobile robots to reach their respective goals without colliding with one another. Figure 4.1 illustrates a simple example of a two-robots coordination problem: robots 1 and 2 respectively try to reach their goal region G_1 and G_2, but have a potential conflict where the black dashed paths intersect.

In the example of Figure 4.1, a possible solution to avoid conflict would be to have robot 2 follow the dotted red path going around robot 1, effectively removing the risk of collision. However, automated driving applications are usually constrained to existing roads and lanes, which highly restricts the applicability of so-called *deconfliction* techniques, which are mostly used for unmanned aerial vehicles (UAVs) [62, 63]. For this reason, the rest of the discussion focuses on coordinating robots along *fixed paths*, for instance the centerline of a road lane; this hypothesis simplifies the coordination problem into a simpler velocity planning one.

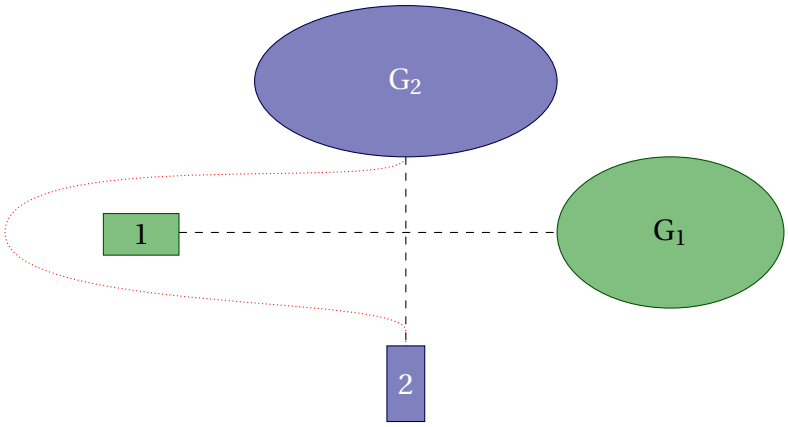


Figure 4.1 – Example of a two-robot coordination problem

Throughout the rest of Part I, we consider that each robot i has a given *reference path* $\gamma_i : \mathbb{R} \to \mathscr{C}_i^f$ which is a (continuous) function of the curvilinear abscissa s such that $\gamma_i(s)$ is the reference configuration of i after traveling a distance s along the path starting from an arbitrary reference point. In this case, $\mathscr{C}_i^f$ is the free portion of the configuration space of robot i when considered alone, *i.e.* we only require that following the reference path does not lead to a collision with a static obstacle; however, reference paths of two distinct robots can be conflicting, as it is the case for the black dashed paths shown in Figure 4.1.

A useful representation to describe collisions between robots is to consider the configuration space for the system of robots, sometimes called *coordination space* (see, *e.g.*, [64, 65]), which we denote by χ. The free portion of the coordination space, χ^f, is a subset of $\prod_{i\in\mathscr{R}} \mathscr{C}_i^f$, but does not contain the sets of configurations leading to collisions between robots. Using this space, we can now define the fixed-path coordination problem as follows:

Problem 1 (Multi-robot coordination along fixed paths)**.** *We consider a (finite) set of mobile robots $\mathscr{R}$ at time t_0, with robot i traveling along a fixed path $\gamma_i : \mathbb{R} \to \mathscr{C}_i^f$. We denote by $x_i^0 \in \gamma_i(\mathbb{R})$ the configuration of robot i at t_0, and assume that each robot has a non-empty goal region $\mathrm{G}_i \subset \gamma_i(\mathbb{R})$.*

The multi-robot coordination problem is that of finding a continuous velocity profile $\mathbf{s} = (s_i)_{i\in\mathscr{R}}$ for the system of robots such that:

i. for all $i \in \mathscr{R}$, $\gamma_i(s_i(t_0)) = x_i^0$,

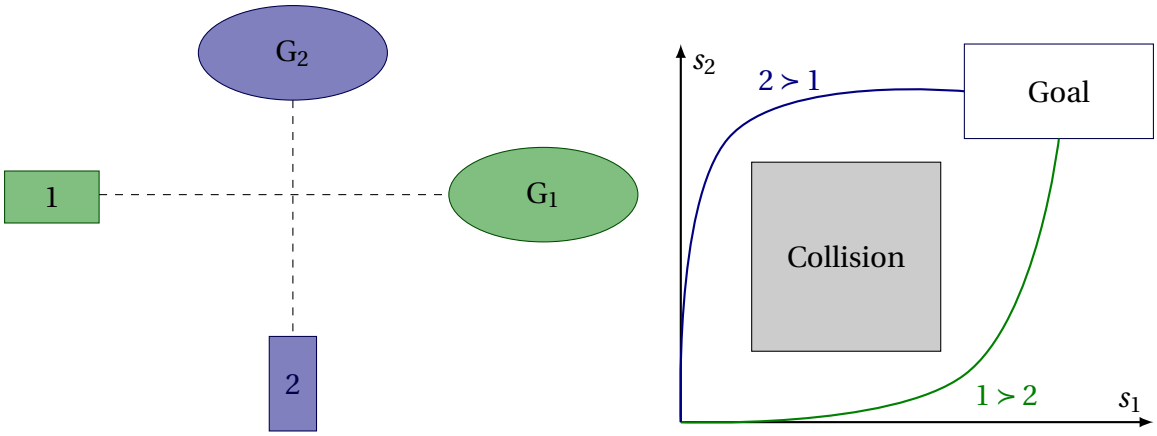
ii. for all $t \geq t_0$, $\big(\gamma_i(s_i(t))\big)_{i\in\mathscr{R}} \subset \chi^f$,

iii. there exists $\mathrm{T} \geq t_0$ such that, for all $t \geq \mathrm{T}$ and all $i \in \mathscr{R}$, $\gamma_i(s_i(t)) \in \mathrm{G}_i$.

We call T *the* completion time *of the problem.*

4.2 Priority and decision-making

Since the reference paths γ_i are given and assumed to be perfectly followed by the robots, Problem 1 can be formulated using curvilinear abscissas, leading to a "path-finding" problem in an abstract space where each dimension corresponds to the curvilinear position of one robot. For the sake of simplicity, we do not make explicit distinctions between



(a) Physical space (b) Coordination space

Figure 4.2 – Correspondence between the physical and the coordination space in a two-robots example

these two representations and we consider curvilinear abscissas and the corresponding robot configurations as equivalent[1].

In this section, we present the so-called *priority framework* introduced in [42], upon which our decision-making algorithm is built, through an example. Figure 4.2 presents the modeling of a real world two-robots coordination problem as a pathfinding problem in the abstract coordination space. The red "Collision" area in Figure 4.2b corresponds to the conflict region at the intersection of the robots' paths and is excluded from the free space χ^f. Paths in the coordination space such as shown in green and blue in can then be mapped to collision-free trajectories for the robots[2].

As was discussed in Chapter 2, the blue and green paths shown in Figure 4.2b correspond to two distinct homotopy classes of paths, namely those avoiding the collision region "from below" (blue path, corresponding to robot 1 crossing before robot 2, and denoted by $1 \succ 2$) or "from above" (green path, corresponding to robot 2 crossing before robot 1 and denoted by $2 \succ 1$). The name "priority" stems from the relationship between these homotopy classes, and relative ordering of the robots. As demonstrated in [42], this observation generalizes to an arbitrary number of robots: **any homotopy class of collision-free paths can be uniquely described by the given of pairwise priority relations between conflicting robots**.

In this thesis, we argue that explicitly using this knowledge about homotopy classes allows designing more efficient motion planners, as they provide a proper description of the decision-making process which has to be undertaken when actually planning a trajectory. The following sections present our proposed approach to treat the challenging question of priority assignment, which remained unanswered in reference [42].

[1] This slight abuse of language may only be problematic for reference paths containing loops, for which the mapping from curvilinear abscissa to configuration is not injective. In this case, additional information is required to map a configuration to a single curvilinear abscissa.

[2] More specifically, for a path $g : [0,1] \to \chi^f$, a collision-free trajectory can be obtained from $g(u(t))$ for any continuous function $u : \mathbb{R} \to [0,1]$. The dynamic feasibility of this trajectory, however, depends on a proper choice of u; this aspect is discussed in Section 4.5.

4.3 Mixed-integer programming

Motion planning usually concerns with finding *efficient* trajectories, and therefore is intimately linked with (constrained) optimization. As was presented in Chapter 2, non-convexity of the search space is one of the main challenges in this regard, and the use of priorities is an effective way of decomposing the difficult, non-convex motion planning problem into smaller (but numerous) convex subproblems. Indeed, as decisions correspond to assignments of pairwise priority relations between robots, the number of possible assignments for N robots can be as large as $2^{\mathrm{N}(\mathrm{N}-1)/2}$; this upper bound is reached, for instance, for N robots on pairwise-distinct paths all intersecting at the same point. However, in more classical situations, many of these possibilities can be discarded as physically unfeasible (*e.g.*, a robot having to go through another) or as inefficient (*e.g.*, introducing unnecessarily long delays by waiting for a robot far away from the conflict point).

A possible way of handling this combinatorial explosion is, therefore, to prune the corresponding decision tree by removing provably suboptimal or infeasible branches. Mixed-integer programming (MIP) is a widely-used framework that allows efficient handling of such combinatorial problems. General MIP problems involving arbitrary functions are very hard, but good techniques exist for a subclass of these problems, called mixed-integer second-order cone programming, or MISOCP [66]. In these problems, a convex quadratic objective function is minimized with quadratic positive semi-definite or affine constraints. A better-known subclass of MISOCP is mixed-integer linear (MILP) or quadratic (MIQP) programming. In this subset of problems, all constraints are required to be linear (possibly involving integer variables), and the objective function is also required to be linear in the MILP variant. These techniques have already been applied to trajectory planning in general, and to the multi-robots coordination problem in particular; a review of existing literature and a comparison with our approach is presented in Chapter 5. A description of MILP/MIQP[3] solving algorithms can be found, for instance, in [67].

Several equivalent formulations of MIQP are commonly found in the literature; in this thesis, we define an MIQP problem as follows:

Problem 2 (MIQP)**.** *We call* mixed-integer quadratic programming *an optimization problem of the form*

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{\textit{Minimize}}} \quad & x^{\mathrm{T}}\mathrm{Q}x + q^{\mathrm{T}}x \\ \textbf{\textit{subject to}} \quad & \mathrm{A}_e x = b_e, \\ & \mathrm{A}_g x + b_g \geq 0, \\ & \forall i \in \mathrm{I}, x_i \in \mathcal{Z}, \end{aligned}$$

where matrix Q *is positive semi-definite and* $\mathrm{I} \subset \{1,\ldots,n\}$ *is used to enforce integrality constraints on certain components of* x*. If* $\mathrm{Q} = 0$*, the above problem is an instance of mixed-integer* linear *programming.*

4.4 Modeling priorities

Due to the success of MIQP methods in multiple fields of operations research, we propose to model the priority relations within the mixed-integer linear-constraints frame-

[3] For brevity purposes and since MILP is a subclass of MIQP, we choose the latter to represent both types of problems, although MILP formulations are quite easier to solve and therefore are frequently preferred.

work in order to leverage these techniques for actual resolution. As introduced in Section 4.2, we only need consider relative orderings within pairs of robots; in what follows, we use the curvilinear abscissa s_i (jointly with the reference path γ_i) to encode the configuration of each robot $i \in \mathscr{R}$, as in Figure 4.2. Moreover, we discretize the problem in time with a fixed time step duration τ, and we denote by s_i^k the position of robot i at time step k (corresponding to time $k\tau$).

4.4.1 Conflict regions

We first define two regions of interest in the coordination space[4]:

Definition 1 (Collision region)**.** *For two robots i and $j \in \mathscr{R}$, we note $C_{ij} \subset \mathbb{R}^2$ the set of positions (s_i, s_j) where the two robots would collide, called* collision region.

In general, C_{ij} can be empty (*e.g.*, for robots on parallel paths) or have one or several connected components (*e.g.*, for paths with multiple intersection points). When C_{ij} has multiple connected components, we denote by C_{ij}^p its p-th component, using the convention $C_{ij}^p = C_{ji}^p$. Figure 4.3 illustrates the shape of the obstacle region[5] in the case of car-like robots traveling in different road configurations.

Definition 2 (Conflict region)**.** *For two robots i and $j \in \mathscr{R}$, we call conflict region between i and j the set of positions $\left\{(s_i, s_j) \in \mathbb{R}^2 \,\middle|\, (\{s_i\} \times \mathbb{R}) \cap C_{ij} \neq \emptyset \text{ or } \left(\mathbb{R} \times \{s_j\}\right) \cap C_{ij} \neq \emptyset\right\}$.*

By this definition, two robots are in their conflict region when there is a possibility of collision between them; of course, the collision region is contained within the conflict region.

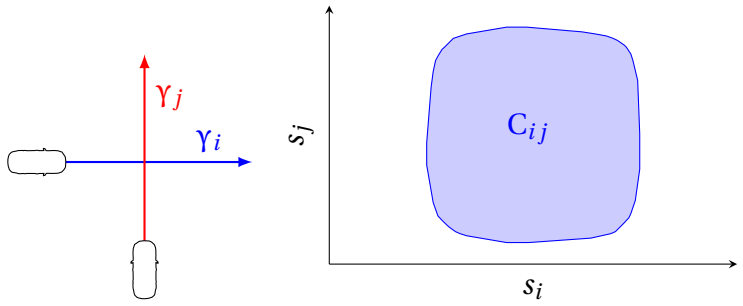
Since the paths and shapes of each robot is known in advance, it is possible to compute the collision regions offline using, *e.g.*, Minkowski difference. Interestingly, each connected component of the collision regions can be approximated efficiently in the shape of a hexagon with edges either horizontal, vertical or parallel to the identity line, denoted by $\hat{C}_{ij}$ in Figure 4.4. Schematically, the horizontal (respectively, the vertical) edge corresponds to having robot j (respectively i) wait for robot i (resp. j) to fully pass the conflict point before moving forward. The diagonal edges correspond to requiring the rear robot to maintain a minimum distance from the lead one, for instance in merging or following situations. Appendix A.1 describes a linear-time algorithm to compute minimum bounding hexagons from a list of points defining the collision regions.

To simplify notations[6], we consider a collision region having a single connected component and drop the superscript p; note that the results still hold in the case of multiple connected components by applying the described method for each component p. We also assimilate the exact collision region C_{ij} with its hexagonal approximation $\hat{C}_{ij}$. We denote by $\underline{s}_{ij}^{\perp}$, $s_{ij}^{\parallel}$ and $\overline{s}_{ij}^{\perp}$ (and their ji counterparts) the coordinates of different vertices of the bounding hexagon, as shown in Figure 4.4.

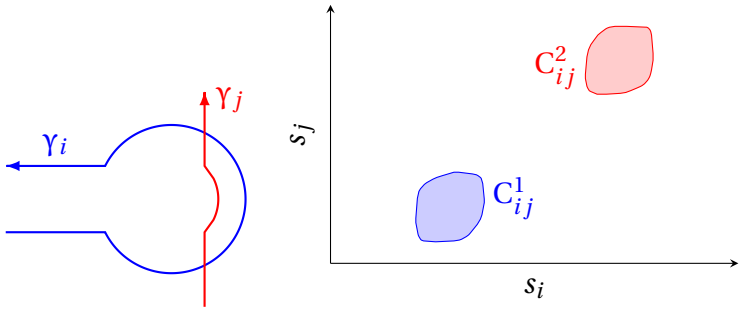
[4] As they only refer to pairs of robots, Definitions 1 and 2 correspond to the projection A of a set $\tilde{A}$ of the form $A \times \mathbb{R}^{n-2} \subset \mathscr{C} \subset \mathbb{R}^n$ (with a possible reordering of the coordinates), onto $\mathbb{R}^2$. To simplify the presentation, we assimilate $\tilde{A}$ with A.

[5] Strictly speaking, Figure 4.3 shows the convex envelope of these regions, which we will use as an approximation of their exact shape.

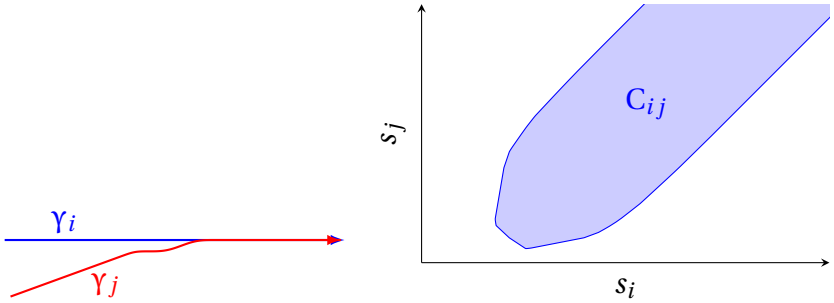
[6] This simplification was implicit in Figure 4.4.



(a) Simple orthogonal intersection situation with example robot shapes



(b) Roundabout situation with multiple connected components



(c) Merging situation

Figure 4.3 – Examples of paths (left) and corresponding collision regions (right) for robots with the polygonal shape shown in Figure 4.3a.

4.4.2 Subregion indicators

To encode priority relations into a mixed-integer programming framework, we propose to constrain the position of each robot i at time step k, denoted by s_i^k. To this end, we introduce a set of subregion indicators, *i.e.* binary variables which we use to decompose $\mathbb{R}^2 \setminus C_{ij}$ into convex regions.

As presented in [68], it is possible to use the so-called "big-M" technique to encode logical constraints in the form of linear inequalities. Consider for instance the linear constraint on variables x and u

$$Mu + x \geq 0, \tag{4.1}$$

with $x \in [a,b]$, $u \in \{0,1\}$ and $M \geq |a|$ a parameter. If $u = 0$, this constraint effectively forces x to be positive; however, if $u = 1$, any value of x satisfies the constraint. Therefore, eq. (4.1) is equivalent to the proposition

$$(u = 0) \Rightarrow (x \geq 0). \tag{4.2}$$

Logical disjunctions between propositions can also be handled in a similar fashion;

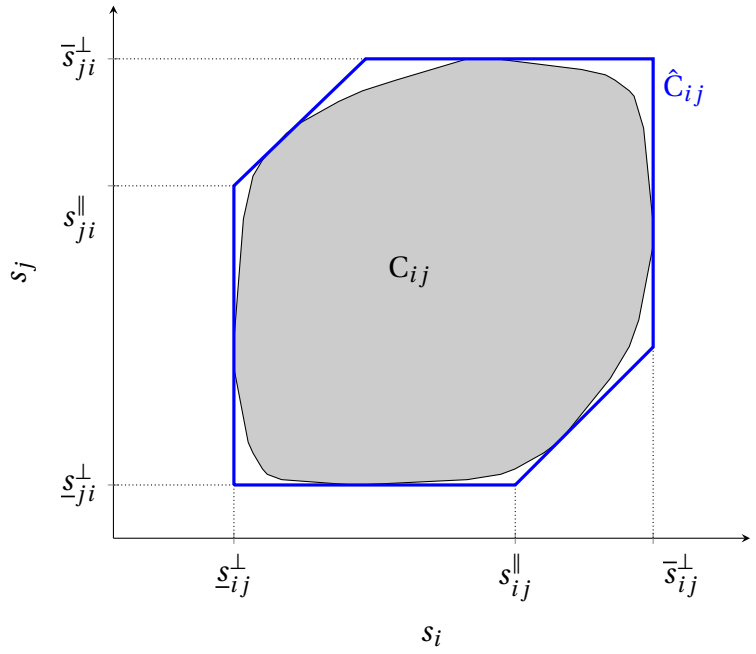


Figure 4.4 – Bounding hexagon approximation of a connected component of a collision region

assume that propositions P and Q are respectively encoded as

$$\mathrm{A_P X} + b_\mathrm{P} \geq 0 \quad \text{and} \tag{4.3a}$$

$$\mathrm{A_Q X} + b_\mathrm{Q} \geq 0, \tag{4.3b}$$

with $\mathrm{A_P}$ and $\mathrm{A_Q}$ matrices and b_P and b_Q vectors of correct dimensions, and X the problem variable. By adding a new binary variable u and a sufficiently large parameter M, disjunction $\mathrm{P} \vee \mathrm{Q}$ (P or Q) can be enforced as

$$\mathrm{A_P X} + b_\mathrm{P} + \mathrm{M}u \geq 0 \quad \text{and} \tag{4.4a}$$

$$\mathrm{A_Q X} + b_\mathrm{Q} + \mathrm{M}(1-u) \geq 0. \tag{4.4b}$$

Note that logical conjunctions do not need such refinements, as they merely correspond to adding additional rows to the constraints matrix.

A slight limitation of this approach is linked to the fact that strict inequalities cannot be enforced in a linear or quadratic optimization framework. Therefore, the negation of a proposition cannot be enforced in a strict sense; a possible way around this limitation is to replace an inequality of the form $\mathrm{A}X + b < 0$ by $\mathrm{A}X + b < -\varepsilon$, with ε a small, positive tolerance parameter. However, since optimization will ultimately be performed numerically, this issue can generally be ignored.

These considerations and results from propositional logic ensure that any proposition involving predicates in the form of linear inequalities can be enforced in an MIQP framework. Therefore, we can introduce the following *subregion indicator* variables:

$$\varepsilon^{\parallel}_{ij}(k) = \mathbb{1}_{\left[s^{\parallel}_{ij}, +\infty\right)}\left(s^k_i\right), \tag{4.5a}$$

$$\varepsilon^{\perp}_{ij}(k) = \mathbb{1}_{\left[\overline{s}^{\perp}_{ij}, +\infty\right)}\left(s^k_i\right), \tag{4.5b}$$

where the set indicator function $\mathbb{1}_\mathrm{X}(x) = 1$ if $x \in \mathrm{X}$, and 0 otherwise[7]. Figure 4.5 presents the different subregions defined by the ε variables; note that, except the bottom-left re-

[7] Note that, due to the limitation regarding strict inequalities, the value of these indicators is actually undefined at their switching point; this limitation does not produce significant problems.

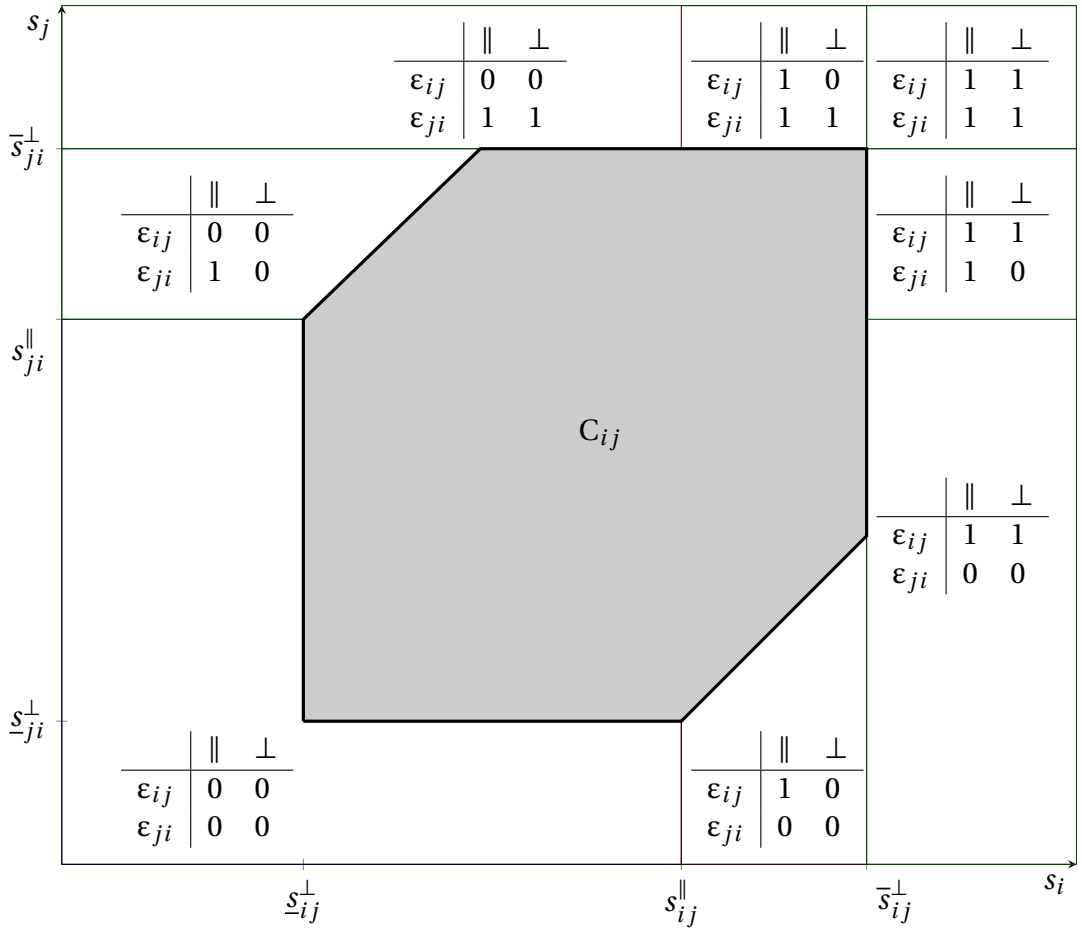


Figure 4.5 – Segmentation of the collision-free portion of the coordination space using the ε indicators variables. The truth tables show the value of each of these variables inside each region.

gion where all variables are zero, all other regions are convex polygons and can therefore be expressed in a matrix inequality form $Ax+b \geq 0$ compatible with an MIQP formulation.

4.4.3 Priority variables

To complete the subregion indicators, we finally introduce a binary *priority variable* between two robots denoted by π_{ij}. This decision variable corresponds to the pairwise priority in the sense of [42]; by convention, we consider that $\pi_{ij} = 1$ corresponds to robot i entering the conflict region before robot j (*i.e.*, $i \succ j$) and we require:

$$\pi_{ij} + \pi_{ji} = 1. \tag{4.6}$$

As illustrated in Figure 4.6, the priority variables are used to restrict the pair of robots to remain on the bottom-right ($\pi_{ij} = 1$) or top-left ($\pi_{ij} = 0$ or, equivalently, $\pi_{ji} = 1$) of the obstacle region.

Coupling the priority variables with the subregions of Figure 4.5 provides a partition of the entire collision-free portion of the coordination space into convex polygons. This

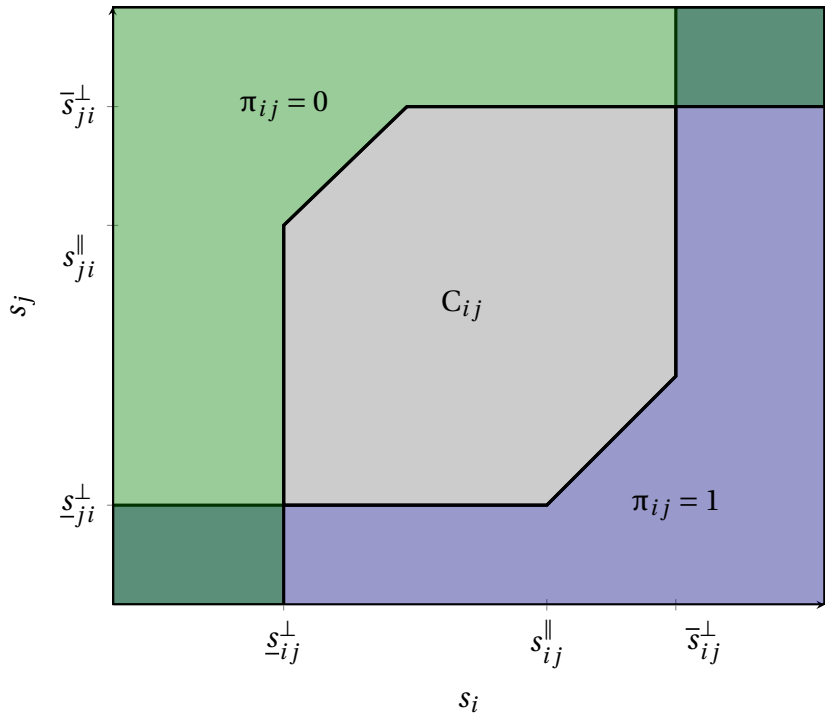


Figure 4.6 – Separation of the coordination space using the priority variables. The blue region is only accessible if $\pi_{ij} = 1$, and the green region if $\pi_{ij} = 0$ (or, equivalently, $\pi_{ji} = 1$). The bottom-left and top-right regions are accessible in both cases.

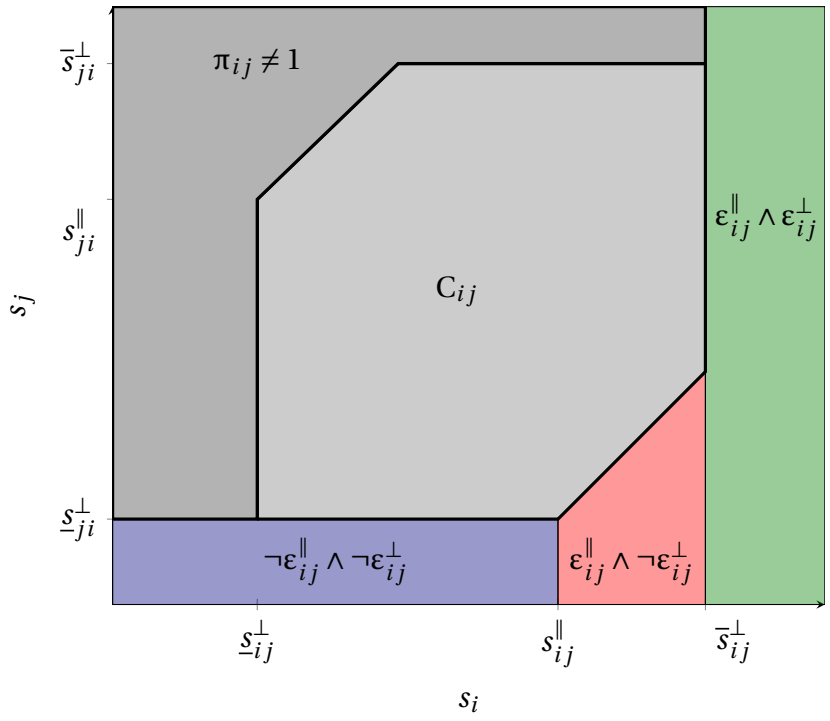


Figure 4.7 – Partition of the coordination space using the priority and ε variables, illustrated in the case $\pi_{ij} = 1$. In this case, collision avoidance is achieved by requiring $s_j \leq \underline{s}^{\perp}_{ji}$ in the blue $\left(\neg\varepsilon^{\parallel}_{ij} \wedge \neg\varepsilon^{\perp}_{ij}\right)$ region (constraint (4.7a)) and requiring to remain below the diagonal line delimiting the red $\left(\varepsilon^{\parallel}_{ij} \wedge \neg\varepsilon^{\perp}_{ij}\right)$ region, *i.e.* $s_j - \underline{s}^{\perp}_{ji} \leq s_i - s^{\parallel}_{ij}$ (constraint (4.7b)). No constraint is necessary in the green $\varepsilon^{\perp}_{ij}$ region.

partition in turn allows implementing the collision avoidance constraints as:

$$\left(\pi_{ij} \wedge \neg \varepsilon^{\parallel}_{ij}(k)\right) \Rightarrow s_j^{k+1} \leq \underline{s}^{\perp}_{ji} \tag{4.7a}$$

$$\left(\pi_{ij} \wedge \varepsilon^{\parallel}_{ij}(k) \wedge \neg \varepsilon^{\perp}_{ij}(k)\right) \Rightarrow s_i^{k+1} - s^{\parallel}_{ij} \geq s_j^{k+1} - \underline{s}^{\perp}_{ji} \tag{4.7b}$$

$$\left(\pi_{ij} \wedge \varepsilon^{\parallel}_{ij}(k) \wedge \neg \varepsilon^{\perp}_{ij}(k)\right) \Rightarrow s_i^{k+1} - s^{\parallel}_{ij} \geq s_j^{k+1} - \underline{s}^{\perp}_{ji} + \frac{\tau}{2}\left(v_j^{k+1} - v_i^{k+1}\right), \tag{4.7c}$$

where v_i^k is the longitudinal velocity of robot i at time step k, and τ is the duration of a time step; operator $\neg$ corresponds to the logical negation. The subregions involved in constraints (4.7a) to (4.7c) are illustrated in Figure 4.7.

Remark 1. *In constraints (4.7a) to (4.7c), the position of each robot at step k induces a constraint at step $k+1$, in order to avoid "corner-cutting" situations. Similarly, constraint (4.7c) is added to avoid possible collisions within the duration of a time step; Appendix A.2 provides additional details about these two issues.*

Remark 2. *Throughout this chapter, we only considered the interactions between pairs of robots. Indeed, a particularity of the multi-robot coordination problem is that the collision region (in the n-dimensional coordination space) is a union of cylinders of the form $\mathbb{R}^{i-1} \times \mathrm{A}_i \times \mathbb{R}^{j-i-1} \times \mathrm{A}_j \times \mathbb{R}^{n-j}$ – or, by reordering variables, $\mathrm{C}_{ij} \times \mathrm{RR}^{n-2}$ – with $\mathrm{C}_{ij} = \mathrm{A}_i \times \mathrm{A}_j$. Therefore, this two-dimensional approach generalizes to the n-dimensional problem simply by considering all pairs of conflicting robots.*

4.5 Dynamic constraints

Constraints (4.7a) to (4.7c) effectively prevent collisions between robots, provided they maintain a constant acceleration over each time step. Moreover, physical robots have limited actuation capacities which do not allow them to change speed instantaneously. To model this behavior, we choose a simple, one-dimensional second-order integrator model for robots dynamics, in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} s_i \\ v_i \end{pmatrix} = \begin{pmatrix} v_i \\ a_i \end{pmatrix}, \tag{4.8}$$

with s_i the curvilinear position, $v_i = \dot{s}_i$ the longitudinal speed and $\ddot{s}_i = a_i \in [\underline{a}_i, \overline{a}_i]$ the (bounded) longitudinal acceleration[8], and we assume that robots are only allowed to move forward ($v_i \geq 0$) with a bounded velocity $\overline{v}_i$. In a time-discretized setting, we approximate the robot dynamics by requiring a constant acceleration over each time step, and we enforce these constraints as

$$s_i^{k+1} - s_i^k = \frac{1}{2}\left(v_i^k + v_i^{k+1}\right)\tau, \tag{4.9a}$$

with the additional requirements

$$0 \leq v_i^k \leq \overline{v}_i \quad \text{and} \tag{4.9b}$$

$$\underline{a}_i \tau \leq v_i^{k+1} - v_i^k \leq \overline{a}_i \tau \tag{4.9c}$$

[8] With $\underline{a}_i < 0 < \overline{a}_i$.

4.6 Chapter conclusion

In this chapter, we leveraged the notion of *priority* as a homotopy invariant to partition the collision-free portion of the coordination space into convex, polygonal subregions by considering the robots of $\mathscr{R}$ pairwise. To this end, we introduced two sets of decision variables: the (time-independent) *priority variables* π_{ij}, encoding which robot should first go through the conflict region, and the time-dependent *subregion indicators* $\varepsilon_{ij}^{\parallel}(k)$ and $\varepsilon_{ij}^{\perp}(k)$, determining when the robots should go from one subregion to another. Moreover, we proposed a simple dynamic model for the robots, which is sufficient to capture a large range of actuating limitations – although important dynamic effects such as skidding in high-velocity curves are neglected.

In doing so, we cast the multi-robots coordination problem as a constraint satisfaction one that can be encoded in a mixed-integer quadratic programming framework provided a suitable objective function is chosen. We argue that using priorities as a means to render the underlying decision-making problem explicit provides interesting properties and allows efficiently solving a large class of problems by simply changing the cost function, as will be illustrated in Chapters 5 and 6.

Chapter 5

Optimal coordination of robots along fixed paths

" The best for the group comes when everyone in the group does what's best for himself and the group. "

John Nash

Contents

5.1 Time-optimal coordination 37
5.2 Problem modeling 39
5.3 MILP formulation 40
5.3.1 Problem-specific constraints 40
5.3.2 Objective function(s) 40
5.3.3 Optimization problem 42
5.4 Simulation results 42
5.4.1 Microscopic simulation 42
5.4.2 Computation time 43
5.5 Some results for traffic management 44
5.5.1 Trajectory-level simulation 44
5.5.2 Number of vehicles 44
5.5.3 Design of near-optimal policies 45
5.6 Chapter conclusion 47

5.1 Time-optimal coordination

In the previous chapter, we used priorities as decision variables to encode the multiple choices arising in the multi-robot coordination problem, and formulated a set of mixed-integer constraints to ensure collision avoidance between robots with a simple second-order dynamic model. This formulation, however, is not complete without an objective function to be chosen according to an actual problem. In this chapter, we focus on *time-optimal* coordination of robots, *i.e.* finding the solution to the multi-robot coordination

problem (problem 1) minimizing the completion time T. Formally, we define the time-optimal coordination problem as:

Problem 3 (Time-optimal coordination along fixed paths)**.** *We consider a (finite) set of mobile robots $\mathscr{R}$ at time t_0, with robot i traveling along a fixed path $\gamma_i : \mathbb{R} \to \mathscr{C}_i^f$. The* time-optimal *coordination problem is that of finding a continuous velocity profile $\mathbf{s}^* = (s_i^*)_{i\in\mathscr{R}}$ for the system of robots such that:*

i. $\mathbf{s}^*$ *is a solution to the coordination problem (Problem 1) with completion time* T^**,*

ii. for any $\mathbf{s}$ *solution of problem 1 with completion time* T*, we have* $\mathrm{T} \geq \mathrm{T}^*$*.*

The time-optimal coordination problem obviously finds applications in robotics, for instance in automated warehouses with robots moving objects along aisles. In the field of automated vehicles, this problem relates to the broader concept of *autonomous intersection management* (AIM), where vehicles cooperate with one another or with the road infrastructure in order to improve safety and efficiency at road intersections. In the rest of this section, we offer a review of the AIM literature – which is more advanced than that on automated warehouses – to illustrate the gains of our approach over existing techniques.

Traffic lights are a means of coordinating vehicles in road intersections by alternating right-of-way between lanes to avoid possible collisions. Extensive research has focused on algorithms to determine good timings of the phases from a statistical perspective [69, 70] off-line; more recently, several authors have proposed using sensors [71, 72] or communication [73] to adapt traffic lights phases and timing in real-time. However, this coordination scheme is in the end limited by its relative coarseness, which can lead to vehicles idling unnecessarily and therefore increase pollution, travel time and congestion.

Naumann et al. are the first to propose further improving intersection management by providing finer control over the vehicles. In this scheme, each vehicle reserves an area of the intersection for a certain duration, and only the vehicles having a valid reservation can enter at a given time [13]; Dresner et al. later perfected this approach to accommodate a larger number of vehicles [74]. A limitation of such algorithms is that the order in which reservations are granted has a significant influence on the efficiency of the autonomous intersection. For instance, reference [74] uses a simple first-come, first-served (FCFS) policy to grant reservations, which is paradoxically known to be *less* efficient than pre-timed traffic lights for higher levels of traffic [75]. Some refinements to the reservation-granting policy have been proposed (see, *e.g.*, [76, 77]), but these techniques remain heuristic and provide no optimality guarantee. As mentioned in the introduction of Part I, these reservation-based algorithms effectively correspond to a job scheduling approach solved using a greedy heuristic.

Other authors have proposed optimization-based algorithms of the job scheduling formulation. A classical assumption is to discretize robots paths into segments of predetermined length (not necessarily equal), and ensure collision avoidance by requiring that two robots cannot simultaneously occupy conflicting segments. In [61], this assumption is translated into a mixed-integer nonlinear programming (MINLP) problem, which cannot be solved in reasonable time; for this reason, approximations of the vehicle dynamics are made to cast the MINLP problem into two simpler mixed-integer linear programming (MILP) problems. Similarly, Digani et al. [78] use a minimum-velocity hypothesis to formulate a (non-convex) quadratic programming problem for multi-robot coordination with kinematic constraints. Despite recent advances in such formulations (see, *e.g.*, [79]),

handling kinematic or dynamic constraints in a spacially-discretized coordination problem remains impractical and requires approximations which can lead to suboptimality or infeasibility issues.

By contrast, few literature entries have considered using motion planning techniques for autonomous intersection management although this approach has been used, *e.g.*, for collision avoidance between spacecrafts [80]. In this chapter, we propose to use the priority-based decision-making framework presented in Chapter 4 in order to solve the time-optimal coordination problem.

5.2 Problem modeling

We consider a set $\mathscr{R}$ of n robots evolving on predetermined paths, and we assume that coordination between robots is only needed inside a bounded region, which we call the *coordination region*. In the case of automated driving, for instance, coordination is primarily needed in the middle of a road intersection (Figure 5.1), while vehicles only need to keep a safe distance from the vehicle in front of them when they are not inside the intersection. In this example, the coordination region would be chosen as the center of the intersection and a portion of the roads leading to, and exiting from this center.

As in Chapter 4, each robot $i \in \mathscr{R}$ is supposed to follow a predetermined path γ_i inside the coordination region with the dynamics described in Section 4.5; we denote by s_i the curvilinear position of robot i along its path and $v_i = \dot{s}_i$ its longitudinal velocity. The velocity is non-negative bounded such that $v_i \in [0, \overline{v}_i]$ and we assume that the longitudinal acceleration a_i of robot i is bounded to an interval $\left[\underline{a_i}, \overline{a_i}\right]$ with $\underline{a_i} < 0 < \overline{a_i}$.

The origin of s_i is chosen so that $s_i = 0$ when the front of the robot enters the coordination region, and $s_i = s_i^{out} > 0$ when it fully exits the coordination region, and we define the goal region $G_i = [s_i^{out}, +\infty)$. s_i can therefore be interpreted as the distance traveled by robot i inside the coordination region, and the goal is reached when exiting this area.

We assume that robot i enters the coordination region at time t_i^{in} with speed $v_i^{in} \in [0, \overline{v}_i]$, and is required to leave the coordination region with speed v_i^{out}; we let t_i^{out} denote the corresponding exit time. Note that v_i^{out} should be properly chosen to avoid collisions outside of the coordination region.

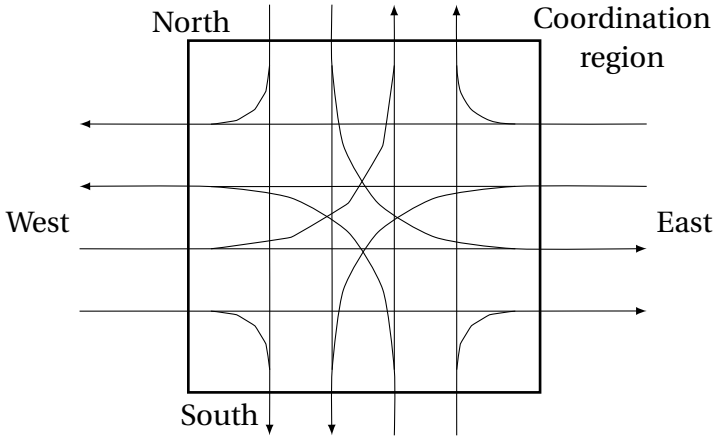


Figure 5.1 – Example of paths inside and outside the coordination region for a two-lanes road intersection.

5.3 MILP formulation

We now formulate the time-optimal coordination problem as a mixed-integer linear program. For the problem to be tractable, we consider a finite horizon $K\tau$ (corresponding to $K+1$ time steps indexed by $k \in \{0,\dots,K\}$), and we require that $K\tau$ is large enough to guarantee that all robots can reach their goal by the end of the horizon.

The results of Chapter 4 allow fully encoding the collision-avoidance and dynamic constraints on the system of robots, and are used as-is in our MILP formulation. To complete this generic framework, we now add a few problem-specific variables and constraints as presented below.

5.3.1 Problem-specific constraints

For each robot i and at each time step k, we introduce a pair of (binary) indicator variables μ_i^k and σ_i^k as:

$$\mu_i^k = \mathbb{1}_{\mathbb{R}_+}(s_i^k) \quad \text{and} \tag{5.1a}$$

$$\sigma_i^k = \mathbb{1}_{G_i}(s_i^k). \tag{5.1b}$$

Therefore $\mu_i^k = 1$ if robot i has entered the coordination region at step k and 0 otherwise, and $\sigma_i^k = 1$ if ro bot i has exited the coordination region at step k (*i.e.*, $s_i^k \geq s_i^{out}$), and $\sigma_i^k = 0$ otherwise.

To account for the initial and terminal constraints on the problem, we enforce:

$$\mu_i^k = 0 \Rightarrow v_i^{k+1} = v_i^{in}, \tag{5.2a}$$

$$\sigma_i^{k+1} = 1 \Rightarrow v_i^k = v_i^{out}, \tag{5.2b}$$

$$v_i^0 = v_i^{in}, \tag{5.2c}$$

$$s_i^0 = -v_i^{in} t_i^{in}, \tag{5.2d}$$

$$s_i^K \geq s_i^{out} \tag{5.2e}$$

$$\tag{5.2f}$$

for all $i \in \mathscr{R}$ and all $k \in \{0,\dots,K-1\}$, where K is the final time step of the problem. Note that by design, constraint (5.2e) guarantees that all robots eventually exit the coordination region; therefore, any feasible solution is sure to be deadlock-free.

5.3.2 Objective function(s)

The optimality criterion in problem 3 corresponds to minimizing k such that, for all $i \in \mathscr{R}$, $\sigma_i^k = 1$. A possible method to design a suitable objective function corresponding to this criterion relies on introducing a set of auxiliary binary variables T_k, with constraints

$$T_k = 1 \Rightarrow \sum_{i \in \mathscr{R}} \sigma_i^k = n, \tag{5.3}$$

and using

$$\Phi_1(\mathbf{X}) = \sum_{k=0}^{K} T_k \tag{5.4}$$

as objective function to be *maximized*, where $\mathbf{X}$ denotes the set of all problem variables.

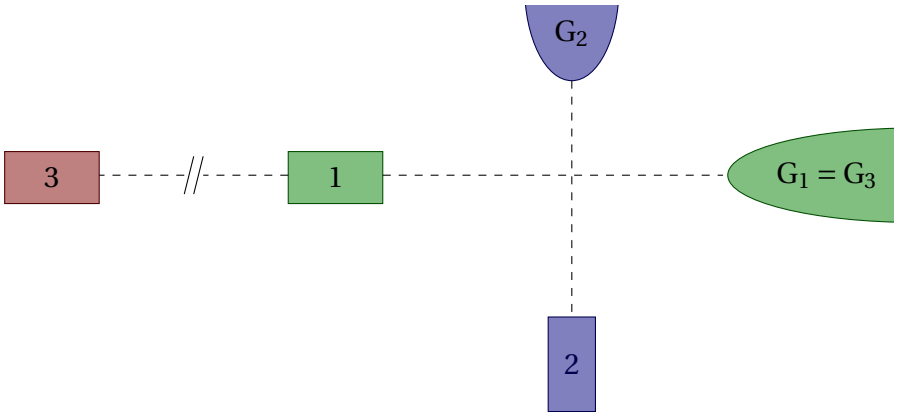


Figure 5.2 – Example of individual suboptimality using global completion time as optimization function. The red robot (3) has a much longer path to travel than the others, and the global completion time therefore almost only depends on that of 3.

A limitation of using objective function Φ_1 is that this optimization criterion can lead to very "inefficient" decisions on the scale of individual robots, as it does not discriminate between solutions having the same global completion time. To illustrate this issue, consider the case depicted in Figure 5.2 where one robot (denoted 3) has a much longer path to travel than the others, *i.e.* Φ_1 almost only depends on the arrival time of robot 3. In this case, Φ_1 will evaluate similarly a situation where robot 2 first crosses, then robot 1 proceeds as soon as possible, and a situation where robot 1 waits for 3 to catch up before moving forward. This may or may not constitute an issue in the case of automated warehouses, depending on whether robots have more tasks to accomplish after the current one. However, this is not an acceptable behavior in the case of autonomous intersection management.

For this reason, it may be desirable to use a finer granularity in the objective function, using

$$\Phi_2(\mathbf{X}) = \frac{1}{n} \sum_{i \in \mathscr{R}} \sum_{k=0}^{\mathrm{K}} \sigma_i^k. \tag{5.5}$$

Maximizing Φ_2 corresponds (as the time step duration τ goes to 0) to minimizing the average completion time of all robots, and therefore optimizes individual travel times within the limits of reaching the same global average minimum. In other words, objective function Φ_2 allows selecting the "best" solution for the group of robots, in the sense of Nash's quote at the head of this chapter.

Despite its merits, Φ_2 may still not be best suited for practical applications, where to value of τ cannot be made arbitrarily small due to the limited computational power available. Indeed, Φ_2 does not distinguish solutions with completion times differing by less than the duration of a time step for at least one robot. To correct this issue, we make use of the fact that maximizing the speed of a robot allows to minimize its (non-discretized) completion time. Therefore, we propose adding an "averaged normalized speed" term to function Φ_2, and define

$$\Phi_3(\mathbf{X}) = \Phi_2(\mathbf{X}) + \frac{1}{n\mathrm{K}} \sum_{k=0}^{\mathrm{K}-1} \frac{v_i^k}{\overline{v_i}} = \frac{1}{n} \sum_{i \in \mathscr{R}} \left(\sum_{k=0}^{\mathrm{K}} \sigma_i^k + \frac{1}{\mathrm{K}} \sum_{k=0}^{\mathrm{K}-1} \frac{v_i^k}{\overline{v_i}} \right). \tag{5.6}$$

Objective function Φ_3 allows selecting the solution with highest average speed and thus smallest continuous completion time. The weighing of the added term ensures that this

solution is still optimal for Φ_2 since $\sum_{k=0}^{K-1} \frac{v_i^k}{\overline{v}_i} \leq K$. Note that this function is linear in the problem variables.

5.3.3 Optimization problem

For the above reasons, we propose to use Φ_3 as objective function; the time-optimal[1] coordination problem can then be expressed under the mixed-integer linear programming framework as:

$$\begin{aligned} &\underset{\mathbf{X}}{\text{Minimize}} && \Phi_3(\mathbf{X}) \\ &\text{subject to} && \text{constraints (4.6), (4.7), (4.9) and (5.2)} \end{aligned} \tag{5.7}$$

Solving eq. (5.7) of course provides the optimal decision regarding robots ordering in the form of pairwise priority relations. However, as our formulation couples the high-level decision-making process with robots dynamics, it is also possible to get the *optimal trajectory* for each robot corresponding to this ordering. As will be discussed in Chapter 10, the tight coupling between dynamics and decision-making seems to be at the core of the algorithm efficiency.

5.4 Simulation results

The use of the above optimization problem to find a time-optimal coordination has been validated by computer simulation on the example of autonomous vehicles in the intersection of Figure 5.1. The simulation is based on the free traffic modeling tool SUMO [81] and uses its path generation algorithm to compute collisions sets. Note that some additional "helper" constraints are introduced to improve computation time, as described in Appendix B.1.

Vehicles are generated either deterministically (Section 5.4.1), or using random Poisson arrival times with normally-distributed entry speeds truncated to a minimum and a maximum speed (Section 5.4.2). Optimization problem (5.7) is then run into the commercial MILP solver Gurobi [67], using its Python interface to generate the constraints. Lastly, if the problem is feasible, the solution trajectories are simulated in SUMO using the TraCI interface to verify that they do not generate collisions.

In all simulations, vehicles are modeled as rectangles of 4 m length by 2 m width, with $\underline{a}_i = -3\,\mathrm{m\,s^{-2}}$ and $\overline{a}_i = 4\,\mathrm{m\,s^{-2}}$. The exit speed for all $i \in \mathscr{R}$ is set as $v_i^{out} = \overline{v} = 15\,\mathrm{m\,s^{-1}}$. The entry speed v_i^{in} is deterministically chosen in the first simulation and is normally distributed with average $12\,\mathrm{m\,s^{-1}}$ and standard deviation $3\,\mathrm{m\,s^{-1}}$, truncated to $[10, 15]\,\mathrm{m\,s^{-1}}$ in Section 5.4.2.

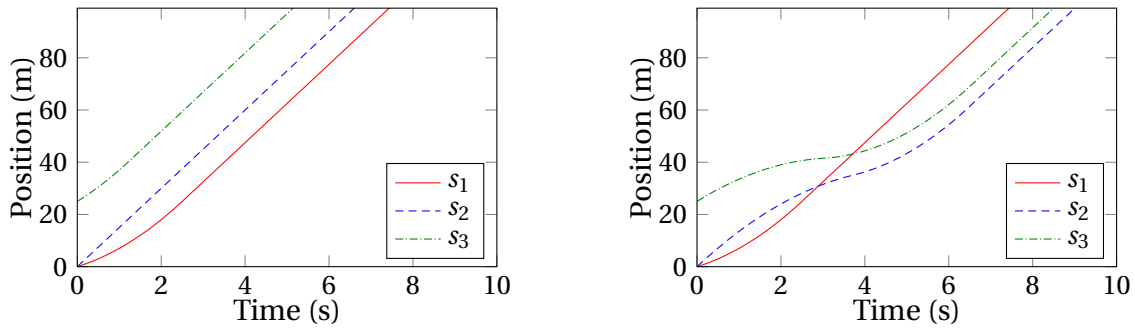
Simulations were performed on a personal computer running on a 3.60 GHz Intel Core i7-4790 CPU with 16 GB of RAM, using version 6.5 of Gurobi. A replay of some of our simulations is available online[2].

5.4.1 Microscopic simulation

We first demonstrate the ability of our approach to find the global optimum on a simple example with three vehicles on the intersection of Figure 5.1: vehicle 1 goes from south

[1] With modified optimality criterion

[2] https://youtu.be/RiW2OFsdHOY



(a) Optimal priorities

(b) Suboptimal priorities

Figure 5.3 – Optimal trajectories within given priority classes, corresponding to a global (left) and local (right) optimum

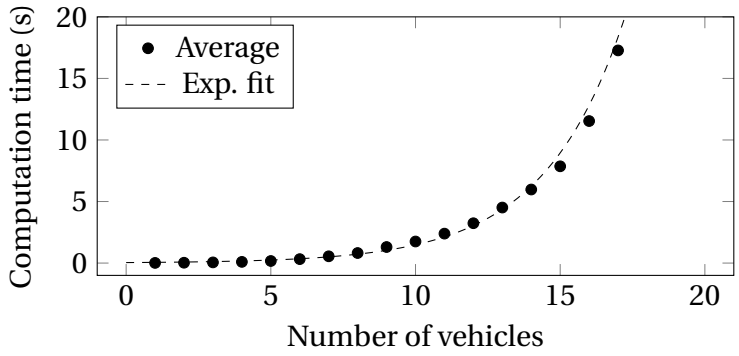


Figure 5.4 – Average computation time over 10 instances for a time step duration of 1 s and a varying number of vehicles.

to west, vehicle 2 from west to east and vehicle 3 from north to south. Vehicles initially start at $\left(s_1^0, s_2^0, s_3^0\right) = (0, 0, 25)$ m with speeds $\left(v_1^{in}, v_2^{in}, v_3^{in}\right) = (5, 15, 10)$ m s^{-1}. Figure 5.3a shows the globally optimal trajectories for each vehicle, which lies in the homotopy class represented by priorities $3 \succ 2$, $2 \succ 1$, $3 \succ 1$. For comparison purposes, Figure 5.3b shows the (locally) optimal trajectories when sub-optimal priorities $1 \succ 3$, $3 \succ 2$, $1 \succ 2$ are enforced. The optimum average completion time found by solving Equation (5.7) is 6.5 s, and the example sub-optimal one is 8.4 s.

5.4.2 Computation time

We know that the motion planning problem we are trying to solve is NP-hard, and the worst-case complexity of mixed-integer programming is[3] $\mathcal{O}(2^b)$ with b the number of binary variables. As formulation Equation (5.7) uses a large number of binary variables for the indicators, b can scale as high as $\mathcal{O}(n^2\mathrm{K})$, thus theoretically leading to prohibitively large computation times for even a few vehicles[4].

To measure the computation time actually required to solve problem (5.7) for the intersection displayed in Figure 5.1, we run the simulator for different numbers of vehicles with a fixed time step of 1 s and a 30 s time horizon (*i.e.*, K = 30); results are displayed

[3] If all integer variables are binary

[4] Evaluating complexity is actually much more difficult, as b actually scales as $c^2\mathrm{K}$ where c is the number of pairwise conflicts and therefore heavily depends on paths geometry. However, the $n^2\mathrm{K}$ bound could be reached, *e.g.* if all robots follow different paths, all of them intersecting at the same point.

in Figure 5.4. Although computation time does scale exponentially as expected from the theoretical analysis, it remains below 1 s for up to 9 vehicles which would make our approach suitable for real-world applications with reasonable levels of performance. Note that the same time horizon is considered across all simulations to provide a fair comparison, while a shorter one could be used for problems involving fewer vehicles, thus further reducing computation time. Moreover, time step duration τ can be adjusted depending on the problem; Appendix B.2 discusses the influence of the chosen value of τ on solution optimality.

Part of this impressive performance – as a theoretical complexity bound would be $2^{9^2 \cdot 30/2} \approx 10^{182}$ – is to put to the credit of the very efficient heuristics leveraged by the solver. Experimentally, we often observe that the solver terminates without needing to branch, *i.e.* linear relaxation of the root node and simple rounding heuristics are sufficient to find the optimal solution. Although the black-box behavior of the solver does not allow studying this question, we argue that intrinsic properties of our formulation, notably a well-established hierarchy between explicit decision variables (especially π_{ij}) and kino-dynamic constraints, are key towards delivering this performance. Further study of the problem structure could be a topic for future research.

5.5 Some results for traffic management

Before concluding this chapter, we propose to use our framework to provide additional insight on autonomous intersection management policies. Indeed, the bijection between priority relations and homotopy classes of collision-free trajectories ensure that *any* AIM policy can be enforced through priorities. In Sections 5.5.1 and 5.5.2, we study the first-come, first-served (FCFS) ordering policy, *i.e.* where vehicle i is given priority over j, if i enters the coordination region before j. In Section 5.5.3, we propose to derive a heuristically "good" policy from our optimal coordination algorithm.

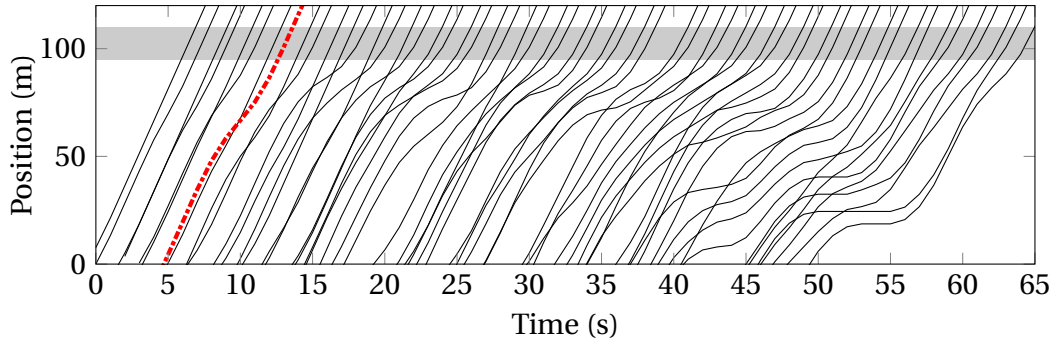
5.5.1 Trajectory-level simulation

In this first set of simulations, we compare the optimal trajectories obtained from Equation (5.7), with those resulting when enforcing a first-come, first-served (FCFS) ordering. Figure 5.5 shows these trajectories, for a given set of 57 vehicles on a single-lane, four-roads intersection. For illustration purposes, vehicles only travel from south to north, and from east to west in this example; the collision area is shown in gray. It can be seen that FCFS ordering causes a higher average completion time for vehicles, and in fact saturates the intersection, as can be seen in the increasing time vehicles remain stopped. By contrast, an optimal ordering allows the same vehicles to pass without stopping. Note that this does not guarantee that an optimal ordering better serves each individual vehicle; for instance, the vehicle with trajectory highlighted in red exits the intersection later in the optimal ordering than in the FCFS ordering. A video of this simulation is available online[5].

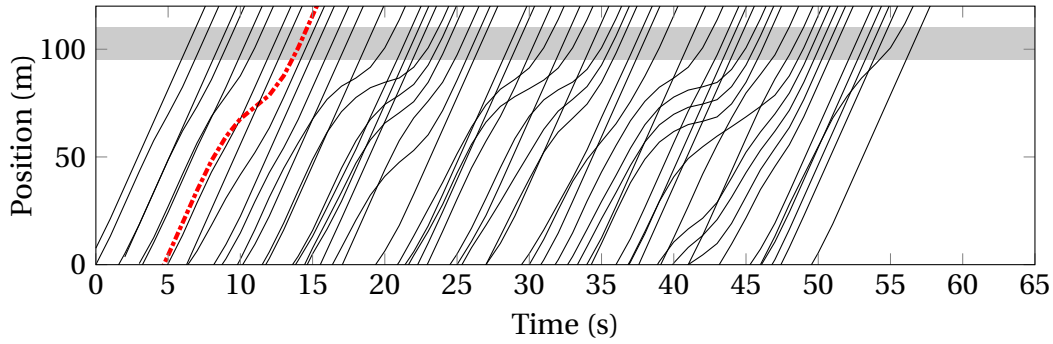
5.5.2 Number of vehicles

It is known that first-come, first-served policies are often efficient for lower volumes of traffic, whereas they lose efficiency when the number of vehicles increases [75]. Fig-

[5] https://youtu.be/U7I7x09Hkeo



(a) First-come, first-served ordering



(b) Optimal ordering

Figure 5.5 – Trajectories of the vehicles in a single-lane intersection for FCFS and optimal ordering of the vehicles. Notice that trajectories do not intersect in the gray area, which corresponds to the collision region.

ure 5.6 shows the influence of the number of vehicles N on the average relative time loss for the optimal priority assignment and FCFS ordering. This indicator is computed as $\frac{t_i^{policy} - t_i^{alone}}{t_i^{alone}}$, where t_i^{alone} is the minimum completion time of vehicle i if there were no other vehicle[6], and t_i^{policy} the completion time of the vehicle in the given policy. This data has been computed over a set of 20 initial configurations on the intersection presented in Figure 5.1.

5.5.3 Design of near-optimal policies

Due to the relatively high computation time of our optimal coordination algorithm, it may be impractical to use it as-is for autonomous intersection management. However, as mentioned in Chapter 2, finding a provably optimal solution may be less important than finding a "good enough" one much faster.

In a MIP framework, it is indeed possible to reduce computation time by using a less strict termination criterion for the solver. Indeed, the branch-and-bound algorithms used in most solvers keep exploring the decision tree until the current solution is very close to the best known bound for the objective function, usually with tolerated relative difference (or gap) of less than 10^{-4}. In our particular formulation, we observe that the solver generally finds a reasonably good solution quite fast, then spends a lot of time improving

[6] Let $t_i^{acc} = \frac{\overline{v}_i - v_i^{in}}{\overline{a}_i}$ be the time needed for i to accelerate to $\overline{v}_i$, and $d_i^{acc} = v_i^n t_i^{acc} + \frac{1}{2}\overline{a}_i t_i^{acc2}$ be the distance covered by i during this acceleration phase. We define $t_i^{alone} = t_i^{acc} + \frac{s_i^{out} - d_i^{acc}}{\overline{v}_i}$.

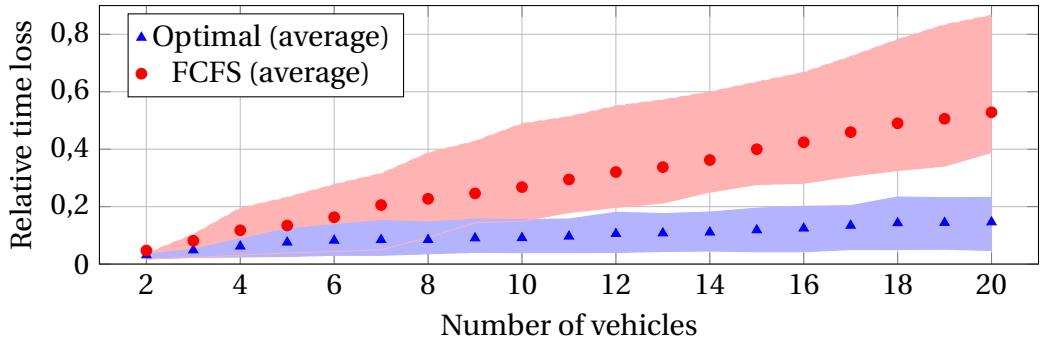


Figure 5.6 – Comparison of average normalized delays for optimal and FCFS ordering for an increasing number of robots. The shaded areas correspond to the [50%, 75%] percentiles of vehicles for each policy.

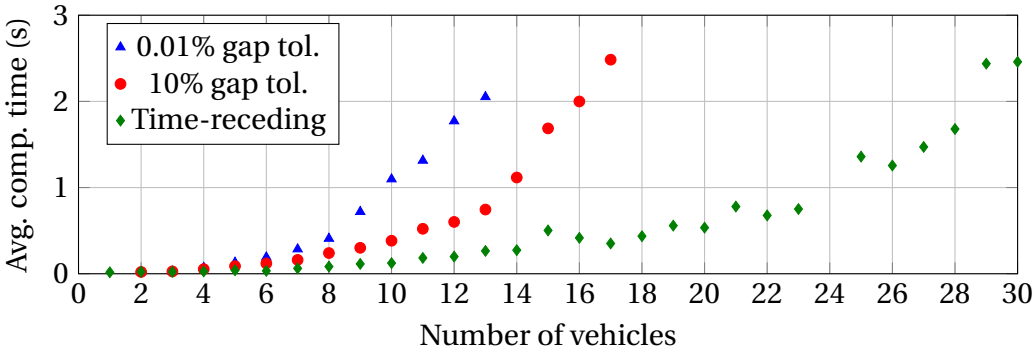


Figure 5.7 – Computation time, depending on the number of vehicles, for a gap tolerance value of 10^{-4} and 10^{-1}, and for the time-receding approach with tolerance value of 10^{-1}.

this solution to gain a few percents of optimality. For real-time applications, a possible approach to decrease computation time is therefore to increase the maximum tolerated gap, leading to a potentially sub-optimal solution in a much shorter time while ensuring an upper bound on the loss of optimality. The red and blue curves in Figure 5.7 compare the average computation times for a gap tolerance of 10^{-4} and 10^{-1} over 10 instances for a time step duration of 1 s; in our test instances, the actual average optimality loss from increasing gap tolerance to 10% is approximately 5%. Using this method, up to 14 vehicles can be treated in less than a second, whereas only 10 vehicles can be treated in the same time span with a gap tolerance of 10^{-4}.

Another possibility for real-time applications is to use a receding time horizon approach. At each time step, the optimization problem can be solved while only considering the vehicles currently inside the coordination region, taking their position and speed at the beginning of the time step as initial conditions. Vehicles inside the coordination region are tasked to adjust their control to comply with the optimal trajectory. This process can then be repeated at the next time step. In this approach, priorities assigned at a given time step can still be modified in the following steps, which should have a very limited impact on optimality. The green curve in Figure 5.7 shows the average computation time needed to converge (with a 10% gap tolerance and 1 s time steps), as a function of the number of vehicles simultaneously present inside the coordination region. Using this approach, up to 24 simultaneous vehicles can be treated in less than a second, which becomes reasonable for real-world applications in medium levels of traffic.

5.6 Chapter conclusion

In this chapter, we used the priority-based decision-making framework presented in Chapter 4 to design a mixed-integer linear programming framework for time optimal coordination of robots along fixed paths. To this end, we introduced a set of additional variables to formulate a linear cost function which allows minimizing the average *completion time*, *i.e.* the time needed for each robot to reach its goal without colliding with another robot. This technique can find application in purely robotics systems such as automated warehouses, but is also of importance for cooperative autonomous vehicles as it allows optimal management of an intersection without traffic lights.

Computer simulations show that, despite a prohibitive theoretical worst-case complexity, our formulation is actually solved quite efficiently by state-of-the-art MILP solvers, and can be used to compute solutions for problems containing up to 10 robots in real time on a standard computer. This number can be further increased to roughly 25 robots simultaneously by allowing a bounded suboptimality of up to 10%.

Note that the solver only has knowledge of the problem through a set of linear inequalities, and is of course not aware of the underlying geometrical properties presented in Chapter 4. Therefore, its performance can only originate from good sparsity and structural properties of the constraint matrix, and the powerful presolving heuristics leveraged by the solver. Although it is difficult to evaluate the exact contribution of both factors, we argue that our structuration of the coordination problem as a decision-making one is an important contributor to the success of this algorithm.

Chapter 6

Supervised semi-autonomy

"An error doesn't become a mistake until you refuse to correct it."

Orlando A. Battista (Chemist)

Contents

6.1 Supervised driving 49
6.2 Supervision problem 51
6.2.1 Modeling 51
6.2.2 Problem statement 54
6.3 Infinite horizon formulation 55
6.3.1 MIQP model 56
6.3.2 Objective function 56
6.3.3 Receding horizon properties 57
6.3.4 Note on multiple paths choices 58
6.4 An equivalent finite horizon problem 58
6.5 Simulation Results 61
6.5.1 Simulation environment 61
6.5.2 Test scenarios 62
6.6 Chapter conclusion 65

6.1 Supervised driving

In the previous chapter, we considered the (optimal) coordination of robots, with applications to fully automated vehicles in the framework of autonomous intersection management. However, the adoption of these technologies will probably be relatively slow, as the median age of the car fleet is roughly 11 years in the US [82]. Therefore, non- and semi-automated vehicles will likely coexist with fully automated systems for years, if not decades.

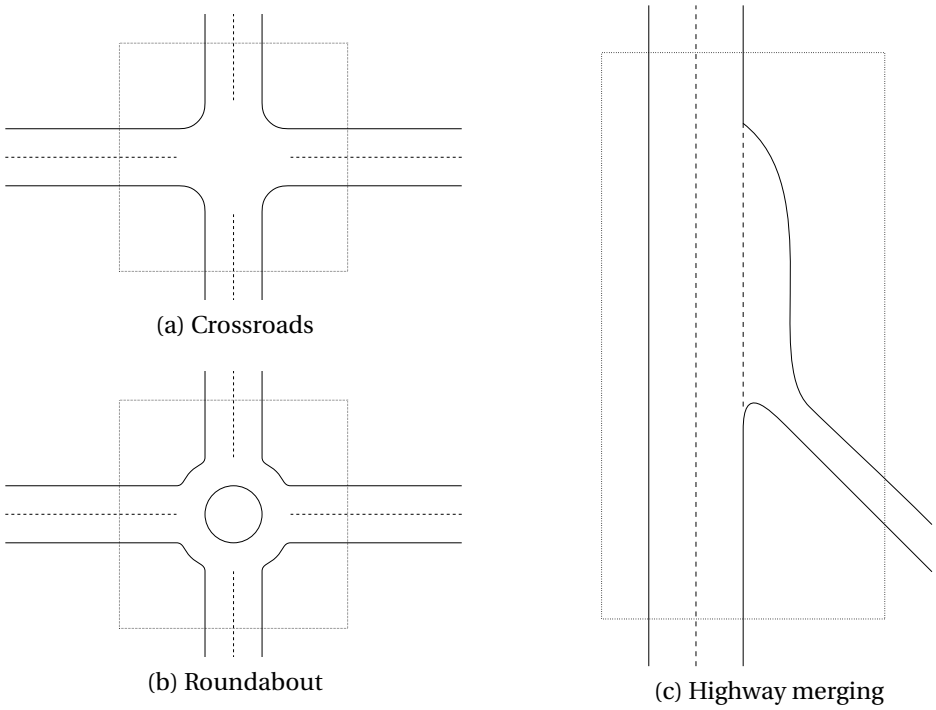
In this chapter, we consider a method to ensure the safety of semi-autonomous vehicles traveling on conflicting paths, for instance crossing an intersection or entering a

highway, while remaining compatible with the presence of human drivers. To this end, and inspired by earlier work in [83, 84], we propose a so-called *Supervisor* which monitors control inputs from each vehicle's driver, and is able to override these controls when they would result in an unsafe situation. More specifically, the role of the supervisor is twofold: first, knowing the current states of the vehicles, the supervisor should determine if the controls requested by the drivers would lead the vehicles into unsafe *inevitable collision states* [85]. In this case, the second task of the supervisor is to compute safe controls – maintaining the vehicles in safe states – which are as close as possible to those actually requested by the drivers. We say that such a control is *minimally deviating*.

The existing literature on semi-autonomous driving assistance is relatively scarce, possibly because the presence of human drivers brings a lot of additional complexity. The goal of a semi-autonomous driving assistant is to help the driver avoid collisions, either by notifying of a potential danger [86] or by taking over vehicle control in dangerous situations. To be accepted by human drivers, such systems should be as unobtrusive as possible and, in particular, should only intervene when necessary. Most of the currently existing literature on semi-autonomous driving mainly focuses on highway driving [87, 88, 89], which presents a relatively low difficulty as vehicle trajectories remain mostly parallel. This chapter aims at bringing semi-autonomy one step further, in order to allow cooperative driving between semi-autonomous vehicles in more complex conflict situations.

Some of these more complex problems have already been studied in the literature. In [90], the authors consider *semi-autonomous* driving at an intersection and propose that human drivers let an automated system control their vehicle while crossing said intersection. However, this scheme is somewhat intrusive as drivers completely relinquish control for a time, and handing back controls to a potentially distracted driver poses problems by itself. Reference [83] introduced the idea of a supervisory instance (called *supervisor*) tasked with preventing the system of vehicles from entering undesirable states by overriding the controls of one or several vehicles. In this more human-friendly approach, overriding only occurs when necessary, *i.e.* if an absence of intervention would result in a crash.

Interestingly, determining if an intervention is necessary is equivalent to finding a feasible solution to the multi-robot coordination problem presented in Chapter 4 and is, as such, NP-hard [91]. The existing literature on supervised driving overwhelmingly rely on the task-scheduling formulation of the coordination problem, and suffer from the usual downsides of such methods. In [83], the proposed supervisor is only suitable for simple intersection geometries with a single conflict point; moreover, no additional property is required from the safe controls used for overriding, which can widely deviate from the desired ones. Reference [92] leverages job-shop scheduling to develop a supervisor that considers several possible conflict points inside the intersection; however, vehicle dynamics are only modeled as first-order integrators; thus, vehicles are allowed to have infinite acceleration capacities which is not realistic in a real-world setting. Reference [84] proposed a Pareto-optimal supervisor leading to a minimally deviating formulation by recursively finding the most constrained vehicle, reserving its optimal crossing time, and scheduling the crossing of the remaining vehicles using the previous schedule as constraints. This method allows to minimize the deviation between the overridden and desired controls, but may be computationally intensive as all possible orderings of the vehicles have to be considered. In the rest of this chapter, we demonstrate that the priority-based decision-making framework developed in Chapter 4 can be used for supervised driving with good performance. Moreover, we provide useful theoretical results regarding guarantees provided by receding-horizon approaches for infinite-horizon safety.



(a) Crossroads

(b) Roundabout

(c) Highway merging

Figure 6.1 – Examples of considered road configurations, and corresponding supervision areas (interior of the dotted rectangles).

6.2 Supervision problem

We consider the problem of safely coordinating multiple semi-autonomous vehicles on the road, in order to prevent collisions and deadlock situations where no vehicle can move forward. Since vehicles are human-driven, a form of outside supervision is necessary to prevent undesirable situations. This section presents our formulation of a so-called Supervision problem generalizing the work of [83]; solving this problem yields a provably safe control, as close as possible to the original intentions of the drivers.

6.2.1 Modeling

6.2.1.1 Supervision area

We consider an isolated portion of a road infrastructure used by semi-autonomous vehicles, where some form of coordination is required to ensure vehicles safety. For instance, this could be a classical road intersection, a roundabout or an entry or weaving lane on a highway. We call this bounded portion of infrastructure the *supervision area* and we assume that vehicles can travel safely outside of this area using only their adaptive cruise control (ACC) capacities. In a real-world setting, different critical portions of infrastructure which are far enough apart can be considered individually, but need to be treated jointly if traffic from one can influence another. Figure 6.1 shows examples of roads configurations and the corresponding possible choice for a supervision area.

In this chapter, we present an embodiment of a Supervisor working over a spatially static supervision area over time, which can be thought of as a dedicated computer on the infrastructure or in the cloud. Vehicles are assumed to establish a connection to the supervisor when they enter the supervision area (using, for instance, V2I communication), and maintain it until they exit this region.

6.2.1.2 A change of paradigm

Contrary to Chapter 5 where we require the time horizon of the problem to be large enough for all robots to exit the coordination region, the supervised driving paradigm prevents us from doing so as drivers could choose to remain stopped for an arbitrary amount of time, *e.g.* in the case of an obstacle on the road. Moreover, we assumed in Chapter 5 that all robots involved in the coordination problem were known in advance; this is no longer the case in this chapter, where we consider continuous arrivals of new vehicles.

For this reason, the rest of this chapter uses a receding horizon approach where we only considered a subset of all the vehicles at any given time, and over a fixed time horizon, and we denote by $\mathscr{R}_t$ the set of vehicles that are located inside the supervision area at a time t. As we will demonstrate in Section 6.3.3, it is still possible to guarantee infinite horizon safety for *all* vehicles even though we only consider a (sufficiently long) finite horizon problem.

Another difficulty arising from this change of paradigm is that deadlock avoidance is no longer guaranteed by the design of the optimization framework, as this was the case in Chapter 5. Therefore, naive algorithms could result in situations where no vehicle is able to move forward, thus requiring additional constraints that are presented in Section 6.2.1.6.

6.2.1.3 Semi-autonomous vehicles

We consider *semi-autonomous* vehicles equipped with advanced driver assistance systems, many of which are already commercially available, and Vehicle to Infrastructure (V2I) communication capacities. In particular, vehicles are assumed to have advanced cruise control, automated braking and lane keeping assistance systems such that accelerating, braking and steering can be actuated by an onboard computer. Moreover, we suppose that vehicles have access to reliable cartographic data and are capable of precisely measuring their current position, orientation and velocity relative to a unique global frame, for instance using GNSS and inertial navigation.

Since the vehicles are not assumed to have advanced environment-sensing capacities (*e.g.*, LIDAR data), they are not able to handle all situations and still require a human driver to safely navigate, for instance in the case of on-road obstacles or loss of GNSS signal. Moreover, lateral collisions or deadlock situations can happen due to human error, justifying the need for supervision.

Moreover, we assume that systems such as lane-keeping assistance allow the vehicles to follow one of several predetermined reference paths with a small bounded lateral error. Therefore, we assume that the fixed-path hypothesis of Chapter 4 still holds and that collision-regions between paths can be computed in advance and inflated to account for the maximum lateral tracking error of the vehicles. We maintain the convention that $s_i = 0$ when the front bumper of i enters the supervision area, and we let $s_i^{out} > 0$ be the position at which the rear bumper of i exits the supervision area.

6.2.1.4 Vehicle dynamics

As in previous chapters, we mostly focus on the longitudinal dynamics of the vehicles and we use the second-order integrator model presented in Section 4.5, and we denote by $x_i = (s_i, v_i)$ the state of vehicle i. As before, we assume that $v_i \in [0, \overline{v}_i]$ (with $\overline{v}_i > 0$) at all times, and that the acceleration $u_i = \ddot{s}_i$ of each vehicle is bounded as $u_i \in [\underline{u}_i, \overline{u}_i]$, with $\underline{u}_i < 0 < \overline{u}_i$. These bounds can differ between vehicles, thus allowing heterogeneous

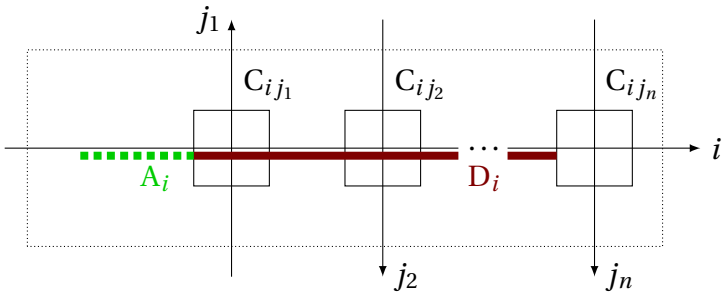


Figure 6.2 – Illustration of the no-stop region D_i and acceleration region A_i inside the supervision area (dotted rectangle).

vehicle performance. At a given time t, we let $\mathbf{U}_t = \prod_{i\in\mathscr{R}_t}[\underline{u}_i, \overline{u}_i]$ be the set of admissible accelerations for the vehicles of $\mathscr{R}_t$. We denote by boldface $\mathbf{x} = (x_i)_{i\in\mathscr{R}_t}$ and $\mathbf{u} = (u_i)_{i\in\mathscr{R}_t}$ the state and control for the system of vehicles.

In what follows, we let $v_{max} > 0$ be a global upper bound for $\overline{v}_i$, $u_a > 0$ a lower bound for $\overline{u}_i$ and $u_b < 0$ an upper bound for $\underline{u}_i$ such that for all $t \geq t_\kappa$ and all $i \in \mathscr{R}_t$, $\overline{v}_i \leq v_{max}$ and $\underline{u}_i \leq u_b < 0 < u_a \leq \overline{u}_i$. Therefore, all vehicles are capable of braking with u_b and accelerating with u_a; finally, we let u_{max} be a global upper bound for $\overline{u}_i$.

6.2.1.5 Collision regions

In this chapter, we consider the possibility of collision regions having multiple connected components, as illustrated in Figure 4.3 (page 30). In this case, we denote by C_{ij}^p its p-th connected component, with the convention $C_{ij}^p = C_{ji}^p$.

6.2.1.6 No-stop regions

To prevent creating deadlock situations, vehicles are not allowed to stop when doing so would block traffic in other directions. To this extent, we define a *no-stop region* D_i (illustrated in Figure 6.2) for each vehicle $i \in \mathscr{R}_t$ as the smallest interval $D_i = \left[\underline{s}_i^\perp, \overline{s}_i^\perp\right]$ containing all $\min\left(\Pi_{s_i} C_{ij}^p\right)$ for all $t' \geq t$, $j \in \mathscr{R}_{t'}$ and all p such that $(0,0) \notin C_{ij}^p$; in this formula, Π_{s_i} is the projection operator on the first coordinate. The no-stop region corresponds to the part of the supervision area where a vehicle may have to yield to another; if C_{ij}^p contains $(0,0)$, then either i or j enters the supervision area behind the other, in which case the relative ordering of the vehicles is given and the C_{ij}^p does not count in D_i.

Note that, although this definition theoretically requires knowledge of all future vehicles, D_i can be computed off-line as a finite intersection of intervals provided that there only exists a finite number of possible paths inside the supervision area. In what follows, we let $v_{min} > 0$ be a minimum allowed speed for any vehicle inside its no-stop region, and we assume that $v_{min} \leq \overline{v}_i$ for all vehicles.

For a no-stop region D_i, we define the corresponding *acceleration region* $A_i = \left[s_i^{acc}, \underline{s}_i^\perp\right]$ such that, if vehicle i is stopped at s_i^{acc}, it can reach a speed v_{min} before reaching $\underline{s}_i^\perp$. More specifically, we require that $0 \leq s_i^{acc} \leq \underline{s}_i^\perp - \frac{{v_{min}}^2}{2u_a}$ for all i. Inside the acceleration region, vehicles are only allowed to accelerate; this condition prevents vehicles from stopping right before the entrance of the no-stop region, leaving them unable to proceed forward due to the minimum speed requirement. Figure 6.2 illustrates an example of the no-stop regions and the corresponding acceleration regions.

6.2.1.7 Time discretization

Drivers continuously change the control input of their vehicle; however, due to computational and communication constraints, it is impractical to handle functions of a continuous variable. For this reason, we choose a constant time step duration $\tau > 0$, and we assume that all vehicles use piecewise-constant controls with step τ, typically 0.5 s. To simplify the formulation, we further assume that vehicles update their control simultaneously at times $t_\kappa = \kappa\tau$ for $\kappa \in \mathbb{N}$, and we denote by $\mathscr{U}_\tau(t_\kappa)$ the set of piecewise-constant admissible controls for the vehicles of $\mathscr{R}_{t_\kappa}$. By definition, for all $t \geq t_\kappa$ and all $\mathbf{u} \in \mathscr{U}_\tau(t_\kappa)$, $\mathbf{u}(t) \in \mathbf{U}_{t_\kappa}$.

6.2.2 Problem statement

Before presenting the so-called *supervision problem*, we first define the safety criterion for the vehicles inside the supervision area at a given time.

Definition 3 (Safe state)**.** *We say that the supervision area is in a safe state* $\mathbf{x}^\kappa$ *at time* t_κ *if there exists an admissible piecewise-constant control* $\mathbf{u} \in \mathscr{U}_\tau(t_\kappa)$ *defined over* $[t_\kappa, +\infty)$ *such that, under this control and starting from* $\mathbf{x}^\kappa$*, for all* $t \geq t_\kappa$ *and all* $i, j \in \mathscr{R}_{t_\kappa}$, $\big(s_i(t), s_j(t)\big) \notin C_{ij}$*. Such a control is said to be a safe control.*

With this definition, the supervision area is in a safe state when all the vehicles inside this region can apply a dynamically admissible, infinite horizon control without risk of collision. This safety condition corresponds to a contraposition of the notion of "inevitable collision state" proposed by Fraichard et al. [85]. In what follows, we denote by $\mathscr{U}_\tau^{safe}(t_\kappa)$ the set of safe and dynamically admissible piecewise-constant controls for the vehicles in $\mathscr{R}_{t_\kappa}$; by definition, a control $\mathbf{u} \in \mathscr{U}_\tau^{safe}(t_\kappa)$ is a piecewise-constant function from $[t_\kappa, +\infty)$ to $\mathbf{U}_{t_\kappa}$. We now define the safety condition for vehicles entering the supervision area.

Definition 4 (Safe entry)**.** *Consider a safe state* $\mathbf{x}^\kappa$ *at time* t_κ *and let* $t_1 > t_\kappa$ *be the first time at which a new vehicle enters the supervision area. We say that the vehicles of* $\mathscr{R}_{t_1} \setminus \mathscr{R}_{t_\kappa}$ *safely enter the supervision area with a margin* τ *if any safe control* $\mathbf{u} \in \mathscr{U}_\tau^{safe}(t_\kappa)$*:*

- *keeps the system of the vehicles of* $\mathscr{R}_{t_1}$ *safe at time* $t_\kappa + \tau$ *and*
- *remains safe over* $[t_\kappa, +\infty)$ *for the vehicles of* $\mathscr{R}_{t_\kappa}$*,*

regardless of the control applied by the vehicles of $\mathscr{R}_{t_1} \setminus \mathscr{R}_{t_\kappa}$ *over* $[t_1, t_1 + \tau]$.

This definition ensures that a safe control computed for the vehicles of $\mathscr{R}_{t_\kappa}$ remains safe after new vehicles enter, *i.e.* the entry of new vehicles does not invalidate previously safe controls. Moreover, we assume that we can safely exclude vehicles departing the supervision area from the safety verification problem, *i.e.* that drivers can follow the previously departed vehicles safely even in the absence of supervision. We will show in Section 6.3.3 that these hypotheses allow discrete-time supervision with continuous vehicle arrival.

In the remainder of this chapter, we consider a centralized supervisor working in discrete time steps of duration τ, and we assume that new vehicles always enter safely with a margin τ. At the beginning of each time step κ, the supervisor receives information about the desired longitudinal control of each vehicle for the next time step, denoted by $u_{i,des}^\kappa$.

The collection of these desired controls for the vehicles of $\mathscr{R}_{t_\kappa}$ defines a constant desired system control $\mathbf{u}^\kappa_{des}$ defined over $[t_\kappa, t_\kappa + \tau)$.

This control may, or may not, lead the system of vehicles into an unsafe state. The supervisor is tasked with preventing the system from entering an unsafe state, by overriding the desired control if necessary. To remain compatible with human drivers, it is desirable that the supervisor has several properties, namely being *least restrictive* and *minimally deviating*. Letting $\mathscr{U}^{safe}_{\tau,\kappa}(t_\kappa)$ be the restriction of the functions of $\mathscr{U}^{safe}_{\tau}(t_\kappa)$ to $[t_\kappa, t_\kappa + \tau)$, we define the *least restrictive* supervision problem:

Definition 5 (Least restrictive supervision)**.** *Consider a safe state $\mathbf{x}^\kappa$ at time $t_\kappa = \kappa\tau$, a desired system control $\mathbf{u}^\kappa_{des}$ and assume that all new vehicles enter the supervision area safely with a margin τ. The least restrictive supervision problem (*SP*) is that of finding a control $\mathbf{u}^\kappa_{safe} \in \mathscr{U}^{safe}_{\tau,\kappa}(t_\kappa)$ such that $\mathbf{u}^\kappa_{safe} = \mathbf{u}^\kappa_{des}$ if $\mathbf{u}^\kappa_{des} \in \mathscr{U}^{safe}_{\tau,\kappa}(t_\kappa)$.*

Note that this definition corresponds to that of [83] in our generalized setting. Such a supervisor is *least restrictive* because overriding only occurs if the initially requested control would lead the vehicles in an unsafe state. However, it is also desirable that the control used for overriding is chosen close to the drivers' desired control. Extending the work in [84], we define the *minimally deviating* supervision problem as follows:

Definition 6 (Minimally deviating supervision)**.** *Consider a safe state $\mathbf{x}^\kappa$ at time $t_\kappa = \kappa\tau$, a desired system control $\mathbf{u}^\kappa_{des}$ and assume that all new vehicles enter the supervision area safely with a margin τ. The minimally deviating supervision problem (*SP**) is that of finding a constant control ${\mathbf{u}^*}^\kappa_{safe}$ such that:*

$$ {\mathbf{u}^*}^\kappa_{safe} = \underset{\mathbf{u} \in \mathscr{U}^{safe}_{\tau,\kappa}(t_\kappa)}{\arg\min} \; ||\mathbf{u}^\kappa - \mathbf{u}^\kappa_{des}|| \tag{6.1} $$

where $||\cdot||$ is a norm defined over $\mathbf{U}_{t_\kappa}$.

Note that, from this definition, any solution to SP* is a solution to SP.

This concept of minimally deviating supervision corresponds to a different fail-safety paradigm that could be found in, *e.g.*, rail transportation where all trains in an area should perform an emergency braking when an incident occurs. The reasoning behind definition 6 is that, to improve efficiency without sacrificing safety, intervention is only performed on vehicles which are actually at risk, and does not necessarily result in a full system stop. However, at the level of individual vehicles, the safe overriding control ${u^*}^\kappa_{safe}$ may differ greatly from the driver's input, *e.g.* braking instead of accelerating.

6.3 Infinite horizon formulation

In this section, we will show that the minimally deviating supervision problem can be solved using mixed-integer quadratic programming (MIQP) using the priority-based decision-making framework presented in Chapter 4.

The sketch of this section is as follows: in Section 6.3.1, we formulate an infinite horizon MIQP problem at the beginning of a time step κ (corresponding to time $t_\kappa = \kappa\tau$). Assuming the state $\mathbf{x}^\kappa$ is safe, we show in Section 6.3.2 that this formulation can be used to find a minimally deviating safe control for the vehicles in $\mathscr{R}_{t_\kappa}$. We will then show in Section 6.3.3 that, if the vehicles of $\mathscr{R}_{t_\kappa}$ follow the corresponding control, our formulation expressed at $t_{\kappa+1} = (\kappa+1)\tau$ remains feasible for the vehicles of $\mathscr{R}_{t_{\kappa+1}}$, provided that

all new vehicles enter safely with a margin τ. These properties ensure that our infinite horizon MIQP formulation can be solved in a receding horizon fashion, to ensure safety for all future vehicles.

6.3.1 MIQP model

The idea behind our MIQP approach stems from the observation that SP* corresponds to an "optimal" motion planning problem for the system of vehicles, where only the first time step is used to evaluate optimality. Therefore, it is possible to use the constraints[1] presented in Chapter 4, applied at all time steps $k \geq \kappa$ and for all vehicles of $\mathscr{R}_{t_\kappa}$, for most of the supervision problem. The additional constraints are described below.

6.3.1.1 Deadlock avoidance

As described in Section 6.2.1.6, we require all vehicles to maintain a minimum speed inside their no-stop region $\mathrm{D}_i = \left[\underline{s}_i^{\perp}, \overline{s}_i^{\perp}\right]$. This requirement is enforced by defining additional binary variables, for all $i \in \mathscr{R}_{t_\kappa}$ and all $k \geq \kappa$:

$$\zeta_i^{acc}(k) = \mathbb{1}_{[\underline{s}_i^{acc}, +\infty)}\left(s_i^k\right) \tag{6.2a}$$

$$\zeta_i^{in}(k) = \mathbb{1}_{[\underline{s}_i^{\perp}, +\infty)}\left(s_i^k\right) \tag{6.2b}$$

$$\zeta_i^{out}(k) = \mathbb{1}_{\left[\overline{s}_i^{\perp}, +\infty\right)}\left(s_i^k\right) \tag{6.2c}$$

$$\eta_i(k) = \mathbb{1}_{[v_{min} - u_a\tau, +\infty)}\left(v_i^k\right) \tag{6.2d}$$

and using the constraints:

$$\left(\zeta_i^{acc}(k) \wedge \neg\zeta_i^{in}(k) \wedge \neg\eta_i(k)\right) \Rightarrow v_i^{k+1} \geq v_i^k + u_a\tau, \tag{6.3a}$$

$$\left(\zeta_i^{in}(k) \wedge \neg\zeta_i^{out}(k)\right) \Rightarrow v_i^k \geq v_{min}. \tag{6.3b}$$

As long as the acceleration regions A_i are large enough, constraint (6.3a) prevents vehicles from remaining blocked due to the minimum speed requirement (6.3b). We will show in the next section that these conditions effectively prevent deadlocks for all future times.

6.3.1.2 Initial conditions

The supervision problem is used in a receding horizon fashion, and we consider that the state of each vehicle of $\mathscr{R}_{t_\kappa}$ at time t_κ is known before solving the problem. Therefore, we use the following initial condition for all $i \in \mathscr{R}_{t_\kappa}$:

$$\left(s_i^\kappa, v_i^\kappa\right) = \left(s_i(t_\kappa), v_i(t_\kappa)\right) \tag{6.4}$$

6.3.2 Objective function

Any piecewise-constant control verifying the constraints of Chapter 4 as well as constraints (6.3a), (6.3b) and (6.4) for all $k \geq \kappa$ is dynamically admissible and prevents collisions for all future times, and is therefore in $\mathscr{U}_{\tau,\kappa}^{safe}(t_\kappa)$. To remain compatible with human driving, we now formulate an objective function allowing to find a least restrictive

[1] To account for multiple connected components in C_{ij}, one variable π_{ij} and one set of variables ε_{ij} are, in this case, introduced for each connected component.

and minimally deviating control given a desired control $\mathbf{u}^{\kappa}_{des} = (u^{\kappa}_{i,des})_{i\in\mathscr{R}_{t_\kappa}}$. In what follows, we let $(w^{\kappa}_i)_{i\in\mathscr{R}_{t_\kappa}}$ be a set of strictly positive weights, $\mathbf{X}$ be the tuple of all the problem variables, and we define:

$$\Phi^{\kappa}(\mathbf{X}) = \sum_{i\in\mathscr{R}_{t_\kappa}} w^{\kappa}_i \left(u^{\kappa}_i - u^{\kappa}_{i,des}\right)^2. \tag{6.5}$$

Noting $\pi_{\mathbf{u}^\kappa}$ the projection operator such that $\pi_{\mathbf{u}^\kappa}(\mathbf{X}) = \mathbf{u}^{\kappa}$, we deduce the following theorem:

Theorem 1. *The solution of the optimization problem:*

$$\begin{aligned} &\underset{\mathbf{X}}{\textit{Minimize}} && \Phi^{\kappa}(\mathbf{X}) \\ &\textit{subject to} && \forall k \geq \kappa, \textit{constraints (4.6), (4.7), (4.9), (5.2), (6.3a), (6.3b) and (6.4)} \end{aligned} \tag{IH-SP}$$

provides a solution $\pi_{\mathbf{u}^\kappa}(\mathbf{X})$ *to the minimally deviating supervision problem* SP* *at time* t_κ*, for the norm associated with* $\Phi^{\kappa}\circ\pi_{\mathbf{u}^\kappa}$.

Note that the weighting terms w^{κ}_i allow distinguishing between different types of traffic participants, for instance to prioritize emergency services or high-occupancy vehicles. More complex cost functions can also be used, for instance to penalize a forced acceleration more than a forced braking.

6.3.3 Receding horizon properties

We now assume that there exists a solution to IH-SP at time t_κ, that the vehicles of $\mathscr{R}_{t_\kappa}$ follow this solution control over $[t_\kappa, t_\kappa + \tau]$, and that the vehicles of $\mathscr{R}_{t_{\kappa+1}}$ enter safely with a margin τ. From Definitions 3 and 4, we have the following theorem:

Theorem 2 (Recursive feasibility)**.** *Let* $\tau > 0$, $\kappa \geq 0$, $t_\kappa = \kappa\tau$ *and* $t_{\kappa+1} = t_\kappa + \tau$. *Assume that:*

- *there exists a solution to IH-SP at time* t_κ *for the vehicles of* $\mathscr{R}_{t_\kappa}$,
- *the vehicles of* $\mathscr{R}_{t_\kappa}$ *follow this solution control over* $[t_\kappa, t_{\kappa+1}]$,
- *the vehicles of* $\mathscr{R}_{t_{\kappa+1}} \setminus \mathscr{R}_{t_\kappa}$ *enter safely with a margin* τ.

Then there exists a solution to IH-SP at time $t_{\kappa+1}$ *for the vehicles of* $\mathscr{R}_{t_{\kappa+1}}$.

Proof. From definitions 3 and 4, and using theorem 1, we know that the first two hypotheses guarantee that the vehicles in $\mathscr{R}_{t_\kappa}$ are in a safe state at time $t_{\kappa+1}$. Moreover, the third hypothesis ensures that the vehicles in $\mathscr{R}_{t_{\kappa+1}}$ also are in a safe state at $t_{\kappa+1}$ regardless of the control applied by the vehicles of $\mathscr{R}_{t_{\kappa+1}} \setminus \mathscr{R}_{t_\kappa}$ up to time $t_{\kappa+1}$. By definition 3, there exists a feasible solution to IH-SP thus proving the theorem. □

We now state that the IH-SP formulation effectively prevents the apparition of deadlocks; the proof of this theorem can be found in Appendix C.1.1.

Theorem 3 (Deadlock avoidance)**.** *Let* $\kappa \geq 0$ *and assume that, for all* $\kappa \leq k \leq \kappa_0$*, the conditions of theorem 2 remain satisfied at time* t_k*. There exists a feasible solution of IH-SP at time* t_{κ_0} *in which all the vehicles in* $\mathscr{R}_{t_{\kappa_0}}$ *exit the supervision area in finite time.*

Note that theorem 3 only ensures that, at all times, there exists a solution where all the vehicles inside the supervision at this particular time eventually exit. However, there is no guarantee that such a solution will actually be selected, for instance if one driver wishes to stop although there is no other vehicle. There is also no fairness guarantee, *i.e.* it is possible that one vehicle is forced to remain stopped for an arbitrarily long time, for instance if there is very heavy traffic coming from another direction. Future developments will focus devising more complex objectives function to take traffic efficiency and fairness into account.

6.3.4 Note on multiple paths choices

The above formulation assumes that the path of each vehicle is known in advance. However, this may not be realistic in the context of semi-autonomous cars where drivers can decide to change paths, for instance to avoid an obstacle on the road or use another itinerary. Using additional variables to indicate the path to which a vehicle is assigned, our formulation can be extended to handle multiple possible paths for each vehicle. As this extension introduces important additional complexity in the notations, we only briefly discuss the case of a single connected component in all C_{ij}.

Assuming that vehicle i can follow R_i possible paths[2], we introduce the binary variable $\Gamma_i^r = 1$ when vehicle i is assigned to path r, and 0 otherwise, and we now constrain the subregion indicator ε depending on the value of the Γ variables:

$$\left(\Gamma_i^{r_i} \wedge \Gamma_j^{r_j}\right) \Rightarrow \varepsilon_{ij}^{\parallel}(k) = \mathbb{1}_{\left[s_{ij,r_i,r_j}^{\parallel},+\infty\right)}\left(s_i^k\right), \tag{6.6a}$$

$$\left(\Gamma_i^{r_i} \wedge \Gamma_j^{r_j}\right) \Rightarrow \varepsilon_{ij}^{\perp}(k) = \mathbb{1}_{\left[\overline{s}_{ij,r_i,r_j}^{\perp},+\infty\right)}\left(s_i^k\right), \tag{6.6b}$$

where parameters s_{ij,r_i,r_j} correspond to the collision region between i and j when i follows path r_i and j follows path r_j. Moreover, we require each vehicle to be assigned to exactly one path, *i.e.*

$$\sum_{r=1}^{R_i} \Gamma_i^r = 1. \tag{6.7}$$

This approach allows the solver to reassign vehicles to a different path to avoid possible collisions. Assuming that each vehicle has a desired path r_i^{des}, adding a term:

$$\sum_i w_i^{\Gamma}\left(1 - \Gamma_i^{r_i^{des}}\right)$$

to the cost function (with weight w_i^{Γ} sufficiently large) ensures that vehicles will be preferentially assigned to their desired path, except in emergency situations.

6.4 An equivalent finite horizon problem

In Section 6.3.3, we presented an infinite horizon formulation to solve the minimally deviating supervision problem. However, due to the infinite number of variables, this formulation is not suitable for practical resolution. In this section, we derive an equivalent

[2]The choice of possible paths is, of course, dependent on the state of vehicle i. Furthermore, we assume that all possible paths are equal up to the current position s_i^k, so that the curvilinear position of the vehicle is the same along each possible path.

finite horizon formulation that can be implemented and solved using standard numerical techniques.

In what follows, we let $K \geq 1$ and we denote by FH-SP$_K$ the restriction of IH-SP at time t_κ to the variables at steps k with $\kappa \leq k \leq \kappa + K$, and we only consider the constraints up to step $\kappa + K$. The objective function is unchanged. A solution to FH-SP$_K$ at time t_κ allows to compute a control preventing collisions up to time $t_\kappa + K\tau$; however, due to the dynamics of the vehicles, the state reached at $t_\kappa + K\tau$ may not be safe. Since FH-SP$_K$ only has a subset of the constraints of IH-SP, we can formulate the following proposition:

Proposition 1. *Let $K \geq 1$ and let $\mathbf{X}$ be a solution of IH-SP at step κ. The restriction of $\mathbf{X}$ to the first $K+1$ time steps is a feasible solution to FH-SP$_K$.*

Using the global bounds u_a, u_b, u_{max} and v_{max} defined in section 6.2.1.4, we will now prove a reciprocal implication to proposition 1: if K is chosen large enough, any solution of FH-SP$_K$ can be used to construct a solution of IH-SP.

The key idea of the proof lies in the choice of a planning horizon long enough to allow any vehicle to fully stop; vehicles can then remained stopped in a safe state for an infinite amount of time. The structure of the demonstration is as follows: lemma 1 gives a lower bound on the time horizon to allow a single isolated vehicle to stop using discrete dynamics, although with a potential risk of rear-end collisions from following vehicles. In proposition 2, we give a slightly higher bound on the time horizon ensuring that all vehicles in a line can safely stop without rear-end collisions. Finally, in proposition 3 we give a bound on K ensuring the recursive feasibility of FH-SP$_K$; this allows formulating theorem 4, stating the equivalence of FH-SP$_K$ and IH-SP. In this section, we only present sketches of proofs for each result; detailed demonstrations can be found in Appendix C.1.2.

Lemma 1. *At a time t_κ, consider a horizon $T = K\tau$ with $T \geq \frac{v_{max}}{|u_b|} + \tau$. Let $i \in \mathscr{R}_{t_\kappa}$ be a vehicle for which there exists a piecewise-constant control $(u_i^k)_{\kappa \leq k < \kappa+K}$ such that, for all $\kappa \leq k < \kappa + K$, $u_i^k \in [\underline{u}_i, \overline{u}_i]$, corresponding to a dynamically feasible trajectory $s_i(t)$ over $[t_\kappa, t_\kappa + T + \tau]$.*
There exists a discrete control $(\tilde{u}_i^k)_{\kappa \leq k \leq \kappa+K}$ such that for all $\kappa \leq k \leq \kappa + K$, $u_i^k \in [\underline{u}_i, \overline{u}_i]$ and $\tilde{u}_i^\kappa = u_i^\kappa$, and for which the corresponding dynamically feasible trajectory $t \mapsto \tilde{x}_i(t) = (\tilde{s}_i(t), \tilde{v}_i(t))$ verifies $\tilde{s}_i(t_\kappa + T + \tau) \leq s_i(t_1 + T)$ and $\tilde{v}_i = 0$ over $[t_\kappa + T, t_\kappa + T + \tau]$.

Sketch of proof. $\frac{v_{max}}{|u_b|}$ is an upper bound on the required time for any vehicle to stop by applying a control u_b, which by definition is dynamically feasible. The additional τ accounts for the fact that we require $\tilde{u}_i^\kappa = u_i^\kappa$ at the first time step. □

In the following proposition and noting $\lceil \cdot \rceil$ the ceiling function, we prove a bound ensuring that a line of vehicles can safely stop before the leader reaches its final computed position at the end of the time horizon, without risk of rear-end collisions:

Proposition 2. *At a time t_κ, suppose that p vehicles of $\mathscr{R}_{t_\kappa}$ (denoted by $1, \ldots, p$ from rear to front) are following one another. Consider a horizon*

$$T_{stop} = K_{stop}\tau \geq \frac{v_{max}}{|u_b|} + (p-1)\left(1 + \left\lceil \frac{u_{max}}{|u_b|} \right\rceil\right)\tau + \tau, \tag{6.8}$$

and assume that every vehicle $i \in \{1, \ldots, p\}$ has a safe discrete control $(u_i^k)_{\kappa \leq k < \kappa+K}$ such that, for all $\kappa \leq k < \kappa + K$, $u_i^k \in [\underline{u}_i, \overline{u}_i]$. We let $t \mapsto x_i(t)$ be the trajectory over $[t_\kappa, t_\kappa + T]$ for vehicle i under control (u_i^k).

For all $i \in \{1,\dots,p\}$, there exists a safe discrete control $(\hat{u}_i^k)_{\kappa \le k \le \kappa+K}$ such that for all $\kappa \le k \le \kappa+K$, $u_i^k \in [\underline{u}_i, \overline{u}_i]$, $\hat{u}_i^\kappa = u_i^\kappa$ and for which the corresponding dynamically feasible and safe trajectory $t \mapsto \hat{x}_i(t) = (\hat{s}_i(t), \hat{v}_i(t))$ verifies $\hat{s}_i(t_\kappa + T + \tau) \le s_i(t_1 + T)$ and $\hat{v}_i = 0$ over $[t_\kappa + T, t_\kappa + T + \tau]$.

Sketch of proof. The worst case that needs to be taken into account corresponds to a situation where the initial states of the vehicles require each of them to accelerate in order to avoid a rear-end collision from the vehicle behind. This rather extreme situation happens when a vehicle goes faster than the one it is following, and the two are too close to allow a safe deceleration. In this case, the rearmost vehicle can always brake with the control from lemma 1, until it decelerates below the speed of the vehicle in front of it. The second rearmost vehicle can then decelerate, then the third and up to the front-most vehicle. The term $\left(1 + \left\lceil \frac{u_{max}}{|u_b|} \right\rceil\right)\tau$ arises from the piecewise-constant control hypothesis, and vanishes as τ goes to 0. Note that the condition $K_{stop}\tau \ge \frac{v_{max}}{|u_b|} + \frac{v_{max}}{u_a} + \tau$ also provides the same guarantees, as the original safe control (u_i^k) cannot lead vehicles to go faster than v_{max}; depending on the value of p, this second bound might be more efficient. □

Remark 3. *The bound from proposition 2 depends on the number of vehicles in a line, and can become quite high when p is large. It can be proven that the condition*

$$K_{stop}\tau \ge \frac{v_{max}}{|u_b|} + \frac{v_{max}}{u_a} + \tau \tag{6.9}$$

also provides the same guarantees; depending on the value of p, this second bound might be more efficient.

We can now prove the recursive feasibility of FH-SP$_K$ for a large enough K, as follows:

Proposition 3. *Consider a time t_κ, and assume that at most p vehicles are following one another at all times $t \ge t_\kappa$. We set $d = \max_{t \ge t_\kappa, i \in \mathscr{R}_t} \left(\overline{s}_i^\perp - s_i^{acc}\right)$ and we let T_{stop} be the stopping horizon from proposition 2 for p vehicles; moreover, we define*

$$T_{rec} = K_{rec}\tau \ge T_{stop} + \frac{v_{min}}{u_a} + \frac{d}{v_{min}} + \tau. \tag{6.10}$$

We assume that all vehicles of $\mathscr{R}_t$ for all $t > t_\kappa$ enter safely with a margin τ.

Problem FH-SP$_{K_{rec}}$ is recursively feasible under the hypotheses of theorem 2, i.e. if there exists a solution to FH-SP$_{K_{rec}}$ at time t_κ for the vehicles of $\mathscr{R}_{t_\kappa}$, there exists a solution at $t_\kappa + \tau$ for the vehicles of $\mathscr{R}_{t_\kappa+\tau}$.

Sketch of proof. The idea between the choice of T_{rec} is to ensure that each vehicle can either stop safely before entering its acceleration region (without generating rear-end collisions), or has already planned to exit its no-stop region safely. Moreover, the safe entering hypothesis ensures that the entry of new vehicles does not invalidate previously safe solutions, which can therefore be extended. □

We obtain the equivalence between IH-SP and FH-SP$_K$:

Theorem 4. *Problems IH-SP and FH-SP$_K$ with $K\tau \ge T_{rec}$ are equivalent, i.e. an optimal solution to one is also an optimal solution to the other.*

Proof. Proposition 1 ensures that any optimal solution to IH-SP is a feasible solution of FH-SP$_K$. Proposition 3 shows that a solution to FH-SP$_K$ (with $K\tau \ge T_{rec}$) can be recursively extended to a solution of IH-SP; therefore, the optimal solution of FH-SP$_K$ is feasible for IH-SP. Using these two results, we deduce the stated theorem. □

An important corollary of theorems 1, 3 and 4 is that the control obtained by solving FH-SP$_K$ with K large enough is also a solution to the minimally deviating supervision problem, and ensures deadlock avoidance as well. Contrary to IH-SP, FH-SP$_K$ is relatively easy to solve with dedicated mixed-integer quadratic programming solvers, as will be demonstrated in the following section.

6.5 Simulation Results

6.5.1 Simulation environment

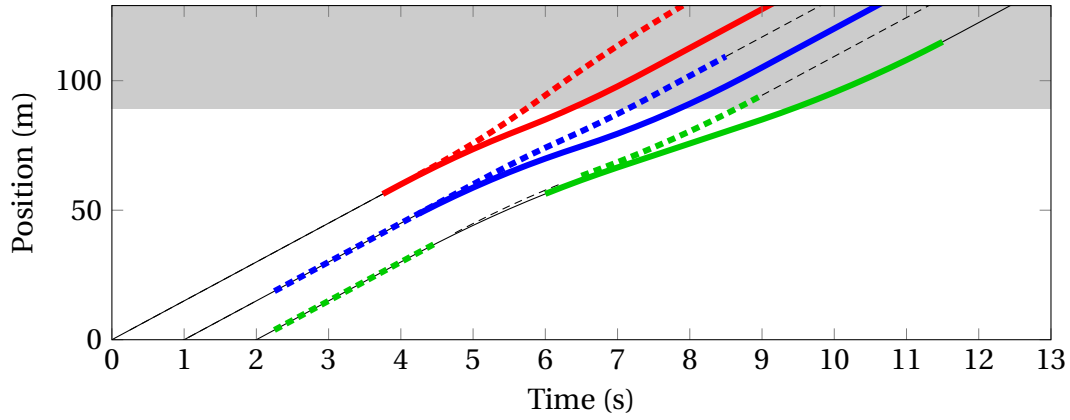
The presented Supervisor framework has been validated using extensive computer simulations on various test scenarios. In the absence of standardized test situations and since no open-sourced implementation of comparable methods [83, 84] is available, this section does not aim at a quantitative comparison with existing algorithms. Since our Supervisor is by design guaranteed to output an optimal[3] safe control, the major evaluation criterion is rather its ability to handle a wider variety of traffic scenarios than existing techniques, which is demonstrated in the rest of this section.

For implementation purposes, the resolution of the supervision problem is performed off-line and simulations are run in two successive phases. In the first phase, we define the geometry of the roads inside the supervision area and the corresponding possible paths, and compute the collision and acceleration regions information for each pair of paths. Since these sets only depend on the geometry of vehicles and paths, the corresponding parameters are computed off-line.

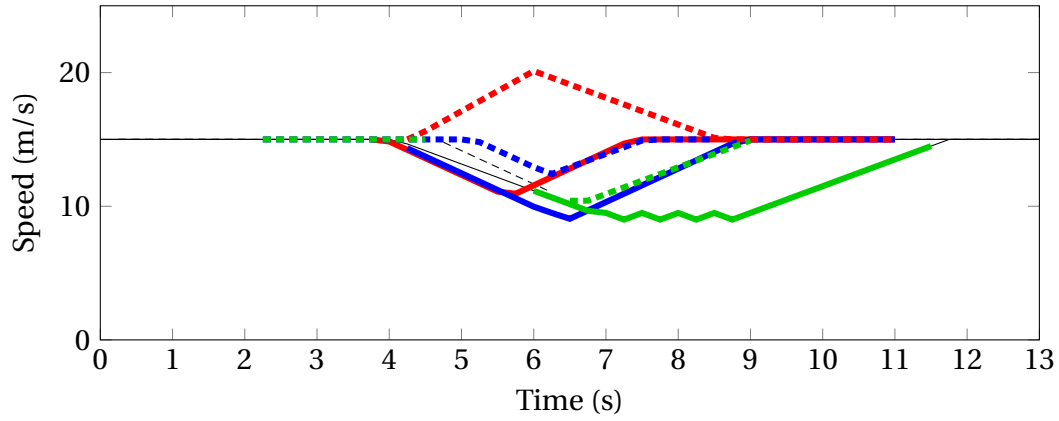
In the second phase, we run the simulation by coupling the high-fidelity traffic and physics simulator PreScan [93] with an external Python implementation of our supervisor. The actual resolution uses the commercial MIQP solver GUROBI [67]; the Python program runs a coarse simulation over a set time horizon with a fixed time step duration. Vehicles are generated using random Poisson arrivals, with a predefined arrival rate for each possible path, while respecting the safe entering condition; the initial velocity of each generated vehicle is chosen randomly according to a truncated Gaussian distribution. At each time step, the finite horizon supervision problem FH-SP is solved for the vehicles inside the supervision area, and yields the best safe control for the set of vehicles. The state of these vehicles at the next time step is then computed according to eq. (4.8).

In parallel, we use PreScan to validate the consistency of this output: from the safe controls computed in the Python supervisor and knowing the reference paths of the vehicles, we compute a target state comprising a desired position, heading and longitudinal velocity for each vehicle. This target state is fed into a low-level controller which outputs a steering and an acceleration or braking control. The vehicle model used in the validation phase takes into account engine response as well as chassis and suspensions dynamics, but does not model road-tire friction. PreScan's collision detection and visualization capacities are then used to validate the absence of collision or dangerous situations. Note that vehicles controllers are designed to ensure a bounded positioning error for any vehicle, relative to their prescribed path and velocity profile. This error is taken into account in the computation of the collision regions, so that the system is robust to control imperfections.

[3] Among the set of piecewise-constant controls with a given time step duration and in the sense of definition 6



(a) Longitudinal positions; the shaded area is the collision region.



(b) Longitudinal velocities

Figure 6.3 – Positions and velocities of the vehicles in the merging scenario; solid lines correspond to vehicles on the entry lane and dashed lines to vehicles starting on the highway. The thick colored portions show overriding intervals.

6.5.2 Test scenarios

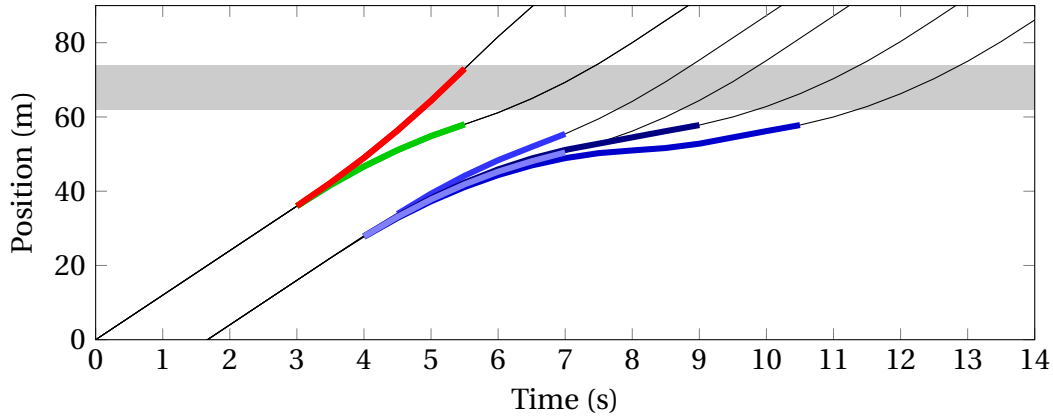
In the rest of this section, we consider three test scenarios – chosen to represent a wide variety of driving situations – consisting of merging on a highway, crossing an intersection or driving inside a roundabout. To showcase the performance of our framework in avoiding accidents and deadlocks, we assume that drivers are "oblivious" and focused on tracking a desired speed, regardless of the presence of other vehicles. A video of the presented simulations is available online[4].

6.5.2.1 Highway merging

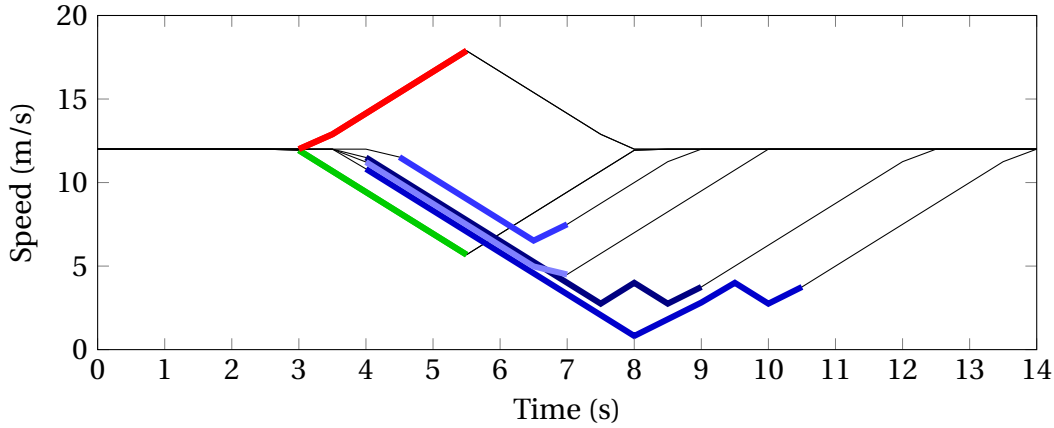
We first consider a very simple highway merging scenario, where an entry lane merges into a single-lane road; the possible paths for the vehicles are the same as in Figure 4.3c. The collision region between a vehicle i in the entry lane and a vehicle j on the highway have a single connected component given as $\underline{s}^{\perp}_{ij} = \underline{s}^{\perp}_{ji} = 89\,\text{m}$ and $s^{\parallel}_{ij} = s^{\parallel}_{ji} = 94\,\text{m}$, taking control errors into account.

To illustrate the action of the supervisor, we consider a set of six vehicles, three of which are on the highway and three on the entry lane. All vehicles are assumed to have "oblivious" drivers maintaining a constant speed, thus resulting in potential collisions.

[4] https://youtu.be/JJZKfHMUeCI



(a) Longitudinal positions; the shaded area is the collision region.



(b) Longitudinal velocities

Figure 6.4 – Positions and velocities of the vehicles in the intersection crossing scenario. The thick colored portions show overriding intervals.

This admittedly unrealistic behavior has been chosen to generate a higher probability of collisions in the absence of supervision. Figure 6.3 shows the longitudinal trajectories of the supervised vehicles; colored (thick) portions of the lines represent intervals of time during which overriding occurs. The area in gray corresponds to the collision region between entering vehicles and vehicles on the highway; thanks to the action of the supervisor, all collisions are successfully avoided.

6.5.2.2 Intersection crossing

The second scenario is the crossing of a + shaped intersection by a total of eight vehicles, with two vehicles per branch. In each branch, the front vehicle goes straight, and the rear vehicle turns left; moreover, all vehicles in front start at the same distance from the center of the intersection, and the same is true for the vehicles in the rear. This scenario illustrates the symmetry-breaking capacities of our framework, which handles this perfectly symmetrical scenario well, as shown in Figure 6.4. The area in gray corresponds to the collision region between vehicles on different branches. A video of a longer, one-hour simulation is also available online[5].

[5] https://youtu.be/cl32nbceZvw

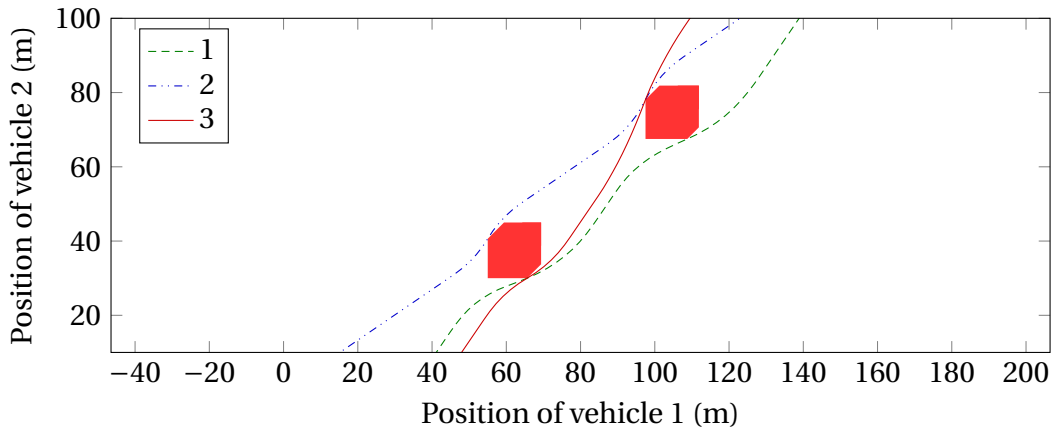


Figure 6.5 – Illustration of three possible classes of trajectories found by the solver, depending on the initial states of the vehicles. Trajectories 1 and 2 correspond to one vehicle passing the two collision points before the other. Trajectory 3 corresponds to the case where the vehicle on the inner lane enters after the other, and overtakes it inside the roundabout.

6.5.2.3 Roundabout driving

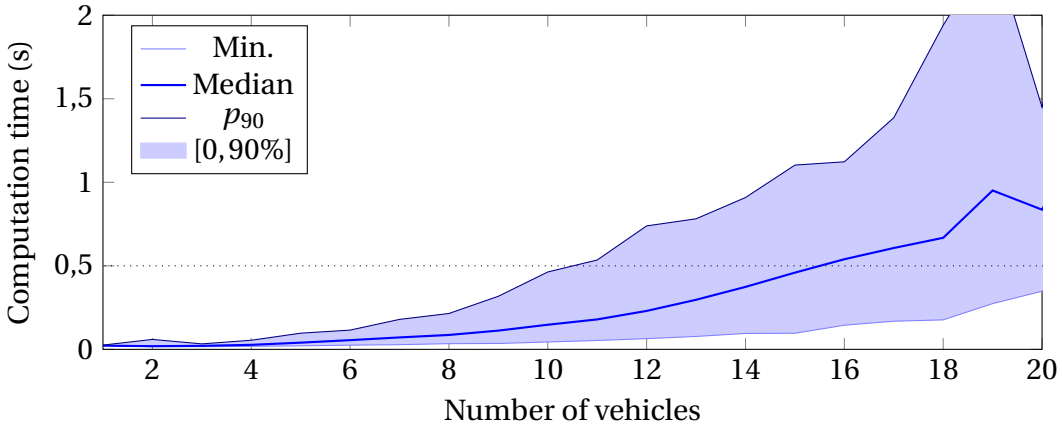
Finally, the third scenario consists of vehicles driving inside a two-lane roundabout. The particularity of this situation is that collision regions can have multiple connected components, for instance for the paths shown in Figure 4.3b. Since our formulation explicitly distinguishes each of these connected components, the supervisor is able to choose an ordering for each point of conflict, as illustrated in Figure 6.5: depending on the initial states and control targets of the vehicles, a different class of solution is chosen. A video of a longer, one-hour simulation is also available online[6].

6.5.2.4 Computation time

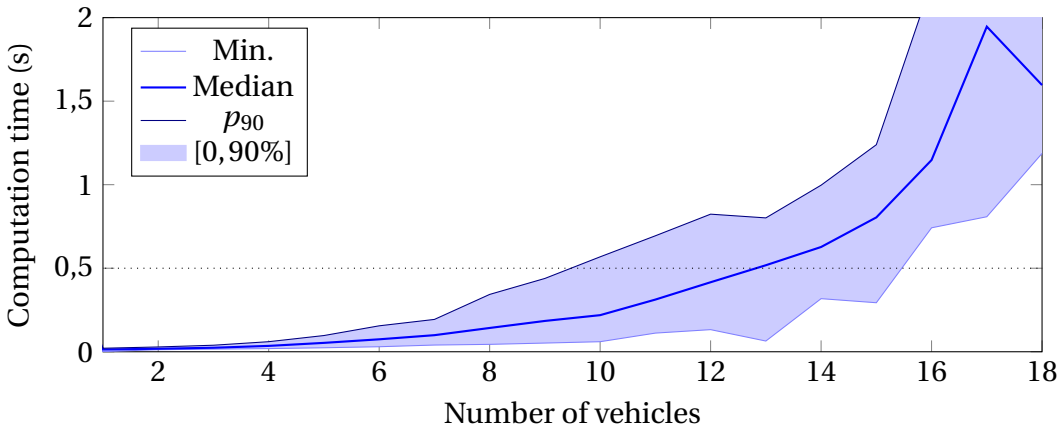
Due to the relatively short time horizon needed to ascertain infinite horizon safety, computation time remains reasonable despite the NP-hardness of the MIQP formulation. Figure 6.6 shows the evolution of the computation time in the intersection crossing and roundabout scenarios; the limited available space in the merging scenario does not allow enough vehicles for a similar diagram. These measurements have been obtained on a computer equipped with an Intel Core i7-6700K CPU clocked at 4 GHz with 16 GB of RAM, using the GUROBI solver in version 7.0. It can be seen that computation time remains below the duration of a time step in 90% of cases for up to approximately ten simultaneous vehicles, thus allowing real-time computation at 2 Hz.

Note that the difficulty of the MIQP problem only loosely depends on the geometry of the paths. However, it is highly dependent on the average number of conflicts per vehicle, which is higher in the case of roundabout driving, thus explaining the longer times reported in Figure 6.6b. Moreover, the implemented algorithm has been devised for readability over efficiency, and can be optimized by removing redundant variables to further reduce computation time. In practice, this refresh rate means that vehicles could apply a new acceleration every 0.5 s, which is faster than the typical reaction time of one second for a human driver, and should therefore be barely perceived. Note that for practical implementation purposes, the input of the supervisor should be predicted states at the end of the computation period instead of current states; since the acceleration of each vehicle

[6] https://youtu.be/pLoG32wFnkE



(a) Intersection scenario



(b) Roundabout scenario

Figure 6.6 – Distribution of computation time depending on the number of vehicles, for $\tau = 0.5$ s. Shaded areas represent the $[0,90\%]$ percentiles.

is assumed to be known to the supervisor, these predictions can be easily performed by forward integration.

6.6 Chapter conclusion

In this chapter, we demonstrated that a slight change to the objective function and constraints in our priority-based decision-making framework can lead to dramatically different applications, while maintaining the interesting properties of our formulation. We argue that the resulting Supervisor offers at least levels of safety comparable to the current state-of-the-art, and we demonstrated its capacity to handle a much wider range of scenarios due to its generic formulation. Although the real-world feasibility of such supervised driving schemes remains to be demonstrated, we provide some arguments in favor of a possible implementation in Appendix C.2.

From a theoretical standpoint, we demonstrated important results regarding the equivalence between safety verification on an infinite and a finite horizon, by providing a lower bound on the verification horizon guaranteeing the safety of the vehicles for all future times.

Conclusion of Part I

In the first part of this thesis, we leveraged previous theoretical results on multi-robot coordination to design a decision-based motion planning framework for a system of cooperative robots. Using the concept of *priority* as decision variables, we proposed a mixed-integer programming approach to encode the non-convex collision avoidance constraints as a set of piecewise-linear inequalities. We then demonstrated that this generic framework could be specialized by adding a few problem-specific constraints and designing a suitable cost function.

In a first example of specialization, we proposed an algorithm for time-optimal coordination of mobile robots, with applications to the field of autonomous intersection management. In the second example of supervised semi-autonomous driving, we presented a method to detect and seamlessly correct critical driver errors that, uncorrected, would lead to a collision. Moreover, we established useful theoretical results regarding the equivalence between an infinite-horizon formulation and a finite-horizon one, provided a lower bound on horizon is satisfied.

The results obtained in this first part demonstrate that proper modeling of the motion planning problem – using fundamental results on the structure of the search space to encode key decisions – helped achieve significant improvements compared to previous state-of-the-art methods. Fundamentally, priority relations are a means to decompose the (non-convex) search space into convex subregions. However, the performance of the solver on an *a priori* intractable formulation leads us to believe that this particular choice of decomposition better captures intrinsic properties of the problem, thus emphasizing the critical role of decision-making in the case of cooperative motion planning.

Part II

Motion planning for autonomous driving

Introduction

In the first part of this thesis, we demonstrated that incorporating an explicit representation of the discrete decisions involved in cooperative driving situations could help design high-performance coordination algorithms. Leveraging previous results on the structure of the solution space in multi-vehicle problems, we were able to represent these decisions in the form of binary "priority variables" in a quite versatile optimization framework. However, the underlying theory relies on the key assumption that vehicles follow fixed paths which are known in advance. This hypothesis, although reasonable in the case of fully automated, cooperative driving, is not suitable for *autonomous* vehicles having to navigate among other traffic participants which may be operated by humans.

In Part II, we focus on such autonomous driving situations where an automated vehicle (which we refer to as the ego-vehicle) has to plan a trajectory avoiding static and dynamic obstacles such as other traffic participants in a structured environment, *i.e.* along a road. Automated systems capable of driving in a single lane while safely following another vehicle are already commercially available, but these systems are generally incapable of making tactical decisions such as changing lanes, overtaking or entering an intersection. Some of these limitations stem from the difficulty of designing reliable perception and scene understanding systems, which are out of the scope of this thesis. Another challenge lies in efficiently handling the large number of discrete decisions that can be made even in simple driving scenarios.

By analogy with Part I, analyzing these decisions as a means to encode classes of collision-free trajectories seems a promising approach. Previous studies already established that the number of classes grew at least exponentially with the number of obstacles [29]. However, the existing literature does not provide a general framework to enumerate possible maneuver choices in a way which would be useful for motion planning. In Part II, we propose a representation of these decisions in the form of a *navigation graph*, which can then naturally be used to plan collision-free trajectories for the ego-vehicle.

Sketch of Part II This part is divided into five chapters. In Chapter 7, we introduce a fast, "decision-free" motion planning technique to drive a vehicle near its operational limits. The existence of such algorithms, although potentially interesting *per se*, demonstrates that trajectory planning is actually easy when decisions are already made, thus justifying our study of decision-based motion planning. In Chapter 8, we present generic considerations on the structure of the set of collision-free trajectories for a generic autonomous driving problem. In Chapter 9, we describe our graph-based approach to represent classes of such trajectories in generic driving situations, provided that the future behavior of all obstacles is known in advance. Finally, Chapter 10 describes a decision-based motion planning algorithm, which uses the navigation graph to explore possible decisions and select the one leading to the "optimal" trajectory.

Chapter 7

Decision-free, near-limits motion planning

" Have you ever noticed that anybody driving slower than you is an idiot, and anyone going faster than you is a maniac? "

George Carlin (Humorist)

Contents

7.1 Aggressive motion planning 73
7.2 A simple second-order integrator model 74
7.3 MPC formulation for trajectory planning 76
7.4 Simulation results 79
7.4.1 Planning without obstacles 80
7.4.2 Planning with obstacle avoidance 81
7.5 Chapter conclusion 83

7.1 Aggressive motion planning

Autonomous vehicles are widely expected to drive smoothly in order not to scare passengers and other road users, and to improve overall traffic safety; thus, automated vehicles should usually follow the road speed limits closely. However, such speed limits may not always exist (*e.g.*, on some dirt roads) or they may not be followed safely due to the road topology (*e.g.*, mountain road with sharp curves) or weather conditions (*e.g.*, icy road). Moreover, some emergency situations may require more aggressive maneuvers to avoid potential accidents. Therefore, the ability to plan *near-limits* safe trajectories, *i.e.* pushing the vehicle to its operational limits while ensuring collision avoidance, is of particular importance but is little studied in the literature.

Indeed, many trajectory planning algorithms rely on an *a-priori* knowledge of a target velocity, that can either be an explicit parameter of the problem [94, 55] or be implicitly given by requiring a set of target positions at fixed times [95, 96]. Some authors have proposed using the road curvature [97] to provide an upper bound on the velocity, which corresponds to a maximum lateral acceleration; this method can be extended to

obstacle avoidance by first planning a collision-free path, then adjusting the velocity consequently [98]. However, as path selection and velocity planning are intrinsically linked, this approach can lead to severe inefficiencies.

For this reason, model predictive control (MPC) techniques are often used in the literature, since they allow to simultaneously compute a feasible trajectory and a sequence of control inputs to track it. However, high-speed trajectory planning requires a complex modeling of the vehicle to account for its dynamic limitations, which mainly come from the complex and highly non-linear [99] tire-road interactions. Adding to this difficulty, wheel dynamics are generally much faster (around 1 ms [100]) than changes of the vehicle's macroscopic state (typically 100 ms). Therefore, the existing literature is generally divided between medium-term (a few seconds) trajectory planning including obstacle avoidance for low-speed applications, mainly relying on simple kinematic models (see, *e.g.*, [101, 102]), and short-term (sub-second) trajectory tracking for high-speed or low-adherence applications using wheel dynamics modeling (see, *e.g.*, [103, 104, 105, 106]). In the second case, obstacle avoidance is generally not considered (with the notable exception of [107]), and the existence of a feasible collision-free trajectory is not guaranteed in the case of an unexpected obstacle. Note that sampling-based approaches [108] have also been proposed in the literature; however, they do not provide optimality guarantees when a finite number of samples is selected, and may have trouble finding a feasible solution in complex scenarios.

In this chapter, we propose a middle-ground approach to allow trajectory planning over a few seconds in highly dynamic situations. This approach relies on the use a simple vehicle model derived from a realistic dynamic modeling of the vehicle body and wheels, which accounts for tire slip effects using carefully estimated bounds on the dynamics. Using this model, we design a non-linear trajectory planning framework, which is able to track a predefined path (*e.g.*, the road centerline) at high speed while avoiding obstacles modeled as semi-infinite regions. As presented in the next sections, this modeling hypothesis ensures that the planner does not have to make discrete decisions regarding which side each obstacle should be avoided from[1]; we thus describe the planning problem as *decision-free*. The high performance of this algorithm, which is able to run in a few milliseconds, illustrates the relative simplicity of decision-free motion planning, thus demonstrating the need to consider these discrete decisions in order to improve trajectory planning.

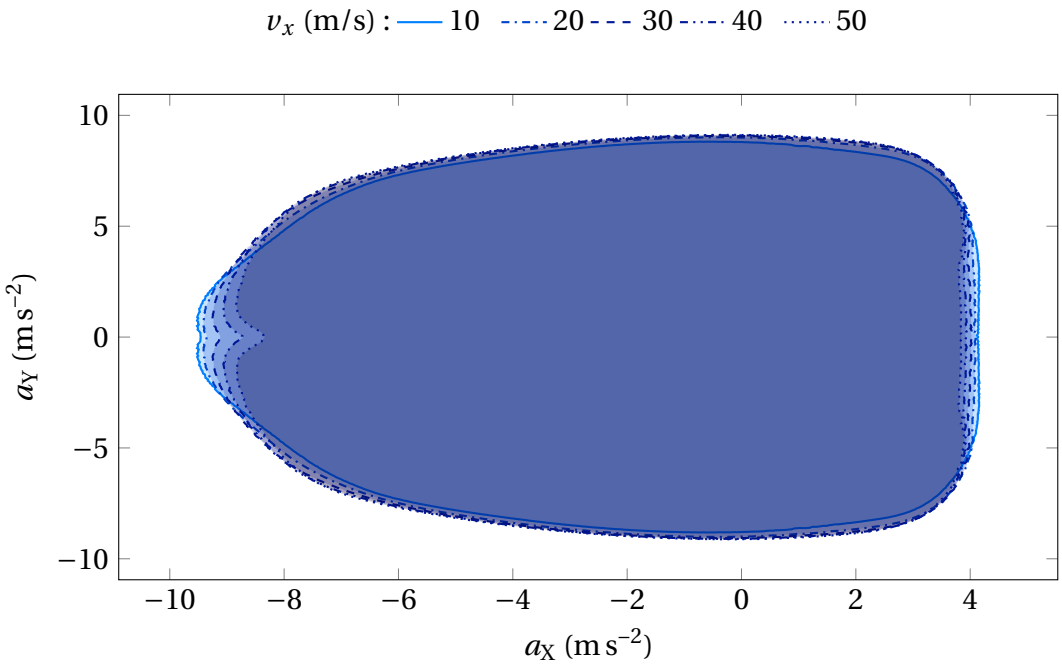
7.2 A simple second-order integrator model

As mentioned in the previous section, high-speed trajectory planning usually involves dynamic behaviors such as load transfer between wheels[2] as well as tire slip or skid, which are ignored in most planning models such as the second-order integrator presented in Section 4.5; Appendix D.2 presents a more realistic, 9-degree-of-freedom (abbreviated as 9DoF) vehicle model, taking such dynamics effects into account.

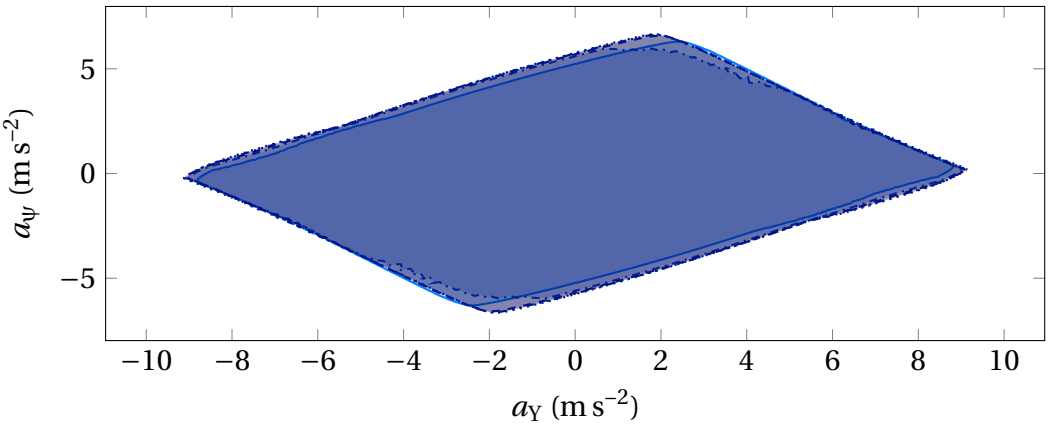
Theoretically, it is possible to use this 9DoF model inside a model predictive control framework to directly compute an optimal trajectory and the corresponding controls. However, standard optimization tools often struggle to solve problems involving highly

[1] Note that this hypothesis is not always sufficient; a short discussion about semi-infinite obstacles and local optima can be found in Appendix D.1

[2] For instance, the inertia of the car body creates a roll moment when turning, resulting in more apparent weight on the outer wheels. Similarly, a pitch moment is created when braking, causing more apparent weight on the front wheels.



(a) Lateral vs. longitudinal



(b) Yaw vs. lateral

Figure 7.1 – Envelope of the sets of feasible accelerations for different longitudinal velocities $v_{x,0}$ and lateral velocity $v_{y,0} \in [-0.2v_{x,0}, 0.2v_{x,0}]$; notice the slight deformation along the a_X axis with increasing $v_{x,0}$.

nonlinear constraints or cost functions, as it is the case in the model presented in Appendix D.2 – notably due to disjunction (D.4) which makes τ non-differentiable. Additionally, wheel dynamics generally occur over very small characteristic times – typically a few milliseconds – which requires choosing a correspondingly small discretization time step, making planning over long horizons impractical at best. For this reason, simplified models are very often preferred. Kinematic bicycle (or single-track) models [109, 101, 102], or even simpler second-order integrator models [55] are therefore common in the trajectory planning literature.

One of the main issues of these simplified models is that they are generally considered to be imprecise when nearing the handling limits of the vehicle. To counter this problem, we propose to use a constrained second-order dynamic model derived from simulation data using the 9DoF model of Appendix D.2.

These sets are obtained using a random sampling method on the 9DoF vehicle model:

we first compute (off-line) an envelope for the set of feasible longitudinal (a_X), lateral (a_Y) and angular (a_ψ) accelerations, as shown in Figure 7.1. Due to the shape of these envelopes, we propose a constrained double integrator model for the vehicle dynamics. This model considers a state vector $\mathbf{x} = [X, Y, \psi, v_x, v_y, v_\psi]^T$ and a control $\mathbf{u} = [u_x, u_y, u_\psi]^T$, with the same notations and reference frames as presented in Appendix D.2. The dynamic equation of the system is $\dot{\mathbf{x}} = f_{2di}(\mathbf{x}, \mathbf{u})$ with

$$f_{2di}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v_x \cos\psi - v_y \sin\psi \\ v_x \sin\psi + v_y \cos\psi \\ [v_\psi, u_x, u_y, u_\psi]^T \end{bmatrix}. \tag{7.1}$$

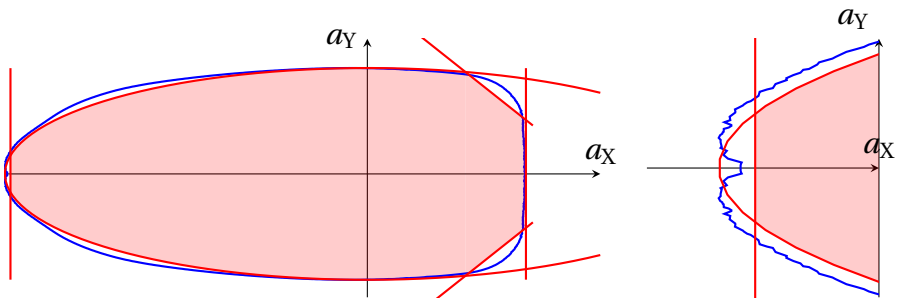
To allow the use of this model in planning, we approximate the sets shown in Figure 7.1 as a set of convex linear and nonlinear constraints in the (a_X, a_Y) plane as shown in Figure 7.2. These constraints are expressed as:

$$\left(\frac{a_X}{\alpha}\right)^2 + \left(\frac{a_Y}{\beta}\right)^2 \leq 1 \tag{7.2}$$

$$a_X^{min}(v_{x,0}) \leq a_X \leq a_X^{max}(v_{x,0}) \tag{7.3}$$

$$A[a_X, a_Y, a_\psi]^T \leq b \tag{7.4}$$

where A is a constant matrix, b a constant vector and a_X^{min}, a_Y^{min} depend on $v_{x,0}$. More detail on the derivation of this model can be found in Appendix D.3.



(a) Feasible set (blue) and approximation (red) (b) Detail

Figure 7.2 – Convex approximation of the feasible accelerations in the (a_X, a_Y) plane. The actual region is shown in blue.

7.3 MPC formulation for trajectory planning

We now use the second-order integrator model developed in the previous section to design a trajectory planner based on model predictive control. In this section, we assume that the vehicle tries to follow a known reference path, for instance the centerline of a given lane. The reference path γ_{ref} is supposed to be given as a set of positions (X_{ref}, Y_{ref}); as obstacle avoidance is part of the MPC formulation, these positions can simply be given through, *e.g.*, high-definition cartography.

In this chapter, we assume that the ego-vehicle has no information on a safe choice of longitudinal speed. Various possible scenarios can lead to this situation: in standard on-road driving, the legal speed limit might be unsafe due to high road curvature (*e.g.*, mountain roads) or low-adherence conditions; a speed limit might not even exist, for instance on certain private roads or for racing applications.

Furthermore, we assume that the vehicle needs to avoid obstacles on the road; we only consider fixed obstacles with known positions and shapes. In this chapter, we do not consider varying road adherence, and we suppose that the tire-road friction coefficient μ is constant and equal for all four wheels.

Algorithm 1: Planning and control

Data: current state $\mathbf{x}(t_0)$, γ_{ref}, horizon T
find $x_{closest}$:= point of γ_{ref} closest to $(X(t_0), Y(t_0))$
set s_0 := curvilinear position of $x_{closest}$
set $hz := [s_0, s_0 + v_x(t_0)T]$
set p_X := fitpolynom$(X_{ref}|_{hz}, s - s_0, 5)$
set p_Y := fitpolynom$(Y_{ref}|_{hz}, s - s_0, 5)$
find obs := list of relevant obstacles
find κ := max(abs(curvature(p_X, p_Y, hz)))
set $v_{max} := \min\left(v_x(t_0) + a_X(B)T, \sqrt{\mu g/\kappa}\right)$
compute $\mathbf{x}_{mpc}$:= MPC($\mathbf{x}(t_0)$, p_X, p_Y, v_{max}, T, obs)
apply low level control to track $\mathbf{x}_{mpc}$

The planning and control scheme works in several steps, as presented in Algorithm 1. We first approximate X_{ref} and Y_{ref} over the next planning horizon T as fifth order polynomials of the curvilinear position, starting at the point of γ_{ref} closest to the vehicle's current position. Using these polynomials, we compute the maximum (in absolute value) of the path curvature over the planning horizon, noted κ, to determine an upper bound $v_{max} = \sqrt{\mu g/\kappa}$ for the speed of the vehicle in order to limit the lateral acceleration to μg (as proposed, *e.g.*, in [97]). Only the relevant obstacles, *i.e.* those for which a risk of collision exists during the next planning horizon T, are effectively considered for collision avoidance; we note $\mathcal{O}$ the set of these obstacles.

In order to avoid creating local optima, we only consider *semi-infinite* obstacles. Fundamentally, this hypothesis amounts to requiring that the decision on which side to avoid each obstacle should be made before the trajectory planning phase. In practice, for each obstacle $o \in \mathcal{O}$ we determine a bounding parabola[3] as shown in Figure 7.3, such that the collision avoidance constraints can be written as $p_o(X,Y) \leq 0$, with p_o a second-order multinomial function. The side of avoidance can, for instance, be chosen heuristically based on whether the obstacle lies to the left or right of the reference path; however, the simplicity of this scheme can result in infeasibilities, for instance when two close-by obstacles are located on different sides of the reference path. Algorithms to choose the side of obstacle avoidance are discussed in the next chapters.

At a time t_0 corresponding to a vehicle state $\mathbf{x}(t_0)$ (with an initial longitudinal speed v_0), we formulate the motion planning problem using model predictive control with time

[3] More information on the choice of these parabolas can be found in Appendix D.1

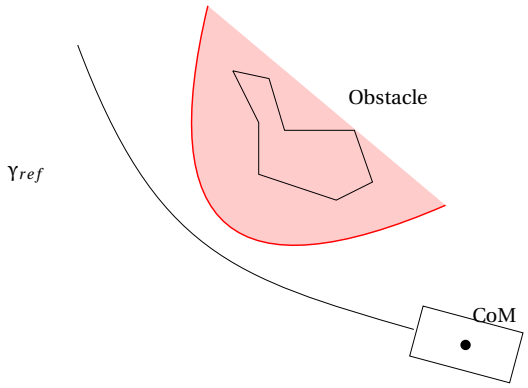


Figure 7.3 – Modeling of an obstacle as a parabola. The vertex and roots of the parabola are chosen with enough margin to ensure that no collision can occur as long as the center of mass is outside of the shaded region containing the obstacle.

step duration h and horizon $\mathrm{T} = \mathrm{K}h$ as follows:

$$\begin{aligned}
\min_{(\mathbf{u}_k)_{k=0\dots\mathrm{K}-1}} \quad & \mathrm{J}\big((\mathbf{x}_k)_{k=0\dots\mathrm{K}}, (\mathbf{u}_k)_{k=0\dots\mathrm{K}-1}\big) && \text{(7.5a)}\\
\text{subj. to} \quad & \mathbf{x}_{k+1} - \mathbf{x}_k = h\, f_{2di}(\mathbf{x}_k, \mathbf{u}_k) && \text{(7.5b)}\\
& \mathbf{x}_0 = \mathbf{x}(t_0) && \text{(7.5c)}\\
& \left(\frac{u_\mathrm{X}}{\alpha}\right)^2 + \left(\frac{u_\mathrm{Y}}{\beta}\right)^2 \leq 1 && \text{(7.5d)}\\
& a_\mathrm{X}^{min}(v_{x,0}) \leq a_\mathrm{X} \leq a_\mathrm{X}^{max}(v_{x,0}) && \text{(7.5e)}\\
& \mathrm{A}[u_\mathrm{X}, u_\mathrm{Y}]^\mathrm{T} \leq b && \text{(7.5f)}\\
& u_\psi = \gamma u_\mathrm{Y} && \text{(7.5g)}\\
& \forall o \in \mathcal{O},\ \ p_o(\mathrm{X}_k, \mathrm{Y}_k) \leq 0 && \text{(7.5h)}\\
\text{for} \quad & k = 0 \dots \mathrm{K}-1.
\end{aligned}$$

Note that, as it is often the case in the planning literature (see, *e.g.*, [87]), collision avoidance (eq. (7.5h)) is actually implemented as soft constraints to avoid infeasibility caused by numerical errors. Additionally, note that our formulation can be slightly modified in order to take moving obstacles into account, by using a different parabola p_o^k for each obstacle and at each time step.

In this chapter, we only focus on minimizing the deviation from the reference trajectory. In most of the existing MPC literature where a reference speed is supposed to be known in advance, the cost function J is expressed as:

$$\mathrm{J}\big((\mathbf{x}_k), (\mathbf{u}_k)\big) = \sum_{k=0}^{\mathrm{K}} \left(\mathrm{X}_k - \mathrm{X}_k^{ref}\right)^2 + \left(\mathrm{Y}_k - \mathrm{Y}_k^{ref}\right)^2. \tag{7.6}$$

In these formulations, X_k^{ref} and Y_k^{ref} implicitly encode the speed at which the reference path should be followed. Since we do not assume that a reference speed is known in advance, this method cannot be applied directly. A possible way to handle this difficulty is to express Y_k^{ref} as a function of X_k (see, *e.g*, [110]). However, this method cannot be applied to all shapes of reference paths, and is notably not suited to sharp turns even when using local instead of global coordinates. For this reason, we introduce an auxiliary state s to denote the curvilinear position of the vehicle along γ_{ref}, so that $\mathrm{X}_k^{ref} = p_\mathrm{X}(s_k)$ and $\mathrm{Y}_k^{ref} = p_\mathrm{Y}(s_k)$. In this chapter, we use a simple first-order integrator dynamic for s

with $\dot{s} = \sqrt{v_x^2 + v_y^2}$. Noting $\mathbf{x}'$ the extended state of the vehicle, we instead use the objective function:

$$\mathrm{J} = \sum_{k=0}^{\mathrm{K}-1} w_v v_{tol_k}^2 + w_\mathrm{X} \mathrm{X}_{tol_k}^2 + w_\mathrm{Y} \mathrm{Y}_{tol_k}^2 + w_o \mathrm{O}_{tol_k}^2 \tag{7.7}$$

where w_v, w_X, w_Y and w_o are positive weighting terms, and we add the following constraints to problem (7.5):

$$\begin{aligned}
v_{tol_k} &\geq |v_{max} - v_{\mathrm{X}k}| && \text{(7.8a)}\\
\mathrm{X}_{tol_k} &\geq |\mathrm{X}_k - p_\mathrm{X}(s_k)| && \text{(7.8b)}\\
\mathrm{Y}_{tol_k} &\geq |\mathrm{Y}_k - p_\mathrm{Y}(s_k)| && \text{(7.8c)}\\
\forall o \in \mathcal{O},\ \mathrm{O}_{tol_k} &\geq p_o(\mathrm{X}_k, \mathrm{Y}_k) && \text{(7.8d)}\\
&\text{for } k = 0 \ldots \mathrm{K}-1.
\end{aligned}$$

7.4 Simulation results

We used the realistic physics simulator PreScan [93] to validate the proposed MPC trajectory planner. The simulator uses the 9 degrees of freedom model presented in Appendix D.2, with the same parameters that were used to obtain the sets of feasible accelerations in Section 7.2. Robustness of the planner to variations of the vehicle parameters is a subject for further study.

The MPC problem (7.5)-(7.8) is solved using the ACADO Toolkit [111]; due to the inner workings of the simulator, the simulation is paused during resolution. The output of the solver is the set of future target longitudinal and lateral accelerations, as well as target future positions and longitudinal velocities for the vehicle in the horizon T. These outputs are fed into two low-level controllers, one being tasked with velocity tracking and the other with steering.

Low-level tracking of the planned trajectory is achieved using PID controllers; the lateral control also uses a τ seconds look-ahead [112]. At time t, the predicted position of the vehicle at $t+\tau$ is computed as $\hat{\mathrm{X}} = \mathrm{X} + \tau v_x \cos(\hat{\psi})$ and $\hat{\mathrm{Y}} = \mathrm{Y} + \tau v_x \cos(\hat{\psi})$, with $\hat{\psi} = \psi + \frac{1}{2}\tau\dot{\psi}$. Instead of tracking the target position at time t, the lateral control uses the error between predicted and desired positions at time $t+\tau$. Using a look-ahead $\tau = 0.2\,\mathrm{s}$, this method was found to provide better performance and stability than a simple PID. The lateral control takes into account a limited angular velocity for the steering wheel of $12\,\mathrm{rad\,s^{-1}}$, which is in the average of recorded steering velocities for human drivers in obstacle avoidance scenarios [113]; we do not consider the dynamics of the engine or brakes, which are supposed to respond instantaneously. Note that, although the PID approach gives an overall satisfying performance, we did observe certain rare occurrences of over-correction sending the vehicle into a spin; therefore, more robust control schemes accounting for the tire's friction circle should be explored.

The reference path used in our simulations is presented in Figure 7.4; the path consists of a 60 m straight line, a half circle with radius 20 m, a 200 m straight line, a Bezier arc corresponding to a 135 degrees turn, a 100 m straight line, a half circle with radius 10 m and a final -135 degrees turn.

In all simulations, the weights are chosen as $w_v = 1$, $w_\mathrm{X} = w_\mathrm{Y} = 10$ and $w_o = 100$, which were found experimentally to provide a good trade-off between speed and precision. The planning horizon is chosen as $\mathrm{T} = 3\,\mathrm{s}$ and the time step duration of the MPC is $h = 200\,\mathrm{ms}$; replanning is performed every 100 ms. To achieve real-time computation speeds, the solver is limited to five SQP iterations, which was experimentally shown to be

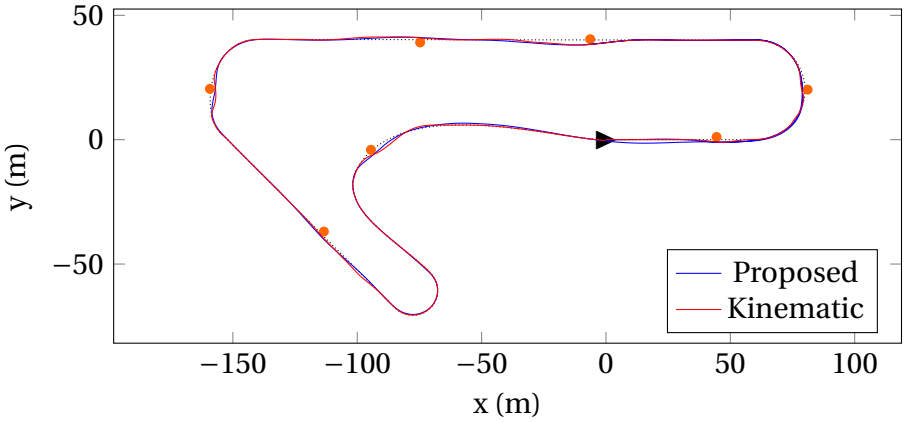


Figure 7.4 – Detail of the reference path (black, dotted) and actual path followed by the vehicle while avoiding obstacles (represented as orange circles), using both planners.

sufficient for a good convergence on our particular problem. Auto-generation techniques can also be used to further reduce computation time to roughly 10 ms [96].

For comparison purposes, we also implemented the same MPC planner with a classical kinematic bicycle model such as presented in [109]; the model is written as follows:

$$\dot{X} = v\cos\left(\psi + \arctan\beta\right) \tag{7.9a}$$
$$\dot{Y} = v\sin\left(\psi + \arctan\beta\right) \tag{7.9b}$$
$$\dot{\psi} = \frac{v}{l_r}\sin\left(\arctan\beta\right) \tag{7.9c}$$
$$\dot{v} = a \tag{7.9d}$$

with v the longitudinal velocity and $\beta = \frac{l_r}{l_r+l_f}\tan\delta$ the side slip angle. The control inputs are the longitudinal acceleration a, and the steering angle of the front wheel δ.

All other parts of the planning and control algorithm are otherwise equal, including the low-level controller. Moreover, this model is only used inside the MPC planner, while the simulation relies on that of Appendix D.2. The maximum and minimum acceleration in the bicycle model are chosen equal to a_X^{min} and a_X^{max} (see Figure 7.2 for the notations) respectively. Note that the solver is slightly faster using this formulation; therefore, the maximum number of SQP iterations is set to 6 for the kinematic model to yield comparable computation times, which increases solution quality.

7.4.1 Planning without obstacles

We first consider trajectory planning without obstacles; Figure 7.5 presents the actual speed of the vehicle during the simulation as a function of its position along the path γ_{ref} for the two MPC planners, as well as the speed bound $v^2 \leq \kappa g$ (with κ the path curvature), corresponding to a centripetal acceleration of $1\,g$ used, *e.g.*, in [97]. First, we notice that both planners have similar performance in the straight portions of the road; however, the planner based on the proposed second-order model systematically achieves higher speeds in curves. Second, no solver reaches the upper bound $\sqrt{g/\kappa}$, thus confirming that including a speed selection phase during planning (as opposed to tracking a predefined velocity solely computed from path curvature) is useful for proper tracking.

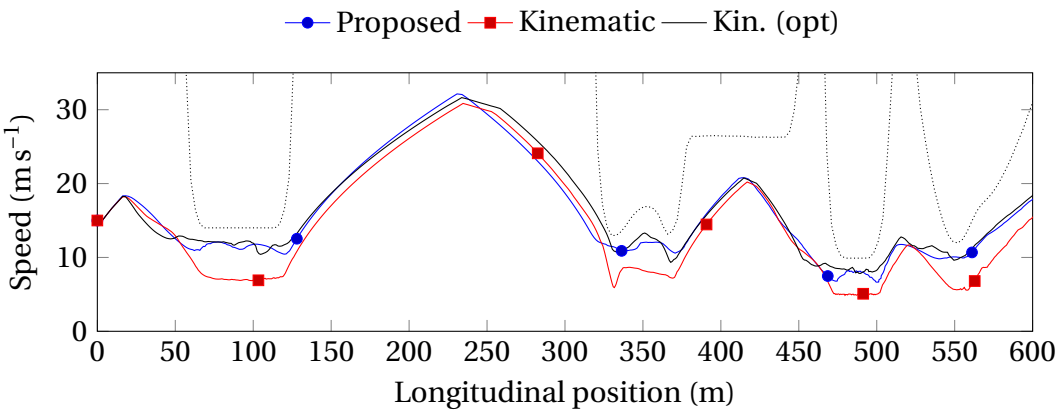


Figure 7.5 – Comparison of achieved vehicle speed with our proposed model (in blue) and for a kinematic bicycle model (in red); the speed for the kinematic model run until convergence is shown in solid black. The dotted curve represents the bound $v^2 \leq \mu g/\kappa$ where κ is the path curvature.

Table 7.1 – Absolute lateral positioning error for both planners.

Model	RMS error (m)	Maximum error (m)
Proposed	0.25	0.70
Kinematic	0.16	0.95

The superior performance of the planner based on the second-order integrator model is likely due to the simpler relation between the optimization variables (the input controls) and the objective value (the deviation from the target state) than in the bicycle model, which allows a much faster convergence towards the solution. Indeed, when allowing the solver to run until convergence with the kinematic bicycle model, the resulting velocity becomes comparable to that obtained with the real-time second-order model (but computation time is above 500 ms).

Figure 7.6 shows the lateral error when tracking the reference path, for both MPC planners. Table 7.1 presents synthetic data about the lateral error of the complete planning and control architecture in both cases, showing satisfying overall performance for high-speed applications. Note that better precision can be achieved (at the cost of speed) by selecting different values for the weighting coefficients. Moreover, a more precise low-level controller can probably achieve better performance.

7.4.2 Planning with obstacle avoidance

In this section, we compare the behavior of both planners in the presence of obstacles, modeled as parabolas as explained in Figure 7.3. Figure 7.4 shows a detail of the actual path followed by the vehicle using the proposed planner while avoiding obstacles. A video of the corresponding simulation can be found online[4].

Figure 7.8 presents the lateral deviation from the reference trajectory while avoiding obstacles using both planners. Generally speaking, the proposed planner allows a smaller deviation from the reference and a higher average speed of 12.7 m s^{-1} compared to 10.2 m s^{-1} using the kinematic bicycle planner (for lack of space, the velocity curves are not shown). More importantly, the kinematic planner is sometimes unable to output a

[4] https://youtu.be/BRpmdIxTz-0

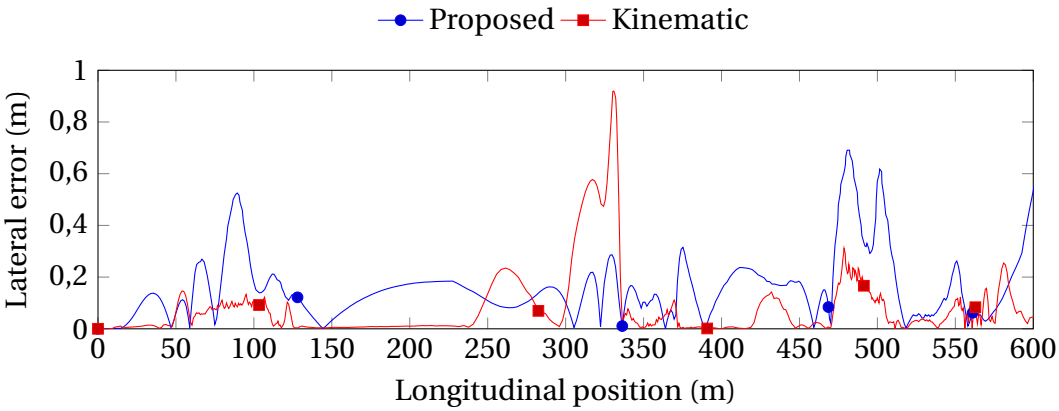


Figure 7.6 – Lateral positioning error for the proposed model (in blue) and for the kinematic bicycle model (in red), without obstacles.

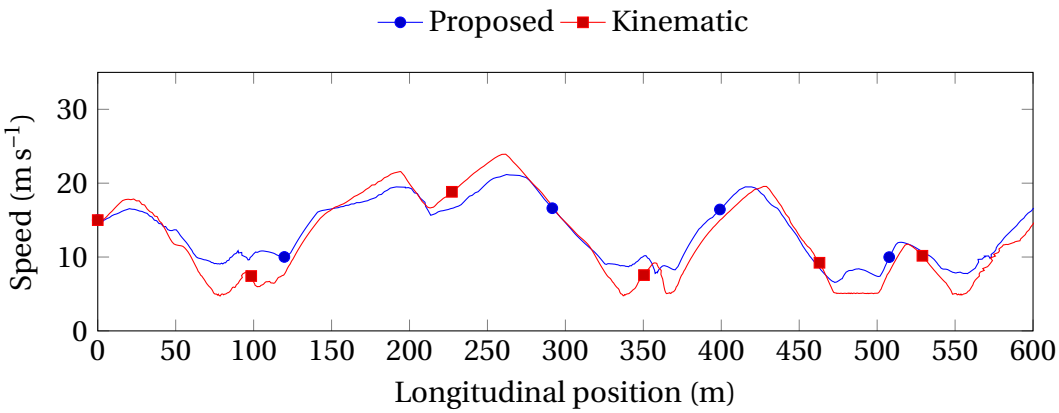


Figure 7.7 – Comparison of achieved vehicle speed with our proposed model (in blue) and for a kinematic bicycle model (in red).

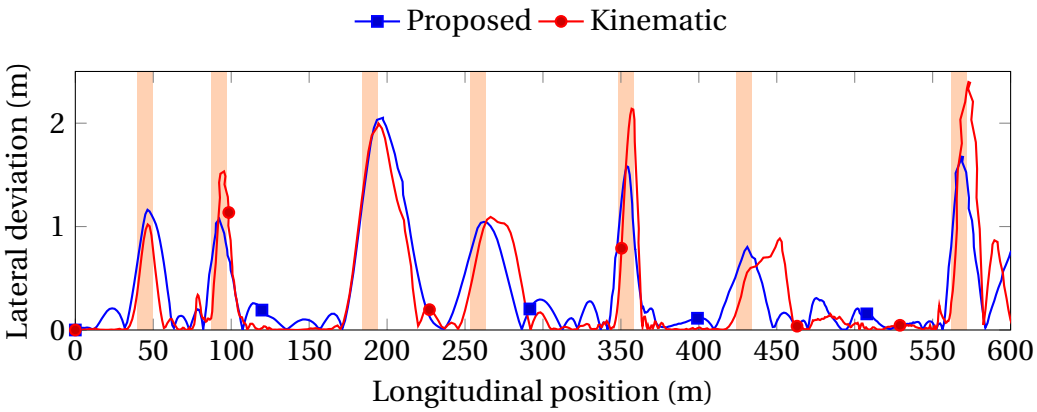


Figure 7.8 – Lateral deviation for the proposed model (in blue) and for the kinematic bicycle model (in red) while avoiding obstacles (shaded regions).

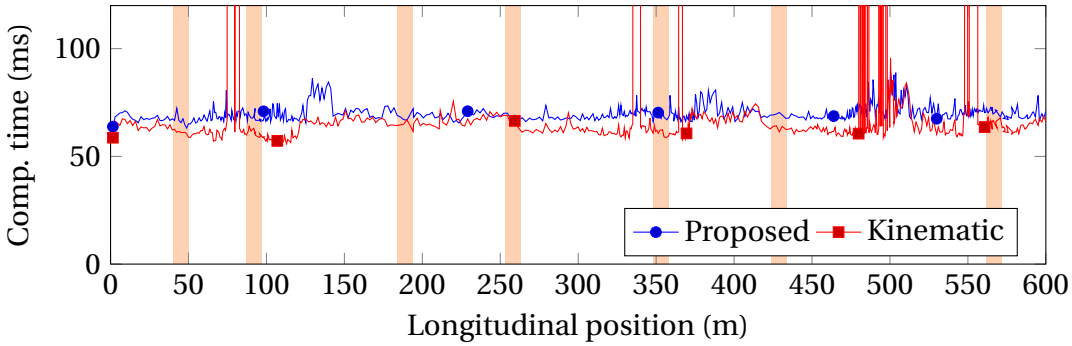


Figure 7.9 – MPC computation time (with obstacles) for the proposed model, and for a kinematic bicycle model starting from the same state.

trajectory in less than 100 ms, as shown in Figure 7.9. This situation happens when approaching obstacles in high-curvature portions of the road; as before, the better behavior of the proposed planner is likely due to the simpler search space since the dynamic model presents much less non-linearity. A more in-depth analysis could provide useful insights on desirable model properties to allow fast and robust convergence.

7.5 Chapter conclusion

In this chapter, we presented a model predictive control framework for near-limits motion planning in the absence of discrete decisions based on a simplified model for vehicle dynamics. Contrary to more traditional MPC formulations, this approach automatically adjusts the ego-vehicle speed to road curvature and obstacles in order to achieve the best possible trade-off between speed and accuracy. We argue that this capacity can prove useful in a variety of situations, notably when the speed limit is not known or not relevant, *e.g.* when driving on slippery or curvy roads, or for applications such as autonomous racing.

Simulation results show that the planner is able to perform in real-time, with computation time remaining below 80 ms; using advanced auto-generation techniques could help reduce this computation time to below 10 ms. This observation tends to demonstrate that motion planning in the absence of discrete decisions – even with speed selection – is, in fact, a relatively "simple" problem from a computational point of view thanks to sophisticated optimization algorithms such as sequential quadratic programming. These considerations, alongside with the results from Part I, motivate our work on decision-based motion planning for more general problems where obstacles are not considered semi-infinite.

Chapter 8

Classes of trajectories in autonomous driving

" To understand is to perceive patterns. "

Isaiah Berlin (Philosopher)

Contents

8.1 Motion planning for autonomous driving 85
8.2 State representation 86
8.3 Free space-time 87
8.4 Maneuver variants and homotopy classes 88
8.5 Chapter conclusion 90

8.1 Motion planning for autonomous driving

In the previous chapter, we showed that motion planning in the absence of discrete decisions was relatively easy to solve using state-of-the-art methods such as model predictive control. Before moving forward with the presentation of our decision-based motion planning algorithms, we propose to discuss some specificities of the trajectory planning problem applied to automated driving.

In some robotics applications in confined areas with mostly static obstacles, it can be conceivable to plan full trajectories from start to goal as described in Chapter 2. However, automated driving usually deals with trajectories that can span dozens or hundreds of kilometers, and where most of the obstacles are mobile and cannot be known in advance, thus making it impossible to plan full trajectories. For this reason, motion planning for on-road automated driving is usually considered in a receding horizon fashion, where the initial state of the vehicle is known but the goal region may only be loosely defined; the framework presented in Chapter 7 can even be considered not to have a goal region.

Control theory can be used to derive conditions guaranteeing the soundness of receding horizon schemes (see, *e.g.*, [39]), and theoretical computations can be made to determine the minimum horizon to be used to ensure recursive feasibility of these problems, as was done in Chapter 6. Empirically, the fact that human drivers – who generally

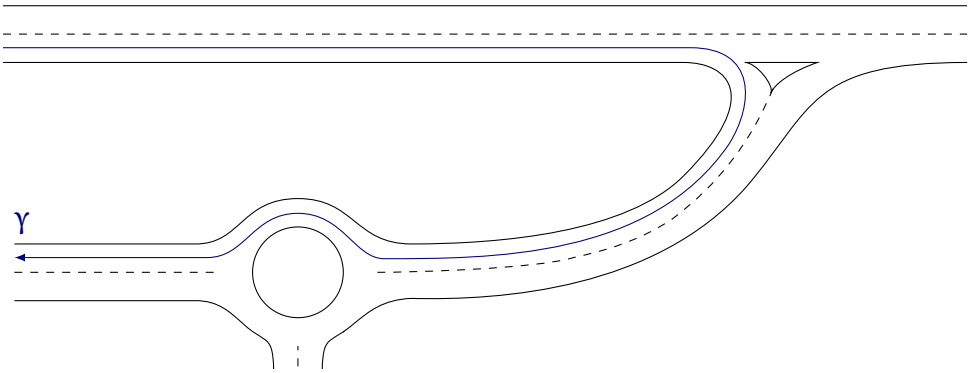


Figure 8.1 – Example of reference path in an on-road driving situation

do not plan more than a few seconds ahead – have a remarkably low rate of collisions also comforts the idea of using receding horizon schemes.

Therefore, the rest of this thesis will only consider motion planning problems with time horizons of 1 s to a few dozen seconds incorporating what is sometimes dubbed *tactical* decisions such as lane change or overtaking maneuvers, or entering an unsignalized intersection. At a higher level, a "mission planning" layer is tasked with providing rough guidance in the form of sub-goals, for instance a sequence of roads to follow in order to reach the destination. In this thesis, we do not consider the mission planning layer and only focus on tactical motion planning – which we abbreviate as "motion planning".

In the rest of this chapter, we present notations and considerations that will be used through the end of this thesis. Some results discussed here are not original to this thesis, and many ideas were already introduced in [29]. Apart from describing useful hypotheses and notations, the aim of this chapter is to demonstrate that classes of trajectories in generic driving situations are much more complex than what was previously assumed, and thus require a more systematic method to be enumerated properly; such an algorithm is presented in Chapter 9.

8.2 State representation

We consider that the input of motion planning is a reference path γ, which can for instance correspond to the center of the rightmost lane of the roads to be followed, as illustrated in Figure 8.1. Assuming that the reference path is sufficiently smooth and that its curvature radii are large enough compared to the local width of the road[1], we can use the Frenet-Serret coordinates (s, r) to represent any point $X = (x, y)$ of the road, where s is the curvilinear abscissa of the point X_γ of γ closest to X, and r is the (signed) distance from X to X_γ, as illustrated in Figure 8.2.

Using the Frenet coordinates allows moving the motion planning problem from a Cartesian geometry where the road is curved, to an abstract space where the road is rectilinear. In the rest of this thesis, we focus on on-road driving of an automated vehicle, where the ego-vehicle is assumed to remain almost parallel to the road. Therefore, we neglect the angle between the heading of the ego-vehicle and the reference path γ, so that we can use the Frenet coordinates (s, r) of one point of the vehicle to fully determine its configuration[2].

[1] See Appendix E.1 for more detail on those conditions.

[2] As described in the following paragraphs, this assumption allows casting the motion planning problem

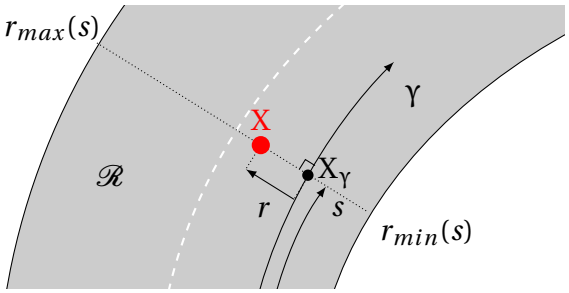


Figure 8.2 – Frenet representation of a point on the road

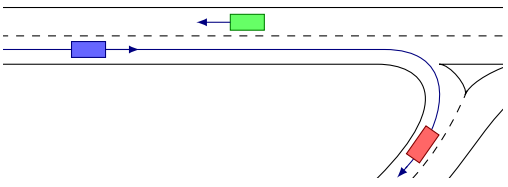
To account for the possibly varying curvature of the road, we denote by $r_{min}(s)$ and $r_{max}(s)$ the extent of the road, such that the vehicle is on the road if and only if $r_{min}(s) \leq r \leq r_{max}(s)$, and we let

$$\mathscr{R} = \left\{(s,r) \in \mathbb{R}^2 \middle| r_{min}(s) \leq r \leq r_{max}(s)\right\}$$

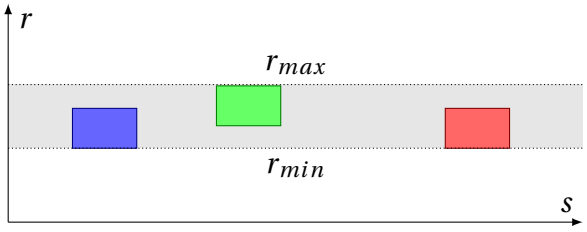
denote the drivable portion of the road. Note that $\mathscr{R}$ does not account for small static obstacles (*e.g.*, a cardboard box on the road), but can be used to transcribe that a lane is being blocked by roadworks.

8.3 Free space-time

We propose to represent the on-road motion planning problem using a configuration space (or C-space) approach as presented in Chapter 2. Contrary to Part I where all traffic participants were part of the problem, we now consider other traffic participants as mobile obstacles; in this chapter, we further assume that the future trajectory of all obstacles is known in advance. At any given time t, we denote by $\mathscr{C}_t$ the configuration space of the ego-vehicle expressed in curvilinear coordinates, and by $\mathscr{C}_t^f \subset \mathscr{R}$ its collision-free part, obtained from the knowledge of the configuration of all other obstacles at time t. Figure 8.3a presents an example situation with three obstacle vehicles with reference path in blue; Figure 8.3b shows the corresponding configuration space, with obstacles represented in the corresponding color.



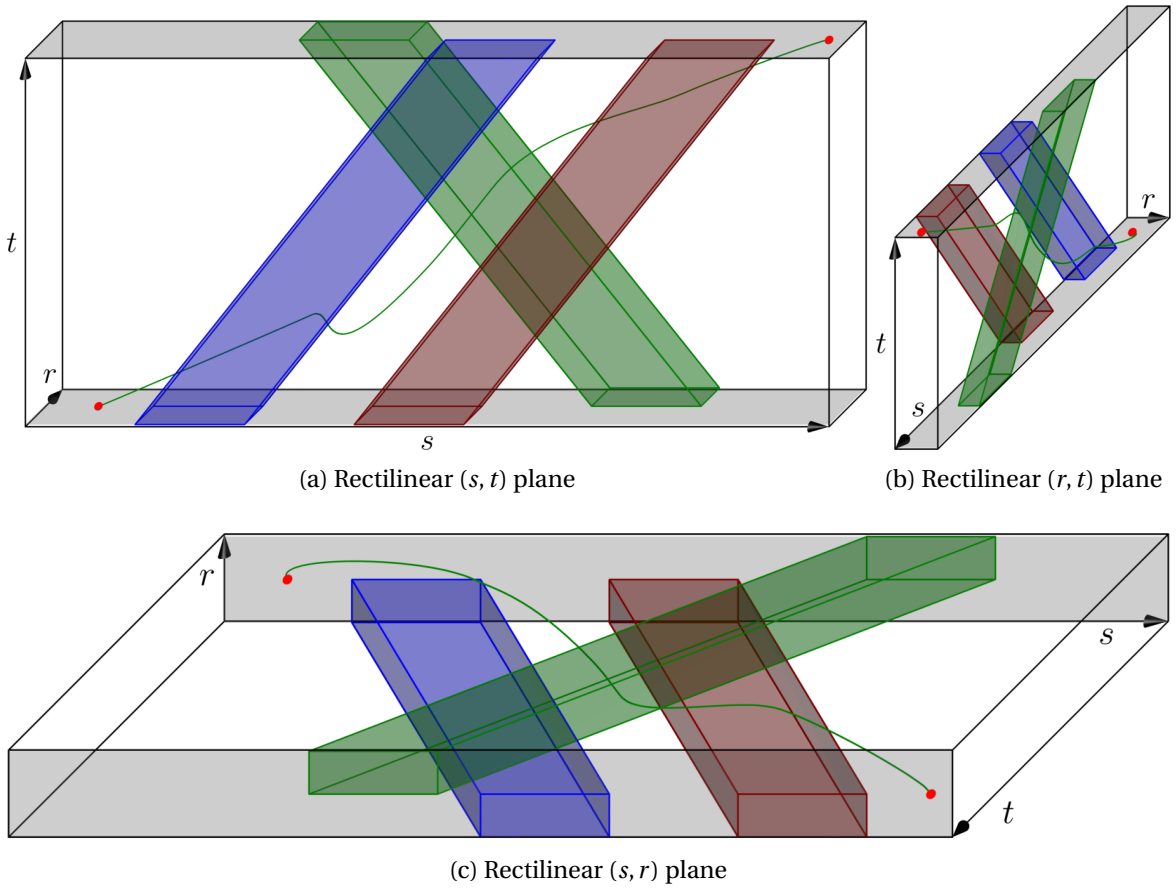
(a) Example situation with three obstacles (colored rectangles).



(b) Configuration space; the grey region represents the free C-space $\mathscr{C}_t^f$; note that obstacle regions are larger than the physical obstacles to account for the dimension of the ego-vehicle.

Figure 8.3 – Physical and configuration space representation of obstacles at a fixed time t.

in a 3D space, thus widely simplifying conceptualization. Moreover, we argue that this assumption is not particularly constraining in the case of on-road driving as vehicles usually keep a small angle with respect to the road direction; however, this may not be the case for some maneuvers such as parking.



(a) Rectilinear (s,t) plane

(b) Rectilinear (r,t) plane

(c) Rectilinear (s,r) plane

Figure 8.4 – Three views of the configuration space-time and an example trajectory for the scenario shown in Figure 8.3. The gray areas represent the extent of the road.

To account for future motion of the obstacles, it is necessary to also take the time dimension into account. To this extent, we define the configuration space-time as $\mathscr{S} = \bigcup_{t\in\mathbb{R}} (\mathscr{C}_t \times \{t\})$, and the free (configuration) space-time as

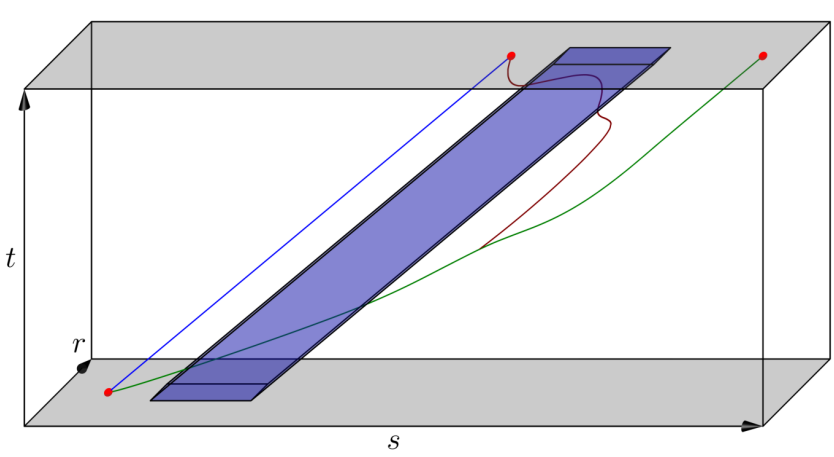
$$\mathscr{S}^f = \bigcup_{t\in\mathbb{R}} \left(\mathscr{C}_t^f \times \{t\}\right).$$

Figure 8.4 illustrates the configuration space in the case of Figure 8.3; the green curve represents a collision-free trajectory for an ego-vehicle (not shown in Figure 8.3a), corresponding to overtaking the blue vehicle from the left, move back to the right lane to let the green vehicle pass, then overtaking the red vehicle.

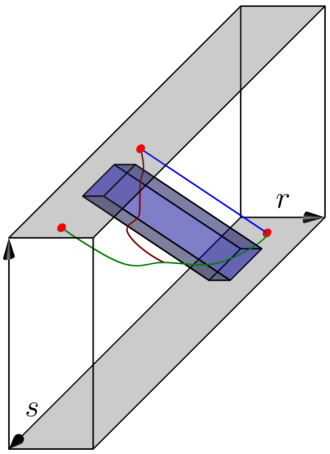
An interesting finding of [29] is that, in this particular scenario, an extended concept of homotopy classes of trajectories can be mapped to *maneuver variants* corresponding to passing vehicles in a particular order and from a certain side. However, the specificity of autonomous driving – notably the fuzzy nature of "maneuvers" and the necessity to only consider finite time horizons – make the classic homotopy theory too restrictive to adequately describe maneuver variants in more general situations, as we present in the next section.

8.4 Maneuver variants and homotopy classes

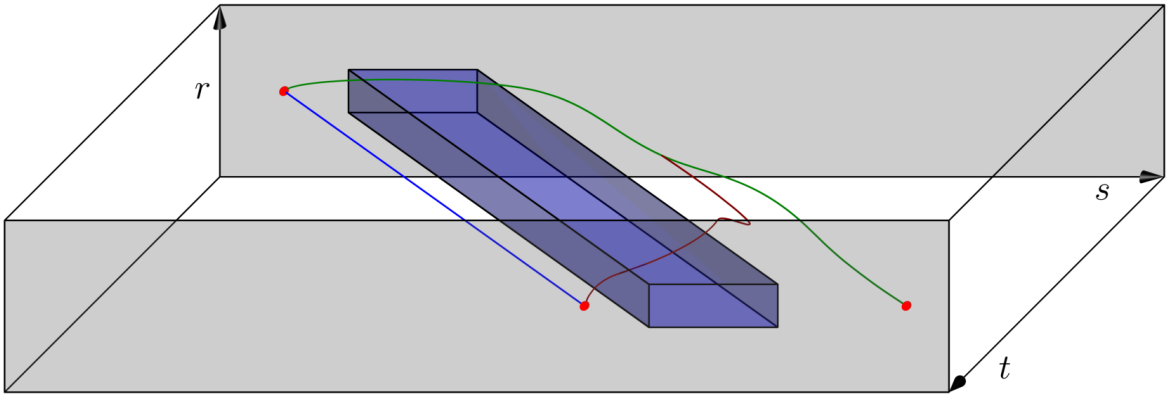
To illustrate these limitations, we propose a simpler scenario with a single obstacle vehicle driving in the middle of a three-lane road. The corresponding configuration space-



(a) Rectilinear (s, t) plane



(b) Rectilinear (r, t) plane



(c) Rectilinear (s, r) plane

Figure 8.5 – Three views of the configuration space-time and three example trajectories around a single obstacle. The gray areas represent the extent of the road.

time is shown in Figure 8.5, alongside with three examples of collision-free trajectories.

The blue trajectory in Figure 8.5 corresponds to the ego-vehicle remaining behind the obstacle over the horizon; the green one corresponds to overtaking the obstacle from the left. The red trajectory first matches the green one, then diverges and terminates at the same point than the blue one, illustrating an aborted overtaking maneuver.

According to the usual homotopy definition, the blue and green trajectories in Figure 8.5 clearly do not belong to the same homotopy class as they do not share the same endpoint. However, the blue and red trajectories do belong to the same class, as they can be continuously deformed into one another.

A first limitation of using homotopy classes to describe maneuver variants lies in the restrictiveness of the same endpoint condition, as a trajectory infinitely close to the blue one but terminating at a slightly different position would not be considered as homotopic, although they arguably correspond to the same maneuver of "remaining behind" the obstacle. To counter this issue, reference [29] introduced the notion of "homotopic relative to a start point" by only requiring the endpoints of the trajectories to be in the same predetermined set, as illustrated in Figure 8.6. However, the choice of this set has an impact on the resulting homotopy classes, and no results have been proposed on how this set should be chosen to reflect maneuver variants adequately.

A second limitation is tied to the fact that motion planning for autonomous driving is

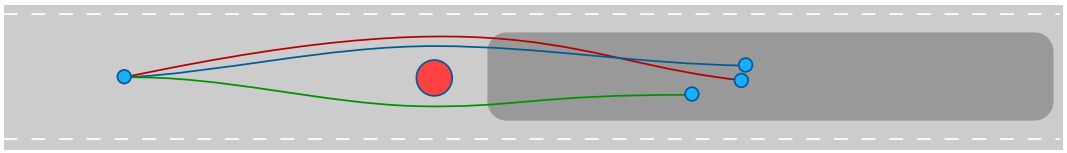


Figure 8.6 – Homotopy classes relative to a region: the blue and red paths are considered homotopic (from [29]).

usually done over a finite (receding) time horizon T. As illustrated by the red and green trajectories in Figure 8.5, the same overall trajectory can correspond to different homotopy classes (even by the extended definition of [29]) depending on the considered time horizon. Indeed, as the two trajectories are equal up to a time t_1, and denoting by t_0 the first instant shown in Figure 8.5, their restriction to $[t_0, t_1]$ are in the same homotopy class although their restriction to $[t_0, \mathrm{T}]$ are not.

From an autonomous motion planning perspective, we also argue that simply providing a homotopy class of trajectory may still not be enough information to properly guide resolution. For instance in the example of Figure 8.3, knowing that the ego-vehicle should overtake the blue vehicle before the green one passes does not help choose the time instant at which the overtaking maneuver should occur. Chapter 10 provides more detail on this particular subject.

8.5 Chapter conclusion

In this chapter, we introduced the specificities of motion planning for autonomous driving and we described a curvilinear state representation for the ego-vehicle which allows casting the planning problem into a rectilinear frame. Using this representation, we then proceeded to show a negative result regarding the use of homotopy classes as decision variables, as we successfully did in Part I. More specifically, although these classes do exist, we argue that they do not match the concept of “driving decisions” or “maneuver variant” well enough to be used in motion planning.

To address this limitation, the next chapter proposes a graph-based classification of trajectories that sacrifices some properties ensured by homotopy theory, but that is much more suitable to encode driving decisions for motion planning.

Chapter 9

Navigation graph

" I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail. "

Abraham H. Maslow (Philosopher)

Contents

9.1 Free space partitioning 91
9.2 A guiding example 92
9.3 Mathematical results 97
9.3.1 Modeling 97
9.3.2 Semantic free-space partitioning 98
9.3.3 Continuous navigation graph 100
9.3.4 Discrete-time partitioning 102
9.4 Navigation graph and homotopy classes 103
9.5 Chapter conclusion 103

9.1 Free space partitioning

In this chapter, we propose to extend previous results on "divide-and-conquer" strategies used in two-dimensional path planning to handle the challenges particular to on-road trajectory planning using a third temporal dimension. Inspired by the work of [43] where the free space is decomposed into convex polygons, we propose to partition the free *space-time* as convex polyhedrons.

Of course, there are infinitely many ways to perform this partition which are equivalent in theory. However, guided by the results in Part I and the considerations of Chapter 8, we argue that performing this partitioning in a semantically meaningful way is preferable than choosing arbitrary partitions. Simulation results presented in Chapter 10 seem to substantiate this claim.

In the case of on-road driving, traffic is usually organized in lanes and constraints on the longitudinal behavior of a vehicle widely differ from those on its lateral behavior. For this reason, we propose to make use of this anisotropy in the design of our semantic partitioning algorithm, as described below.

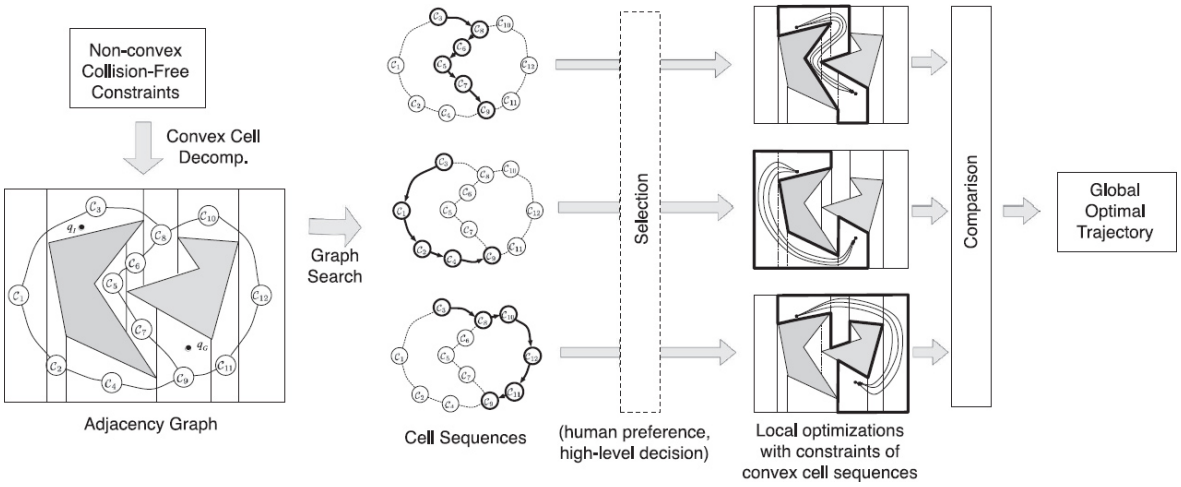


Figure 2.5 – Partition of the free space for 2D path planning, from [43] (repeated from page 14)

9.2 A guiding example

The goal of this section is to give an intuition of the main mathematical results, using the example scenario shown in Figure 9.2; the formal mathematical theory is developed in the next section. In our example, we consider an ego-vehicle navigating on a road with two other vehicles (obstacles); vehicles are modeled as rectangles driving parallel to the side of the road. Intuitively, the ego-vehicle has three classes of maneuvers to choose from: either it can remain behind vehicle 1, overtake it before vehicle 2 passes, or overtake it after vehicle 2 has passed. As we consider a straight axis-align road, the curvilinear and cartesian coordinates match in this example.

Assuming that the future trajectory of the obstacles is known in advance, we use the same notations as in the previous chapter and denote by $\mathscr{S}^f$ the free region of the space-time (or free space-time), which is illustrated in Figure 9.3.

As presented earlier, we propose to decompose $\mathscr{S}^f$ in convex subregions having adjacency relations. First, we partition horizontal planes (corresponding to fixed time instants) using relative positions with respect to each obstacle as illustrated in Figure 9.4. Each subset of the partition corresponds to positions where the ego-vehicle is either located in front (f), to the left (l), behind (b) or to the right (r) of each obstacle. Using the additional information given by road boundaries, this partitioning technique yields four subsets denoted by lb, lf, br and fr, indicating the relative position of the ego-vehicle

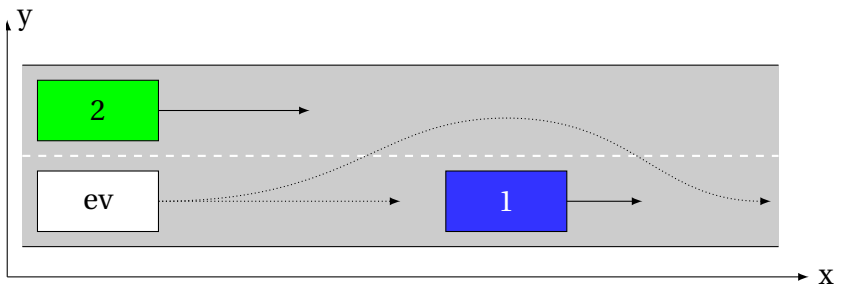


Figure 9.2 – Example driving situation involving multiple maneuver choices for an ego-vehicle (labeled as ev): overtake the slower (blue, denoted 1) vehicle before the green vehicle (2) passes or wait behind the blue vehicle, possibly overtaking after the green vehicle has passed. Solid arrows represent the velocity of each vehicle; dotted arrows represent possible ego-vehicle trajectories.

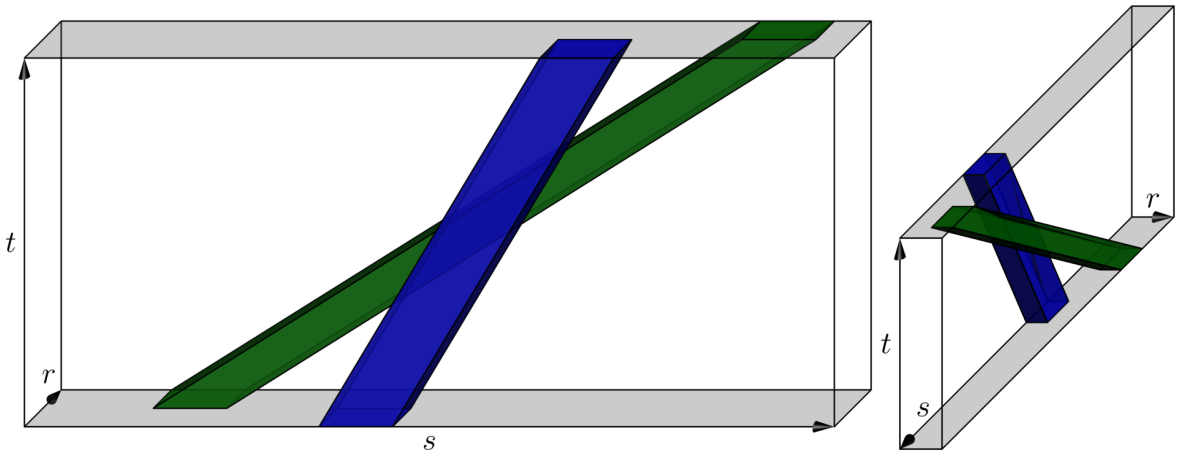


Figure 9.3 – Free space-time $\mathscr{S}^f$ (in white) corresponding to the situation of Figure 9.2. Obstacles are pictured in the color of the corresponding vehicle, light-gray planes represent the road extent.

Table 9.1 – Validity sets Adj(A, B)

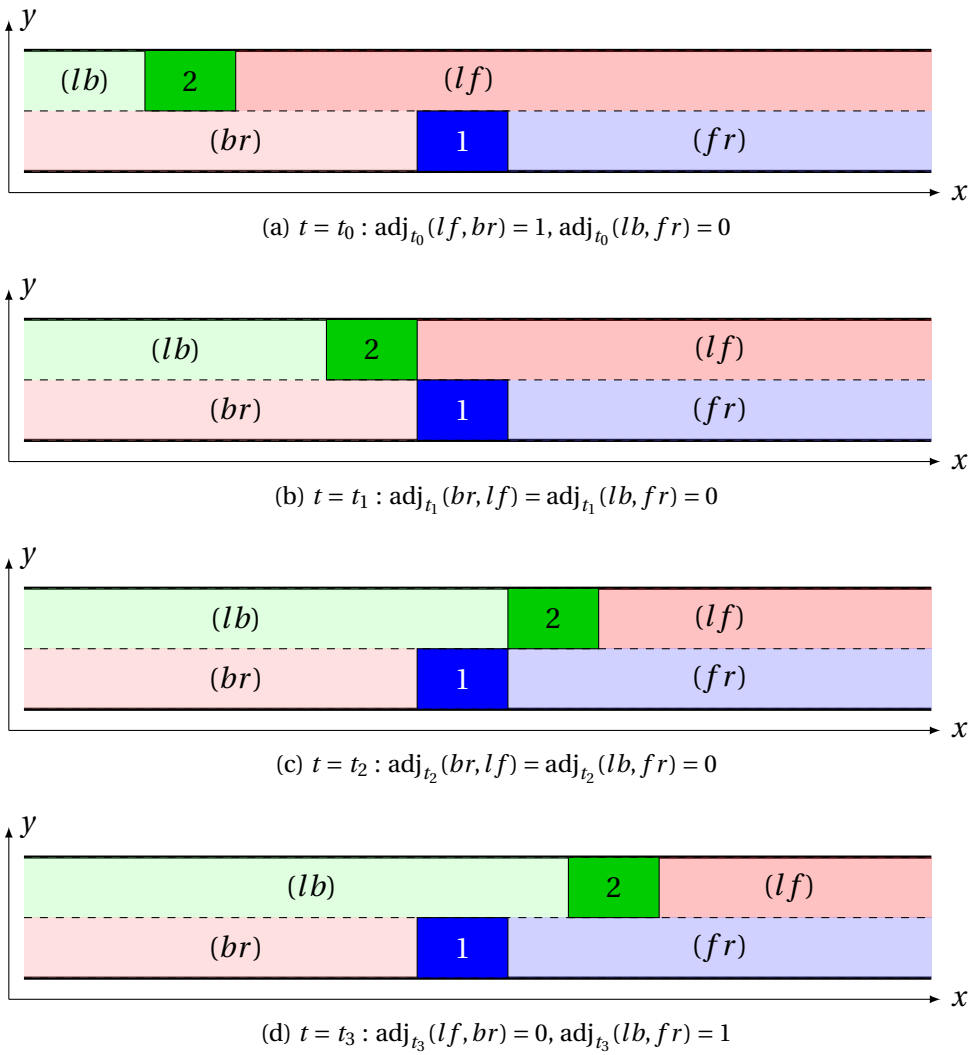
	br	fr	lb	lf
br	$[t_0, +\infty)$	$\emptyset$	$[t_0, +\infty)$	$[t_0, t_1)$
fr	$\emptyset$	$[t_0, +\infty)$	$(t_2, +\infty)$	$[t_0, +\infty)$
lb	$[t_0, +\infty)$	$(t_2, +\infty)$	$[t_0, +\infty)$	$\emptyset$
lf	$[t_0, t_1)$	$[t_0, +\infty)$	$\emptyset$	$[t_0, +\infty)$

from obstacle 1 and 2 in this order. We call these labels *signature* of each subset. Additionally, for two such subsets A and B at a given time t, we can define an adjacency relation adj_t (related to that of [43]), such that $\text{adj}_t(\text{A}, \text{B}) = 1$ if the intersection of their closures is not empty, *i.e.* $\bar{\text{A}} \cap \bar{\text{B}} \neq \emptyset$.

This partitioning method can be generalized to the three-dimensional space-time by using unions of regions sharing the same signature, as shown in Figure 9.5. The notion of adjacency described above can be extended, and we let Adj(A, B) be the set of times t such that $\text{adj}_t(\text{A}, \text{B}) = 1$. We call the set Adj(A, B) the *validity set* of the transition from A to B, corresponding to time periods for which a collision-free trajectory from A to B exists. The validity sets in this example are given in Table 9.1, with initial time t_0.

Using Table 9.1, we can build a directed graph (that we call *navigation graph*) representing all the possible transitions between cells of the partition as shown in Figure 9.6: each vertex of this graph corresponds to a partition cell, and we add the edge A → B if $\text{Adj}(\text{A}, \text{B}) \neq \emptyset$. Additionally, we associate with each edge of the graph the corresponding validity set. A path in this graph is given as a succession of edges and associated transition times within the validity set of each edge, for instance $\big((br \to lb, t_1), (lb \to fr, t_2)\big)$ corresponding to the maneuver of waiting for the green vehicle (2) to pass before overtaking the blue one (1). Between these explicit transition times, the ego-vehicle is supposed to remain inside the last reached cell.

Using this graph-based representation also allows computing a risk metric associated to a maneuver, that we call *time margin*. This measure is defined as the time which remains to the ego-vehicle to perform a particular maneuver, before the most constrained transition becomes impossible. To illustrate this notion (which is formally defined in Section 9.3), we present example time margins for a selection of paths in Table 9.2.



(a) $t = t_0$: $\mathrm{adj}_{t_0}(lf, br) = 1$, $\mathrm{adj}_{t_0}(lb, fr) = 0$

(b) $t = t_1$: $\mathrm{adj}_{t_1}(br, lf) = \mathrm{adj}_{t_1}(lb, fr) = 0$

(c) $t = t_2$: $\mathrm{adj}_{t_2}(br, lf) = \mathrm{adj}_{t_2}(lb, fr) = 0$

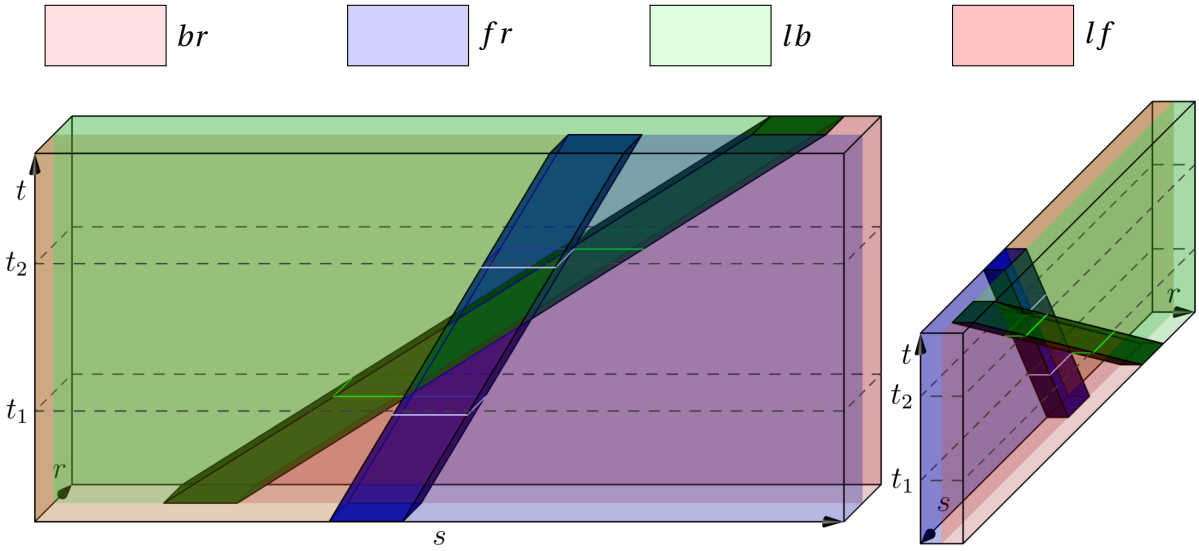
(d) $t = t_3$: $\mathrm{adj}_{t_3}(lf, br) = 0$, $\mathrm{adj}_{t_3}(lb, fr) = 1$

Figure 9.4 – Partitioning of the 2D space at different times in our example scenario, and adjacency relations adj. In this example, $\mathrm{adj}_t(lb, br) = \mathrm{adj}_t(lf, fr) = 1$ and adj_t is symmetrical at all times.

Although this continuous approach is mathematically interesting, it is not necessarily suited for practical computer implementation, which is generally based on time sampling. For this reason, we also propose a discrete partitioning as shown in Figure 9.7: for a discretization time step $\tau > 0$, we approximate the free space as a union of disjointed cuboids[1] of the form $A \times [t_0 + k\tau, t_0 + (k+1)\tau)$ where A is a subset in the partition at time $t_0 + k\tau$. Using this time-discretized partition, we can adapt the notion of adjacency to design a time-discretized navigation graph, as shown in Figure 9.8. In this graph, a path can be simply given as a list of successive vertices, thus allowing to use classic exploration algorithms. The time margin of an edge in the graph can also be easily computed, as shown in Section 9.3. Note that it is also possible to perform an event-based (instead of constant-time-based) partition, which has the advantage of exactly matching the time instants when the adjacency of two cells changes. However, since the partitioning will ultimately be used using the ego-vehicle's feasible dynamics – which are not easily integrated into an event-based framework – the constant-time discretization is preferred.

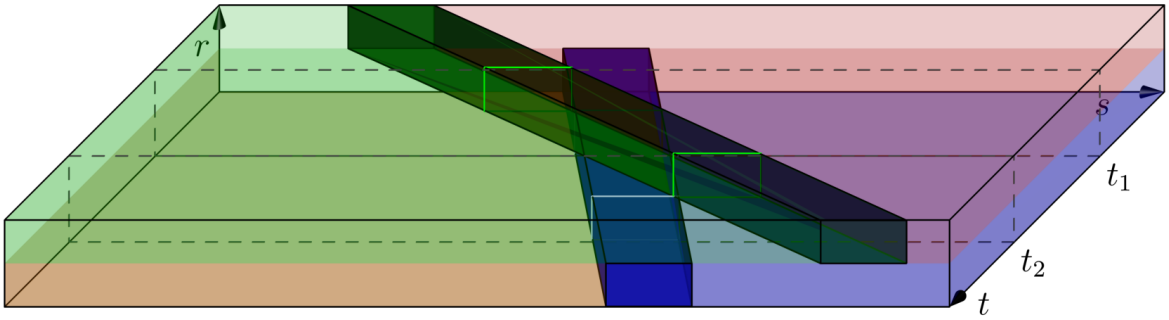
We argue that the proposed graph-based representation has two main advantages. First, the combinatorial part of the trajectory planning problem, consisting in choosing

[1] Note that it is also possible to consider a constant velocity for each obstacle over each time step, leading to rhombohedral cells.



(b) Rectilinear (s, t) plane

(c) Rectilinear (r, t) plane



(d) Rectilinear (s, r) plane

Figure 9.5 – Partitioning of the free space-time of Figure 9.3 into four cells. The legend gives the signature of each cell, with blue obstacle first. Lighter rectangles represent the intersection of the obstacle sets with the t_1 and t_2 planes.

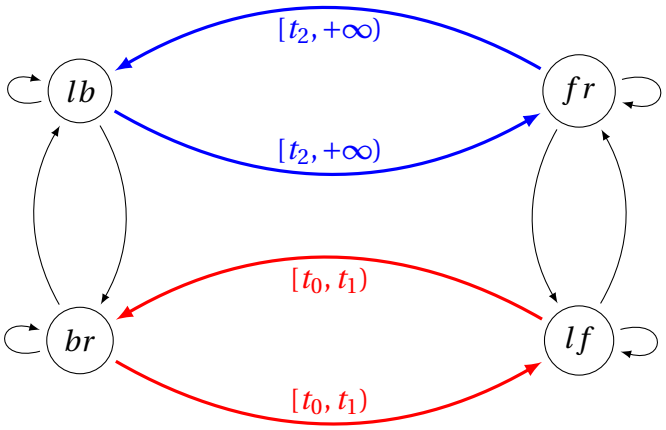
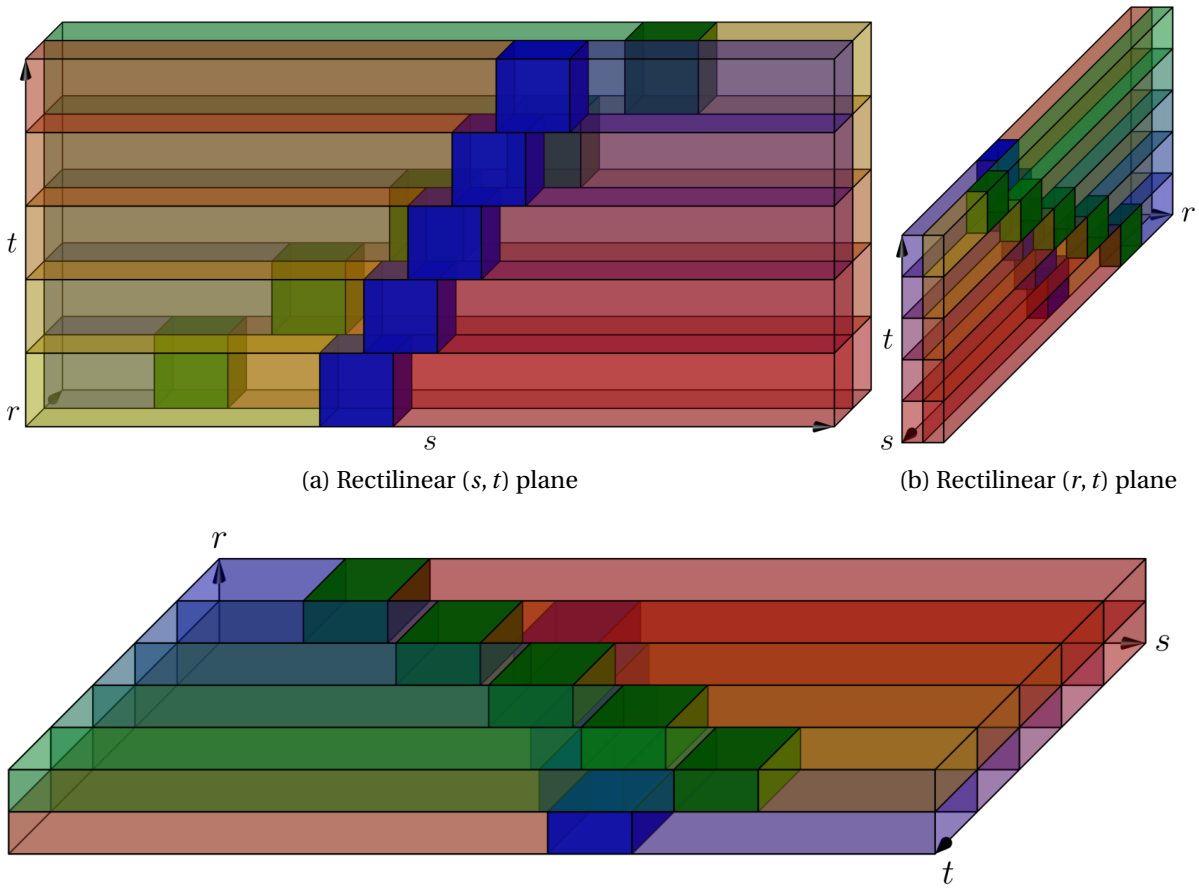


Figure 9.6 – Navigation graph corresponding to Figure 9.5, with validity set of each edge. Thinner edges shown in black have a validity set $[t_0, +\infty)$ (omitted for readability).



(a) Rectilinear (s,t) plane

(b) Rectilinear (r,t) plane

(c) Rectilinear (s,r) plane

Figure 9.7 – Discrete partitioning of the free space-time of Figure 9.3, with $t_0 = 0$.

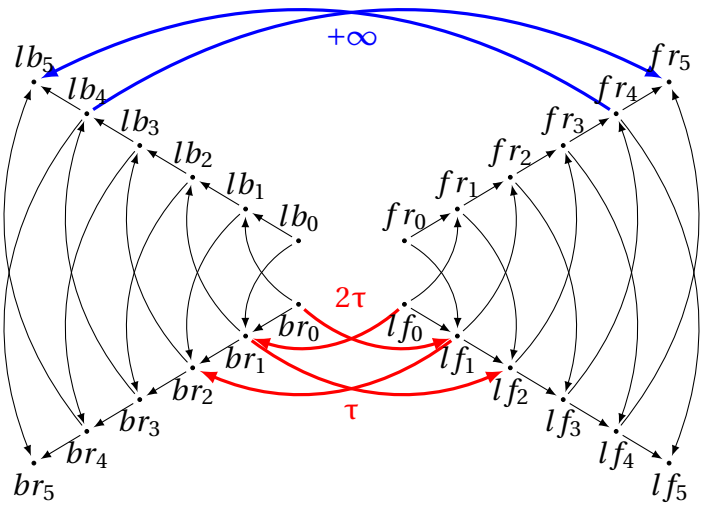


Figure 9.8 – Discrete-time navigation graph and time margins corresponding to the partition of Figure 9.7. Vertex A_k corresponds to the ego-vehicle being in set A at time $t_0 + k\tau$.

Table 9.2 – Time margins of example paths

Path	Most constr. trans.	Margin
$((br \to br, t_0))$	$br \to br$	$+\infty$
$\big((br \to lb, t_1), (lb \to fr, t_3)\big)$	$lb \to fr$	$+\infty$
$\big((br \to lf, t_0), (lf \to fr, t_1)\big)$	$br \to lf$	$t_1 - t_0$

a feasible maneuver around the obstacles, is reduced to selecting a path in a navigation graph. We will show in Chapter 10 that, once such a path is given, computing a corresponding optimal trajectory becomes relatively simple for a large class of cost functions. Second, the graph approach makes it easy to take into account safety margins by avoiding exploration of time-constrained edges, which can be useful to handle uncertainty in trajectory estimation. Additional metrics can also be computed (see, *e.g.*, [114]) for spatial constraints, in order to account for control or positioning error.

9.3 Mathematical results

9.3.1 Modeling

We now proceed to theorize and generalize the intuitions exposed in the previous section. We consider an autonomous ego ground vehicle, driving on a road in the presence of obstacles, which can either be fixed or mobile, and we make the assumptions of Chapter 8 so that we can represent the configuration of the ego-vehicle using curvilinear coordinates (s, r), with $r_{min}(s) \leq r \leq r_{max}(s)$ defining the lateral extent of the road $\mathscr{R}$.

We denote by $\mathscr{O}$ the set of obstacles (considered as open sets) existing on the road around the ego-vehicle, and by $\mathrm{N} = |\mathscr{O}|$ the number of obstacles. At a given time t_0, we consider a time horizon T and we assume that an estimation of the trajectory of each obstacle $o \in \mathscr{O}$ is available over $[t_0, t_0 + \mathrm{T}]$. This estimation could come, for instance, be performed through machine learning techniques as presented in Chapter 7. We let $\mathscr{S} = \mathscr{R} \times [t_0, t_0 + \mathrm{T}]$ the set of space-time points for which the vehicle is on the road, and we define the free portions of the space (respectively, of the space-time) as follows:

Definition 7 (Free space)**.** *The (collision-)free space at time t is the set*

$$\mathscr{C}_t^f = \mathscr{R} \setminus \bigcup_{o \in \mathscr{O}} o.$$

The (collision-)free space-time over $[t_0, t_0 + \mathrm{T}]$ is the set

$$\mathscr{S}^f = \left\{ \mathscr{C}_t^f \times \{t\} \ : \ t \in [t_0, t_0 + \mathrm{T}] \right\}.$$

We call obstacle space-time $\mathscr{S}^o$ the complement of $\mathscr{S}^f$; to simplify the rest of the presentation, we consider that for all $t_1 \in [t_0, t_0 + \mathrm{T}]$, the intersection of $\mathscr{S}^o$ is a union of (potentially rotated) rectangles; due to the roughly rectangular shape of classical vehicles, this assumption does not excessively sacrifice precision. Moreover, we assume that the road boundary functions r_{min} and r_{max} are piecewise-linear and continuous. In the following subections, we present our approach to partition $\mathscr{S}^f$ into semantically meaningful subsets using a two-step algorithm: first, we partition planes corresponding to a fixed time $t_1 \in [t_0, t_0 + \mathrm{T}]$ in Section 9.3.2; second, we deduce a partition of $\mathscr{S}^f$ in Sections 9.3.3 and 9.3.4.

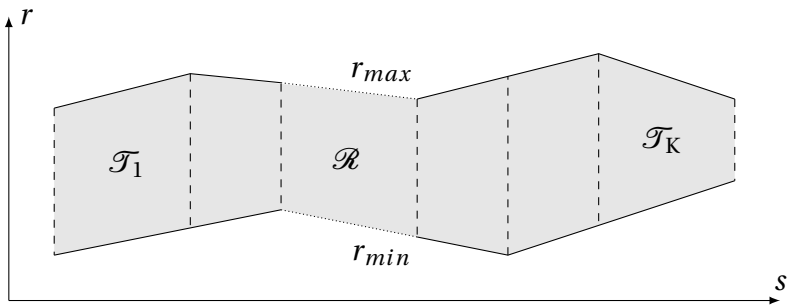


Figure 9.9 – Decomposition of the road (in grey) in trapezes.

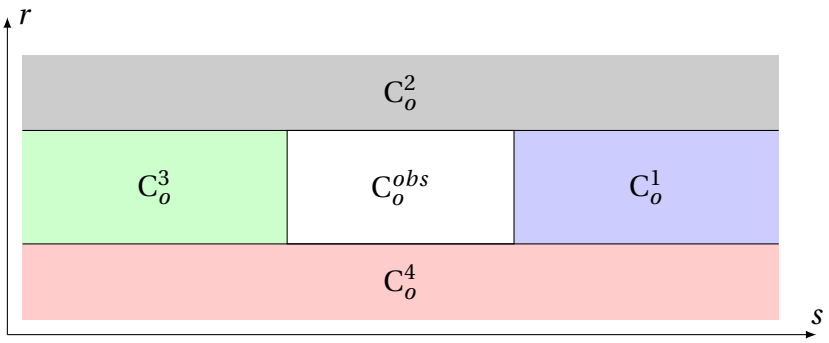


Figure 9.10 – Partitioning of the 2D space around a single obstacle (C_o^{obs}) into four collision-free regions C_o^i.

In what follows, we systematize the approach presented in Section 9.2 in a two-step partitioning algorithm: first, we partition planes corresponding to a fixed time $t_1 \in [t_0, t_0 + T]$ in Section 9.3.2; second, we deduce a partition of $\mathscr{S}^f$ in Sections 9.3.3 and 9.3.4.

9.3.2 Semantic free-space partitioning

First, note that we can use trapeze decomposition to partition the road in convex regions using r_{min} and r_{max} as shown in Figure 9.9; since the road profile does not depend on time, this decomposition allows to fully partition χ by using cylinders with trapezoidal base. Each trapeze $\mathscr{T}_k$ (with $k \in \{1..K\}$) can be defined by a set of linear constraints[2], in the form $A_k X \leq b_k$ with A_k a 4-by-2 matrix, $X = [s, r]^T$ and b_k a vector of $\mathbb{R}^4$. This approach allows modeling varying roadway width and curvature, but may require an important number of trapezes to handle sharp bends correctly.

For a single rectangular obstacle o at time t_1, we define four regions $C_o^i \subset \mathbb{R}^2$ ($i \in \{1..4\}$) as illustrated in Figure 9.10; as in Section 9.2, these regions can be identified as positions where the ego-vehicle is located in front, to the left, behind or to the right of the obstacle. Similarly, we let C_o^{obs} be the obstacle region corresponding to o. Since all obstacles are assumed rectangular, each of the C_o^i regions is defined by a set of linear constraints[3] in the form $A_o^i X \leq b_o^i$, with A_o^i a two-column matrix, $X = [s, r]^T$ and b_o^i a vector having the same number of lines as A_o^i. The partition of the free space $\mathscr{C}_{t_1}^f$ can be built recursively according to Algorithm 2; Theorem 5 ensures the validity of this algorithm; we let $\mathscr{P}^{t_1}$ be the partition of $\mathscr{C}_{t_1}^f$ obtained by Algorithm 2.

[2] To ensure the sets are disjoint, some of the inequalities should be strict. In practice, we use non-strict inequalities with a small tolerance ε.

[3] With the same restrictions as before.

Algorithm 2: Partitioning of $\mathscr{C}^f_{t_1}$

```
set 𝒫_0 := {𝒯_k}_{k=1..K}                          // Initialize 𝒫_0 as a partition of ℛ
set N = |𝒪|
for n = 1..N do
    /* Loop over all obstacles o_n                                              */
    set 𝒫_n = {}
    forall C ∈ 𝒫_{n-1} do
        /* Loop over cells C in 𝒫_{n-1}                                         */
        for j = 1..4 do
            if C^j_{o_n} ∩ C ≠ ∅ then
                set 𝒫_n = 𝒫_n ∪ {C^j_{o_n} ∩ C} /* Partition C \ C^obs_{o_n}    */
    /* This loop ensures the invariant:  ∀e ∈ 𝒫_n  C^j_{o_n} ⊃ e ≠ ∅             */
set 𝒫^{t_1} = 𝒫_N
```

Theorem 5 (Partition). *$\mathscr{P}^{t_1}$ is a partition of $\mathscr{C}^f_{t_1}$.*

Proof. We will prove that, for all $0 \leq n \leq \mathrm{N}$, $\mathscr{P}_n$ is a partition of $\mathscr{R}_n = \mathscr{R} \setminus \bigcup_{i=1}^{n} \mathrm{C}^{obs}_{o_i}$. First, this property is verified for $\mathscr{P}_0$ which is a partition of $\mathscr{R}$. Second, the loop preserves the following invariants for all $n \geq 1$ and $e \in \mathscr{P}_n$:

- $e \neq \emptyset$ and $\exists e' \in \mathscr{P}_{n-1}$ such that $e \subset e'$;
- for all $1 \leq i \leq n$, $\exists j \in \{1..4\}$ such that $e \subset \mathrm{C}^j_{o_i}$.

Thus, $\mathscr{P}_n = \left\{ e \cap \mathrm{C}^j_{o_n} \middle| e \in \mathscr{P}_{n-1}, j \in \{1..4\}, e \cap \mathrm{C}^j_{o_n} \neq \emptyset \right\}$. Since the sets $\left(\mathscr{C}^j_{o_n}\right)_{j=1..4}$ define a partition of $\mathbb{R}^2 \setminus \mathrm{C}^{obs}_{o_n}$ and since all $e \in \mathscr{P}_0$ is a subset of $\mathscr{R}$, we deduce by induction that all elements of $\mathscr{P}_n$ are nonempty subsets of $\mathscr{R}_n$.

Reciprocally, for all any $q \in \mathscr{R}_n = \mathscr{R}_{n-1} \setminus \mathrm{C}^{obs}_{o_n}$ there exists $j \in \{1..4\}$ such that $q \in \mathscr{R}_{n-1} \cap \mathrm{C}^j_{o_n}$. Since $\mathscr{P}_0$ is a partition of $\mathscr{R}$, inductive reasoning yields $\mathscr{R}_n \subset \bigcup_{e \in \mathscr{P}_n} e$. □

From the previous proof, we deduce that our partitioning of $\mathscr{C}^f_{t_1}$ bijectively corresponds to relative positions from all N obstacles in the free space at time t_1. Thus, each element in the partition can be uniquely defined by a signature, as stated in Corollary 1 and Definition 8:

Corollary 1 (Semantization). *For all $e \in \mathscr{P}^{t_1}$, there exists a unique tuple*

$$\sigma_{t_1}(e) = (k, j_1, \ldots, j_{\mathrm{N}}) \in \{1..\mathrm{K}\} \times \{1..4\}^{\mathrm{N}}$$

such that $e = \mathscr{T}_k \cap \bigcap_{n=1..\mathrm{N}} \mathrm{C}^{j_n}_{o_n}$.
Using $\Sigma = \{1..\mathrm{K}\} \times \{1..4\}^{\mathrm{N}}$, σ_{t_1} is a bijection from Σ to $\mathscr{P}^{t_1} \cup \emptyset$.

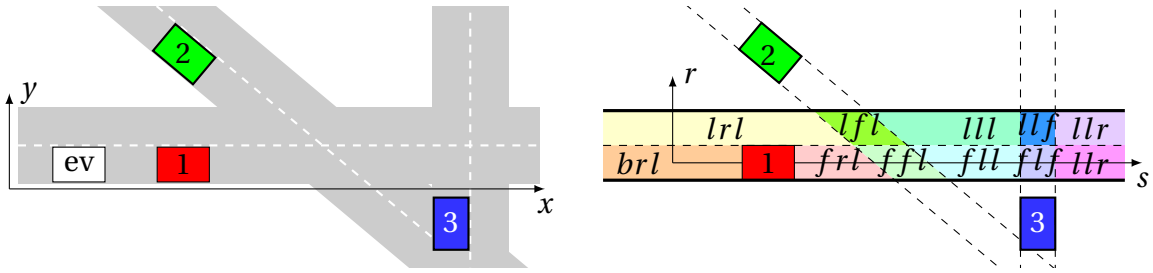
Definition 8 (Signature). *We call $\sigma_{t_1}(e) \in \Sigma$ from Corollary 1 the signature of subset e.*

Moreover, there is a finite number of elements in the partition which is bounded by $\mathrm{K4}^{\mathrm{N}}$ for K trapezes and N obstacles. Additionally, all elements $e \in \mathscr{P}^{t_1}$ also are convex polygons (or *cells*), which can be fully described using a single (matrix, vector) pair that

can easily be stored in computer memory. Figure 9.11 and fig. 9.11b illustrate our partitioning in a more complex scenario[4] with 3 vehicles; note that, for clarity purposes, we respectively used f, l, b, r instead of $1, 2, 3, 4$ as defined in Definition 8. Also remark that, although Figure 9.11a is shown in world coordinates (x, y), Figure 9.11b uses Frenet coordinates (s, r). In order to encode the relation between elements of the partition, we introduce the notion of adjacency as follows:

Definition 9 (Adjacency). *For $e_1, e_2 \in \mathscr{P}^{t_1}$, we say that e_1 and e_2 are adjacent if, and only if the intersection of their closures is not empty,* i.e. $\overline{e_1} \cap \overline{e_2} \neq \emptyset$.
For $\sigma, \sigma' \in \Sigma$, we let $\mathrm{adj}_{t_1}(\sigma, \sigma') = 1$ *if* $\sigma_{t_1}^{-1}(\sigma)$ *and* $\sigma_{t_1}^{-1}(\sigma')$ *are adjacent, and* 0 *otherwise.*

Note that this definition considers cells whose closures intersect at single point as adjacent; this situation could happen, *e.g.*, in the case shown in Figure 9.4b between br and lf. From a theoretical standpoint, the hypothesis that obstacles are open sets allows to overcome this issue since, in this case, the vehicle can actually perform the maneuver; in practice, the use of safety margins around the obstacles, and constraints on time margin (see Section 9.3.4) prevent this question from becoming an issue. The main reason for choosing this slightly looser criterion compared, *e.g.*, to that of [43] – requiring the intersection to be a non-singleton segment – is that it can be verified in polynomial time using the matrix inequality representation and linear programming.



(a) Geometric obstacles

(b) Semantic partition and subset signature

Figure 9.11 – A more complex example with 3 obstacles.

9.3.3 Continuous navigation graph

We now proceed to generalize these results to the free space-time $\mathscr{S}^f$; to define a partition of this space we use the union (over time) of cells sharing the same signature:

Definition 10 (Space-time cell). *For $\sigma \in \Sigma$, we let*

$$\mathrm{E}_\sigma = \bigcup_{t_1 \in [t_0, t_0 + \mathrm{T}]} \sigma_{t_1}^{-1}(\sigma) \times \{t_1\}$$

be the space-time cell corresponding to σ, i.e. *the set of all points in the free space-time sharing this signature.*

Since the set of non-empty elements of $\sigma_t^{-1}(\Sigma)$ defines a partition of $\mathscr{C}_t^f$, the set $\mathscr{P} = \{\mathrm{E}_\sigma \,|\, \sigma \in \Sigma, \mathrm{E}_\sigma \neq \emptyset\}$ defines a partition of $\mathscr{S}^f$ (see Figure 9.5). The notion of adjacency can then be generalized as follows:

[4] Obstacle regions 1, 2, 3 in Figure 9.11b are computed for a point-mass ego-vehicle, in order to match the vehicle shapes shown in Figure 9.11.

Definition 11 (Validity set)**.** *For $\sigma, \sigma' \in \Sigma$, we define the validity set*

$$\mathrm{Adj}(\sigma, \sigma') = \left\{ t \in [t_0, t_0 + \mathrm{T}] \,\middle|\, \mathrm{adj}_t(\sigma, \sigma') = 1 \right\}.$$

Using this notion, we can now define a navigation graph (see Figure 9.6) as follows:

Definition 12 (Continuous navigation graph)**.** *The continuous navigation graph is the directed graph $\mathcal{G}^c = (\mathcal{V}^c, \mathcal{E}^c, \mathrm{Adj})$ with vertex set $\mathcal{V}^c = \{\sigma \in \Sigma \,|\, \mathrm{E}_\sigma \neq \emptyset\}$, edges set*

$$\mathcal{E}^c = \left\{ (\sigma_1, \sigma_2) \in \Sigma^2 \,\middle|\, \mathrm{Adj}(\sigma_1, \sigma_2) \neq \emptyset \right\}$$

and associated validity set Adj.

The motivation for introducing this graph is that any collision-free maneuver corresponds to a unique *path* in $\mathcal{G}^c$. To account for the temporal aspect of this graph, such a path is defined as follows:

Definition 13 (Path in $\mathcal{G}^c$)**.** *A path in $\mathcal{G}^c$ is given by a list of vertices $(\sigma_1, \ldots, \sigma_{m+1}) \in \mathcal{V}^c$ so that for all $i \leq m$, $(\sigma_i, \sigma_{i+1}) \in \mathcal{E}^c$, and a list of strictly increasing transition times $(t_1, \ldots, t_m)$ such that:*

i. for all $i \in \{1..m\}$, $t_i \in \mathrm{Adj}(\sigma_i, \sigma_{i+1})$ and

ii. for all $i \in \{0..m\}$, $[t_i, t_{i+1}) \subset \mathrm{Adj}(\sigma_{i+1}, \sigma_{i+1})$.

In other words, a path in $\mathcal{G}^c$ is a sequence of cells and time instants corresponding to the transition time between two successive cells, and where the ego-vehicle can remain in the last occupied cell until the next transition.

Remark 4 (Uniqueness)**.** *Note that in definition Definition 13, it is possible to repeat a single cell E_σ infinitely many times as long as $\mathrm{Adj}(\mathrm{E}_\sigma, \mathrm{E}_\sigma) \neq \emptyset$, which still effectively results in the same effective path. For instance, path $(\sigma, \emptyset)$ (corresponding to the ego-vehicle staying in cell E_σ indefinitely) is equivalent to $\big((\sigma, \sigma), (t_1)\big)$, which corresponds to the ego-vehicle starting in cell E_σ, then "moving" from E_σ to E_σ at time t_1 and then remaining there. To ensure representations are unique, we always assume that two consecutive signatures in the representation of a path in $\mathcal{G}^c$ are distinct.*

For a given path, we can now define the corresponding time margin as the time left for the vehicle to perform the most constrained transition before it becomes impossible:

Definition 14 (Time margin along a path)**.** *Consider a path in $\mathcal{G}^c$ given as*

$$\pi^c = \big((\sigma_1, .., \sigma_{m+1}), (t_1, .., t_m)\big).$$

The time margin along π^c is $\mathrm{V}(\pi^c) = \min_{i=1\ldots m} \left(\sup \left\{ t - t_i \;\middle|\; [t_i, t) \subset \mathrm{Adj}(\sigma_i, \sigma_{i+1}) \right\} \right)$.

Finally, for a given collision-free trajectory, we associate a navigation graph path as follows:

Definition 15 (Correspondance of navigation graph paths)**.** *Let*

$$\pi_0 = \big((\sigma_1, \ldots, \sigma_{m+1}), (t_1, \ldots, t_m)\big)$$

be a path in $\mathcal{G}^c$ and $x(t)$ be a collision-free trajectory for the ego-vehicle. We say that π_0 corresponds to x if, noting $\psi(t) = (x(t), t)$:

i. for all $i \in \{1..m\}$, $\psi\big([t_{i-1}, t_i)\big) \subset \overline{\mathrm{E}_{\sigma_i}}$ and

ii. $\psi\big([t_m, \mathrm{T})\big) \subset \overline{\mathrm{E}_{\sigma_{m+1}}}$.

9.3.4 Discrete-time partitioning

Due to the potentially complex trajectories followed by the obstacles, there is no guarantee regarding the topology of the subsets E_σ, which can for instance have multiple connected components; similarly, $\mathrm{Adj}(\sigma_1,\sigma_2)$ is in general a union of disjointed intervals. To make practical applications easier, we also propose a temporal discretization of the free space-time with a time step duration τ (with $\mathrm{T}=\mathrm{P}\tau$). Note that the value of τ depends on a trade-off between acceptable computation time, length of the planning horizon and required precision in vehicle dynamics; Section 6.5 provides some performance reports on the influence of this parameter. We approximate E_σ as a union of box-shaped cells:

Definition 16 (Discrete space-time cell)**.** *For $\sigma\in\Sigma$ and $p\in\{0\dots\mathrm{P}\}$, we let*

$$\mathrm{E}_\sigma^p=\sigma_{\theta_p}^{-1}(\sigma)\times[\theta_p,\theta_{p+1})$$

be the discrete space-time cell corresponding to σ at step p, with $\theta_p=t_0+p\tau$.

In the rest of this section, we assume[5] that $\mathrm{E}_\sigma^p\subset\mathscr{S}^f$ for all $\sigma\in\Sigma$ and $p\in\{0..\mathrm{P}\}$. Since $\sigma_{\theta_p}^{-1}(\sigma)$ is a convex (or empty) set, E_σ^p is either empty or convex, and fully defined by a set of linear inequalities in the form $\mathrm{A}[\mathrm{X},t]^\mathrm{T}\leq b$ (the comments of footnote 2 page 98 also apply here). Finally, we define a partition of the free space-time $\mathscr{S}^f$ in convex box-shaped cells:

$$\mathscr{P}^\tau=\left\{\mathrm{E}_\sigma^p\;\middle|\;p\in\{0..\mathrm{P}\},\sigma\in\Sigma,\mathrm{E}_\sigma^p\neq\emptyset\right\}$$

as illustrated in Figure 9.7, and we associate a discrete navigation graph:

Definition 17 (Discrete navigation graph)**.** *The discrete navigation graph is the directed graph $\mathscr{G}^d=(\mathscr{V}^d,\mathscr{E}^d)$ with vertex set $\mathscr{V}^d=\mathscr{P}^\tau$ and edges set*

$$\mathscr{E}^d=\left\{\left(\mathrm{E}_{\sigma_1}^p,\mathrm{E}_{\sigma_2}^{p+1}\right)\;\middle|\;\mathrm{E}_{\sigma_1}^p,\mathrm{E}_{\sigma_2}^{p+1}\in\mathscr{V}^d,\mathrm{adj}_{\theta_p}(\sigma_1,\sigma_2)=1\right\}.$$

Therefore, each vertex of $\mathscr{G}$ corresponds to a certain cell of the partition $\mathscr{P}^\tau$ at a given time θ_p, and the edge $v_1\to v_2$ exists if v_1 and v_2 represent two adjacent cells (possibly twice the same) at two consecutive time steps. Paths in $\mathscr{G}^d$ comply with the usual definitions of graph theory and can be given as a set of vertices. Finally, we define the time margin for paths in $\mathscr{G}^d$ as:

Definition 18 (Time margin in the discrete graph)**.** *For a path $\pi^d=\left(\mathrm{E}_{\sigma_0}^0,\dots,\mathrm{E}_{\sigma_{m+1}}^{m+1}\right)$ in $\mathscr{G}^d$, the time margin is $v(\pi^d)=\min\limits_{i=0\dots m}\max\limits_{p=i\dots m}\tilde{v}(i,p)$ with*

$$\tilde{v}(i,p)=\left\{\tau(p-i+1)\;\middle|\;\forall q\in\{i..p\},\mathrm{adj}_{\theta_q}(\sigma_i,\sigma_{i+1})=1\right\}.$$

Note that $\tilde{v}(i,p)$ is the discrete-time equivalent of the set $\left\{t-t_i\;\middle|\;[t_i,t)\subset\mathrm{Adj}(\sigma_i,\sigma_{i+1})\right\}$ from Definition 14, and this discrete time margin is analogous to that of Definition 14. Finally, as in Definition 15, we associate collision-free trajectories to paths in the discrete navigation graph as follows:

Definition 19 (Correspondance of discrete navigation graph paths)**.** *Let $\tau>0$ be a discretization time step, $\pi_0^d=(\mathrm{E}_{\sigma_0}^0,\dots,\mathrm{E}_{\sigma_{m+1}}^{m+1})$ be a path in $\mathscr{G}^d$ and $x(t)$ be a collision-free trajectory for the ego-vehicle. We say that π_0^d corresponds to x if, noting $\psi(t)=(x(t),t)$, we have for all $p\in\{0..m\}$:*

i. $\psi(\theta_p)\in\overline{\mathrm{E}_{\sigma_p}^p}$ *and*

ii. $\psi([\theta_p,\theta_{p+1}))\subset\overline{\mathrm{E}_{\sigma_p}^p}\cup\overline{\mathrm{E}_{\sigma_{p+1}}^p}$.

[5] This assumption can be avoided by using rhombohedral cells instead of cuboids.

9.4 Navigation graph and homotopy classes

In Chapter 8, we showed that the notion of homotopy classes was not sufficient to properly describe the multiple decisions arising from autonomous on-road driving. In this chapter, we therefore proposed a slightly different approach that we argue is more suitable to encode driving decisions for motion planning, as we will show in Chapter 10.

Looking back to the example of Figures 9.5 and 9.6, there obviously are similitudes between homotopy classes of trajectories and paths in the navigation graph. In the case of 2D path planning, reference [43] even showed that non-looping paths in a similarly constructed graph could be mapped bijectively to homotopy classes assuming the underlying free-space partitioning is *minimal*.

However, this is not the case in our navigation graph approach, as we purposely aim to build a more discriminating notion. For instance in Figure 9.11b, the graph paths (frl, ffl, fll) and $(frl, ffl, lfl, lll, fll)$ both correspond to homotopic trajectories. As hinted in [43], the non-equivalence lies in the choice of a non-minimal partition which accounts for the "decision" of changing lanes, even though this decision ultimately has no impact. However, the navigation graph is at least as discriminating as homotopy classes, as stated in Theorem 6. In order to introduce this result, we first define a notion of *equivalence* between navigation graph paths:

Definition 20 (Equivalence)**.** *Consider two paths*[6] π_1 *and* π_2 *in* $\mathcal{G}^c$*, with*

$$\begin{aligned}\pi_1 &= \big((\sigma_1^1,..,\sigma_{m+1}^1),(t_1^1,..,t_m^1)\big) \quad \textit{and}\\ \pi_2 &= \big((\sigma_1^2,..,\sigma_{n+1}^2),(t_1^2,..,t_n^2)\big).\end{aligned}$$

We say that π_1 *and* π_2 *are equivalent, written* $\pi_1 \equiv \pi_2$*, if* $m = n$ *and, for all* $i \in \{1,\dots,m\}$, $\sigma_i^1 = \sigma_i^2 = \sigma_i$ *and either* $\big[t_i^1, t_i^2\big] \subset \mathrm{Adj}(\sigma_i, \sigma_{i+1})$ *or* $\big[t_i^2, t_i^1\big] \subset \mathrm{Adj}(\sigma_i, \sigma_{i+1})$.

We can now use this notion (which does define an equivalence relation) to state the following "injectivity" theorem:

Theorem 6 (Homotopy of trajectories having equivalent navigation graph paths)**.** *Let* x_1 *and* x_2 *be two collision-free trajectories over* $[t_0, t_0 + \mathrm{T}]$ *respectively corresponding to paths* π_1 *and* π_2 *in* $\mathcal{G}^c$*. If* $\pi_1 \equiv \pi_2$*, then* x_1 *and* x_2 *are homotopic.*

Sketch of proof. The main idea behind the theorem is that the equivalence condition

$$\big[t_i^1, t_i^2\big] \subset \mathrm{Adj}(\sigma_i^1, \sigma_{i+1}^1)$$

guarantees that trajectories use the same "driving corridor", which remains open at least over the $[t_i^1, t_i^2]$ interval. Therefore, it is possible to continuously deform one trajectory into the other. A detailed proof is available in Appendix F. □

9.5 Chapter conclusion

In this chapter, we first presented an algorithm to partition the collision-free portion of the configuration space-time into convex spatiotemporal cells which are semantically meaningful in terms of driving. We then proposed a *navigation graph* approach building upon this semantic partitioning, which is used to represent all possible driving decisions

[6] Assuming that consecutive signatures are distinct, as per Remark 4.

for an ego-vehicle alongside with the temporal validity of the corresponding maneuvers. Moreover, we showed that this navigation graph representation extends the notion of homotopy classes, which we showed in Chapter 8 to be insufficient to describe maneuvers for generic on-road driving.

We argue that the main advantage of our graph-based approach is that it allows decoupling the on-road motion planning problem into two simpler subproblems: a decision-making phase (selecting a path on the graph, which is has a "polynomial" complexity) followed by a continuous (potentially convex) optimization problem of finding the best trajectory corresponding to this graph path. As presented in Chapter 7, the latter problem can be solved efficiently using techniques such as MPC.

However, the motion planning problem does remain NP-hard, and the polynomial complexity of path-finding should not obscure the fact that the navigation graph has a number of nodes growing *exponentially* with the number of obstacles. Another difficulty is that paths in the navigation graph do not always correspond to dynamically feasible trajectories. For this reason, efficient exploration strategies in the navigation graph are critical to the overall usability of this approach.

Chapter 10

Graph-based decision-making

" An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem. "

John Tukey (mathematician)

Contents

10.1 Decision-aware motion planning 105

10.2 Motion planning on the navigation graph 106

10.3 Global optimum search 107

10.4 Heuristics 111

10.5 Chapter conclusion 112

10.1 Decision-aware motion planning

In the previous chapter, we proposed a graph-based approach to enumerate the possible driving decisions for on-road motion planning, in an attempt to overcome limitation of earlier approaches to decision-based planning.

Indeed, it is possible to use mixed-integer quadratic programming to directly make the driving decisions as we did in Part I. For instance, reference [55] describes a possible MIQP implementation using so-called *logical constraints* to encode these decisions. However, this formulation may be too constraining to be applied for motion planning, for instance in the case of uncertain predictions of the future behavior of other traffic participants. Similarly, it may be difficult to encode the "desirability" of a candidate decision using only a convex quadratic function, and a pure MIQP approach will often result in the ego-vehicle brushing past obstacles[1]. For this reason, we argue that the ability of a planning framework to provide more expressive power – at the cost of slightly increased computation time – may be desirable.

A second advantageous property of the navigation graph approach is to simplify the use of risk metrics, which can be directly taken into account at the graph exploration

[1] Although it is possible to add margins around the obstacles to counter this particular issue, objectives such as "maximizing the distance to obstacles" cannot be implemented easily.

phase; in [114], the authors used a similar partitioning technique to design a "space margin" metric for 2D path planning. In the previous chapter, we introduced a complementary *time margin* metric, corresponding to a temporal tolerance to execute a particular maneuver, that can be easily computed from our graph representation. This measure is related to the notion of "gap acceptance", commonly used in stochastic decision-making (see, *e.g.*, [115]). We believe that combining a temporal margin (notably accounting for uncertainty in predicting the future trajectory of moving obstacles) as well as a spatial margin (accounting for perception and control errors) is key for trajectory planning and tracking in real-world situations, for instance coupled with MPC or Linear Quadratic Gaussian motion planning and control [116].

Our navigation graph approach generalizes state-machine-based techniques [117, 118] which rely on a predefined set of maneuvers (such as *track lane* or *change lane*) that needs to be manually adapted to the driving situation. By contrast, our method can be applied in many scenarios (including highway and urban driving, for instance crossing an intersection) with the same formalism. Although spatio-temporal graphs have already been used for the control of AGVs [119, 120], no existing approach provides the same desirable properties, and notably to easily account for margins in planning.

In the rest of this chapter, we demonstrate how the navigation graph can be used for on-road motion planning in order to take into account more complex evaluation criteria compared to the MIQP approach of [55]. Moreover, we discuss possible approaches that could be explored in order to improve performance using different heuristics since finding the global optimum may not be as important as quickly converging to a good solution, as mentioned in Part I.

10.2 Motion planning on the navigation graph

As presented in Chapter 9, a path in the navigation graph encodes a family of trajectories that can be followed by the ego-vehicle without colliding with any obstacle, provided they are dynamically feasible. Reciprocally, any collision-free trajectory x corresponds to a unique path $\pi(x)$ in the navigation graph. Noting $\pi_{-1}(\pi_0)$ the set of trajectories corresponding to the path π_0, we deduce the following theorem:

Theorem 7 (Motion-planning equivalence)**.** *Let* $\mathrm{J}(x)$ *be a cost function for a given trajectory* $x(t)$, X *the set of collision-free trajectories, and* Π *the set of paths in* $\mathcal{G}^c$*. Then:*

$$\min_{x\in\mathrm{X}} \mathrm{J}(x) = \min_{\pi_0\in\Pi}\left(\min_{x\in\pi^{-1}(\pi_0)} \mathrm{J}(x)\right) \tag{10.1}$$

Proof. From Definition 15, for any collision-free trajectory $x \in \mathrm{X}$ there exists $\pi_0 \in \Pi$ such that $\pi(x) = \pi_0$. Therefore, $\min_{x\in\mathrm{X}} \mathrm{J}(x) \leq \min_{\pi_0\in\Pi}\left(\min_{x\in\pi^{-1}(\pi_0)} \mathrm{J}(x)\right)$. Reciprocally, for all $\pi_0 \in \Pi$, any $x \in \pi^{-1}(\pi_0)$ is guaranteed to be collision-free, leading to the reciprocal inequality. □

In other words, it is equivalent to find an optimal trajectory for the ego-vehicle, and to find an optimal path in the navigation graph $\mathcal{G}^c$ and then the optimal trajectory corresponding to this path; this result can of course be extended to paths in the discrete graph $\mathcal{G}^d$.

An interesting feature of this decomposition of the trajectory planning problem is that we effectively separate the discrete choice of a maneuver variant, and the search for an optimal control corresponding to this maneuver as was obtained in [43] for path-planning.

As presented in Chapter 7, this second problem can be solved efficiently under certain assumptions on the vehicle dynamics. Moreover, we can get polynomial time computation in certain cases, as presented below.

We now assume that the ego-vehicle follows the discrete linear dynamics $x_{p+1} = \mathrm{A}x_p + \mathrm{B}u_p$ for a state x_p and a control $u_p \in \mathrm{U}$ (with U a convex polyhedron) with a discretization time step τ, where A and B have constant coefficient. For a positive semi-definite matrix Q, a line vector L and defining $\mathrm{X}_p = [x_p^{\mathrm{T}}, u_p^{\mathrm{T}}]^{\mathrm{T}}$, we consider the generic quadratic cost function $\mathrm{J}(x,u) = \sum_p \frac{1}{2}\mathrm{X}_p^{\mathrm{T}}\mathrm{Q}\mathrm{X}_p + \mathrm{L}^{\mathrm{T}}\mathrm{X}_p$. In this case, the optimal trajectory corresponding to a path π_0 in $\mathscr{G}^d$ can be computed in polynomial time:

Theorem 8 (Polynomial time computability)**.** *Let π_0 be a path in $\mathscr{G}^d$ and x_0 an initial state for the ego-vehicle. The optimal trajectory (and associated control sequence) starting from x_0 and realizing*

$$\min_{\substack{(x_p)\in\pi^{-1}(\pi_0)\\ \forall p\ u_p\in\mathrm{U}}} \mathrm{J}(x,u)$$

can be computed in polynomial time in the number of obstacles and time steps.

Proof. We will show that this problem is an instance of convex quadratic programming (QP), which has a complexity $\mathscr{O}(n^3)$ where n is the number of constraints [121]. First, the cost function J is quadratic and convex. Second, vehicle dynamics and control bounds can be encoded as linear constraints. Moreover, the condition $(x_p) \in \pi^{-1}(\pi_0)$ corresponds to a set of $\mathscr{O}(\mathrm{PN})$ linear constraints, leading to a QP problem with complexity $\mathscr{O}\left((\mathrm{PN})^3\right)$ for N obstacles over P time steps, thus proving the announced result. □

However, the graphs $\mathscr{G}^c$ and $\mathscr{G}^d$ do have a number of vertices scaling exponentially with the number of obstacles. An advantage of our approach is that exhaustive exploration is not required to ensure global optimality, especially when considering safety margins such as a minimum required time or space validity (see [114]). Moreover, tailored exploration heuristics could also be developed.

10.3 Global optimum search

To showcase the advantages of the navigation graph approach and the axes for improvement, we first present simulation results in the scenario of Figure 9.2. For illustration purposes, we consider a very simple second-order dynamics for the ego-vehicle as:

$$\mathrm{X}_{p+1} = \begin{pmatrix} 0 & \mathrm{I}_2 \\ 0 & 0 \end{pmatrix}\mathrm{X}_p + \begin{pmatrix} 0 \\ \mathrm{I}_2 \end{pmatrix} u_p$$

with

$$\mathrm{X}_p = [s, r, \dot{s}, \dot{r}]_p^{\mathrm{T}},$$

$$u_p = [a_{lon}, a_{lat}]_p^{\mathrm{T}}$$

and I_2 denotes the $\mathbb{R}^2$ identity. To account for the nonholonomic constraints, we bound the lateral velocity as $|\dot{r}_p| \leq \alpha \dot{s}_p$ with $\alpha > 0$ a parameter, and we also require a_{lon} and a_{lat} to be bounded. The objective function is chosen as

$$\mathrm{J} = \sum_p \left(\dot{s}_p - \dot{s}_p^{ref}\right)^2 + \dot{r}_p^2 + r_p^2,$$

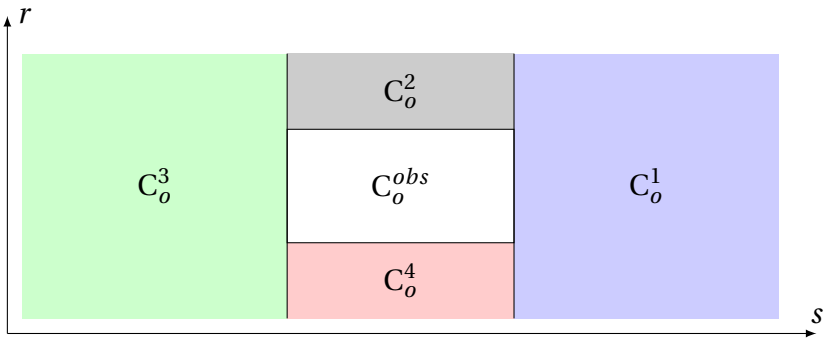


Figure 10.1 – Another, less meaningful partitioning of the 2D space.

where $\dot{s}_p^{ref}$ is a reference target velocity, and we use a planning horizon of 10 s with a time step $\tau = 1$ s, and a minimum time margin of 1 s. Despite its simplicity, this model is generally suitable when driving on low-curvature roads (see, *e.g.*, [55], which has been experimentally validated on a ring of radius *circa* 20 m).

We propose to compare three different approaches to plan the maneuver of the ego-vehicle in this scenario. The first one is the pure mixed-integer quadratic programming (MIQP) method from [55], which we use as a reference; this approach has been proven to be computationally efficient, but is not suited to take risk metrics such as the time margin into account. We then evaluate our semantic partition-based approach with a custom graph exploration algorithm; to demonstrate the advantage of using the semantic partitioning of Chapter 9, we also present results obtained with the same exploration algorithm on another free space-time partition shown in Figure 10.1, which is not as relevant to driving decisions.

As mentioned earlier, one difficulty in the navigation graph approach is the need to ensure that there exists a dynamically feasible trajectory corresponding to a given graph path. In this implementation, we use a naive approach consisting in computing the optimal trajectory corresponding to each path; as stated in Theorem 8, this computation can be performed relatively fast, and in feasibility can be proven mathematically. Additionally, we use a branch-and-bound technique to prune infeasible or poor quality branches. The exploration algorithm is described in Algorithm 3; note that when performing motion planning, the initial position of the ego-vehicle is known, and thus the starting cell in the discrete navigation graph is uniquely determined. As a result, the decision-making process becomes a *tree* instead of a *graph* exploration problem; therefore, Algorithm 3 is formulated in terms of *branches*.

The trajectories from both algorithms are shown in Figure 10.2; Table 10.1 reports the computation time for all steps of our algorithm using a standard desktop computer. For comparison purposes, we added an implementation of the same problem using a pure mixed-integer quadratic programming (MIQP) method from [55]. We use an implementation of Seidel's linear programming algorithm [122] to perform the partitioning, and Gurobi [67] in version 7.5 to solve quadratic programming (trajectory optimization) and MIQP (from [55]) problems. Reported times are obtained on a desktop Intel i7-6700K CPU.

As shown in Table 10.1, the pure MIQP approach is faster than our navigation graph one; indeed, the decoupling between decision-making and ego-vehicle dynamics does not allow as efficient a resolution. However, the graph-based decision-making approach has much more expressive power since it allows adding arbitrary high-level criteria – such as time margin – to the optimization objective. In Figure 10.2, such considerations trans-

Algorithm 3: Decision-based motion planning

Data: Discrete navigation graph $\mathcal{G}^d$, initial position (s_0, r_0), QP cost function J_q, decision cost function $J_d \geq 0$

```
set σ_0 = locateCell(s_0, r_0)                                  // Starting cell
set visitList = {(0, σ_0)}                                      // History of visited cell
set prevDepth = 0                                               // Previous exploration depth
set bestVal = +∞                                                // Best objective value so far
set cellConstraints = {}                                        // List of cell constraints
set bestBranch = {σ_0}                                          // Best branch found sor far
while visitList ≠ ∅ do
    /* Depth-first search :  pop the first element of visitList          */
    set (k, σ_k) = popHead(visitList)
    for i = k..prevDepth do
        /* Remove constraints for steps i ∈ {k,..,prevDepth}            */
        removeCellConstraints(k)
    /* Add constraints for the current step and update current depth    */
    addCellConstraints((s_k, r_k) ∈ σ_k)
    prevDepth = k
    /* Minimize J under the given cell constraints and taking ego-vehicle
       dynamics constraints
       Returns the optimal value if feasible, or +∞ otherwise           */
    set branchVal = solveConstrainedQP(J_q) + J_d(cellConstraints)
    /* If the problem is infeasible or the current branch has been proven
       sub-optimal, discard and move to another branch                  */
    if branchVal >= bestVal then
        continue
    /* We have reached a terminal node with a lower cost; update the best
       solution and continue exploring                                  */
    if k == numSteps-1 then
        bestVal = branchVal
        updateBestBranch()                          // update the best branch found
        continue
    /* Otherwise, loop over reachable cells from σ_k and add them if the time
       margin is sufficient                                             */
    forall σ_{k+1} ∈ reachable(k, σ_k) do
        if validity(σ_k, σ_{k+1}) ≥ minValidity then
            visitList = {visitList, (k+1, σ_{k+1})}
return bestBranch
```

Table 10.1 – Summary of computation time for 10 time steps (average over 1000 repetitions)

Algorithm	Comp. time	Objective val.
Partitioning	2.0 ms	-
Optimal path computation	< 1 ms	-
MIQP (no margin)	10.5 ms	13.58
Graph exploration (semantic partitioning, no margin)	28.3 ms	13.58
Graph exploration (other partitioning, no margin)	32.8 ms	13.58
Graph exploration (1 s margin)	27.5 ms	13.89

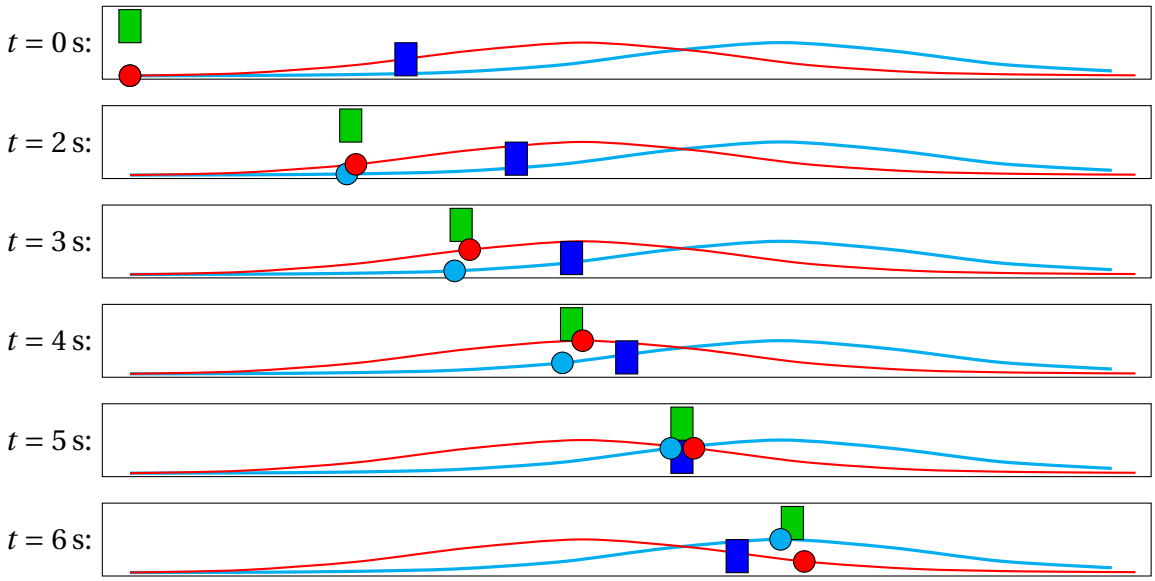


Figure 10.2 – Trajectories computed by both algorithms (MIQP in red, our graph-based approach in thicker cyan) for a time margin of 1 s. The rectangles correspond to the vehicles of Figure 9.2; the circles to the ego-vehicle position.

late into different maneuver choices for the same initial conditions between the MIQP solution, where the ego-vehicle tries to overtake the blue vehicle (1) before the green one (2), and our graph-based approach where this higher-risk maneuver is excluded; when removing the time margin constraints, both algorithms of course converge to the same optimum.

Another interesting result from Table 10.1 is that the same exploration algorithm is roughly 25% slower when using another free-space partition with the same number of cells, but for which the geometry is not as well suited to describe the underlying driving decisions. This effect becomes much more dramatic when increasing the number of time steps, as shown in Table 10.2 for 20 time steps of 0.5 s.

Note that the computation times reported in Tables 10.1 and 10.2 are obtained using a single-thread implementation; due to the highly parallel nature of the graph exploration, performance gains can be expected by splitting the exploration over multiple threads. In practice, splitting the work across 4 threads only results in a slight speedup for 10 time steps, and roughly 20% lower computation times for 20 time steps with our semantic partitioning. However, computation time decreases by roughly 60% in the case of the partitioning shown in Figure 10.1 – but is still more than 4 times higher than obtained with our semantic partitioning.

Table 10.2 – Summary of computation time for 20 time steps (average over 1000 repetitions)

Algorithm	Comp. time	Objective val.
MIQP (no margin)	26.7 ms	26.28
Graph exploration (sem. part., no margin)	142.6 ms	26.28
Graph exploration (sem. part., no margin, 4 threads)	124.3 ms	26.28
Graph exploration (other part., no margin)	1718.6 ms	26.28
Graph exploration (other part., no margin, 4 threads)	609.7 ms	26.28

10.4 Heuristics

The previous section described an exact search algorithm which is guaranteed to converge in finite (but exponential) time towards the globally optimal trajectory. However, this search may be time consuming, and the algorithm does not provide guarantees regarding real-time computability. For practical applications, heuristic exploration algorithms with bounded computation time may be preferable even though they can potentially sacrifice optimality. In this exploratory section, we present a brief overview of possible search algorithms; their implementation and evaluation is, however, out of the scope of this thesis but can serve as leads to be pursued in future work.

Heuristic initialization A first possible approach is to provide a feasible solution to the solver in order to obtain a known upper bound for the minimum cost. This bound can help the exploration algorithm cut branches off earlier, thus reducing the total amount of calls to the QP solver. For this reason, we believe that techniques such as end-to-end imitation or reinforcement learning, which have the advantage of constant-time complexity but usually cannot be validated, could form a good complement to our decision-based approach by serving as the initializer for the exploration.

Multi-scale exploration A second approach to reduce computation time is to use multi-scale time discretization to prune dynamically infeasible branches faster. Indeed, knowing that transition (σ_i, σ_j) is (dynamically) infeasible with a time step duration τ guarantees that any transition $(\sigma_i, \sigma_k, \sigma_j)$ is infeasible[2] with a time step duration $\frac{\tau}{2}$. Similarly, knowledge of a good solution with a coarser time step can be used to initialize the search with a finer time step, as presented above.

Event-based approach A potentially promising approach, that would allow combining the advantages of both time and space discretization, is to use the continuous navigation graph instead of its discretized version, and perform event-based computation. In this case, transition times themselves would be considered as optimization variables to be optimized, possibly using gradient-based methods.

Monte-Carlo tree search and reinforcement learning Finally, building upon the success of learning agents such as Deepmind's AlphaGo [16, 19], techniques such as reinforcement learning coupled with Monte-Carlo tree search could also be evaluated to handle the most complex situations.

[2] Note, however, that adjacency relations can be modified when using a different time step.

10.5 Chapter conclusion

In this chapter, we proposed a decision-based motion planning algorithm building upon the (discrete) navigation graph presented in Chapter 9. Contrary to previous algorithm which simultaneously optimize decision-making and continuous trajectory planning using mixed-integer programming, our approach separates both problems to allow a much wider range of evaluation functions. In particular, safety metrics such as minimum time margins can be used easily; high-level considerations stemming from traffic rules, for instance to avoid overtaking from the right, can also be taken into account.

Of course, this improved expressiveness comes at a cost in terms of computational complexity. To allow practical application in real time, we proposed several possible heuristics that can be investigated in order to reduce computation time while providing high-quality – if not globally optimal – trajectories.

Conclusion of Part II

In Part II, we first showed that – provided driving decisions were dictated by external considerations – autonomous motion planning could be performed efficiently using gradient-based optimization techniques in frameworks such as model predictive control. This observation, coupled with the encouraging results from Part I, motivated our research for an efficient decision-making framework for autonomous driving.

Second, we illustrated the negative result that the mathematical notion of homotopy classes was not sufficient to properly encode driving decisions in autonomous motion planning. Indeed, contrary to cooperative motion planning where each vehicle could be assumed to roughly follow a fixed path, which allowed us to cast the problem into a two-dimensional space which did not have an explicit time dependency as all vehicles could collaborate. In the case of autonomous motion planning where the fixed-path assumption cannot be made, the search space is three-dimensional and the motion of exo-vehicles over time is prescribed; therefore, the temporal dimension of the problem cannot be abstracted. These key differences require more expressiveness than what is permitted by the usual notion of homotopy classes in order to classify possible maneuvers.

To this effect, we proposed a novel representation of driving decisions in the form of a navigation graph which generalizes the notion of homotopy classes: any collision-free trajectory can be represented by a path in this graph, and two trajectories having the same graph representation are guaranteed to be homotopic. As opposed to existing literature, we build this navigation graph upon a semantic partitioning of the collision-free space-time; we showed that this semantic partitioning leads to faster computations.

Finally, we proposed a decision-based motion planning framework, which explores the navigation graph in order to guide the optimization. Compared to existing decision-aware frameworks, for instanced based on mixed-integer programming, this graph-based approach allows much more expressiveness in the choice of cost function and constraints, at the cost of increased computation time. In order to mitigate this disadvantage, we proposed several heuristics that could be explored in future research in order to compute *high-quality* (but potentially sub-optimal) trajectories in real-time.

Another axis for future research is the generalization of our navigation graph approach to probabilistic predictions of obstacle trajectories. Indeed, most of the current framework can be used in a stochastic setting, for instance by choosing a probability threshold above which a region is considered occupied. However, this approach may prove too simplistic when longer planning horizons are considered as the existence of multiple possible behaviors for an exo-vehicle can lead to bifurcating predictions – for instance when an exo-vehicle can change lanes. To handle such scenarios, the proposed partitioning algorithm may have to be refined.

This last remark shows an important limitation of many motion planning algorithms, which often consider perfect predictions on future obstacle states that by definition cannot be obtained except in simulation. Based on this observation, Part III focuses on bridging the gap between simulations and practical implementations.

Part III

Beyond simulations: bridging the gaps

Introduction

In the first two parts of this thesis, we focused on designing cooperative (Part I) and autonomous (Part II) motion planning algorithms for automated driving. Although the proposed frameworks can accommodate – to some extent – to a stochastic environment, they still rely on assumptions that are not easily met in real-world conditions. Moreover, motion planning algorithms do not exist in a vacuum; for practical implementation, the link between perception and planning should also be considered.

The aim of Part III is to present possible approaches to bridge the gap between the simulated results of Parts I and II, and a practical implementation. As the state of the art of cooperative driving is not as advanced as that of autonomous driving, we focus on planning trajectories for a single automated vehicle.

In particular, motion planning algorithms require predictions on future obstacle trajectories which in practice are not easily obtained, and are by essence stochastic rather than deterministic. In turn, adapting existing algorithms to handle this stochasticity also constitutes an important challenge.

Sketch of Part III This shorter and somewhat more engineering-oriented part is divided in two chapters, each proposing exploratory approaches to tackle the above-mentioned challenges. In Chapter 11, we present possible methods that can be used in order to perform either stochastic or deterministic trajectory predictions for moving obstacles. In Chapter 12, we describe a real-world implementation of a simplified decision-making framework for autonomous driving on an automated vehicle prototype.

Chapter 11

Trajectory prediction

" There are many methods for predicting the future. For example, you can read horoscopes, tea leaves, tarot cards, or crystal balls. [...] Or you can put well-researched facts into sophisticated computer models, more commonly referred to as 'a complete waste of time.' "

Scott Adams (Cartoonist)

Contents

11.1 Trajectory and behavior prediction 119

11.1.1 Digression on probabilistic representations 120

11.1.2 Literature review 120

11.2 Physics-based Monte-Carlo estimation 121

11.2.1 Inputs 121

11.2.2 Monte-Carlo approach 122

11.2.3 State-machine-based behavioral model 124

11.2.4 Interaction awareness 124

11.3 A neural network approach 128

11.3.1 Data and features 128

11.3.2 Learning model 131

11.3.3 Results 132

11.4 Chapter conclusion 136

11.1 Trajectory and behavior prediction

In most situations, experienced human drivers are able to accurately infer future behaviors for the surrounding vehicles, which is critical when making tactical driving decisions such as overtaking or crossing an unsignalized intersection. Current generations of driving assistance systems such as adaptive cruise control lack these predictive capacities; as such, they usually act in a purely reactive fashion and are unable to make tactical

driving decisions which are usually left to the driver. Similarly, autonomous vehicle prototypes which lack predictive capacities have to behave very conservatively in the presence of other traffic participants; failure to properly estimate obstacle trajectories with any amount of risk-taking has already caused accidents involving self-driving car prototypes (see, *e.g.*, [123]). Therefore, reliable motion prediction of surrounding vehicles is a critical feature for safe and efficient autonomous driving.

11.1.1 Digression on probabilistic representations

Trajectory prediction for motion planning applications (such as presented in Part II) presents important challenges, some of which are inherent to the task of state estimation/prediction, while others are particular to the context of automated driving.

Fundamentally, the generic task of trajectory prediction consists in determining a probability distribution for the future state of the considered system given a sequence of past observations. However, when considering highly-maneuverable systems such as cars or pedestrians, there is a large difference between (physically) *possible*, *plausible* and *probable* future states. Although several authors may always consider worst-case scenarios (see, *e.g.*, [124]) leading to "100% safe" systems, we argue that doing so would be impractical in real-life as this would result in overly cautious and thus inefficient driving. Therefore, knowing where the probability distribution is non-zero is not sufficient for practical applications.

This consideration shows a limit in the above definition of the trajectory prediction problem, as it considers future system states as a random variable. This modeling is sensible in the context of probability theory, but falls somewhat short when applied to traffic participants who can usually be expected to behave "reasonably". Therefore, their future trajectory heavily depends on *a priori* knowledge on their behavior (including, but vastly exceeding dynamic constraints) that may be highly difficult to account for. Moreover, other traffic participants will react to the very driving decisions made by the ego-vehicle based on their own prediction of the ego-vehicle's behavior. In practice, this feedback loop may prove difficult to take into account as it involves complex equilibrium concepts that are close to those of game theory – although drivers are in fact both cooperating with, and competing against one another; an iterative approach to take such feedback into account is presented in [125].

For the above reasons, determining the actual probability density function (PDF) for future states of other traffic participants seems out of reach. For this reason, the rest of this chapter focuses on determining functions that are expected to approximate peaks and valleys of the actual PDF, but may not themselves be proper densities.

11.1.2 Literature review

Notwithstanding these considerations, many approaches to motion prediction have been proposed in the literature; a more comprehensive survey can be found in [126]. Although some of them (such as the one we propose in Section 11.2) can be parametrized by hand from expert knowledge, most of the existing techniques rely on automated learning.

As in many learning applications, these techniques can be split between *classification* or *regression* methods. When applied to motion prediction, classification problems consist in determining a high-level behavior (or *intention*), for instance *lane change left*, *lane change right* or *lane keeping* for highway driving or *turn left*, *turn right* or *go straight* in an intersection. Many techniques have already been explored for behavior prediction,

such as hidden Markov models [127, 128], Kalman filtering [129], Support Vector Machines [130, 131] or directly using a vehicle model [132]; more recently, artificial neural network approaches have also been proposed [133, 134, 135].

The main advantage of predicting behaviors is that the discrete outputs make it easier to train models and evaluate their performance. However, they only provide rough information on future vehicle states, which is not easy to use when planning a trajectory for the self-driving ego-vehicle. Some authors have proposed using a generative model, for instance Gaussian Processes [127] or neural networks [133] based upon the output of the behavior prediction, but this approach requires complex training and is only as robust as the classifier accuracy. Regression problems, on the other hand, aim at directly obtaining a prediction for future positions of the considered vehicle, which can then be used for motion planning. Many regression algorithms could be used for this problem, such as regression forests [136]; more recently, artificial neural networks have attracted the most attention in the field of trajectory prediction for cars [137, 138], cyclists [139] or pedestrians [140, 141]. A potential downside of such approaches is that the output of many regression algorithms is a single "point" (*e.g.*, a single predicted trajectory) without providing a measure of confidence. To counter this issue, well-established approaches such as Monte Carlo sampling or k-fold validation [142] can be used to provide error estimates; more recently, dropout estimation techniques have also been proposed for applications using neural networks [143]; these techniques can therefore be employed to output rough probability distributions.

11.2 Physics-based Monte-Carlo estimation

One of the major limitations to learning-based approaches to trajectory prediction is that they typically require large amounts of data recorded in "relevant" situations. Moreover, it is still unclear how well results obtained on a specific location (for instance, an instrumented intersection) can be generalized.

On the other hand, physics-based approaches that mostly rely on *a priori* knowledge on object dynamics do generalize well, but are more suited to computing possible – rather than likely – trajectories. In this section, we propose a relatively simple Monte-Carlo estimation scheme that can be used for trajectory prediction with little knowledge on the dynamic objects of the scene. Coupled with some hand-crafted parameters, this approach was successfully used in actual driving conditions for a prototype of a level 4 automated vehicle using 2D LiDAR data. Figure 11.1 shows an example of such a point cloud, and a frame from a front-facing camera showing the scene at the same time instant. Although the proposed approach can of course generalize, the rest of this section will focus on the roundabout scenario of Figure 11.1.

11.2.1 Inputs

This algorithm is aimed at being implemented easily with a minimal amount of a priori knowledge and hypotheses on the perception layer. We consider a set of dynamic objects $\mathscr{O}$ that can evolve inside a navigable region $\mathscr{R} \subset \mathbb{R}^2$. We assume that the perception layer is capable of correctly detecting and tracking object $o \in \mathscr{O}$, and to determine its type, for instance as a *car*, a *bicycle* or a *pedestrian*. As the perception layer has tracking capabilities, a history of past positions $(x_k^o, y_k^o)_k$ is available for each obstacle o, for instance locating the center of the bounding box corresponding to o in the LiDAR point cloud in a ground frame. The dimensions of each object are also supposed to be provided, as well as

(a) LiDAR point cloud (b) Camera frame

Figure 11.1 – Raw data for the estimation algorithm

its longitudinal velocity and heading; a possible approach to compute this information is to use extended Kalman filtering with a nonlinear point-mass model. Error estimates on input parameters can also be provided.

For each class of moving objects, a dynamic forward simulation model, with a corresponding set of feasible controls, is supposed to be provided from expert knowledge. To simplify the rest of this section, we only consider car-like objects following the kinematic bicycle model presented in Equation (7.9); note that a better-suited model should probably be used for other types of traffic participants such as pedestrians or trailer trucks. Finally, we assume that the drivable region $\mathscr{R}$ is known, for instance based on subtracting detected static objects from an off-line (HD) map. Figure 11.2 shows an overlay of the LiDAR point cloud on a binary representation of the navigable region, based on the estimated ego-vehicle location from the perception layer.

11.2.2 Monte-Carlo approach

The basic idea of our Monte-Carlo approach is to randomly sample feasible controls for each object, and to propagate the observed state (or an initial state sampled within the error ellipsoid around this initial state) forward in time, resulting in a possible trajectory. To filter out the most unlikely trajectories, the sequence of control is discarded if the obstacle exits its navigable region. The use of a bicycle model, where controls are a longitudinal acceleration and a steering angle, allows taking into account reasonable car dynamics. This forward simulation process is repeated a predetermined number of times for each object, and regions more frequently visited are therefore associated with a higher value of occupancy.

In practice, we represent the driving area around the ego-vehicle as a 2D grid with fixed spatial resolution, and we use a fixed discretization time step; the occupancy probability over a prediction horizon T is therefore represented by a piecewise-constant function defined over a 3D grid, where the third dimension represents time. Algorithm 6 presents the sampling algorithm used to evaluate future occupancy distribution for a single object.

The output of Algorithm 6 (applied to all dynamic obstacles) is a discrete function generally taking higher values in cells having a higher probability of being occupied at a given time. As objects are considered individually, several of them can contribute to the

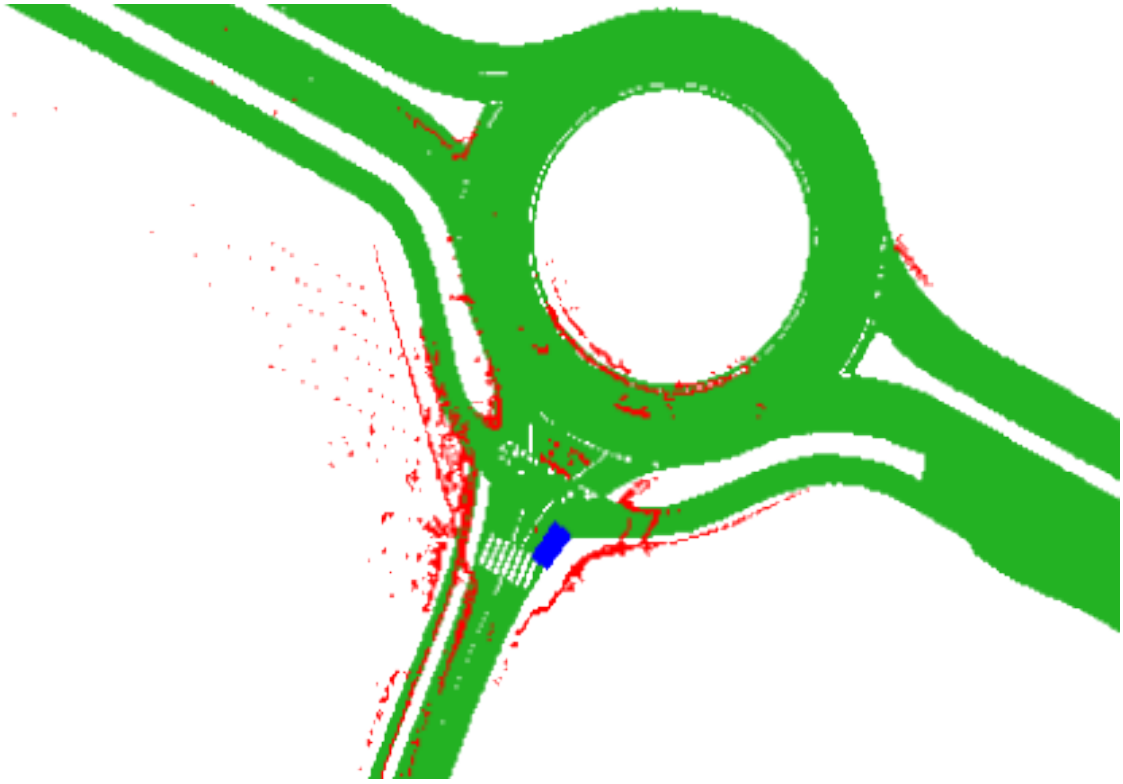


Figure 11.2 – Overlay of the point cloud of Figure 11.1 (in red) on the navigable region (in green). The location of the ego-vehicle estimated by the perception layer is shown in blue.

Algorithm 4: Trajectory prediction for obstacle o

Input: Initial state X_0 with error (covariance) matrix Σ, (static) navigability grid G, number of sampled controls N, number of prediction time steps K

Output: Probabilistic occupancy grids G^k at time steps $k = 1..K$

for $i = 1..N$ **do**

  /* Randomly sample an initial state within the uncertainty ellipsoid */

  set $\tilde{X} = \mathcal{N}(X_0, \Sigma)$

  /* NonNavigable(G,X) returns true if any point of the obstacle shape, when in state X, is in a non-navigable cell of G */

  for $k = 1..K$ **do**

    if *NonNavigable*$(G,\tilde{X})$ **then**

      continue // Abort unlikely trajectory

    foreach $(i,j) \in$ *OccupiedCells*$(G,\tilde{X})$ **do**

      /* Increment by $\frac{1}{N}$ all cells occupied by the obstacle shape when in state X at step k */

      $G^k[i,j] = \min\left(1, G^k[i,j] + \frac{1}{N}\right)$

    set u = RandomControl($\tilde{X}$) // Randomly sample a feasible control for object o

    set $\tilde{X}$ = IntegrateDynamics($\tilde{X}, u$) // Compute resulting new state

value of a single cell; the min function in the assignment is used to bound this value to 1. Although the value of a cell does not correspond to an actual probability, the rest of this thesis will refer to it using the improper term of *occupancy probability*; an illustration of such probabilities in the situation of Figure 11.1 is shown in Figure 11.3.

11.2.3 State-machine-based behavioral model

A key element that vastly conditions the quality of trajectories predicted by Algorithm 6 is the RandomControl function. Indeed, uniformly sampling the space of dynamically feasible controls will admittedly yield *possible* trajectories, but that may be highly unlikely, corresponding for instance to a vehicle cutting straight across the central island in the roundabout. Statistical or machine learning techniques can of course be used to estimate the distribution of controls actually applied by human drivers, but require potentially large amounts of training data to do so, and may be difficult to generalize to other scenarios.

Another approach, that may be less precise but possibly more generalizable is to use expert knowledge to handcraft behavioral models depending on a high-level semantic understanding of the driving situation. For instance, drivers in a roundabout will usually either turn around the central island, or move outwards to reach an exit.

The main advantage of using such maneuvers is that the probability distribution of likely controls conditioned by a maneuver choice can be hypothesized to be much simpler than in the absence of conditioning. Moreover, we conjecture that better generalization can be achieved by parameterizing controls depending on certain well-chosen semantic elements; for instance, steering angle when turning around the roundabout will mostly depend on the roundabout radius, with slight driver-dependent variations.

State machines (or finite automata) have been widely used for behavior selection, as they allow a compact representation of multiple (and possibly hierarchical) choices [144]. In this simple implementation, we use such a state machine to parametrize the RandomControl function based on the predetermined semantic considerations – in the case of Figure 11.3, we only consider whether the exo-vehicle is inside the roundabout, or in one of the straight parts of the road.

In both cases, we assume that driver inputs, *i.e.* longitudinal acceleration and steering angle, have a Gaussian distribution truncated to a minimum and maximum value. However, the average value and standard deviation of the distributions depend on the current exo-vehicle state; for instance, the steering angle has a centered distribution in straight parts of the road, which is not the case inside the roundabout. Although quite simple, this approach proves useful in situations where training data is scarce, for instance when approaching an unknown road. For this reason, we believe that developing more generic and comprehensive state-machine-based models is a promising topic for future research.

11.2.4 Interaction awareness

Another limitation of Algorithm 6 is that each object is considered individually, although interactions between traffic participants is a key aspect of driving. A possible approach to overcome this issue is to combine our simple sampling strategy with an extended notion of “priorities”.

In many situations, traffic rules avoid ambiguity by requiring one vehicle to yield to another based on high-level concepts such as priority to the right; thus, priority relations in the sense of [42] can be assigned within certain pairs of traffic participants. Other pairs

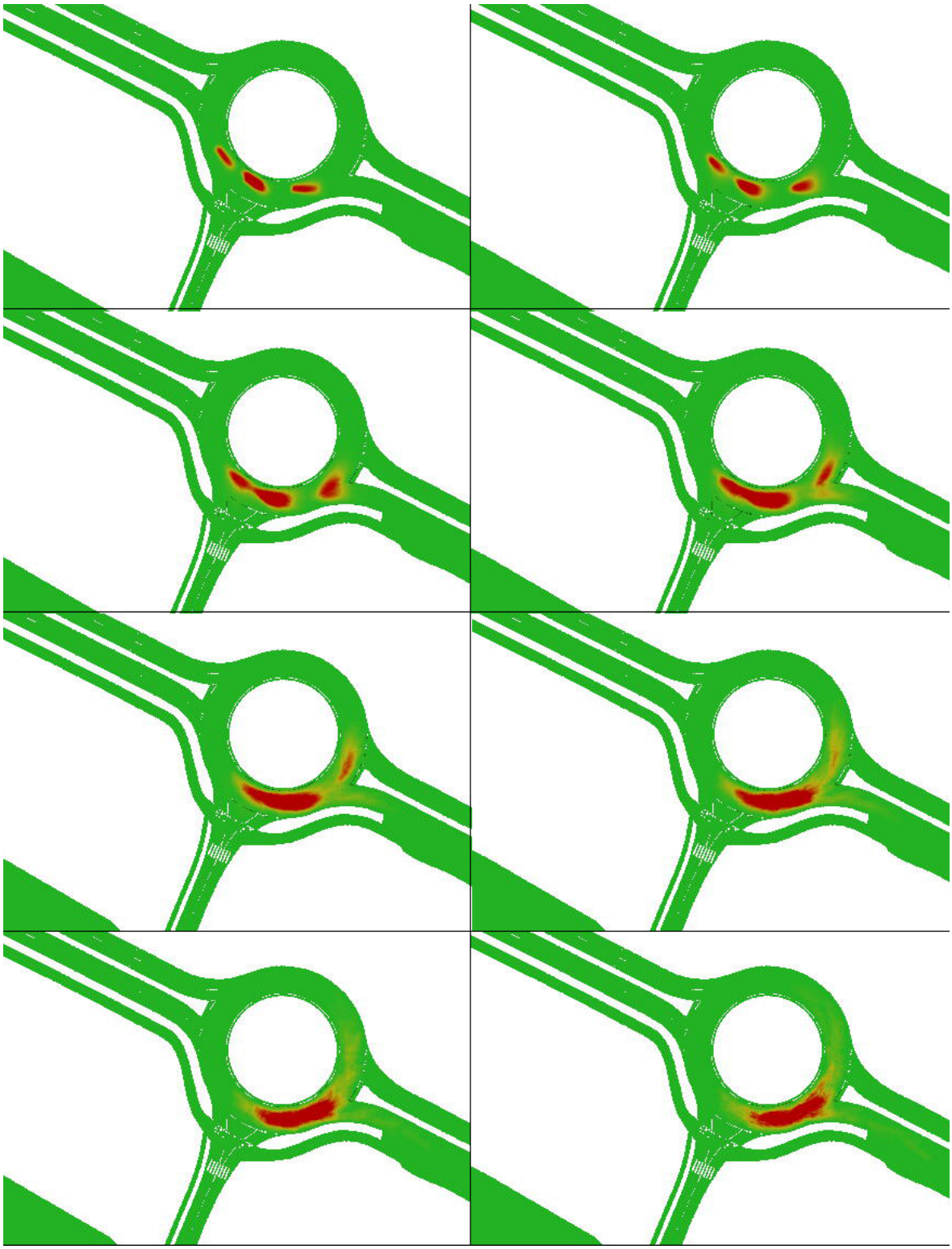


Figure 11.3 – Predicted occupancy predictions in the situation of Figure 11.1 with Algorithm 6; higher occupancy probability is shown in red. Increasing time is from left to right, and from top to bottom.

may not have explicit precedence relations, *e.g.* for two vehicles driving side-by-side on a straight road, in which case we add another "equivalent" relation $\approx$ to complete the precedence relation $\succ$.

In order to take interactions between obstacles having an equivalent priority, we propose to adapt Algorithm 6 by *simultaneously* choosing a random control for all vehicles with undetermined priority. Corresponding trajectories are discarded if they exit the navigable region, or result in a collision within two obstacles with equivalent priority.

Assuming that precedence relations in a given situation are acyclic, there exists one or several "maximum elements" corresponding to obstacles having the highest priority (which are pairwise equivalent). As these obstacles are not required to yield, but should only avoid collisions within themselves, their likely future occupancy can be computed first using this simultaneous sampling technique. This process is then repeated for obstacles having lower priority, taking the occupancy probability computed previously as a constraint, *i.e.* discarding trajectories which enter cells with non-zero occupancy probability, with the same probability. This process is described in Algorithm 5. A limitation of this joint estimation approach is that many samples risk being discarded, which may require increasing the number of samples to get sufficient coverage. This remark emphasizes even more the importance of the RandomControl function, which may mitigate this issue by considering, *e.g.*, repulsive forces between obstacles.

Algorithm 5: Interaction-aware trajectory prediction for an obstacle set $\mathcal{O}$

Input: For each objet $i \in \mathcal{O}$: initial state X_0^i with error (covariance) matrix Σ^i, (static) navigability grid G, number of sampled controls N, number of prediction time steps K

Output: Probabilistic occupancy grids G^k at time steps $k = 1..\mathrm{K}$

```
/* Sort objects by decreasing level of priority                          */
set L = SortByPriority(O)
while L ≠ ∅ do
    set i_0 = L[0]          // Get the highest priority obstacle i_0 remaining in L
    set L_eq = {i ∈ L | i ≈ i_0}  // Get the obstacles with priority equivalent to i_0
    set X_0 = (X_0^i)_{i∈L_eq}          // Consider the system of all obstacles of L_eq
    set Σ = diag(Σ^i)_{i∈L_eq}                          // Assume independant noise
    /* Randomly sample an initial system state                           */
    set X̃ = N(X_0, Σ)
    /* NonNavigable(G,X) returns true if any point of an obstacle shape, when
       the system of obstacles is in state X, is in a non-navigable cell */
    /* PairwiseCollision(X) returns true if state X corresponds to a collision
       between two objects in L_eq                                       */
    /* OccupancyProb(G^k,X) returns the maximum of the previously computed
       occupancy probability in G^k, over the cells occupied by the system of
       obstacles in state X                                              */
    for k = 1..K do
        /* Abort unlikely trajectories:                                  */
        if NonNavigable(G,X̃) then
            continue                                          // Non-navigable
        if PairwiseCollision(X̃) then
            continue          // Collision between equivalent priority obstacles
        if Rand() < OccupancyProb(G^k,X̃) then
            continue       // Probable collision with a higher-priority obstacle
        foreach (i, j) ∈ OccupiedCells(G,X̃) do
            /* Increment by 1/N all cells occupied by each obstacle shape when
               the system of obstacles is in state X at step k           */
            G^k[i, j] = min(1, G^k[i, j] + 1/N)
        set u = RandomControl(X̃)  // Randomly sample a feasible system control
        set X̃ = IntegrateDynamics(X̃, u)          // Compute resulting new state
    for i ∈ L_eq do
        remove i from L
```

11.3 A neural network approach

The previous section described a mostly hand-crafted approach to motion prediction which can be applied when little *a priori* knowledge is available. In this section, we focus on a larger scale, data-driven approach to trajectory prediction using long short-term memory (LSTM) neural networks [145], which are a particular implementation of recurrent neural networks. Because they are able to keep a memory of previous inputs, LSTMs are considered particularly efficient for time series prediction [146] and have been widely used in the past few years for pedestrian trajectory prediction [140, 141] or to predict vehicle destinations at an intersection [135, 147]. We propose an LSTM network to predict car trajectories on highways, which is notably critical for safe autonomous overtaking or lane changes, and for which very little literature exists.

A particular challenge for this problem is that highway driving usually comprises a lot of constant velocity phases with rare punctual events such as lane changes, which are therefore hard to learn correctly. For this reason, many authors rely on purposely recorded [131] or handpicked [148, 149] trajectory sets which are not representative of actual, average driving. Therefore, the real-world performance of trained models can be significantly different. A second particularity of our approach is that we train and validate our model using the entire NGSIM US101 dataset [150] without *a-priori* selection, and show that we can predict future trajectories with a satisfying average RMS error below 0.7 m (laterally) and 2.5 m s^{-1} (longitudinally) when predicting 10 s ahead.

11.3.1 Data and features

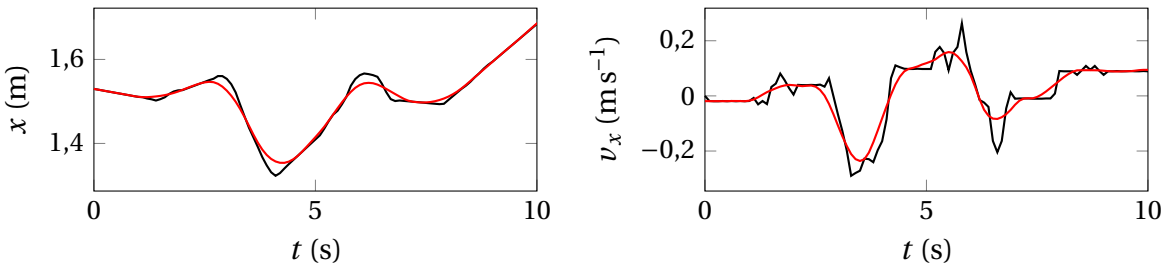
11.3.1.1 Dataset

In this chapter, we use the Next Generation Simulation (NGSIM) dataset [150], collected in 2005 by the United States Federal Highway Administration, which is one of the largest publicly available source of naturalistic driving data and, as such, has been widely studied in the literature (see, *e.g.*, [137, 133, 151, 135]). More specifically, we consider the US101 dataset which contains 45 minutes of trajectories for vehicles on the US101 highway, between 7:50 am and 8:35 am during the transition from fluid traffic to saturation at rush hour. In total, the dataset contains trajectories for more than 6000 individual vehicles, recorded at 10 Hz.

The NGSIM dataset provides vehicle trajectories in the form of (X, Y) coordinates of the front center of the vehicle in a global frame, and of local (x, y) coordinates of the same point on a road-aligned frame. Since our problem is mostly invariant by a rigid transformation, we use the local coordinates system (dataset columns 5 and 6), where x is the lateral position of the vehicle relative to the leftmost edge of the road, and y its longitudinal position. Moreover, the dataset contains each vehicle's lane identifier at every time step, as well as information on vehicle dimensions and type (motorcycle, car or truck). Finally, the data also contains the identifier of the preceding vehicle for every element in the set (when applicable).

11.3.1.2 Data preparation

One known limitation of the NGSIM set is that vehicle positioning data was obtained from video analysis, and the recorded trajectories contain a significant amount of noise [152]. Velocities, which are obtained from numerical differentiation, suffer even more from this noise. For this reason, we used a first order Savitzky-Golay filter [153] – which



(a) Lateral position

(b) Lateral speed

Figure 11.4 – Smoothing of the lateral position and speed.

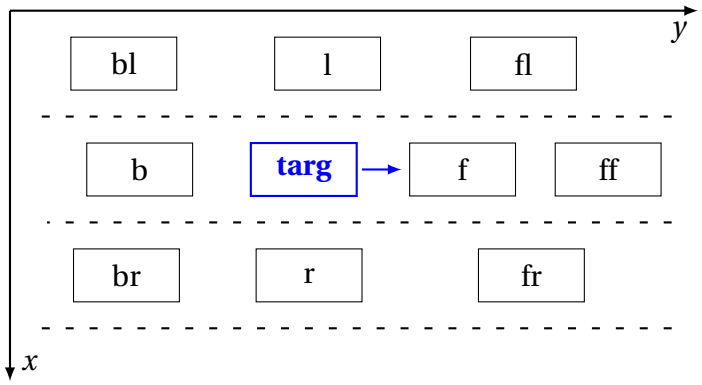


Figure 11.5 – Vehicles of interest around the target vehicle and local axis system. The blue arrow represents traffic direction.

performs well for signal differentiation – with window length 11 (corresponding to a time window of 1 s) to smooth the longitudinal and lateral positions and compute the corresponding velocities, as illustrated in Figure 11.4.

We hypothesize that the future behavior of a target vehicle can be reliably predicted by using local information on the vehicles immediately around it; a similar hypothesis was successfully tested in [154] to detect lane-change intent. For a target vehicle, we consider 9 vehicles of interest, which we label according to their relative position with respect to the target vehicle targ, as shown in Figure 11.5. By convention, we let r (respectively l) be the vehicle which is closest to the target vehicle in a different lane with $x > x_{targ}$ (respectively $x < x_{targ}$). We respectively denote by fl, f, fr and ff the vehicle preceding l, targ, r and f; similarly, vehicles bl, b and br are chosen so that their leader is respectively l, targ and r. During the data preprocessing phase, we compute the identifier of each vehicle of interest and perform *join* requests to append their information to the dataset. When such a vehicle does not exist, the corresponding data columns are set to zero.

Note that the rationale behind the inclusion of information on ff is that only observing the state of the vehicle directly in front is not always sufficient to correctly determine future traffic evolution. For instance, in a jam, knowing that vehicle ff is accelerating can help infer that f, although currently stopped, will likely accelerate in the future instead of remaining stopped. The obvious limit to increasing the number of considered vehicles is the ability to realistically gather sufficient data using onboard sensors; for this reason, we restrict the available information to these 9 vehicles.

11.3.1.3 Features

In this chapter, we aim at only using features which can be reasonably easily measured using on-board sensors such as GNSS and LiDAR, barring range or occlusion issues. For this reason, we consider a different set of features for the target vehicle (for which we want to compute the future trajectory) and for its surrounding vehicles as described above.

For the target vehicle, we define the following features:

- local lateral position x_{targ}, to account for different behaviors depending on the driving lane,
- local longitudinal position y_{targ}, to account for different behaviors when approaching the merging lane,
- lateral and longitudinal velocities $v_{x\,targ}$ and $v_{y\,targ}$,
- type (motorcycle, car or truck), encoded respectively as -1, 0 or $+1$.

For each vehicle $p \in$ {bl, b, br, l, f, r, fl, f, fr, ff}, we define the following features:

- lateral velocity $v_{x\,p}$,
- longitudinal velocity relative to $targ$: $\Delta v_{y\,p} = v_{y\,targ} - v_{y\,p}$,
- lateral distance from $targ$: $\Delta x_p = x_p - x_{targ}$,
- longitudinal distance from $targ$: $\Delta y_p = y_p - y_{targ}$,
- signed time-to-collision with $targ$: $\mathrm{TTC}_p = \frac{\Delta y_p}{\Delta v_{y\,p}}$,
- type (motorcycle, car or truck), encoded respectively as -1, 0 or $+1$.

These features are scaled to remain in an acceptable range with respect to the activation functions; in this first implementation, we simply divide longitudinal and lateral distances (expressed in SI units), as well as longitudinal velocities by 10, which results in values generally contained within $[-2,2]$. Note that in the case of missing data (*e.g.*, when the left vehicle does not exist), the corresponding values of Δ can become higher (in absolute value).

This choice of features was made to replicate the information a human driver is likely to base its decisions upon: the features from surrounding vehicles are all relative to the target vehicle, as we expect drivers to usually make decisions based on perceived distances and relative speeds rather than their values in an absolute frame. Features regarding the target vehicle's speed are given in a (road-relative) absolute frame as drivers are generally aware of speedometer information; similarly, we use road-relative positions since the driver is usually able to measure lateral distances from the side of the road visually, and knows its longitudinal position. The choice of explicitly including time-to-collision as a feature comes from the high importance of this metrics in lane-change decisions [155]; furthermore, neurosciences seem to indicate that animal and human brains heavily rely on time-to-collision estimations to perform motor tasks [156].

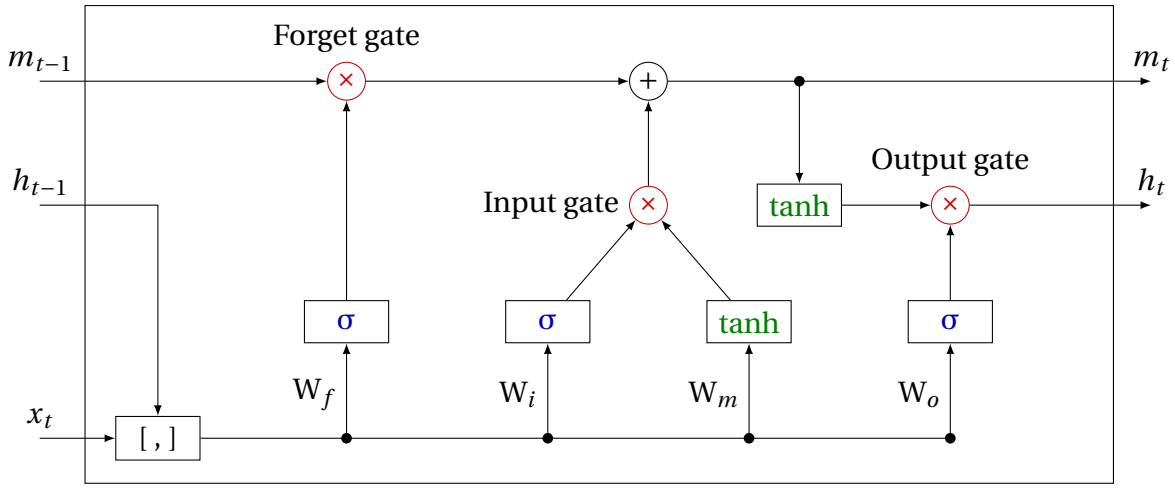


Figure 11.6 – Internal structure of an LSTM cell as used in the Keras framework. σ and tanh respectively denote a sigmoid and hyperbolic tangent activation functions; the [,] node on the lower right operates a concatenation of the new input x_t and the previous output h_{t-1}.

11.3.1.4 Outputs

Our goal is to predict the future trajectory of the target vehicle. Since the region of interest spans roughly 1 km longitudinally, the values of the longitudinal position can become quite large; for this reason, we prefer to predict future longitudinal velocities $\hat{v}_{y\,targ}$ instead. Since the lateral position is bounded, we directly use $\hat{x}_{targ}$ for the output. In order to have different horizons of prediction, we choose a vector of outputs $[\hat{x}^k_{targ}, \hat{v}_y{}^k_{targ}]_{k=1...K}$ consisting in values taken k seconds in the future.

11.3.2 Learning model

Contrary to many existing frameworks for intent or behavior prediction, which can be modeled as *classification* problems, our aim is to predict future (x, y) positions for the target vehicle, which intrinsically is a *regression* problem. Due to their success in many applications, we choose to use an artificial neural network for our learning architecture, in the form of a Long Short-Term Memory (LSTM) network [145]. LSTMs are a particular implementation of recurrent neural networks (RNN), which are particularly well suited for time series; in this implementation, we used the Keras framework [157], which implements the extended LSTM described in [146], presented in Figure 11.6. Compared to simpler *vanilla* RNN implementations, LSTMs are generally considered more robust for long time series [145]; future work will focus on comparing the performance of different RNN approaches on our particular dataset.

An interesting feature of LSTM cells is the presence of an internal state which serves as the cell's memory, denoted by m_t in Figure 11.6. Based on a new input x_t, its previous state m_{t-1} and previous output h_{t-1}, the cell performs different operations using so-called "gates":

- forget: uses the inputs to decide how much to "forget" from the cell's previous internal state m_{t-1};
- input: decides the amount of new information to be stored in a memory based on x_t and h_{t-1};

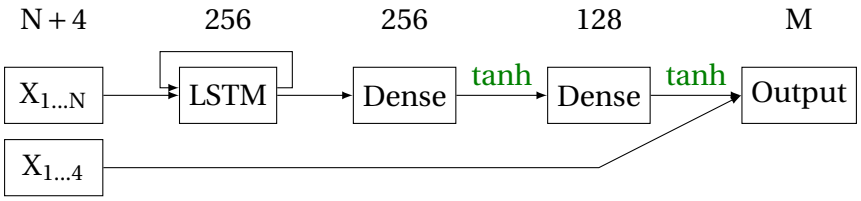


Figure 11.7 – Network architecture used as reference design. The four repeated inputs $X_{1...4}$ correspond to the current target vehicle states (positions and speeds), and are directly fed to the (dense) output layer.

- output: computes the new cell output from a mix of the previous states and output of the input gate.

This particular feature of LSTMs allows a network to learn long-term relations between features, which makes them very powerful for time series prediction.

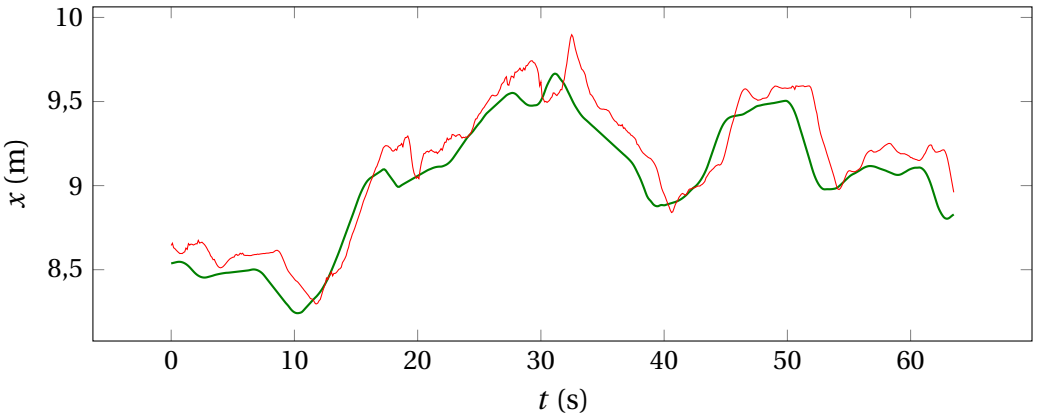
Due to their recurrent nature, even a single layer of LSTM nodes can be considered as a "deep" neural network. Although such layers may theoretically be stacked in a fashion similar to convolutional neural networks to learn higher-level features, previous studies [135] and our own experiments (see Section 11.3.3) seem to indicate that stacked layers of LSTM do not provide improvements over a single layer in our application. We use the network presented in Figure 11.7 as our reference architecture, and we compare a few variations on this design in Section 11.3.3. The reference architecture uses a first layer of 256 LSTM cells, followed by two dense (fully connected) and time-distributed layers of 256 and 128 neurons and a final dense output layer containing as many cells as the number of outputs. In this simple architecture, the role of the LSTM layer is to abstract a meaningful representation of the input time series; the two dense layers then combine these higher-level "features" in order to produce the output, in this case the predicted future states.

Additionally, the first four input features of the network – corresponding to the absolute state of the target vehicle – are repeated and directly fed to the (dense) output layer, thus bypassing the LSTMs. The motivation behind this bypass is to allow the recurrent layer to focus on variations from the current states, rather than modeling the steady state of driving at constant speed on a given lane. In practice (see Section 11.3.3), the use of this bypass seems to slightly improve prediction quality.

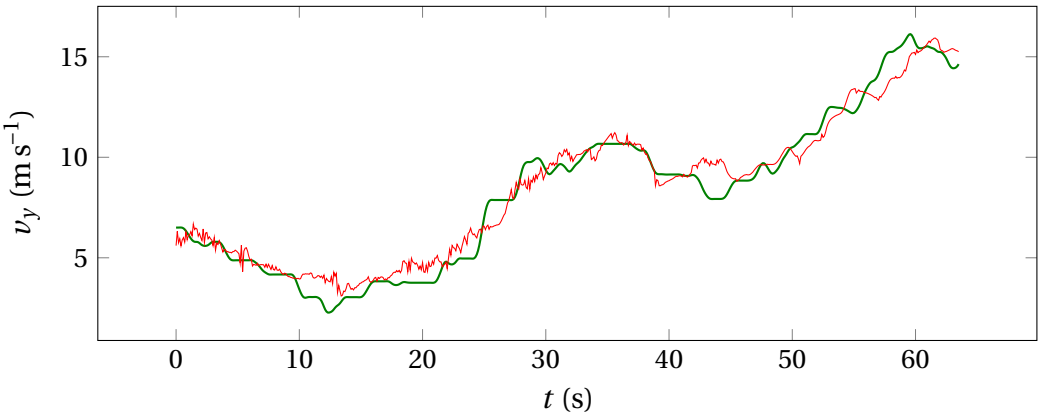
11.3.3 Results

In this section, we use the previously described deep neural network to predict future trajectories sampled from the US101 dataset. To assess the learning performance of the model and its ability to generalize over different drivers, we first randomly select 80% of vehicles (4892 trajectories) for training, and withhold the remaining 20% of vehicles (1209 trajectories) for testing; this later 20% is not used during the training phase.

In this chapter, we aim at designing a network which is capable of understanding medium-term (up to 10 s) relations for prediction. To avoid backpropagation-related issues that can arise with long time series, we trained the network using windows of 100 inputs, representing a total of 10 s past observations. One such window is taken every 10 data points; therefore, two consecutive windows have 9 s of overlap. Additionally, vehicles are grouped by batches of 500 (except for the final batch), and data is shuffled within batches. As a result, the data actually fed to the network for a batch of vehicles is a tridimensional tensor of shape $B \times 100 \times N$ where $B \approx 20000$ and $N \approx 50$ are respectively the



(a) Lateral position



(b) Longitudinal velocity

Figure 11.8 – Example of a predicted trajectory (2 s forecast) for a vehicle in the test set (in red); the thick green line corresponds to the reference.

total number of time windows in the batch, and the number of features. The training is performed on GPU using the TensorFlow backend with a batch size of 32; the model is trained for 5 epochs on each set of 500 vehicles and the whole dataset is processed 20 times, resulting in 100 effective epochs.

For the test set, we directly feed the input features for the whole trajectory, without processing the data by time windows. For each vehicle, we then compute the Root Mean Squared error (RMSE) between the network prediction and the actual expected value. In Figure 11.8, we present the prediction outputs of the network of Figure 11.7 for one of the vehicles in the test set. For comparison purposes, we tested the following variations of the reference design:

- Reference design of Figure 11.7,
- Using vehicle type information,
- Without using information on vehicle ff,
- Without using a bypass,
- Using bypass before the first dense layer (only bypass the LSTMs),
- Using a linear activation for the 128 dense layer,

Table 11.1 – RMS error for the tested models

(a) Lateral position (errors are in m)

	Prediction horizon						
Model	1 s	2 s	3 s	4 s	6 s	8 s	10 s
Reference*	**0.11**	**0.25**	**0.33**	**0.40**	0.53	0.60	0.73
With type information*	0.39	0.39	0.44	0.48	0.53	0.63	0.69
No data on vehicle ff*	0.14	0.24	0.33	0.41	0.54	0.65	0.76
No bypass	0.80	0.82	0.85	0.88	0.93	0.97	1.03
Bypass only LSTM	0.33	0.38	0.43	0.46	0.52	0.61	0.68
Linear activation	1.38	1.39	1.40	1.42	1.46	1.51	1.56
2 LSTM layers	1.25	1.26	1.28	1.29	1.33	1.37	1.41
3 dense layers*	0.34	0.38	0.44	0.50	0.59	0.70	0.72
[138]	**0.11**	0.32	0.71	not available			
Bagged	0.17	**0.25**	**0.33**	**0.40**	**0.46**	**0.57**	**0.65**

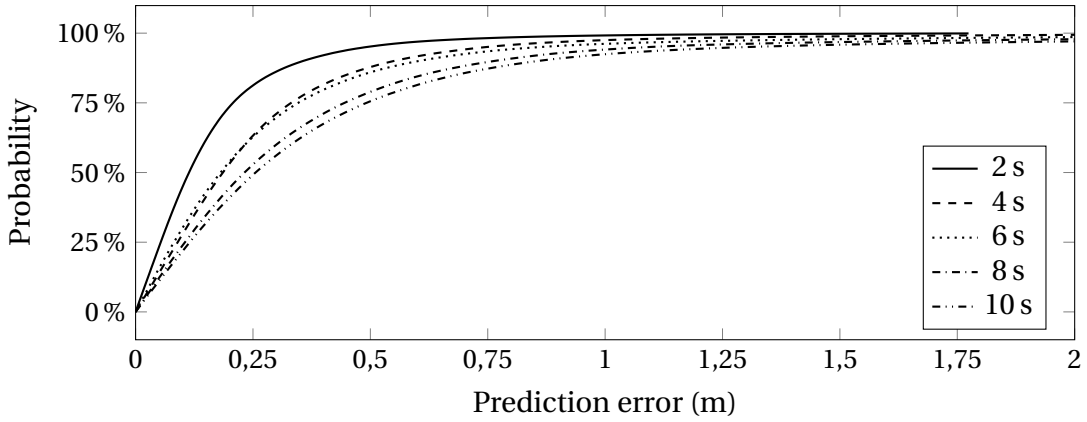
(b) Longitudinal speed (errors are in m s^{-1})

	Prediction horizon						
Model	1 s	2 s	3 s	4 s	6 s	8 s	10 s
Reference*	0.71	0.99	1.25	1.49	2.10	2.60	2.96
With type information*	0.65	0.88	1.05	**1.25**	**1.75**	**2.28**	**2.74**
No data on vehicle ff*	0.67	0.91	1.16	1.44	1.98	2.43	2.84
No bypass	1.50	1.50	1.55	1.66	2.05	2.50	2.89
Bypass only LSTM	0.78	0.90	1.06	1.26	1.76	2.30	2.78
Linear activation	0.77	1.10	1.34	1.56	2.08	2.58	2.94
2 LSTM layers	0.76	1.14	1.42	1.71	2.22	2.72	3.17
3 dense layers*	0.73	**0.87**	**1.04**	**1.25**	1.76	2.30	2.77
Bagged	**0.64**	**0.81**	**0.98**	**1.18**	**1.63**	**2.08**	**2.48**

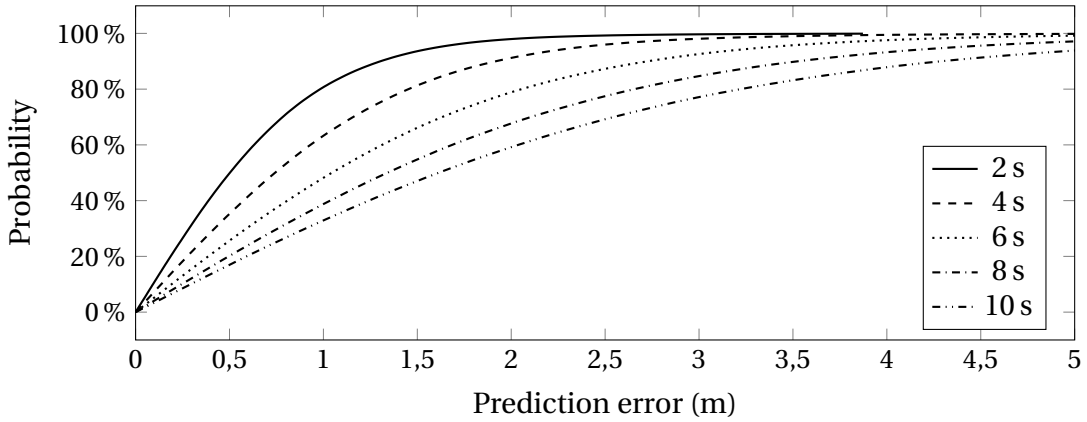
- Adding another LSTM layer after the first,
- Adding a third dense layer of 64 nodes;

Table 11.1 presents the average RMS error across all networks for various prediction horizons. In an effort to further improve accuracy, we used a light bagging technique consisting in using the average of the outputs from the four best models (denoted by a * in Table 11.1); this bagged predictor almost always perform best over the testing data. For comparison purposes, we also report results from [138] which chose a related approach using a multi-layer perceptron (which does not have a recurrent layer). The higher prediction errors for longer horizons seem to show that the use of LSTMs provides better results for longer prediction horizons.

As can be seen in Table 11.1, the architecture of Figure 11.7 provides the best overall results for lateral position prediction, but is less precise for velocity prediction. Interestingly, providing vehicle type information does not improve predictions of lateral movement but allows more precise forecasting of longitudinal speed, probably due to the difference in acceleration capacities. In what follows, we focus on this reference design to provide more



(a) Lateral position



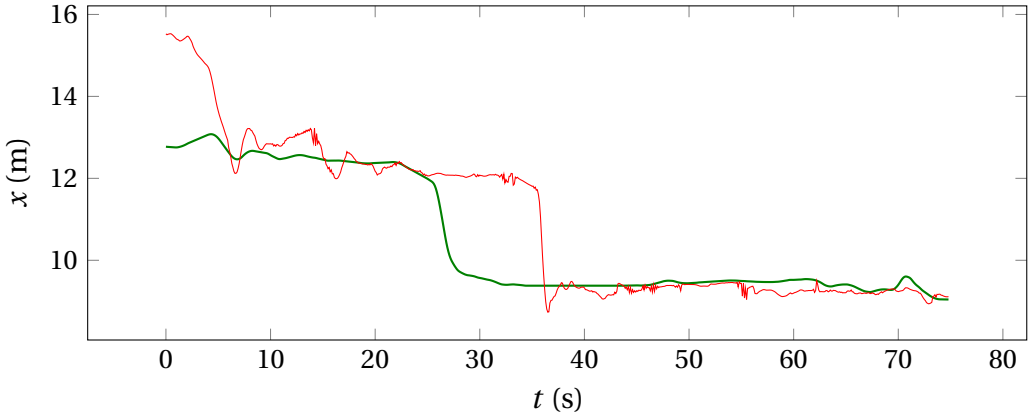
(b) Longitudinal velocity

Figure 11.9 – Distribution of error on the test set for the bagged predictor.

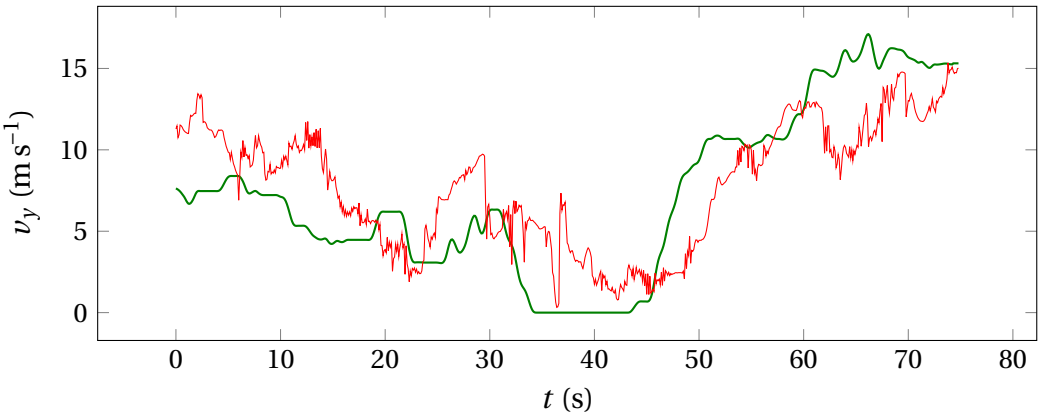
insight on error characterization. Figure 11.9 presents the distribution of prediction error over the test set for the bagged predictor.

Note that the above results mostly use the RMSE and error distributions to evaluate the quality of prediction. However, such aggregated metrics may not be the best suited for this particular application, notably due to the over-representation of sequences consisting in straight driving at constant speed, which highly outnumber discrete events such as lane changes or sudden acceleration. An illustration of this limitation is that we sometimes observe that the prediction reacts with a delay, such as shown in Figure 11.10; this effect mostly happens for longer prediction horizons, and is not properly accounted for using RMSE. In the worst cases (such as depicted in Figure 11.10), this delay can reach up to 8 s or 9 s for a prediction horizon of 10 s, thus demonstrating that the model is sometimes unable to interpret observed behaviors.

Experimentally, separately training each network output seems to yield better results, at the cost of an increased overall training time; training one model per vehicle type, or using wider networks could also be possible ways of improvement, as well as using different time windows durations for training. Besides providing an improvement to the model, future work can focus on designing better-suited metrics related to correct detection of meaningful traffic information, for instance lane changes, overtaking events or re-acceleration and braking during stop-start driving, which could help further improve predictions. Moreover, a more careful analysis of cases showing large deviations should be performed to compare model predictions with human-made estimations.



(a) Lateral position



(b) Longitudinal speed

Figure 11.10 – Delay between prediction (in red) and reference (in green, thick line) for a prediction horizon of 10 s. Note that although a large delay is observed on lateral position prediction, it is much smaller for longitudinal speed.

11.4 Chapter conclusion

In this chapter, we presented exploratory results on potential solutions to bridge the gap between perception (*i.e.*, processing raw sensor data to extract scene elements) and motion planning by predicting future behavior of traffic objects. We first presented a stochastic prediction method based on hand-coded expert knowledge and Monte-Carlo sampling, refined using an extended notion of priority to take interactions between traffic participants into account. This approach is useful in an unknown environment or when available training data is scarce, but may be too conservative.

For this reason, we proposed a second estimation technique based on deep learning, and in particular long short-term memory networks. In this approach, a recurrent neural network is trained to recognize driving patterns on recorded data, in order to infer future states of a target object. This technique has been shown to perform satisfactorily for short-term predictions (up to 2 to 3 s), but additional work is needed to perform longer-term estimations. In our opinion, the major advantage of this approach is its (theoretical) ability to provide the most likely trajectory instead of a fuzzy probability distribution. However, trajectory prediction using learning algorithms is still an open research topic, and we do acknowledge that the approach presented in this section is far from scalable. Indeed, a model trained on a finite set of data may not always be able to generalize to new situations such as outlier trajectories, or different road topology.

Interestingly, the two approaches presented in this section are almost polar opposites in strengths and weaknesses. A possible area for future research could be to combine the modeling power of machine learning with the predictive capacity of generative models.

Chapter 12

A simplified implementation: velocity planning in the real world

" In theory, theory and practice are the same. In practice, they are not. "

Apocryphally attributed to Albert Einstein

Contents

12.1 Autonomous roundabout entry 139
12.2 Decision-making for velocity planning 140
12.3 Trajectory generation 142
12.4 Experimental results 143
12.5 Chapter conclusion 146

12.1 Autonomous roundabout entry

As mentioned in the introduction to Part III, a huge gap exists between assumptions made in state-of-the-art motion planning and decision-making techniques, and the actual performance of state-of-the-art perception algorithms. Although this gap is slowly being bridged by various techniques such as those of Chapter 11, direct application of algorithms such as our decision-making motion planning framework (Chapter 10) remains out of reach.

The aim of this final chapter is to describe an approach based on the theoretical insights developed throughout this thesis for a practical implementation in the real world, on a prototype automated vehicle. The objective of the experiment was to perform real-time decision-making in order for the ego-vehicle to enter a roundabout autonomously, in the midst of an actual traffic flow and in the absence of *a-priori* data on exo-vehicle behavior.

Decision-wise, entering a roundabout is a problem quite similar to merging into a highway [14], as it requires choosing the correct time instant to move forward using only probabilistic estimations of exo-vehicle behaviors. Case-based reasoning (*i.e.*, if-then-else techniques), for instance based on gap acceptance policies, have already been proposed in this setting [158, 159], but lack scalability as they require manually programming

the behavior of the ego-vehicle in all possible situations. More recently, partially observable Markov decision processes (POMDPs) have been used for decision-making under uncertainty (see, *e.g.*, [160, 161, 162]); although they are promising in theory, these approaches require *a-priori* data to be correctly parametrized, and usually rely on abstract state representations which are impractical to use in actual applications.

In the rest of the chapter, we instead propose to leverage the theoretical knowledge that we obtained on driving decisions, in order to design a pragmatic motion planning framework capable of dealing with our roundabout scenario, while avoiding case-based reasoning.

12.2 Decision-making for velocity planning

In this experiment, the ego-vehicle is tasked with following a predetermined path which corresponds to driving in the center of the rightmost lane. This reference path includes a target velocity, which takes into account speed limitations and road curvature in order to avoid uncomfortable lateral accelerations.

With this assumption, the motion planner of the ego-vehicle is mostly tasked with performing velocity planning and trajectory generation, in order to compensate for sensor and control error.

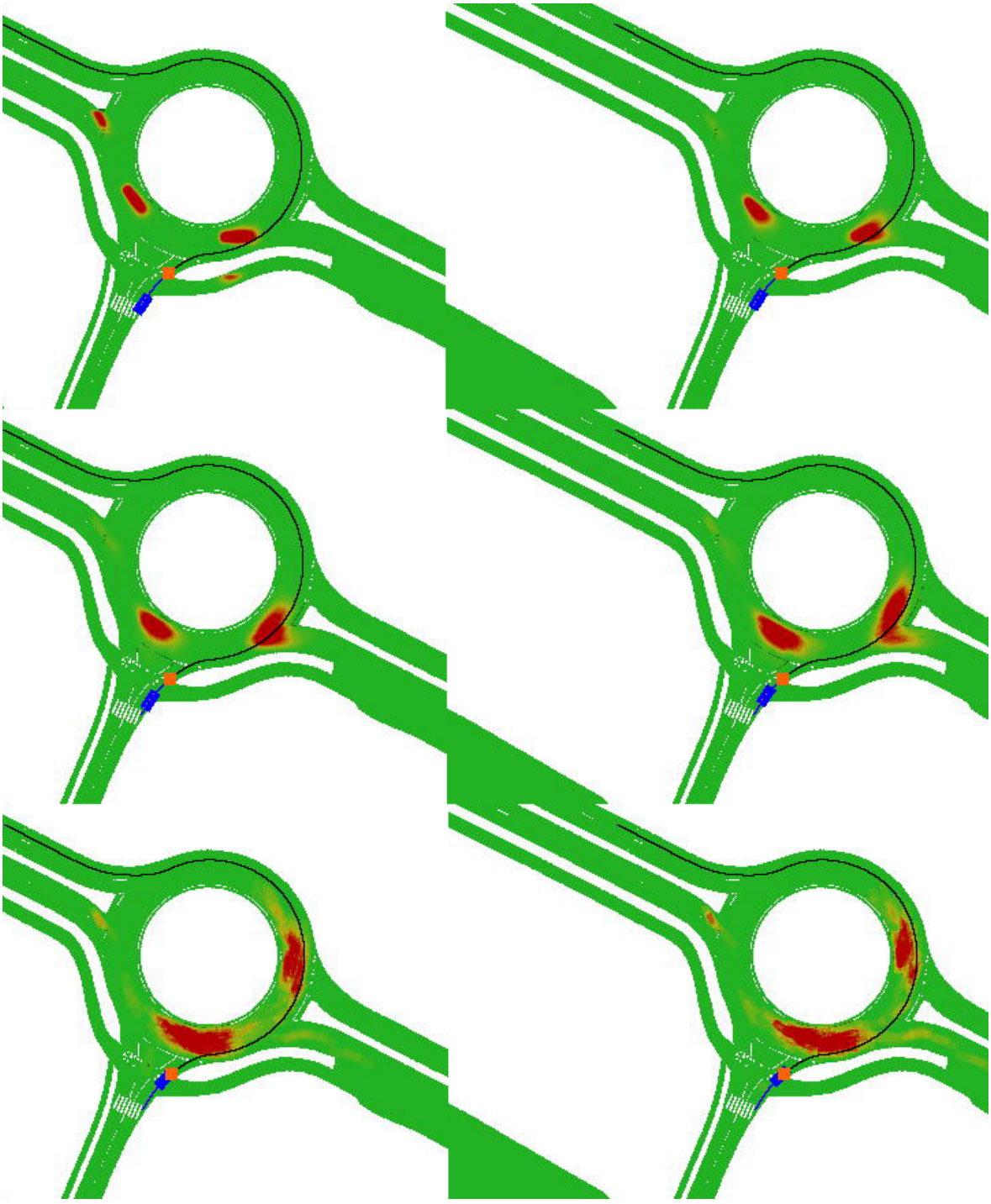
Using the (known) reference path and probabilistic occupancy predictions such as shown in Figure 11.3, it is possible to transform the distributions $p^k(x,y)$ (denoting the estimated probability of a cell (x,y) being occupied k time steps into the future) into a path-following one given as $\tilde{p}^k(s)$, denoting the estimated probability of a segment s on the reference path being occupied k time steps into the future. Figure 12.1 presents the resulting distribution, where the horizontal axis in Figure 12.1b corresponds to the space coordinate, and the vertical axis the time coordinate. Thus, the first column in Figure 12.1b corresponds to occupancy probability of the path segment currently being occupied by the ego-vehicle.

Using this path-coordinates representation, the velocity planning problem becomes that of selecting a "green diagonal" passing between obstacles. Partitioning techniques such as presented in Chapter 9 could be used, but the complex shape of the predicted occupancy regions makes this approach impractical. Instead, our implementation uses a decision-tree approach, where a finite set of *actions* $(a_i)_{i=1\dots n}$, corresponding to a choice of longitudinal acceleration, is considered. Knowing the initial position of the ego-vehicle along its path, and its longitudinal velocity, it is possible to propagate this initial state for a particular action, leading to a decision tree in the form of Figure 12.2.

By exploring this decision tree and for an arbitrary cost function, it is therefore possible to determine an optimal sequence of actions. For a sequence of actions $(a^k)_{k=1\dots \mathrm{K}}$, we propose a simple cost function of the form:

$$\mathrm{J}\left(\mathrm{X}^0, a^1, \dots, a^{\mathrm{K}}\right) = \sum_{k=1}^{\mathrm{K}} w_a {a^k}^2 + w_v \left(v^k - v_r^k\right)^2 + w_m \max\left(0, v^k - v_r^k\right)^2 + w_o \mathrm{O}^k \tag{12.1}$$

where v^k is the longitudinal speed of the ego-vehicle at the beginning of step k, v_r^k the reference speed at the point occupied at step k, and w_a, w_v, w_m and w_o are positive weights respectively penalizing longitudinal acceleration, deviation from the reference speed, exceeding the speed limit and performing a risky maneuver. The term O^k corresponds to the estimated probability of the ego-vehicle colliding with another obstacle between step



(a) Cartesian (x, y) coordinates; the black line represents the reference path, and the orange square corresponds to the planned stopping point of the ego-vehicle.

(b) Path coordinates; the horizontal axis corresponds to the spatial dimension, and the vertical axis to the time.

Figure 12.1 – Representations of occupancy predictions in cartesian and in path coordinates.

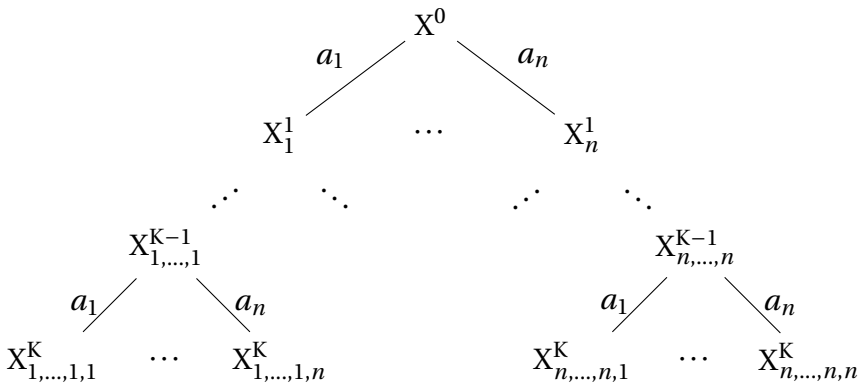


Figure 12.2 – Decision tree for the velocity planning problem. Each node corresponds to a state given by a longitudinal position and velocity; edges correspond to possible levels of longitudinal accelerations.

k and $k+1$; in practice, we choose:

$$O^k = \max_{s^k \leq s \leq s^{k+1}} p^k(s).$$

Using this cost function with a depth-first exploration of the decision tree, it is therefore possible to find the best possible decision in a given situation. An important difficulty of this approach, however, is that the number of branches to explore grows exponentially with the depth of the tree, *i.e.* the number of future time steps considered.

In practice, we choose $n = 6$ possible accelerations of $-5\,\mathrm{m\,s^{-2}}$ (emergency braking), $-3\,\mathrm{m\,s^{-2}}$ (braking), $-1\,\mathrm{m\,s^{-2}}$ (deceleration), $0\,\mathrm{m\,s^{-2}}$, $1\,\mathrm{m\,s^{-2}}$ (mild acceleration) and $3\,\mathrm{m\,s^{-2}}$ (acceleration), and a depth of $K = 9$ future actions with a time step duration of 0.5 s for a total prediction horizon of 5 s, sufficient to bring the ego-vehicle to a full stop. Therefore, the decision tree has a total of $n^K \approx 10^7$ nodes; for efficiency purposes, we therefore use branch-and-bound techniques to prune poor quality branches, allowing real-time computation at approximately 10 Hz on an on-board computer in nominal situations. However, in some occasions – notably when confronted with a seemingly inevitable collision due to the sudden appearance of an obstacle right in front of the ego-vehicle due to errors in the perception layer – exploration time can reach up to 1 s.

A possible way around this particular issue is to fully explore the *emergency braking* branch first, and abort exploration after a maximum time has elapsed, returning the best solution found so far. More generally, many methods could be experimented with in order to reduce computation time spent in the tree search, in particular those presented at the end of Chapter 10. Additionally, heuristically pre-ordering the actions list (*e.g.*, first exploring branches corresponding to braking if the ego-vehicle expects to stop) can be used to further improve performance.

12.3 Trajectory generation

The output of the decision-making process is a list of actions that can be mapped to target future states using the reference path. We then use a classical Model Predictive Control approach such as presented in Chapter 7 to compute a smooth trajectory and a sequence of controls (longitudinal acceleration and steering rate) to be fed to the vehicle. Note that in order to stabilize the vehicle despite sensing and control uncertainty, the reference waypoints for the MPC are computed starting from the measured ego-vehicle

state, and progressively return towards the reference path. As stated in Chapter 7, the trajectory generator runs in a few milliseconds, which theoretically allows replanning at the same rate as the sensors. However, in our first implementation, the trajectory generator runs at the same rate of the decision-making algorithm.

12.4 Experimental results

The presented algorithm has been extensively tested in simulation using the proprietary environment, sensor and vehicle simulator 4D Virtualiz [163], as well as on real-time data from the prototype vehicle. During real-world testing, a human driver was piloting the vehicle normally, and sensor data were fed to the motion planning software in real-time. The corresponding driving decisions and planned trajectories were recorded for later comparison, but were not communicated to the driver.

Simulated data The proposed decision-making and motion planning framework was first evaluated in a simulated environment built from a georeferenced, high-definition LiDAR scan of the testing area. Besides handling the ego-vehicle dynamics, the software simulates GPS and IMU data, as well as camera frames and laser scans for the sensors used in the actual prototype vehicle. Therefore, the perception and motion planning algorithms use the same sort of raw data, although precise noise profiles are not taken into account.

In the simulation, the output from the MPC trajectory planner is a list of timestamped target future states for the ego-vehicle, alongside with the corresponding controls (longitudinal acceleration and steering rate). A low-level, open-loop controller working at a higher refresh rate is used to integrate these commands and send them as a longitudinal velocity and steering angle to the vehicle.

Figure 12.3 presents snapshots from the simulator in a roundabout entering scenario, with estimated future occupancy shown as color gradients. In the first row, the ego-vehicle (in black, circled in blue) approaches the busy roundabout from the east and decides to wait until the car circled in red has passed. In the second and third rows, the ego-vehicle is still waiting for incoming vehicles (circled in red and white) to clear the way before entering; on the second column of the third row, the white car (circled in cyan) is detected as exiting the roundabout (no more predicted occupancy), which prompts the ego-vehicle to pass and then accelerate sufficiently to avoid rear-end collision from the vehicle circled in green.

Overall, the behavior of the system on simulated data is satisfactory, although slightly conservative. A video of some simulated scenarios (including that of Figure 12.3) is available online[1].

Actual sensor data During the testing on actual sensor data, the controller is disabled and a human driver is in charge of maneuvering the vehicle. In this experiment, real-time sensor data is sent to the perception and planning software, and the corresponding decisions and target states are recorded. This information is not communicated to the driver who drives normally in the right lane along the reference path; the quality of the motion planner is evaluated as the difference between actual driver inputs and planned trajectory.

[1] https://youtu.be/WsaHglp3XXo

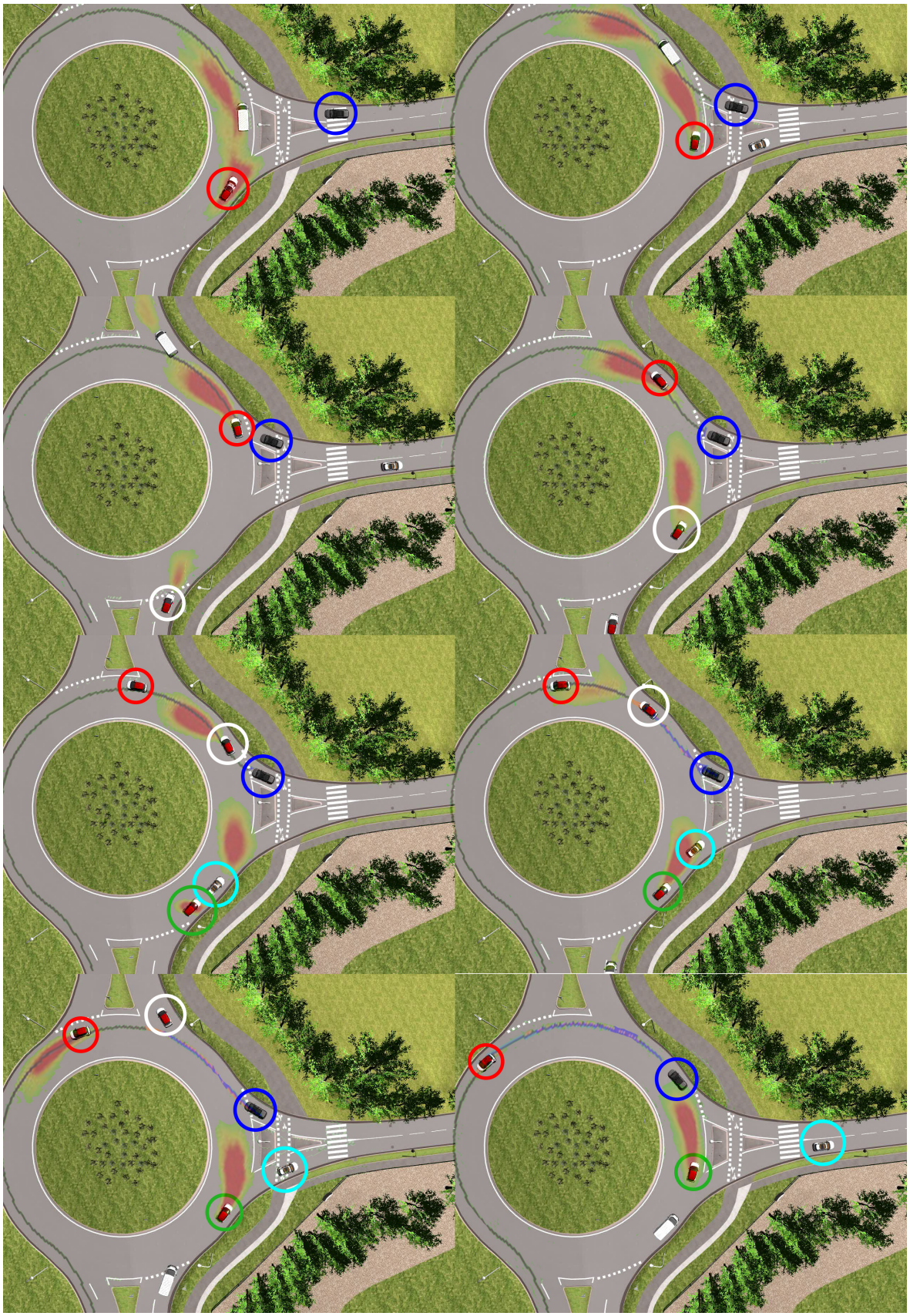


Figure 12.3 – Simulation results, with the ego-vehicle circled in blue; increasing time is from left to right and top to bottom. Color gradients represent estimated future occupancy.

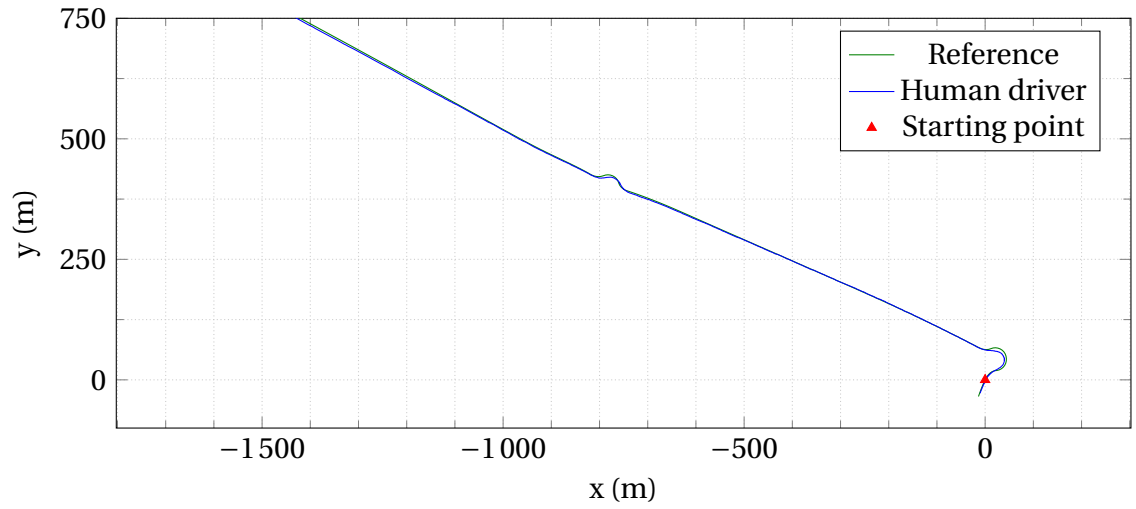


Figure 12.4 – Reference path (in green) and actual driver path (in blue) as returned by the GPS.

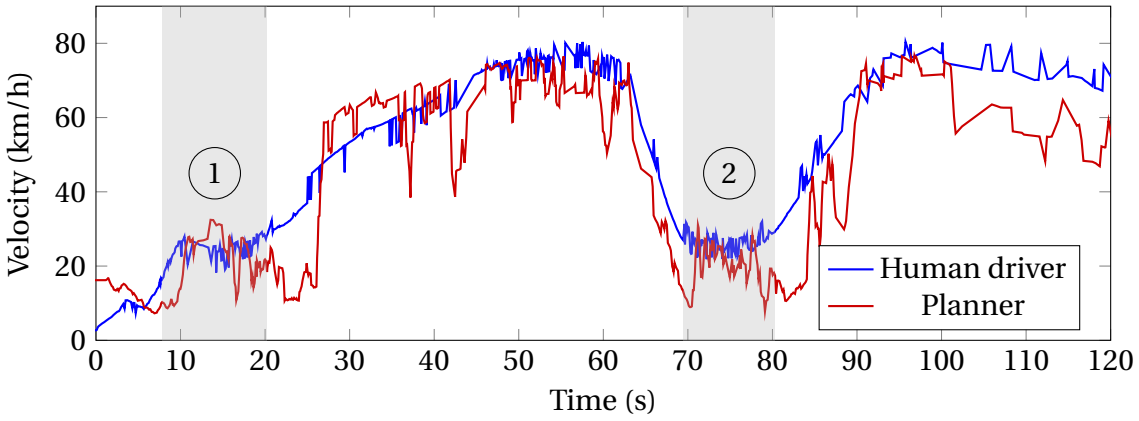


Figure 12.5 – Comparison of human driver input (longitudinal velocity) and output from the velocity planner. The location of the roundabouts is indicated with gray shading, and the horizontal dotted line represents the maximum velocity for the planner.

The test drive is illustrated in Figure 12.4, and starts with the vehicle stopped a few meters before the entrance of a first roundabout, which is used as the origin of the Cartesian frame. The ego-vehicle then turns left and proceeds to drive on a roughly straight road for approximately 800 m before entering a second roundabout, where it goes straight. Note that localization is provided by an inertial-aided differential GPS, with a precision of roughly 0.5 m.

Figure 12.5 presents a comparison between the actual longitudinal velocity chosen by the human driver, and the target velocity computed by our motion planner for the same timestamp; the gray regions correspond to the two roundabouts.

Overall, we observe quite a good correlation between the human driver's actions and those suggested by our planner, with a root-mean-square (RMS) error of approximately 10.5 km h^{-1}. In particular, the driver and the motion planner took the decision to actually enter both roundabouts at roughly the same time, although the planner is more cautious and uses a lower speed. However, many occurrences of sudden drops of the planned velocity can be observed in straight lines, for instance around 23 s, 37 s or 42 s. These drops correspond to false-positive detections of objects on the road, which are interpreted as obstacles appearing in front of the ego-vehicle. Similarly, the planned velocity is noticeably lower than that executed by the human driver starting from the 100 s mark; this effect

is due to the important lateral positioning error of the ego-vehicle of approximately 2 m, causing vegetation on the side of the road to be categorized as an obstacle. Improvements to ego-localization and object detection and tracking is expected to result in much better performance of the motion planner.

12.5 Chapter conclusion

In this final chapter, we leveraged the results obtained throughout this thesis to design a decision-making and trajectory planning algorithm capable of operating in real-world conditions using actual sensor data. By casting the motion planning problem into a simpler velocity planning one, we proposed a basic tree-based approach to make tactical driving decisions such as entering a roundabout. Although this implementation has downsides, notably that tree exploration time is not deterministic, we argue that it serves as a proof-of-concept of the advantages of incorporating an explicit decision phase during motion planning.

Conclusion and perspectives

Conclusion

In this thesis, we proposed a *decision-based* approach to the problem of planning motion for cooperative or autonomous vehicles in a structured environment such as a road network. Indeed, one of the main challenges of motion planning in these settings is the handling of obstacles that can be either static (*e.g.*, a stone fallen on the road) or dynamic, such as other traffic participants.

Geometrically, any collision-free trajectory is known to correspond to a discrete *class* depending on the way it avoids obstacles, which is linked to the topological notion of homotopy. In this thesis, we built upon this knowledge to generalize the notion of homotopy classes in order to encode *driving decisions* for motion planning. By explicitly formulating the decision-making aspect of the problem, we proposed new techniques to handle real-time cooperative or autonomous motion planning.

In Part I, we studied the problem of cooperative motion planning, consisting in coordinating multiple vehicles through critical parts of the road infrastructure such as intersections or roundabouts in order to avoid collisions while increasing efficiency over legacy traffic management schemes such as traffic lights. Using the notion of *priority*, corresponding to the relative order in which conflicting vehicles should cross, we showed that an optimal coordination could be computed by mixed-integer programming techniques in near real-time for a reasonable number of vehicles. Performance-wise, we hypothesize that our use of binary decision variables to encode priority relations provides the solver with enough structural information on the problem to quickly converge to a solution despite a theoretical complexity scaling exponentially in the square of the number of vehicles. Another advantage of our formulation is its ability to handle a wide range of cost functions, which allows applications both in fully and partially automated driving. To showcase this advantage, we introduced a new *supervised driving* algorithm capable of seamlessly correcting driver errors to avoid traffic accidents. In doing so, we demonstrated the important theoretical result – usually taken for granted – that the existence of a collision-free (system) trajectory over a sufficiently long time horizon actually ensures that the system is in a *safe* state, *i.e.* that there exists at least one infinite horizon, collision-free trajectory. The demonstration provides two bounds on the minimum time horizon that should be used to provide these guarantees.

In Part II, we shifted our focus to the task of *autonomous* motion planning, consisting in generating a "good", collision-free trajectory for an automated ego-vehicle. By proposing a framework for near-limits motion planning, we first demonstrated that generating efficient trajectories even in high dynamics situations was a fairly simple problem using state-of-the-art techniques such as model predictive control, provided driving decisions have been made beforehand. This observation motivated our study of the driving decisions involved in autonomous motion planning, which somewhat differ from those of cooperative planning in that the temporal aspects of the problem add further challenges. In particular, we showed that the notion of homotopy classes was too restrictive to be actually useful in autonomous motion planning where a receding time horizon is usu-

ally considered. Instead, we introduced a so-called *navigation graph* which generalizes the notion of homotopy classes by explicitly taking the time dimension into account. We then presented a decision-making algorithm based on this graph representation, which allows using a much wider range of cost functions and quality criteria for the generated trajectories compared to the previous state-of-the-art. Moreover, we showed that using this graph-based approach to represent semantically meaningful driving decisions – instead of arbitrary geometric considerations – led to significantly improved performance, thus further highlighting the relevance of explicit decision-making in motion planning.

Finally, in Part III, we briefly studied the interface between perception and motion planning for autonomous driving. In particular, many planning algorithms rely on the assumption that perfect knowledge on the future behavior of other traffic participants is provided, which is obviously not the case in practice. From this remark, we first described a pragmatic approach to trajectory prediction in the absence of available calibration data based on objects physics and hand-encoded behaviors. In scenarios where training data were available, we proposed a machine learning algorithm based on recurrent neural networks to predict future vehicle trajectories based on past observed behavior, with satisfying results up to a few seconds ahead. Finally, we described a real-world implementation of the ideas presented in this thesis in order to perform decision-making and motion planning for a prototype vehicle in real traffic conditions, in this case to drive through a roundabout autonomously. By constraining the trajectory to roughly follow a predetermined path, we proposed a decision-based motion planner providing satisfactory results on simulated and actual sensor data in regard to the quality of the perception layer.

Perspectives

The results presented in this thesis open many promising areas for future research, in order to effectively bring these ideas into a fully autonomous vehicle.

Dealing with uncertainty

From a theoretical standpoint, we hope that the proposed navigation graph approach could serve as a basis to build a good representation of the extremely wide range of possible driving decisions to be considered. Although the current construction of the graph can accommodate uncertainty to a certain extent (*e.g.*, by setting various thresholds on occupancy probability), additional research is needed to build an equivalent graph from probabilistic predictions such as those proposed in Part III. Moreover, representing unknown elements such as possible objects coming from a blind corner is also problematic in the current formulation.

Efficiency of exploration

The second major axis for improvement is related to efficiently making the driving decision, *i.e.* explore the navigation (or related) graph in real-time. Although we proposed a convenient way to enumerate and represent these decisions, the motion planning problem still remains NP-hard and has exponential complexity incompatible with the real-time requirements of autonomous driving that will not vanish from the simple increase of available computational power. Recent advances in artificial intelligence, notably through deep and reinforcement learning (alongside with related techniques such as Monte-Carlo tree search), seem a promising avenue to drastically reduce computation time while maintaining high quality solutions.

Combining approaches

As mentioned in the introduction, the new trend of *end-to-end* learning is emerging in many scientific communities, including motion planning. These approaches consist in using machine learning techniques to determine a direct mapping from raw sensor data (typically, camera frames) to driving actions (*e.g.*, action on the steering wheel), either by imitating a human driver or by trial-and-error through simulation. Although they are popular, applying these these methods to actual production vehicles seems delicate as they operate as "black boxes" and may behave catastrophically in case of failure. Nevertheless, we believe that end-to-end techniques could be beneficially combined with some of the concepts presented in this thesis, as they could for instance be used as a heuristic initialization for a decision-making algorithm in order to provide a form of *intuition* which is usually specific to humans.

Bibliography

[1] J. Dargay, D. Gately, and M. Sommer, "Vehicle Ownership and Income Growth, Worldwide: 1960-2030," *The Energy Journal*, vol. 28, no. 4, pp. 143–170, 2007. Cited on page 3.

[2] National Highway Traffic Safety Administration, "Summary of motor vehicle crashes (Final edition): 2015 data," tech. rep., National Center for Statistics and Analysis, Washington, DC, October 2017. Cited on page 4.

[3] S. Singh, "Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey," Tech. Rep. DOT HS 812 115, National Highway Traffic Safety Administration, February 2015. Cited on page 4.

[4] N. Kalra and S. M. Paddock, "Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?," *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, dec 2016. Cited on page 4.

[5] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, jul 2015. Cited on page 4.

[6] J. Anderson, N. Kalra, K. Stanley, P. Sorensen, C. Samaras, and O. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*, vol. 1. RAND Corporation, 2016. Cited on pages 4 and 5.

[7] J. Ploeg, A. F. Serrarens, and G. J. Heijenk, "Connect & drive: design and evaluation of cooperative adaptive cruise control for congestion reduction," *Journal of Modern Transportation*, vol. 19, no. 3, pp. 207–213, 2011. Cited on page 4.

[8] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008. Cited on page 4.

[9] T. Litman, "Autonomous vehicle implementation predictions," *Victoria Transport Policy Institute*, vol. 28, 2014. Cited on page 5.

[10] SAE On-Road Automated Driving Committee, "SAE J3016. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," tech. rep., SAE International, 2016. Cited on page 5.

[11] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected Vehicles: Solutions and Challenges," *IEEE Internet of Things Journal*, vol. 1, pp. 289–299, aug 2014. Cited on page 6.

[12] O. Gehring and H. Fritz, "Practical results of a longitudinal control concept for truck platooning with vehicle to vehicle communication," in *Proceedings of Conference on Intelligent Transportation Systems*, pp. 117–122, IEEE, 1998. Cited on page 6.

[13] R. Naumann, R. Rasche, and J. Tacken, "Managing autonomous vehicles at intersections," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 3, pp. 82–86, 1998. Cited on pages 6 and 38.

[14] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," *IEEE Intelligent Vehicles Symposium, Proceedings*, no. Iv, pp. 450–457, 2014. Cited on pages 6, 13, and 139.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012. Cited on page 6.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. Cited on pages 6 and 111.

[17] C. Hubschneider, A. Bauer, M. Weber, and J. M. Zollner, "Adding Navigation to the Equation: Turning Decisions for End-to-End Vehicle Control," *2017 IEEE International Conference on Intelligent Transportation Systems*, no. October, pp. 148–155, 2017. Cited on page 6.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. Cited on page 7.

[19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017. Cited on pages 7 and 111.

[20] A. Athalye and I. Sutskever, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017. Cited on page 7.

[21] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012. Cited on page 9.

[22] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 57, pp. 65–100, jan 2010. Cited on page 9.

[23] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416–442, 2015. Cited on page 9.

[24] T. Lozano-Perez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 100, pp. 108–120, 1983. Cited on page 10.

[25] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006. Cited on page 10.

[26] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 49–60, Oct 1987. Cited on page 11.

[27] T. Fraichard, "Dynamic trajectory planning with dynamic constraints: A'state-time space'approach," in *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 2, pp. 1393–1400, IEEE, 1993. Cited on page 12.

[28] T. Fraichard, "Trajectory planning in a dynamic workspace: a'state-time space'approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1998. Cited on page 12.

[29] P. Bender, O. S. Tas, J. Ziegler, and C. Stiller, "The Combinatorial Aspect of Motion Planning: Maneuver Variants in Structured Environments," 2015. Cited on pages 12, 13, 18, 71, 86, 88, 89, and 90.

[30] D. Gonzalez, J. Perez, V. Milanes, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2015. Cited on page 12.

[31] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, mar 2016. Cited on page 12.

[32] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996. Cited on page 12.

[33] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," techreport TR 98-11, Iowa State University, 1998. Cited on page 12.

[34] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, pp. 1–7, 2005. Cited on page 12.

[35] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE transactions on robotics*, vol. 21, no. 6, pp. 1077–1091, 2005. Cited on page 12.

[36] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 1433–1440, IEEE, 2016. Cited on page 12.

[37] V. J. Lumelsky and A. A. Stepanov, "Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorithmica*, vol. 2, no. 1, pp. 403–430, 1987. Cited on page 12.

[38] T. Howard, C. Green, and A. Kelly, "Receding horizon model-predictive control for mobile robot navigation of intricate paths," in *Field and Service Robotics*, pp. 69–78, Springer, 2010. Cited on page 13.

[39] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. Cited on pages 13 and 85.

[40] E. C. Kerrigan and J. M. Maciejowski, "Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control," in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 5, pp. 4951–4956, IEEE, 2000. Cited on page 13.

[41] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992. Cited on page 13.

[42] J. Gregoire, *Priority-based coordination of mobile robots*. PhD thesis, Mines ParisTech, 2014. Cited on pages 13, 15, 17, 23, 27, 32, 124, and C2.

[43] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015. Cited on pages 13, 14, 18, 91, 92, 93, 100, 103, and 106.

[44] J. P. Laumond, M. Taix, and P. Jacobs, "A motion planner for car-like robots based on a mixed global/local approach," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, pp. 765–773 vol.2, Jul 1990. Cited on page 13.

[45] M. G. Laidlaw and C. M. DeWitt, "Feynman functional integrals for systems of indistinguishable particles," *Physical Review D*, vol. 3, no. 6, p. 1375, 1971. Cited on page 13.

[46] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The International Journal of Robotics Research*, vol. 5, no. 3, pp. 72–89, 1986. Cited on page 14.

[47] F. Altche, X. Qian, and A. de La Fortelle, "Time-optimal coordination of mobile robots along specified paths," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5020–5026, IEEE, oct 2016. (Not cited.)

[48] F. Altché and A. de La Fortelle, "Analysis of optimal solutions to robot coordination problems to improve autonomous intersection management policies," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2016-August, pp. 86–91, IEEE, jun 2016. (Not cited.)

[49] F. Altché, X. Qian, and A. de La Fortelle, "Least restrictive and minimally deviating supervisor for Safe semi-autonomous driving at an intersection: An MIQP approach," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2520–2526, IEEE, nov 2016. (Not cited.)

[50] F. Altche, X. Qian, and A. de La Fortelle, "An Algorithm for Supervised Driving of Cooperative Semi-Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3527–3539, dec 2017. (Not cited.)

[51] F. Altché, P. Polack, and A. de La Fortelle, "A Simple Dynamic Model for Aggressive, Near-Limits Trajectory Planning," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1–6, 2017. (Not cited.)

[52] F. Altche, P. Polack, and A. de La Fortelle, "High-speed trajectory planning for autonomous vehicles using a simple dynamic model," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–7, IEEE, oct 2017. (Not cited.)

[53] F. Altche and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 2126–2133, IEEE, dec 2017. (Not cited.)

[54] F. Altche and A. de La Fortelle, "An LSTM network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 353–359, IEEE, oct 2017. (Not cited.)

[55] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 205–210, IEEE, nov 2016. Cited on pages 73, 75, 105, 106, and 108.

[56] X. Qian, F. Altché, J. Gregoire, and A. de La Fortelle, "Autonomous Intersection Management Systems: Criteria, Implementation and Evaluation," *IET Intelligent Transport Systems*, feb 2017. (Not cited.)

[57] P. Polack, F. Altche, B. D'Andrea-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 812–818, IEEE, jun 2017. (Not cited.)

[58] P. Polack, F. Altché, B. D'Andréa-Novel, and A. de La Fortelle, "Guaranteeing Consistency in a Motion Planning and Control Architecture Using a Kinematic Bicycle Model," in *Proceedings of the American Control Conference*, jun 2018. (Not cited.)

[59] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *52nd IEEE Conference on Decision and Control*, pp. 3944–3950, IEEE, dec 2013. Cited on page 23.

[60] J. Gregoire, X. Qian, E. Frazzoli, A. De La Fortelle, and T. Wongpiromsarn, "Capacity-aware backpressure traffic signal control," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, pp. 164–173, 2015. Cited on page 23.

[61] J. Peng and S. Akella, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, pp. 295–310, apr 2005. Cited on pages 23 and 38.

[62] P. Scerri, S. Owens, B. Yu, and K. Sycara, "A decentralized approach to space deconfliction," in *2007 10th International Conference on Information Fusion*, pp. 1–8, IEEE, jul 2007. Cited on page 25.

[63] M. Lizarraga and G. H. Elkaim, "Spatially deconflicted path generation for multiple UAVs in a bounded airspace," in *2008 IEEE/ION Position, Location and Navigation Symposium*, pp. 1213–1218, IEEE, 2008. Cited on page 25.

[64] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998. Cited on page 26.

[65] T. Siméon, S. Leroy, and J.-P. Laumond, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, 2002. Cited on page 26.

[66] F. Alizadeh and D. Goldfarb, "Second-Order Cone Programming," *Math. Programming - Ser. B.*, vol. 95, no. 3 – 51, pp. 3–51, 2003. Cited on page 28.

[67] Gurobi Optimization, Inc., "Gurobi optimizer reference manual," 2015. Cited on pages 28, 42, 61, and 108.

[68] A. Richards and J. How, "Mixed-integer programming for control," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 2676–2683, IEEE, 2005. Cited on page 30.

[69] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 823–839, 2013. Cited on page 38.

[70] Y. Yin, "Robust optimal traffic signal timing," *Transportation Research Part B: Methodological*, vol. 42, no. 10, pp. 911–924, 2008. Cited on page 38.

[71] D. McKenney and T. White, "Distributed and adaptive traffic signal control within a realistic traffic simulation," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 574–583, 2013. Cited on page 38.

[72] S. Faye, C. Chaudet, and I. Demeure, "A distributed algorithm for adaptive traffic lights control," *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1572–1577, 2012. Cited on page 38.

[73] C. C.-N. Pandit Kartik, Ghosal Dipak, "Adaptive Traffic Signal ControlWith Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 4, pp. 1459–1470, 2013. Cited on page 38.

[74] K. Dresner and P. Stone, "Multiagent traffic management: a reservation-based intersection control mechanism," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004.*, pp. 530–537, IEEE Computer Society, July 2004. Cited on page 38.

[75] M. Vasirani and S. Ossowski, "Evaluating Policies for Reservation-based Intersection Control," in *Proceedings of the 14th Portuguese Conference on Artificial Intelligence*, pp. 39–50, 2009. Cited on pages 38 and 44.

[76] A. de La Fortelle, "Analysis of reservation algorithms for cooperative planning at intersections," in *Proceedings of the 13th International IEEE Annual Conference on Intelligent Transportation Systems*, September 2010. Cited on page 38.

[77] A. R. Bill, "A Next-Generation Intersection Control Algorithm for Autonomous Vehicles," in *Transportation Research Board 92nd Annual Meeting*, 2013. Cited on page 38.

[78] V. Digani, M. A. Hsieh, L. Sabattini, and C. Secchi, "A Quadratic Programming approach for coordinating multi-AGV systems," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 600–605, IEEE, aug 2015. Cited on page 38.

[79] M. G. Plessen, P. F. Lima, J. Martensson, A. Bemporad, and B. Wahlberg, "Trajectory Planning Under Vehicle Dimension Constraints Using Sequential Linear Programming," in *IEEE Conference on Intelligent Transportation Systems, Proceedings*, pp. 108–113, 2017. Cited on page 38.

[80] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft Trajectory Planning with Avoidance Constraints Using Mixed-Integer Linear Programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002. Cited on page 39.

[81] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012. Cited on page 42.

[82] E. Martin, S. Shaheen, and J. Lidicker, "Impact of carsharing on household vehicle holdings: Results from north american shared-use vehicle survey," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2143, pp. 150–158, 2010. Cited on page 49.

[83] A. Colombo, "A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections," in *2014 11th International Symposium on Wireless Communications Systems (ISWCS)*, pp. 449–453, IEEE, aug 2014. Cited on pages 50, 51, 55, and 61.

[84] G. R. Campos, F. Della Rossa, and A. Colombo, "Optimal and least restrictive supervisory control: Safety verification methods for human-driven vehicles at traffic intersections," in *2015 54th IEEE Conference on Decision and Control*, pp. 1707–1712, IEEE, dec 2015. Cited on pages 50, 55, and 61.

[85] T. Fraichard and H. Asama, "Inevitable Collision States. A Step Towards Safer Robots?," *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, no. October, pp. 388–393, 2003. Cited on pages 50 and 54.

[86] R. Vasudevan, V. Shia, Yiqi Gao, R. Cervera-Navarro, R. Bajcsy, and F. Borrelli, "Safe semi-autonomous control with enhanced driver modeling," in *2012 American Control Conference*, pp. 2896–2903, IEEE, jun 2012. Cited on page 50.

[87] A. Gray, Y. Gao, J. K. Hedrick, and F. Borrelli, "Robust Predictive Control for semi-autonomous vehicles with an uncertain driver model," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, no. 1239323, pp. 208–213, IEEE, jun 2013. Cited on pages 50 and 78.

[88] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, and F. Borrelli, "Stochastic predictive control for semi-autonomous vehicles with an uncertain driver model," in *16th International IEEE Conference on Intelligent Transportation Systems*, pp. 2329–2334, IEEE, oct 2013. Cited on page 50.

[89] C. Liu, A. Gray, C. Lee, J. K. Hedrick, and J. Pan, "Nonlinear stochastic predictive control with unscented transformation for semi-autonomous vehicles," in *2014 American Control Conference*, pp. 5574–5579, IEEE, jun 2014. Cited on page 50.

[90] T.-C. Au, S. Zhang, and P. Stone, "Autonomous Intersection Management for Semi-Autonomous Vehicles," *Handbook of Transportation*, pp. 88–104, 2015. Cited on page 50.

[91] S. A. Reveliotis and E. Roszkowska, "On the Complexity of Maximally Permissive Deadlock Avoidance in Multi-Vehicle Traffic Systems," *IEEE Transactions on Automatic Control*, vol. 55, pp. 1646–1651, jul 2010. Cited on page 50.

[92] H. Ahn and D. Del Vecchio, "Semi-autonomous intersection collision avoidance through job-shop scheduling," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pp. 185–194, ACM, 2016. Cited on page 50.

[93] TASS International, "http://www.tassinternational.com/prescan." Cited on pages 61 and 79.

[94] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Towards urban driverless vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 6, no. 1/2, p. 4, 2008. Cited on page 73.

[95] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical Model Predictive Control framework for autonomous ground vehicles," *2008 American Control Conference*, pp. 3719–3724, 2008. Cited on page 73.

[96] J. Frasch, A. Gray, M. Zanon, H. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," *European Control Conference (ECC), 2013*, pp. 4136–4141, 2013. Cited on pages 73 and 80.

[97] Y. Koh, K. Yi, H. Her, and K. Kim, "A speed control race driver model with on-line driving trajectory planning," in *The Dynamics of Vehicles on Roads and Tracks: Proceedings of the 24th Symposium of the International Association for Vehicle System Dynamics (IAVSD 2015), Graz, Austria, 17-21 August 2015*, p. 67, CRC Press, 2016. Cited on pages 73, 77, and 80.

[98] D. Kim, J. Kang, and K. Yi, "Control strategy for high-speed autonomous driving in structured road," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 186–191, IEEE, oct 2011. Cited on page 74.

[99] H. Pacejka, *Tire and vehicle dynamics*. Elsevier, 2005. Cited on pages 74, D4, and D5.

[100] C. V. Altrock, "Fuzzy logic technologies in automotive engineering," in *WESCON/94. Idea/Microelectronics. Conference Record*, pp. 110–117, Sep 1994. Cited on page 74.

[101] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with Nonlinear Model Predictive Control of autonomous vehicles," in *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, IEEE, may 2014. Cited on pages 74 and 75.

[102] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza, "A Model-Predictive Motion Planner for the IARA Autonomous Car," *arXiv preprint arXiv:1611.04552*, nov 2016. Cited on pages 74 and 75.

[103] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive Active Steering Control for Autonomous Vehicle Systems," *IEEE Transactions on Control Systems Technology*, vol. 15, pp. 566–580, may 2007. Cited on page 74.

[104] J.-M. Park, D.-W. Kim, Y.-S. Yoon, H. J. Kim, and K.-S. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," *Proc. of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, 2009. Cited on page 74.

[105] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive Control of Autonomous Ground Vehicles With Obstacle Avoidance on Slippery Roads," in *ASME 2010 Dynamic Systems and Control Conference, Volume 1*, pp. 265–272, ASME, 2010. Cited on page 74.

[106] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015. Cited on page 74.

[107] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A Multi-Stage Optimization Formulation for MPC-Based Obstacle Avoidance in Autonomous Vehicles Using a LIDAR Sensor," in *ASME 2014 Dynamic Systems and Control Conference*, ASME, oct 2014. Cited on page 74.

[108] J. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *2013 American Control Conference*, pp. 188–193, IEEE, jun 2013. Cited on page 74.

[109] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1094–1099, IEEE, jun 2015. Cited on pages 75 and 80.

[110] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2/3/4, p. 265, 2005. Cited on page 78.

[111] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011. Cited on page 79.

[112] J. Kosecka, R. Blasi, C. Taylor, and J. Malik, "Vision-based lateral control of vehicles," in *Proceedings of Conference on Intelligent Transportation Systems*, vol. 46, pp. 900–905, IEEE, 1997. Cited on page 79.

[113] J. Breuer, "Analysis of driver-vehicle-interactions in an evasive manoeuvre - results of "moose test" studies," *Proceedings: International Technical Conference on the Enhanced Safety of Vehicles*, vol. 1998, pp. 620–627, 1998. Cited on page 79.

[114] A. Constantin, J. Park, and K. Iagnemma, "A margin-based approach to threat assessment for autonomous highway navigation," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, vol. 12, pp. 234–239, IEEE, jun 2014. Cited on pages 97, 106, and 107.

[115] S. Lefevre, C. Laugier, and J. Ibanez-Guzman, "Evaluating risk at road intersections by detecting conflicting intentions," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4841–4846, IEEE, oct 2012. Cited on page 106.

[116] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, pp. 895–913, jun 2011. Cited on page 106.

[117] Q. Wang, T. Weiskircher, and B. Ayalew, "Hierarchical Hybrid Predictive Control of an Autonomous Road Vehicle," in *2015 Dynamic Systems and Control Conference*, ASME, oct 2015. Cited on page 106.

[118] Q. Wang, B. Ayalew, and T. Weiskircher, "Optimal assigner decisions in a hybrid predictive control of an autonomous vehicle in public traffic," *Proceedings of the American Control Conference*, vol. 2016-July, pp. 3468–3473, 2016. Cited on page 106.

[119] M. Ono, G. Droge, H. Grip, O. Toupet, C. Scrapper, and A. Rahmani, "Road-following formation control of autonomous ground vehicles," in *2015 54th IEEE Conference on Decision and Control (CDC)*, no. Cdc, pp. 4714–4721, IEEE, dec 2015. Cited on page 106.

[120] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and Jin-Woo Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 250–256, IEEE, sep 2015. Cited on page 106.

[121] S. A. Vavasis, *Complexity theory: quadratic programming Complexity Theory: Quadratic Programming*, pp. 451–454. Boston, MA: Springer US, 2009. Cited on page 107.

[122] R. Seidel, "Small-dimensional linear programming and convex hulls made easy," *Discrete & Computational Geometry*, vol. 6, no. 1, pp. 423–434, 1991. Cited on page 108.

[123] Google, "Google self-driving car project monthly report," tech. rep., https://www.google.com/selfdrivingcar/files/reports/report-0216.pdf, Feb. 2016. Cited on page 120.

[124] G. Schildbach, M. Soppert, and F. Borrelli, "A collision avoidance system at intersections using Robust Model Predictive Control," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 233–238, IEEE, jun 2016. Cited on pages 120 and C5.

[125] J. Ziehn, M. Ruf, D. Willersinn, B. Rosenhahn, J. Beyerer, and H. Gotzig, "A tractable interaction model for trajectory planning in automated driving," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1410–1417, IEEE, nov 2016. Cited on page 120.

[126] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, p. 1, dec 2014. Cited on page 120.

[127] C. Tay, K. Mekhnacha, and C. Laugier, "Probabilistic Vehicle Motion Modeling and Risk Estimation," in *Handbook of Intelligent Vehicles*, pp. 1479–1516, Springer London, 2012. Cited on page 121.

[128] T. Streubel and K. H. Hoffmann, "Prediction of driver intended path at intersections," *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 134–139, 2014. Cited on page 121.

[129] A. Carvalho, Y. Gao, S. Lefevre, and F. Borrelli, "Stochastic predictive control of autonomous vehicles in uncertain environments," in *12th International Symposium on Advanced Vehicle Control*, 2014. Cited on page 121.

[130] H. M. Mandalia and M. D. D. Salvucci, "Using Support Vector Machines for Lane-Change Detection," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, pp. 1965–1969, sep 2005. Cited on page 121.

[131] P. Kumar, M. Perrollaz, S. Lefevre, and C. Laugier, "Learning-based approach for online lane change intention prediction," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 797–802, IEEE, jun 2013. Cited on pages 121 and 128.

[132] A. Houenou, P. Bonnifait, V. Cherfaoui, and Wen Yao, "Vehicle trajectory prediction based on motion model and maneuver recognition," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4363–4369, IEEE, nov 2013. Cited on page 121.

[133] S. Yoon and D. Kum, "The multilayer perceptron approach to lateral motion prediction of surrounding vehicles for autonomous vehicles," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2016-August, pp. 1307–1312, IEEE, jun 2016. Cited on pages 121 and 128.

[134] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2267–2272, IEEE, nov 2016. Cited on page 121.

[135] D. J. Phillips, T. A. Wheeler, and M. J. Kochenderfer, "Generalizable Intention Prediction of Human Drivers at Intersections," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1665–1670, 2017. Cited on pages 121, 128, and 132.

[136] B. Volz, H. Mielenz, R. Siegwart, and J. Nieto, "Predicting pedestrian crossing using Quantile Regression forests," in *2016 IEEE Intelligent Vehicles Symposium*, pp. 426–432, IEEE, jun 2016. Cited on page 121.

[137] R. S. Tomar and S. Verma, "Safety of Lane Change Maneuver Through A Priori Prediction of Trajectory Using Neural Networks," *Network Protocols and Algorithms*, vol. 4, no. 1, pp. 4–21, 2012. Cited on pages 121 and 128.

[138] Qiang Liu, B. Lathrop, and V. Butakov, "Vehicle lateral position prediction: A small step towards a comprehensive risk assessment system," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 667–672, IEEE, oct 2014. Cited on pages 121 and 134.

[139] S. Zernetsch, S. Kohnen, M. Goldhammer, K. Doll, and B. Sick, "Trajectory prediction of cyclists using a physical model and an artificial neural network," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 833–838, IEEE, jun 2016. Cited on page 121.

[140] Yanjie Duan, Yisheng Lv, and Fei-Yue Wang, "Travel time prediction with LSTM neural network," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1053–1058, IEEE, nov 2016. Cited on pages 121 and 128.

[141] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 961–971, IEEE, jun 2016. Cited on pages 121 and 128.

[142] T. Fushiki, "Estimation of prediction error by using k-fold cross-validation," *Statistics and Computing*, vol. 21, no. 2, pp. 137–146, 2011. Cited on page 121.

[143] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, PMLR, 20–22 Jun 2016. Cited on page 121.

[144] U. Ozguner, C. Stiller, and K. Redmill, "Systems for safety and autonomous behavior in cars: The darpa grand challenge experience," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 397–412, 2007. Cited on page 124.

[145] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. Cited on pages 128 and 131.

[146] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, pp. 2451–2471, oct 2000. Cited on pages 128 and 131.

[147] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long Short Term Memory for Driver Intent Prediction," *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1484–1489, 2017. Cited on page 128.

[148] C. Ding, W. Wang, X. Wang, and M. Baumann, "A neural network model for driver's lane-changing trajectory prediction in urban traffic flow," *Mathematical Problems in Engineering*, vol. 2013, 2013. Cited on page 128.

[149] J. Zheng, K. Suzuki, and M. Fujita, "Predicting driver's lane-changing decisions using a neural network model," *Simulation Modelling Practice and Theory*, vol. 42, pp. 73–83, 2014. Cited on page 128.

[150] U.S. Federal Highway Administration, "US Highway 101 dataset," 2005. Cited on page 128.

[151] J. Morton and T. A. Wheeler, "Project Report Deep Learning of Spatial and Temporal Features for Automotive Prediction," pp. 1–9, 2016. Cited on page 128.

[152] M. Montanino and V. Punzo, "Making ngsim data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2390, pp. 99–111, 2013. Cited on page 128.

[153] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. Cited on page 128.

[154] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-d. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 108–114, IEEE, jun 2014. Cited on page 129.

[155] J. Schlechtriemen, F. Wirthmueller, A. Wedel, G. Breuel, and K. D. Kuhnert, "When will it change the lane? A probabilistic regression approach for rarely occurring events," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2015-August, pp. 1373–1379, 2015. Cited on page 130.

[156] D. T. Field and J. P. Wann, "Perceiving time to collision activates the sensorimotor cortex," *Current Biology*, vol. 15, no. 5, pp. 453–458, 2005. Cited on page 130.

[157] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015. Cited on page 131.

[158] A. Kanaris, E. Kosmatopoulos, and P. Loannou, "Strategies and spacing requirements for lane changing and merging in automated highway systems," *IEEE Transactions on Vehicular Technology*, vol. 50, no. 6, pp. 1568–1581, 2001. Cited on page 139.

[159] J. Wei, J. M. Dolan, and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 201–207, IEEE, jun 2013. Cited on page 139.

[160] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X* (E. Frazzoli, T. Lozano-Perez, N. Roy, and D. Rus, eds.), (Berlin, Heidelberg), pp. 475–491, Springer Berlin Heidelberg, 2013. Cited on page 140.

[161] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 392–399, IEEE, oct 2014. Cited on page 140.

[162] V. Sezer, T. Bandyopadhyay, D. Rus, E. Frazzoli, and D. Hsu, "Towards autonomous navigation of unsignalized intersections under uncertainty of human driver intent," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3578–3585, IEEE, sep 2015. Cited on page 140.

[163] 4D Virtualiz, "http://www.4d-virtualiz.com." Cited on page 143.

[164] A. Richards and J. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, vol. 3, pp. 1936–1941, IEEE, 2002. Cited on page A2.

[165] D. Younger, "Minimum feedback arc sets for a directed graph," *IEEE Transactions on Circuit Theory*, vol. 10, pp. 238–245, Jun 1963. Cited on page C1.

[166] W. Zhan, C. Liu, C.-y. Chan, and M. Tomizuka, "A non-conservatively defensive strategy for urban autonomous driving," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 459–464, IEEE, nov 2016. Cited on page C6.

[167] X. Qian, J. Gregoire, F. Moutarde, and A. De La Fortelle, "Priority-based coordination of autonomous and legacy vehicles at intersection," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1166–1171, IEEE, 2014. Cited on page C6.

[168] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich, "Design of 5.9 ghz dsrc-based vehicular safety communication," *IEEE Wireless Communications*, vol. 13, pp. 36–43, oct 2006. Cited on page C6.

[169] S. Demmel, A. Lambert, D. Gruyer, A. Rakotonirainy, and E. Monacelli, "Empirical IEEE 802.11p performance evaluation on test tracks," in *2012 IEEE Intelligent Vehicles Symposium*, pp. 837–842, IEEE, jun 2012. Cited on page C6.

[170] R. Guntur and S. Sankar, "A friction circle concept for dugoff's tyre friction model," *International Journal of Vehicle Design*, vol. 1, no. 4, pp. 373–377, 1980. Cited on page D4.

[171] E. Siampis, E. Velenis, and S. Longo, "Torque Vectoring Model Predictive Control with Velocity Regulation Near the Limits of Handling," *Vehicle System Dynamics*, pp. 2553–2558, 2015. Cited on page D4.

[172] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2016-June, pp. 1433–1440, IEEE, may 2016. Cited on page D9.

[173] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Muller-Bessler, and B. Huhnke, "Up to the limits: Autonomous Audi TTS," in *2012 IEEE Intelligent Vehicles Symposium*, pp. 541–547, IEEE, jun 2012. Cited on page D10.

[174] H. Federer, "Curvature measures," *Transactions of the American Mathematical Society*, vol. 93, no. 3, pp. 418–491, 1959. Cited on page E1.

[175] C. Thäle, "50 years sets with positive reach-a survey-.," *Surveys in Mathematics & its Applications*, vol. 3, 2008. Cited on page E1.

Part IV

Appendices

Appendix A

Complements on Chapter 4

A.1 Computation of minimum bounding hexagons

We consider a set of n points $\mathbf{X} = (x_i, y_i)_{i=1\dots n}$ (considered as a $n \times 2$ matrix), and we search the minimal convex hexagon with edges parallel to the horizontal (x) or vertical (y) axis, or the diagonal $y = x$ in the same order as that illustrated in Figure 4.4 (repeated here for legibility). We note $\mathscr{H}(\mathrm{X})$ the set of such convex hexagons containing X.

We let $x_{min} = \min_{i=1\dots n}(x_i)$, $y_{min} = \min_i(y_i)$, $x_{max} = \max_i(x_i)$ and $y_{max} = \max_i(y_i)$. Moreover, we let i_1 and i_2 be such that, for all $i = 1\dots n$,

$$y_{i_1} - x_{i_1} \leq y_i - x_i \leq y_{i_2} - x_{i_2}$$

and we define $x^{\parallel} = x_{i_1} - (y_{i_1} - y_{min})$ and $y^{\parallel} = y_{i_2} - (x_{i-1} - x_{min})$. The values of x_{min}, x_{max}, $x^{\parallel}$ and their y equivalents can be computed in a single pass over the points of X.

We define vertices A to F having the following coordinates:

$$\mathrm{A} = \left(x_{min}, y_{min}\right), \qquad \mathrm{B} = \left(x^{\parallel}, y_{min}\right),$$
$$\mathrm{C} = \left(x_{max}, y_{min} + (x_{max} - x^{\parallel})\right), \qquad \mathrm{D} = \left(x_{max}, y_{max}\right)$$
$$\mathrm{E} = \left(x_{min} + (y_{max} - y^{\parallel}), y_{max}\right), \qquad \mathrm{F} = \left(x_{min}, y^{\parallel}\right),$$

and we denote by $h_m(\mathrm{X})$ the ABCDEF hexagon. We then have the following theorem:

Lemma 2. *$h_m(\mathrm{X})$ is the minimum element of $\mathscr{H}(\mathrm{X})$,* i.e. *$h_m(\mathrm{X}) \in \mathscr{H}(\mathrm{X})$ and for all $h \in \mathscr{H}(\mathrm{X}), h_m(\mathrm{X}) \subset h$.*

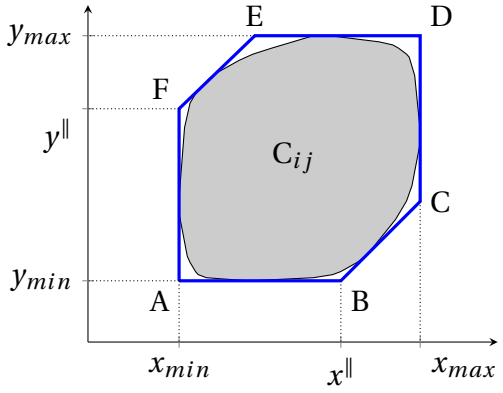


Figure 4.4 – Bounding hexagon approximation of a collision region (repeated from page 31)

Proof. First, the definition of $h_m(\mathrm{X})$ as hexagon ABCDEF trivially implies that it is a convex hexagons with the required edges slopes, since AB and DE are parallel to the horizontal axis, CD and FA to the vertical axis, and BC and EF to the $y = x$ diagonal.

Second, by definition of the various vertices of $h_m(\mathrm{X})$ and denoting e_x and e_y the canonical basis for $\mathbb{R}^2$, we know that $\mathrm{X} \subset \mathrm{A}+\mathbb{R}_+ e_x+\mathbb{R}_+ e_y$ and $\mathrm{X} \subset \mathrm{D}+\mathbb{R}_- e_x+\mathbb{R}_- e_y$, and so all points in X are contained in the rectangle defined by prolonging lines AB, CD, DE and FA. Moreover, any point $(x_i, y_i) \in \mathrm{X}$ lies above the BC line having equation $y - y_\mathrm{B} = x - x_\mathrm{B}$; indeed, we know that $y_i - x_i \geq y_{i_1} - x_{i_1}$ from the definition of i_1. Therefore, $y_i - x_i \geq y_{min} - x^{\parallel}$ and thus $y_i - y_\mathrm{B} \geq x_i - x_\mathrm{B}$. Similarly, all points of X are below the EF line, which proves that $\mathrm{X} \subset h_m(\mathrm{X})$ and thus $h_m(\mathrm{X}) \in \mathscr{H}(\mathrm{X})$.

Third, consider a hexagon $h \in \mathscr{H}(\mathrm{X})$; we denote its vertices by A' to F' in the same order as in $h_m(\mathrm{X})$. Since $h \subset \mathrm{A}' + \mathbb{R}_+ e_x + \mathbb{R}_+ e_y$, $x_{\mathrm{A}'} \leq x_{min}$ and $y_{\mathrm{A}'} \leq y_{min}$ so A' is to the bottom-left of A; similarly, $x_{\mathrm{D}'} \geq x_{max}$ and $y_{\mathrm{D}'} \geq y_{max}$, and D' is to the top-right of D. Since edges $\mathrm{A}'\mathrm{B}'$ and $\mathrm{C}'\mathrm{D}'$ are respectively horizontal and vertical, we get that $y_{\mathrm{B}'} = y_{\mathrm{A}'}$ and $x_{\mathrm{C}'} = x_{\mathrm{D}'}$. Additionally, as $(x_{i_1}, y_{i_1}) \in \mathrm{X} \subset h$, we deduce that $y_{i_1} - y_{\mathrm{B}'} \geq x_{i_1} - x_{\mathrm{B}'}$, leading to $y_{min} - y_{\mathrm{B}'} \geq x^{\parallel} - x_{\mathrm{B}'}$ *i.e.* $y_\mathrm{B} - y'_\mathrm{B} \geq x_\mathrm{B} - x_{\mathrm{B}'}$, and $x_{\mathrm{C}'} - x_{\mathrm{B}'} \geq x_{i_1} - x_{\mathrm{B}'} \geq x_\mathrm{B} - x_{\mathrm{B}'}$. As a result, B and thus C are above the $\mathrm{B}'\mathrm{C}'$ line; a symmetric reasoning applied to points E and F show that they are below the $\mathrm{E}'\mathrm{F}'$ line. Using our knowledge on the shape of h, we conclude that all the points A to F are within the convex hexagon h and therefore that $h_m(\mathrm{X}) \subset h$, thus completing the proof. □

A.2 Sub-timestep collision avoidance

Consider the collision-avoidance constraints introduced in Chapter 4:

$$\left(\pi_{ij} \wedge \neg \varepsilon^{\parallel}_{ij}(k)\right) \Rightarrow s_j^{k+1} \leq \underline{s}^{\perp}_{ji} \tag{4.7a}$$

$$\left(\pi_{ij} \wedge \varepsilon^{\parallel}_{ij}(k) \wedge \neg \varepsilon^{\perp}_{ij}(k)\right) \Rightarrow s_i^{k+1} - s^{\parallel}_{ij} \geq s_j^{k+1} - \underline{s}^{\perp}_{ji} \tag{4.7b}$$

$$\left(\pi_{ij} \wedge \varepsilon^{\parallel}_{ij}(k) \wedge \neg \varepsilon^{\perp}_{ij}(k)\right) \Rightarrow s_i^{k+1} - s^{\parallel}_{ij} \geq s_j^{k+1} - \underline{s}^{\perp}_{ji} + \frac{\tau}{2}\left(v_j^{k+1} - v_i^{k+1}\right). \tag{4.7c}$$

As mentioned when they were first introduced, slight complications have been introduced to constraints (4.7a) to (4.7c) in order to guarantee collision-avoidance within the duration of one time step. First, the subregion indicators at step $\mathbf{k}$ are used to constrain robot position at step $\mathbf{k+1}$, which we presented as a means to avoid "corner cutting" phenomena. Second, we introduced constraint (4.7c) to avoid potential collisions between robots having "follower constraints" within a time step.

A.2.1 Corner cutting

To illustrate the "corner cutting" phenomenon, we consider the case of rectangular robots on paths intersecting at straight angles; in this case, the collision region is a rectangle. We describe by "corner cutting" the fact that a trajectory with control points all outside of the collision region might still enter it between two consecutive time steps, as illustrated in Figure A.2, and encountered *e.g.* in [164]. By constraining the positions at step $k+1$, we prevent this problem from occurring while keeping to a minimum the amount of suboptimality introduced as the overhead goes to 0 as the time step duration τ decreases.

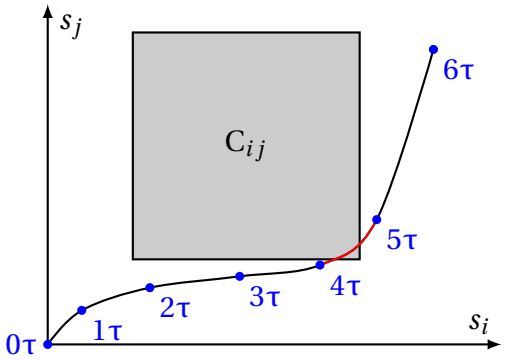


Figure A.2 – Example of corner-cutting phenomenon: all points (corresponding to successive time steps) are outside of the collision region, but the resulting trajectory intersects with C_{ij}. Blue labels indicate the corresponding time instant.

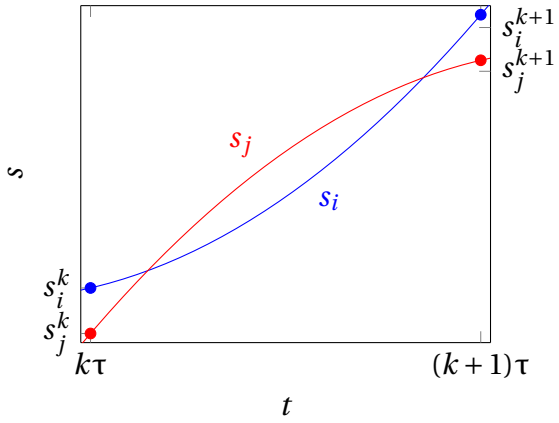


Figure A.3 – Possible sub-timestep collision between following vehicles despite collision-avoidance constraints being verified at integral time steps

A.2.2 Follower constraints

As in the case of corner cutting, collision avoidance in the case of robots following one another also requires additional constraints in discrete time. Consider two robots i and j on the same path, with i following j. In continuous time, collision avoidance requires that $s_i(t) \geq s_j(t) + d_{ji}$ for all t, where s_0 is an offset corresponding, among others, to the size of the robots. Figure A.3 illustrates that punctual verification of collision avoidance at integral time steps is not sufficient to guarantee safety between consecutive time steps.

Assuming that each robot follows a constant acceleration a_j over $[k\tau, (k+1)\tau]$ (and choosing $k = 0$ for simplicity), the inter-robot distance is

$$d(t) = s_i(t) - s_j(t) = s_i^0 - s_j^0 + \left(v_i^0 - v_j^0\right) t + \frac{1}{2}\left(a_i - a_j\right) t^2. \tag{A.2}$$

In the case $a_i \leq a_j$, d is concave and thus reaches its extrema over $[0, \tau]$ at the edges of the interval; therefore, verifying safety at integer time steps is sufficient.

We now focus on the case $a_i > a_j$: in this case, d reaches a unique minimum over $\mathbb{R}$,

$$d_m = s_i^0 - s_j^0 - \frac{1}{2}\frac{\left(v_i^0 - v_j^0\right)^2}{a_i - a_j},$$

at $t_m = -\frac{v_i^0 - v_j^0}{a_i - a_j}$, and is monotonous over $]-\infty, t_m[$ and $]t_m, +\infty[$. We distinguish three cases:

- If $v_i^0 - v_j^0 \geq 0$ then $t_m \leq 0$ and the monotonicity of d ensures that verifying safety at 0 and τ is sufficient (*i.e.* constraint (4.7b) guarantees safety).
- The same reasoning applies if $v_i^0 - v_j^0 \leq -\tau(a_i - a_j)$ *i.e.* $t_m \geq \tau$, and constraint (4.7b) guarantees safety.
- Finally, if $0 < -\frac{v_i^0 - v_j^0}{a_i - a_j} < \tau$ then $v_i^0 - v_j^0 < 0$ and thus $0 > -\frac{\left(v_i^0 - v_j^0\right)^2}{a_i - a_j} > \tau\left(v_i^0 - v_j^0\right)$,. By substituting into eq. (A.2), we deduce

$$d_m > s_i^0 - s_j^0 + \frac{1}{2}\tau\left(v_i^0 - v_j^0\right);$$

therefore, constraint (4.7c) guarantees safety.

Appendix B

Complements on Chapter 5

B.1 Helper constraints

In order to slightly reduce complexity, we propose to use additional "helper" constraints to our MILP formulation Equation (5.7). These constraints essentially arise from high-level considerations that are easily derived from the coordination problem, but may be difficult to obtain from the set of constraints of Chapter 5.

First, robots starting on the same paths have their relative priority constrained by their initial positions, which allows prescribing the value of the corresponding π variable. Second, in the situation illustrated in Figure B.1 where robot j follows i on the same path and conflict with robot k, then priority $k \succ i$ implies that $k \succ j$, *i.e.* $\pi_{ki} \Rightarrow \pi_{kj}$. Similarly, we have $\pi_{jk} \Rightarrow \pi_{ik}$.

Note that the "black-box" behavior of the solver does not allow determining which internal heuristics are used to further reduce complexity. An in-depth analysis of the structure of the problem could certainly lead to more efficient constraints being added to the formulation and lower computation time.

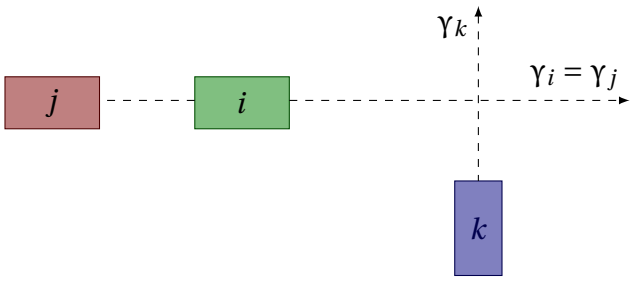


Figure B.1 – Example of a 3-robot situation where helper constraints can be used

B.2 Influence of time step duration on optimality

The discretization time step has a double influence on the solution: first, we assume constant acceleration during one time step. Second, the safety constraints require one robot of each conflicting pair to leave the conflict area one time step before the other enters.

In Figure B.2 we show the average optimality loss caused by choosing larger time step durations for a random set of 85 initial configurations of 15 vehicles. For each instance, the minimum average time t^{τ}_{opt} is computed for time step durations τ ranging from 0.125 s to 5 s. The relative loss of optimality is computed as

$$\frac{t^{\tau}_{opt} - t^{0.125}_{opt}}{t^{0.125}_{opt}}.$$

Interestingly, the averaged values fit closely to an affine function with slope 7.2% per second for the above set of parameters. The loss of optimality remains less than 6% when the time step is smaller than 1 s; moreover, the solution of (5.7) converges as the time step duration vanishes.

The kinodynamic parameters, *i.e.* the maximum speed $\overline{v}$ and the acceleration bounds $\underline{a_i}$ and $\overline{a_i}$ influence the magnitude of the loss of optimality. Figure B.3 shows a comparison of the losses of optimality for three different scenarios, namely "reference", "lower speed" and "'higher acceleration". Parameters for each scenario are presented in Table B.1.

The same set of entry times and vehicles paths is used for those three scenarios, and we only vary the kinodynamic parameters. We find that an increase of time step duration causes higher losses of optimality in instances with more dynamic robots (higher speeds or higher absolute values of acceleration bounds) than those with less dynamic ones. As a result, the time step duration can be chosen according to the dynamic characteristics of robots, *e.g.* by choosing a longer time step for slower robots to allow a more distant time horizon.

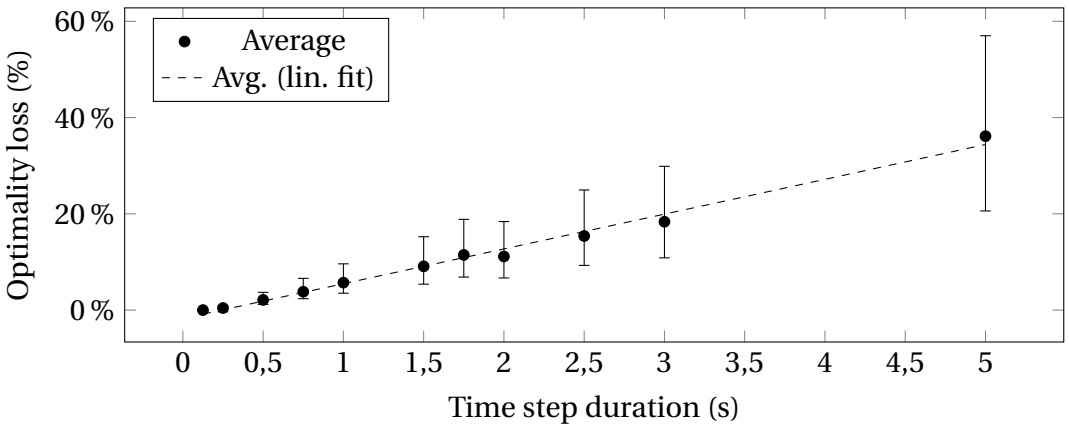


Figure B.2 – Average relative optimality loss (compared to a 0.125 s time step), depending on time step duration, for 85 instances of 15 vehicles. Error bars correspond to 1 standard deviation for instances above or below average.

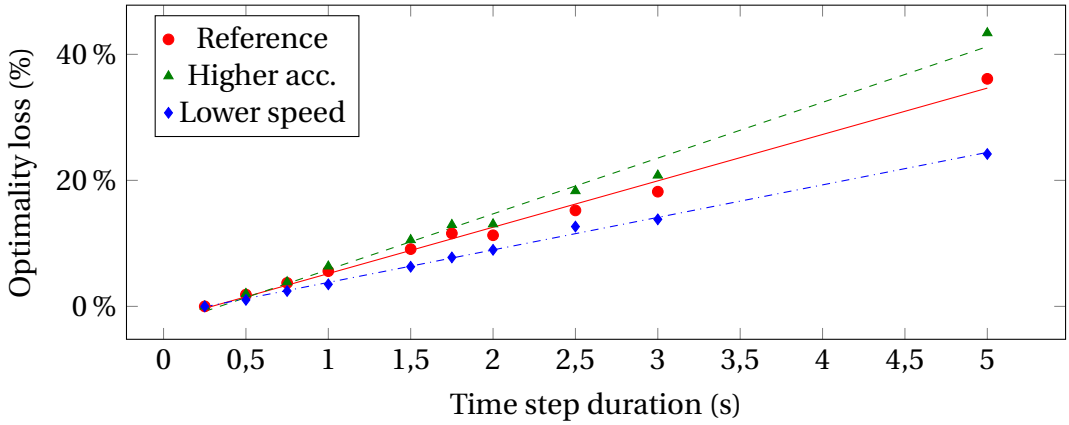


Figure B.3 – Average relative optimality loss (compared to a 0.25 s time step), depending on time step duration, for 85 instances of 15 vehicles with reference parameters (red), lower maximum speed of 10 m s^{-1} (blue) and higher absolute values of accelerations $[\underline{a_i}, \overline{a_i}] = -6\,\mathrm{m\,s^{-2}}$ to $8\,\mathrm{m\,s^{-2}}$ (green).

Table B.1 – Parameters used in the studied scenarios

Scenario	$\overline{v}$ (m s^{-1})	$\underline{a_i}$ (m s^{-2})	$\overline{a_i}$ (m s^{-2})
Reference	15	−3	+4
Higher acceleration	15	−6	+8
Lower speed	10	−3	+4

Appendix C

Complements on Chapter 6

C.1 Detailed demonstrations

C.1.1 Proofs for Section 6.3

C.1.1.1 Preliminary lemma

Before proving Theorem 3, we introduce the following lemma stemming from graph theory:

Lemma 3. *Let $\mathcal{G} = (\mathrm{V}, \mathrm{E})$ a directed graph with vertices set V and edges set E. All cycles in $\mathcal{G}$ can be removed by reversing a set of edges, each of them contributing to at least one cycle.*

Proof. The proof is based on the existence of minimum feedback arc sets [165], *i.e.* a minimum set $\mathrm{E}_{feedback} \subset \mathrm{E}$ such that $\mathcal{G}' = (\mathrm{V}, \mathrm{E} \setminus \mathrm{E}_{feedback})$ is acyclic. By minimality of $\mathrm{E}_{feedback}$, any $e \in \mathrm{E}_{feedback}$ belongs to at least one cycle of $\mathcal{G}$. Moreover, it can be seen that reversing the edges of $\mathrm{E}_{feedback}$ also leads to an acyclic graph, thus proving the lemma. □

C.1.1.2 Theorem 3

Proof of Theorem 3. Note that the only constraints requiring a vehicle to stop are constraints (4.7a) to (4.7c), forcing a vehicle j to wait for a vehicle i with $\pi^p_{ij} = 1$. From the hypotheses and theorem 2, there exists a solution $\mathbf{X}$ to IH-SP at time t_{κ_0}. We define a directed *priority graph* $\mathcal{G}_{\mathbf{X}} = (\mathrm{V}, \mathrm{E})$ with $\mathrm{V} = \mathscr{R}_{t_{\kappa_0}}$ and where an edge $i \to j$ belongs to E if there exists p such that $\pi^p_{ij} = 1$. Using this representation, a cycle in $\mathcal{G}_{\mathbf{X}}$ corresponds to a chain of q conflicting vehicles $i_1, i_2, \ldots, i_q, i_{q+1} = i_1$ for which there exists a connected component $\mathrm{C}^{p_n}_{i_n i_{n+1}}$ such that $\pi^{p_n}_{i_n i_{n+1}} = 1$ for all $n = 1 \ldots q$.

If $\mathcal{G}_{\mathbf{X}}$ is acyclic, it defines a (partial) topological order, and it is always possible to admit the vehicles one by one in that order. Therefore, there exists a feasible solution where all the vehicles of $\mathscr{R}_{t_{\kappa_0}}$ exit the supervision area in finite time.

We now assume that there exists at least one cycle in $\mathcal{G}_{\mathbf{X}}$. If all the vehicles involved in the cycle can exit in finite time, the result of the theorem is proven. Otherwise, we note $\mathscr{R}_{dead} \subset \mathscr{R}_{t_{\kappa_0}}$ a set of vehicles corresponding to a cycle in $\mathcal{G}_{\mathbf{X}}$: all of these vehicles are stopped at infinity, and are prevented to move further by a constraint of form (4.7a), for a certain $j \in \mathscr{R}_{dead}$. Moreover, the no-stop condition (6.3b) ensures that, for all $i \in \mathscr{R}_{dead}$ and all $k \geq \kappa$, $s^k_i \leq \underline{s}^{\perp}_i$.

From lemma 3, we know that it is possible to change the values of the variables π_{ij}^p for $i, j \in \mathscr{R}_{dead}$ to render $\mathscr{G}_{\mathbf{X}}$ acyclic. Using the fact that $s_i^k \leq \underline{s}_i^{\perp}$ for all of these vehicles, we know that modifying these priorities does not violate constraints (4.7a) to (4.7c). Therefore, we can build a solution $\mathbf{X}'$ for which the corresponding priority graph is acyclic, which proves the theorem as [42] has already proven that deadlocks could only occur with cyclic priority graphs. □

C.1.2 Proofs for Section 6.4

C.1.2.1 Lemma 1

Proof of Lemma 1. The proof is trivial if we consider continuous-time dynamics, as a single vehicle can always apply a control lower or equal to u_b starting from time $t_k + \tau$, which ensures it is stopped for $t \geq t_k + \tau + \frac{v_{max}}{u_b}$. A slight additional complexity happens at the final braking time step when considering piecewise-constant controls, applying u_b for a duration τ might result in a negative velocity, which is not allowed in our framework. We now proceed to the formal proof, as below.

Let (u_i^k) be the control corresponding to trajectory s_i, and let us define a control (w_i^k) as: $w_i^\kappa = u_i^\kappa$, $w_i^k = \min(u_b, u_i^k)$ for $\kappa < k < \kappa + K$, and $w_i^{\kappa+K} = u_b$. We construct $(\tilde{u}_i^k)$ iteratively as $\tilde{u}_i^\kappa = u_i^k$ and, for $k \geq \kappa + 1$,

$$\tilde{u}_i^k = \begin{cases} w_i^k & \text{if } \tilde{v}_i^k + w_i^k \tau \geq 0 \\ -\frac{v_i^k}{\tau} & \text{otherwise} \end{cases},$$

where $\tilde{v}_i^k$ is the speed of vehicle i at time t_k under control $(\tilde{u}_i^k)$.

As $\tilde{v}_i^{\kappa+1} = v_i^{\kappa+1} \leq v_{max} \leq (K-1)\tau|u_b|$ from the hypothesis, there exists a minimal value of $k_0 \geq \kappa$ such that $\tilde{v}_i^{k_0} \leq |u_b|\tau$; moreover, the condition on K ensures that

$$\tilde{v}_i^{\kappa+1} - (K-2)|u_b|\Delta_t \leq |u_b|\Delta_t,$$

and so $k_0 \leq (\kappa+1) + (K-2) = \kappa + K - 1$.

From the definition of $(\tilde{u}_i^k)$, we know that for all $k_0 + 1 \leq k \leq \kappa + K$, $\tilde{v}_i^k = 0$. Since $\tilde{u}_i^k \leq u_i^k$ for $\kappa \leq k \leq k_0 - 1$, we also know that $\tilde{s}_i^{k_0} \leq s_i^{k_0}$ and $\tilde{v}_i^{k_0} \leq v_i^{k_0}$. Finally, $\tilde{u}_i^{k_0}$ is the minimal admissible control starting from $\tilde{x}_i^{k_0}$; therefore, $\tilde{s}_i^{k_0+1} \leq s_i^{k_0+1} = s_i^{\kappa+K}$ which proves the above lemma. □

C.1.2.2 Proposition 2

Proof of Proposition 2. We first consider the continuous-time case to give an intuition of the proof. We build upon the fact that the rearmost vehicle in a line can always brake with acceleration u_b until it fully stops. However, some the initial conditions may require vehicles in front to accelerate in order to avoid collisions, for instance if the rearmost vehicle is too fast. However, even in this case, we know that the second rearmost vehicle can brake with u_b as soon as it has matched the speed of the rearmost vehicle, and by induction this is true for all the vehicles in the line. In the continuous-time case, all of these vehicles can therefore stop in a time bounded by $\frac{v_{max}}{|u_b|}$.

When considering piecewise-continuous controls, an additional source of complexity arises from the fact that vehicles may match speed between two time steps, resulting in an "overshoot" in velocity. We use the hypotheses on the acceleration to bound this

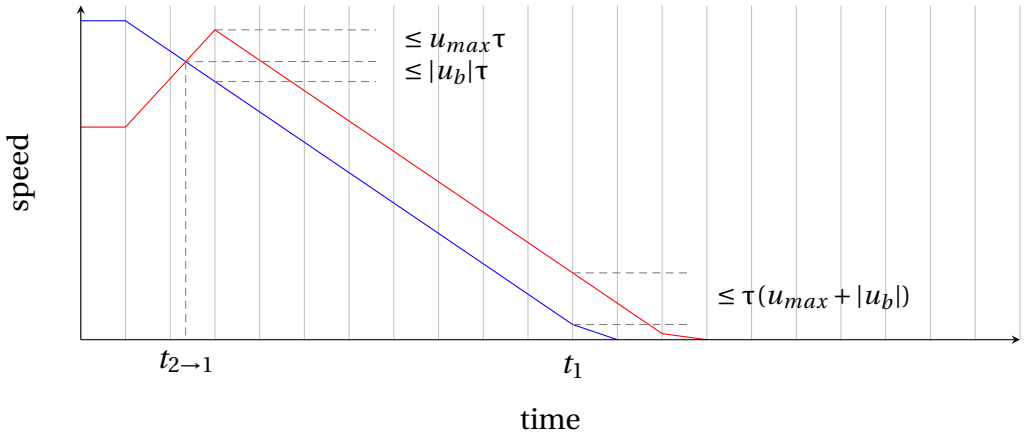


Figure C.1 – Illustration of the "overshoot" phenomenon; the vertical grid correspond to integer multiples of the time step.

overshoot, as illustrated in Figure C.1: we consider a fast vehicle (noted 1, blue curve) following a slower vehicle (noted 2, red curve). To avoid collisions, vehicle 2 is required to accelerate; vehicles match speed at time $t_{2\to1}$; however, due to the time discretization, the overshoot phenomenon can occur. Using the bounds on the acceleration, noting $k_{2\to1}$ the time step immediately following $t_{2\to1}$, we know that

$$v_2^k \leq v_1^k + \tau(u_{max} + |u_b|).$$

Moreover, after step k, vehicle 2 can brake with a control u_b up to time t_1, corresponding to the first integer time step k_1 when $v_1^{k_1} \leq \tau|u_b|$. Since both vehicles 1 and 2 have the same acceleration of $[t_{2\to1}, t_1]$, we know that

$$v_2^{k_1} \leq \tau|u_b| + \tau(|u_b| + u_{max}).$$

Therefore, noting $k_2 = k_1 + 1 + \left\lceil \frac{u_{max}}{|u_b|} \right\rceil$, we get $v_2^{k_2} \leq \tau|u_b|$. The same reasoning can then be repeated for the vehicles preceding vehicle 2. We will now formalize this recursion, as follows.

We will prove by induction that, for $i \in \{1, \dots, p\}$, there exists a dynamically feasible control $(\hat{u}_i^k)$ with $\hat{u}_i^\kappa = u_i^\kappa$ and $\hat{u}_i^k \leq u_i^k$ for $k \geq \kappa$ such that the corresponding vehicle speed $(\hat{v}_i^k)$ verifies $\hat{v}_i^{\kappa+K_i} \leq |u_b|\tau$ with

$$K_i\tau = \left\lceil \frac{v_{max}}{|u_b|} \right\rceil + (i-1)\left(1 + \left\lceil \frac{u_{max}}{|u_b|} \right\rceil\right)\tau.$$

First, for the rearmost vehicle $i = 1$, the proof of lemma 1 provides the result with $\hat{u}_i^k = \tilde{u}_i^k$.

We now let $i \geq 2$ and assume that every vehicle $j \in \{1, \dots, i-1\}$ follows its corresponding control $(\hat{u}_j^k)$. We note $\hat{s}_j^k$ and $\hat{v}_j^k$ the position and speed of vehicle j at step k under this control. Since $\hat{u}_j^k \leq u_j^k$ for these vehicles, we deduce from the monotony of the system that the original control solution for vehicle i $(u_i^k)_{\kappa \leq k < \kappa+K}$ prevents rear-end collisions if vehicle $i-1$ applies $(\hat{u}_{i-1}^k)$. Therefore, any dynamically feasible extension of (u_i^k) is safe over $[\kappa\tau, (\kappa+K+1)\tau)$. As a result, the set $U_i^{safe}(u_i^\kappa, [\kappa, \kappa+K])$ of all admissible controls $(\hat{u}_i^k)_{\kappa \leq k \leq \kappa+K}$ for vehicle i such that $\hat{u}_i^\kappa = u_i^\kappa$ and $\underline{u}_i^k \leq \hat{u}_i^k \leq u_i^k$ for $\kappa \leq k < \kappa+K$ is not empty. We note $(\hat{u}_i^k)$ a minimum element of this set (and so $\hat{u}_i^k \leq u_i^k$); we will prove that $\hat{v}_i^{\kappa+K_i} \leq |u_b|\tau$.

If for all $k \geq \kappa$, $\hat{v}_i^k \leq \tilde{v}_{i-1}^k$, we conclude that vehicle i stops before vehicle $i-1$ which proves the result from the induction hypothesis. Otherwise, we let $k_0^i \geq \kappa$ be the minimum k such that $\hat{v}_i^k \geq \tilde{v}_{i-1}^k$, and we know that

$$\hat{v}_i^{k_0^i} \leq \hat{v}_{i-1}^{k_0^i-1} + u_{max}\tau.$$

For all $k \geq k_0^i$, we know from the monotony of the system that the control $\min(\hat{u}_{i-1}^k, u_i^k, u_b)$ prevents rear-end collisions; we deduce that, for $k \geq k_0^i$,

$$\hat{v}_i^k \leq \hat{v}_{i-1}^{k-1} + u_{max}\tau.$$

Therefore,

$$\hat{v}_i^{\kappa+\mathrm{K}_{i-1}+1} \leq \hat{v}_{i-1}^{\mathrm{K}_{i-1}} + u_{max}\tau$$

and we deduce from the induction hypothesis that

$$\hat{v}_i^{\kappa+\mathrm{K}_{i-1}+1} \leq u_{max}\tau + |u_b|\tau.$$

Therefore, we obtain the recursion relation

$$\mathrm{K}_i = \mathrm{K}_{i-1} + 1 + \left\lceil \frac{u_{max}}{|u_b|} \right\rceil$$

which yields the announced result.

Finally, we conclude that vehicle i can fully stop (without rear-end collisions) at step $\kappa + \mathrm{K}_i + 1$; therefore the set of p vehicles can safely stop before the beginning of step $\kappa + \mathrm{K}$ if

$$\mathrm{K} \geq \mathrm{K}_p = \frac{v_{max}}{|u_b|\tau} + (p-1)\left\lceil \frac{u_{max}}{|u_b|} \right\rceil + 1.$$

Since the recursion ensures that for all i and k, $\hat{u}_i^k \leq u_i^k$, we deduce that $s_i^{\kappa+\mathrm{K}} \leq \hat{s}_i^{\kappa+\mathrm{K}}$ which proves the proposition.

Note that the time needed for vehicles to match speeds can also be bounded by $\frac{v_{max}}{u_a}$ regardless of the value of p. Therefore, all vehicles can also fully stop within the time horizon if

$$\mathrm{T} = \mathrm{K}\tau \geq \frac{v_{max}}{|u_b|} + \frac{v_{max}}{u_a} + 2\tau.$$

Depending on the value of p, this bound may be better than the previously demonstrated one. □

C.1.2.3 Proposition 3

Proof of Proposition 3. We consider a time $t_\kappa = \kappa\tau$, and we let

$$\mathrm{T}_{rec} = \mathrm{K}_{rec}\tau \geq \mathrm{T}_{stop} + \frac{v_{min}}{u_a} + \frac{d}{v_{min}} + \tau.$$

Consider a solution $\mathbf{X}$ of FH-SP$_{\mathrm{K}_{rec}}$ for the vehicles of $\mathscr{R}_{t_\kappa}$, defined for steps $\kappa \leq k \leq \kappa + \mathrm{K}$. We will first show that this solution can be extended to a solution of FH-SP$_{\mathrm{K}+1}$ for the vehicles in $\mathscr{R}_{t_\kappa}$. Note that the only constraints which can be unfeasible are the safety constraints (4.7a) to (4.7c) and the minimum velocity constraints (6.3a) and (6.3b). Consider a vehicle $i \in \mathscr{R}_{t_\kappa}$: using the control corresponding to this solution, two cases can arise:

- $s_i(\mathrm{T}_{stop}) \leq s_i^{acc}$, in which case proposition 2 ensures that i and all the vehicles behind it can fully stop before reaching s_i^{acc}, and can remain stopped up to step $\mathrm{K}_{rec}+1$. Since we also require that $s_j^{acc} \geq s_i^{acc}$ if j follows i, this ensures that keeping i and its followers stopped satisfies all the above constraints;
- otherwise, $s_i(\mathrm{T}_{rec}) \geq \overline{s}_i^{\perp}$, in which case the crossing and minimum velocity constraints (4.7a), (6.3a) and (6.3b) are satisfied for all conflicting vehicle j up to step K_{rec}. The requirement $\mathrm{T}_{rec} \geq \mathrm{T}_{stop}$ and proposition 2 ensure that the safe following constraints (4.7b) and (4.7c) involving vehicle i remain satisfiable for the vehicles of $\mathscr{R}_{t_\mathrm{K}}$ up to step $\mathrm{K}_{rec}+1$.

Indeed, if $\underline{s}_i^{\perp} \geq s_i(\mathrm{T}_{stop}) > s_i^{acc}$, condition (6.3a) ensures that vehicle i accelerates at least with acceleration u_a until reaching speed v_{min}, which takes at most a time $\frac{v_{min}}{u_a}$. The vehicle is then required to maintain speed v_{min} until reaching $\overline{s}_i^{\perp}$, which takes at most a time $\frac{d}{v_{min}}$. Therefore, vehicle i necessarily reaches $\overline{s}_i^{\perp}$ by time T_{rec}; the additional τ accounts for vehicles reaching doing so between two time steps.

The above considerations ensure that the solution of FH-SP$_\mathrm{K}$ at time t_K can be prolongated to a solution of FH-SP$_{\mathrm{K}+1}$ for the vehicles of $\mathscr{R}_{t_\mathrm{K}}$. Finally, noting that the safe entry hypothesis ensures that this solution remains safe even when taking the vehicles of $\mathscr{R}_{t_\mathrm{K}+\tau} \setminus \mathscr{R}_{t_\mathrm{K}}$ into consideration. By definition, there also exists a safe control (and therefore a solution to FH-SP$_\mathrm{K}$) for these vehicles at time $t_\mathrm{K}+\tau$. As a result, there exists a solution to FH-SP$_\mathrm{K}$ at time $t_\mathrm{K}+\tau$ for the vehicles of $\mathscr{R}_{t_\mathrm{K}+\tau}$ which proves the stated result. □

C.2 Discussion on implementation

In Chapter 6, we presented an optimization-based algorithm for the supervision of semi-autonomous vehicles; we now briefly discuss obstacles and possible solutions for actual implementation. First and foremost, not all vehicles will be equipped with the required communication capacities at the same time; therefore, the ability to deal with unequipped vehicles and other traffic participants is key to envision actual applications. Second, this work assumes perfect communication and control, and in general ignores uncertainties arising from real-world constraints.

C.2.1 Dealing with unequipped vehicles

As with all innovations, the penetration rate of such a system would gradually increase overtime, but remain below 100 % for years, yet the formulation proposed in Chapter 6 requires all vehicles to be equipped with supervision capacities. Although a detailed study on the integration of unequipped vehicles in our framework is out of the scope of this paper, we present a possible technique to handle these vehicles provided that they can avoid longitudinal collisions with the leading vehicle, and have a bounded reaction time.

First, note that it is always possible to consider unequipped vehicles conservatively as proposed in [124]: at a given step k, we compute the minimum and maximum curvilinear position that can be reached at time t_k by the unequipped vehicle i_u, denoted by $s_{i_u,min}^k$ and $s_{i_u,max}^k$ respectively. Using the same notations as in Chapter 6, we then define:

$$\varepsilon_{i_u j,p}^{\parallel}(k) = \mathbb{1}_{\left[s_{i_u j,p}^{\parallel},+\infty\right)}\left(s_{i_u,max}^k\right), \tag{C.1}$$

$$\varepsilon_{i_u j,p}^{\perp}(k) = \mathbb{1}_{\left[\overline{s}_{i_u j,p}^{\perp},+\infty\right)}\left(s_{i_u,min}^k\right). \tag{C.2}$$

Therefore, the unequipped vehicle is considered as occupying the conflict region at step k when there exists a control (maximum acceleration) for which it could be inside this region at step k. Similarly, the vehicle is only considered as liberating the conflict region when, even by applying a maximum braking, it would exit it. The collision avoidance constraints (4.7b) and (4.7c) are also modified to use $s_{i_u,min}^{k+1}$ and $v_{i_u,min}^{k+1}$, where $v_{i_u,min}^{k+1}$ is the minimum speed reachable by i_u at $k+1$. Other traffic participants such as cyclists (and, to a lesser extent, pedestrians) could also be taken into account in this fashion. Recently proposed "non-conservatively defensive strategies" [166] could also be applied.

A limitation of this simple approach is that it can lead equipped vehicles to often yield right-of-way to unequipped vehicles, which may be problematic and can slow the acceptance of the system. A possible method (introduced in [167]) to reduce this problem while improving the global level of safety is to use the existing equipped vehicles to force the unequipped ones to stop when required. Suppose that an unequipped vehicle (denoted by i_u) follows an equipped one (i_e), both crossing the path of another equipped vehicle j_e. By setting $\pi_{i_e j_e} = 0$ (thus requiring j_e to pass before i_e), we effectively force the unequipped vehicle i_u to also pass after j_e; the reaction time of the unequipped vehicle can be taken into account by adjusting the lower bound on the longitudinal acceleration of vehicle i_e.

Note that this approach still guarantees that no collision can happen between an unequipped and an equipped vehicle; moreover, as the penetration rate of equipped vehicles increases, additional rules may be enforced to reduce the number of occurrences in which conflicting unequipped vehicles are simultaneously allowed in the conflict region, thus increasing safety even for the unequipped vehicles. Future work will study the impact of penetration rate on safety and efficiency for both equipped and unequipped vehicles.

C.2.2 Practical implementation

In a first iteration, we propose a centralized implementation where a roadside computer (*supervisor*) with communication capacities is added to the infrastructure, and is tasked with repeatedly solving FH-SP$_{\mathrm{K}}$. Note that resolution could also be performed using cloud computing, possibly allowing much faster computation without necessitating fully dedicated hardware. The supervisor is also assumed to be equipped with a set of sensors (*e.g.*, cameras), so that the arrival of new vehicles in the supervision area can be monitored (in order to account for unequipped vehicles and other traffic participants). Equipped vehicles are supposed to regularly communicate their current state, including position, velocity and driver's control input, and receive instructions (the safe acceleration sequence (u_i^k) solution of FH-SP$_{\mathrm{K}}$) from the roadside supervisor. The vehicle's onboard computer then uses these instructions to override the driver's control inputs when needed. We argue that the main sources of uncertainty, *i.e.* communication, sensing and control errors, can be taken into account by using safety margins when computing collision regions.

Communications are assumed to have similar performance to current 802.11p specifications; we use the figures provided in [168, 169] as reference, with latency below 20 ms, and packet loss probability of less than 30 % under 300 m. To account for network congestion, we use more conservative values than those reported experimentally in [169]. Moreover, using the additional roadside sensors, we estimate that uncertainty in each vehicle's localization could be reduced to below 1 m longitudinally.

First, the 20 ms latency corresponds to less than 1 m at highway speed. Second, since they do not require exchanging a lot of data, such messages can be sent much more fre-

quently than the refresh rate of the supervisor. Considering messages can be sent at 20 Hz, the probability of a message not being received in 0.25 s is roughly 0.2 %, and 6×10^{-6} after 0.5 s. Since they receive a whole sequence of safe accelerations, individual vehicles can keep executing this sequence until a new one is successfully received. A worst-case scenario would be having one vehicle using acceleration u_a (maximum acceleration) where it should have used u_b (maximum braking): after a duration t, the corresponding positioning error is $\frac{1}{2}t^2(u_a + |u_b|)$, which is roughly 30 cm after 0.25 s and 1.3 m after 0.5 s for typical values of u_a and $|u_b|$ of 5 m s^{-2}. More robust contingency protocols could likely be developed, and will be the subject of future work, but these values can be used as safety margins without compromising performance.

Similarly, positioning and control uncertainty can be accounted for as margins in the collision regions, provided they can be bounded. In this work, we assume that vehicle self-positioning can be improved using the roadside sensors from the supervisor (which can be precisely calibrated), which could provide relatively tight bounds on error.

Appendix D

Complements on Chapter 7

D.1 Semi-infinite obstacles and local optima

In Chapter 7, we claim to use semi-infinite obstacles to avoid creating local optima in the motion planning problem. Although this is true in the application considered in this particular chapter, the semi-infinite obstacle hypothesis is in general *not* sufficient to guarantee the existence of a unique optimum even for convex cost functions. For instance, the single semi-infinite obstacle shown in Figure D.1 creates two locally optimal solutions illustrated as the green and blue paths.

However, obstacles in autonomous driving situations generally are objects with a finite size, for which an infinity of semi-infinite bounding regions exist. Moreover, the question of determining the "best" class of functions to represent obstacles for automated driving is widely debated, and is out of the scope of this thesis. In Chapter 7, we proposed using bounding parabolas as they provide good flexibility without introducing excessive computational complexity. To avoid the issue depicted in Figure D.1, each obstacle o is represented by a parabola p_o with directrix parallel to the reference path at its point closest to o; we then choose the parameters of p_o such that it is the minimal parabola containing all vertices of o.

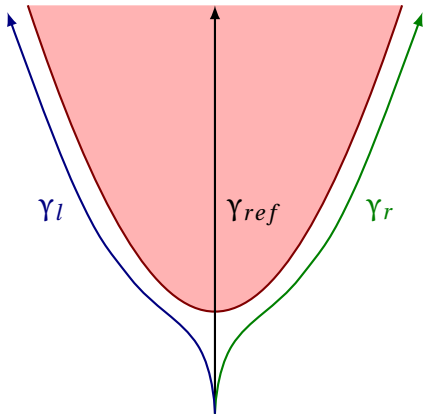


Figure D.1 – Example situation where a single semi-infinite obstacle (in red) can create multiple local optima.

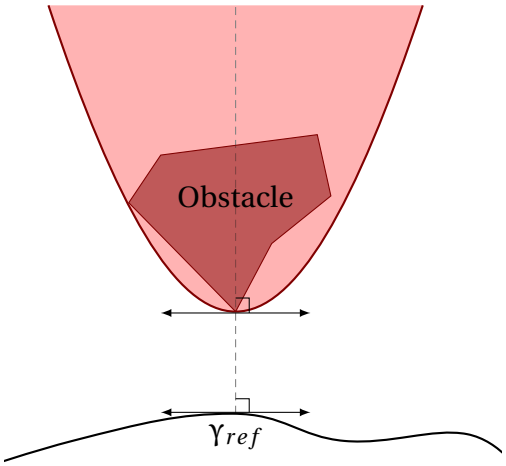


Figure D.2 – Modeling of an obstacle as a semi-infinite parabola with directrix parallel to the reference path at its closest point.

D.2 Simulation model

We consider a 9-degree-of-freedom modeling (DoF) of the vehicle body, which provides a satisfying balance between accuracy and computational cost. Alongside with the usual 2D state $[X, Y, \psi]$ (with ψ the yaw angle) of the vehicle, the model takes into account roll and pitch movements, wheel dynamics and coupling of longitudinal and lateral tire slips. Being a chassis model, it does not take into account the dynamics of the car engine or brakes. The control inputs of the vehicle are the torque T_i applied to each wheel i and the steering angle of the front wheels, δ. We use uppercase letters (*e.g.*, X, Y) to denote coordinates in the ground (global) frame, and lowercase letters for coordinates in the vehicle (local) frame; the x coordinate in the local frame corresponds to the longitudinal component. The notations are given in Table D.1 and illustrated in Figure D.3; subscript $i \in \{1,\dots,4\}$ refers to each of the four wheels of the vehicle in the following order: front left (fl), front right (fr), rear left (rl) and rear right (rr).

Table D.1 – Notations for the 9DoF model

X, Y, Z	Position of the vehicle's center of mass (ground frame)
θ, ϕ, ψ	Roll, pitch and yaw angles of the car body
V_x, V_y	Longitudinal and lat. vehicle speed (vehicle frame)
V_{xw_i}	Longitudinal speed of wheel i (wheel frame)
ω_i	Angular velocity of wheel i
ζ_i	Displacement of suspension i
δ	Steering angle of the front wheels
T_{ω_i}	Total torque applied to wheel i
F_{xw_i}, F_{yw_i}	Longitudinal and lateral forces on wheel i (wheel frame)
F_{x_i}, F_{y_i}	Longitudinal and lateral forces on wheel i (vehicle frame)
F_{z_i}	Normal ground force on wheel i
F_{aero}	Air drag force on the vehicle
M_T	Total mass of the vehicle
I_x, I_y, I_z	Roll, pitch and yaw inertia of the vehicle
I_{r_i}	Inertia of wheel i around its axis
l_f, l_r	Distance between the front/rear axle and the center of mass
l_w	Half-track of the vehicle
r_w	Effective radius of the wheels
k_s, d_s	Suspensions stiffness and damping

We make the assumptions that the body of the vehicle rotates around its center of mass, and that the aerodynamic forces do not create a moment on the vehicle. Moreover, we assume that the road remains horizontal, and any slope or banking angle is neglected; this assumption could be relaxed using a slightly more complex vehicle model. Under

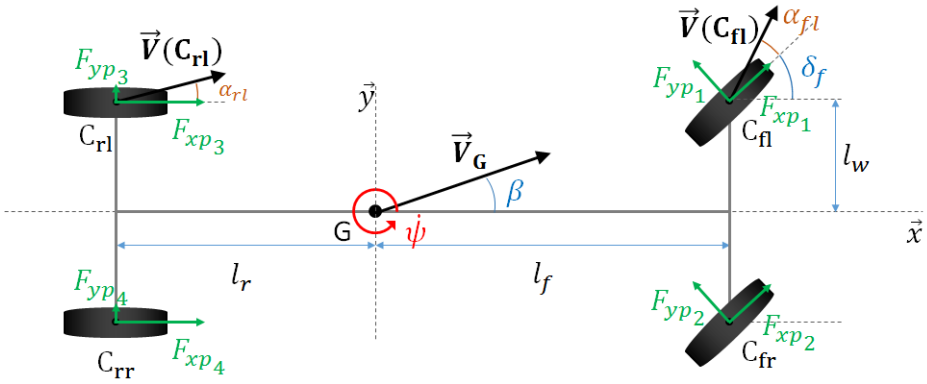


Figure D.3 – Simulation model of the vehicle in the (x, y) plane

these hypotheses, the dynamics of the vehicle's center of mass are written as:

$$\dot{X} = V_x \cos\psi - V_y \sin\psi \tag{D.1a}$$

$$\dot{Y} = V_x \sin\psi + V_y \cos\psi \tag{D.1b}$$

$$\dot{V}_x = \dot{\psi} V_y + \frac{1}{M_T} \sum_{i=1}^{4} F_{x_i} - F_{aero} \tag{D.1c}$$

$$\dot{V}_y = -\dot{\psi} V_x + \frac{1}{M_T} \sum_{i=1}^{4} F_{y_i}, \tag{D.1d}$$

where F_{x_i} and F_{y_i} are respectively the longitudinal and lateral tire forces generated on wheel i, expressed in the local vehicle frame (x, y). The yaw, roll and pitch motions of the car body are computed as:

$$I_z \ddot{\psi} = l_f (F_{y_1} + F_{y_2}) - l_r (F_{y_3} + F_{y_4}) + l_w (F_{x_2} + F_{x_4} - F_{x_1} - F_{x_3}) \tag{D.2a}$$

$$I_x \ddot{\theta} = l_w (F_{z_1} + F_{z_3} - F_{z_2} - F_{z_4}) + Z \sum_{i=1}^{4} F_{y_i} \tag{D.2b}$$

$$I_y \ddot{\phi} = l_r (F_{z_3} + F_{z_4}) - l_f (F_{z_1} + F_{z_2}) - Z \sum_{i=1}^{4} F_{x_i} \tag{D.2c}$$

where $F_{z_i} = -k_s \zeta_i(\theta, \phi) - d_s(\dot{\zeta}_i)(\theta, \phi)$, with $\zeta_i(\theta, \phi)$ the displacement of suspension i for the given roll and pitch angles of the car body. The variation of F_z models the impact of load transfer between tires. Finally, the dynamics of each wheel i can be written as

$$I_r \dot{\omega}_i = T_{\omega_i} - r_w F_{xw_i}. \tag{D.3}$$

In general, the longitudinal and lateral forces F_{xw_i} and F_{yw_i} depend on the longitudinal slip ratio τ_i, the side-slip angle α_i, the reactive normal force F_{z_i} and the road friction coefficient μ. The slip ratio of wheel i can be computed as

$$\tau_i = \begin{cases} \frac{r_w \omega_i - V_{xw_i}}{r_w \omega_i} & \text{if } r_w \omega_i \geq V_{xw_i} \\ \frac{r_w \omega_i - V_{xw_i}}{V_{xw_i}} & \text{otherwise.} \end{cases} \tag{D.4}$$

The lateral slip-angle α_i of tire i is the angle between the wheel's orientation and its velocity, and can be expressed as

$$\alpha_f = \delta - \frac{V_y + l_f \dot{\psi}}{V_x \pm l_w \dot{\psi}} \tag{D.5a}$$

$$\alpha_r = -\frac{V_y - l_r \dot{\psi}}{V_x \pm l_w \dot{\psi}} \tag{D.5b}$$

where f and r denote the front and rear wheels respectively.

In this thesis, we use Pacejka's combined slip tire model (equations (4.E1) to (4.E67) in [99]), which takes into account the interaction between longitudinal and lateral slips, thus encompassing the notion of friction circle [170]. For concision purposes, we do not reproduce the complete set of equations.

D.3 Deriving a simpler dynamic model

In this section, we only consider the dynamic response of the car body, and in particular we do not model engine response precisely. Instead, we assume that the engine can deliver a torque comprised between 0 N m and $2\mathrm{T}_{max} > 0$, and that the brakes can apply a negative torque between $\mathrm{T}_{min} < 0$ and 0 N m on each wheel. The engine torque is equally split between the two front wheels, and braking torques are supposed to be equal for wheels on a same axle. Note that torque vectoring [171], in which the accelerating and braking torques are not equally divided between the wheels of an axle, can also be treated using the same method. The steering angle of the front wheels is supposed to be bounded between $\delta_{min} < 0$ and $\delta_{max} > 0$. With these hypotheses, we note $\mathcal{U} = [\mathrm{T}_{min}, \mathrm{T}_{max}] \times [\mathrm{T}_{min}, 0] \times [\delta_{min}, \delta_{max}]$ the set of admissible controls and $u = [\mathrm{T}_f, \mathrm{T}_r, \delta] \in \mathcal{U}$ a control, where T_f is the torque applied on each of the front wheels, T_r the torque on the rear wheels, and δ the steering angle for the front wheels.

Using the dynamic model of Appendix D.2 and starting from a known system state $\mathbf{x}_0$, it is possible to compute future states of the vehicle under a known control input using numerical integration. To do so, we use a fourth order Runge-Kutta integration scheme with a time step duration Δt of 1 ms, which appears to be sufficient to correctly handle the wheel dynamics. We compute an approximation of the set of feasible accelerations starting from $\mathbf{x}_0$ as presented in Algorithm 6. In the algorithm, the function fitpolynom(st, $\star$, 2) returns the coefficients of the best fitting polynom of order 2 for the component $\star$ of st, with leading coefficient first. Therefore, the variable `feas` contains the set of resulting accelerations in the X and Y directions (noted a_X and a_Y) as well as the yaw rate acceleration $\ddot{\psi}$ (noted a_ψ), all expressed in the ground coordinates frame. In what follows, we present outputs from Algorithm 6 for varying conditions. The control bounds are chosen as $\mathrm{T}_{min} = -1500$ N m, $\mathrm{T}_{max} = 1250$ N m and $\delta_{max} = -\delta_{min} = 30°$.

Algorithm 6: Sampling of the feasible regions

Data: state ξ_0, num. of samples n, horizon T, step Δt
set `feas` := []
for $i = 1 \dots n$ **do**
  randomly choose $u \in \mathcal{U}$
  for $k = 1 \dots \mathrm{T}/\Delta_t$ **do**
    set ξ_k := RK4 (ξ_{k-1}, u, Δt)
  set $st := (\xi_k)_{k=0\dots\mathrm{T}/\Delta t}$
  set p_X := fitpolynom(st, X, 2)
  set p_Y := fitpolynom(st, Y, 2)
  set p_ψ := fitpolynom(st, ψ, 2)
  append to: `feas`, $2 \cdot [p_\mathrm{X}(1), p_\mathrm{Y}(1), p_\psi(1)]$

D.3.1 Longitudinal velocity

In Figure D.4, we present the computed shapes of the set of reachable accelerations in the (a_X, a_Y), (a_X, a_ψ) and (a_Y, a_ψ) planes for a standard berline car ($l_f = 1.17\,\text{m}$, $l_r = 1.77\,\text{m}$, $l_w = 0.81\,\text{m}$, $M_T = 1820\,\text{kg}$, front-wheel drive), over a horizon T of 0.1 s and for various initial longitudinal velocities $v_{x,0}$. The initial state of the vehicle is taken with all angles and initial velocities (except the longitudinal one) equal to zero for the car body, and the wheels are initially rolling without slipping (*i.e.* $\omega_i = v_{x,0}/r_w$ for all $i = 1\ldots4$). The friction coefficient for the road-tire contact is chosen equal to 1. Note that this technique assumes a constant control over a time interval of 0.1 s; therefore, the impact of ABS or ESP cannot be measured, and may be the cause of the concavity at maximum braking observed in Figures D.4a and D.4b at higher velocities.

Remarkably, the projections of this set on the (a_X, a_Y) and (a_X, a_ψ) planes remain very similar throughout the whole speed range. In the (a_Y, a_ψ) plane (Figure D.4c), the projections are all located along the same line, except for high lateral accelerations at high speed in which over- and understeering can occur. We will use these properties to derive efficient bounds in the next section.

D.3.2 Lateral velocity

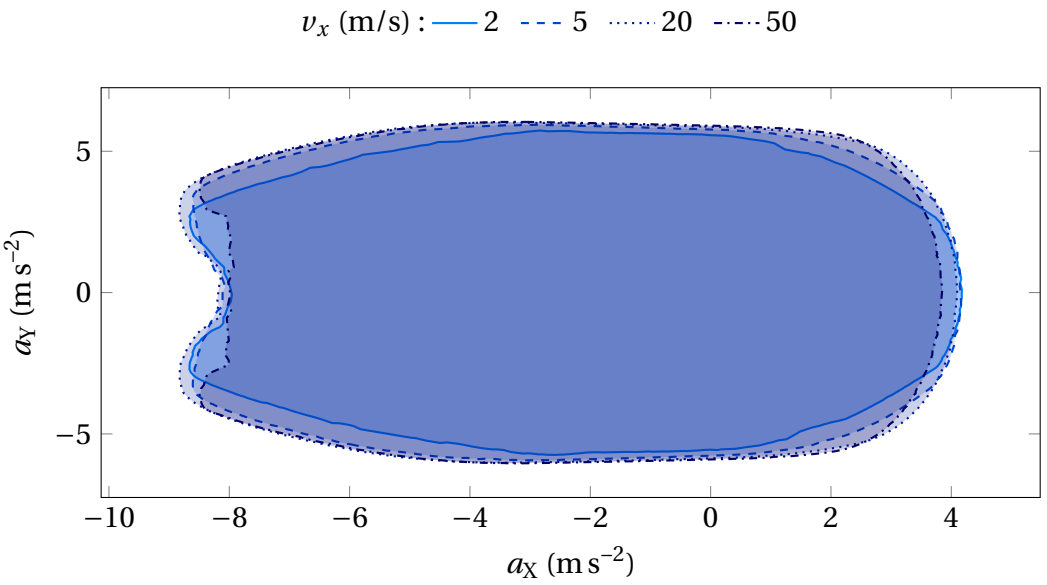
In Figure D.5, we present similarly computed shapes for the sets of reachable accelerations when varying the initial lateral velocity $v_{y,0}$ with an initial longitudinal velocity $v_{x,0} = 20\,\text{m}\,\text{s}^{-1}$. Initial wheel velocities are chosen, as before, as $\omega_i = v_{x,0}/r_w$ for all $i = 1\ldots4$ and initial angles and angular velocities for the car body are chosen as 0.

As the initial lateral velocity increases, the sets in the (a_X, a_Y) and (a_X, a_ψ) planes is shifted mostly along the a_Y and a_ψ axes respectively. Note that we also observe a slight gain in longitudinal acceleration, which corresponds to the fact that part of the initial lateral velocity can be "redirected" into longitudinal velocity by turning the vehicle, though this gain is very marginal. Moreover, the sets are progressively skewed as the lateral velocity increases. Interestingly, we note that increasing lateral velocity further than $0.2\,v_x$ does not provide additional acceleration performance, and instead reduces the commandability of the vehicle thus motivating to avoid these regions during planning.

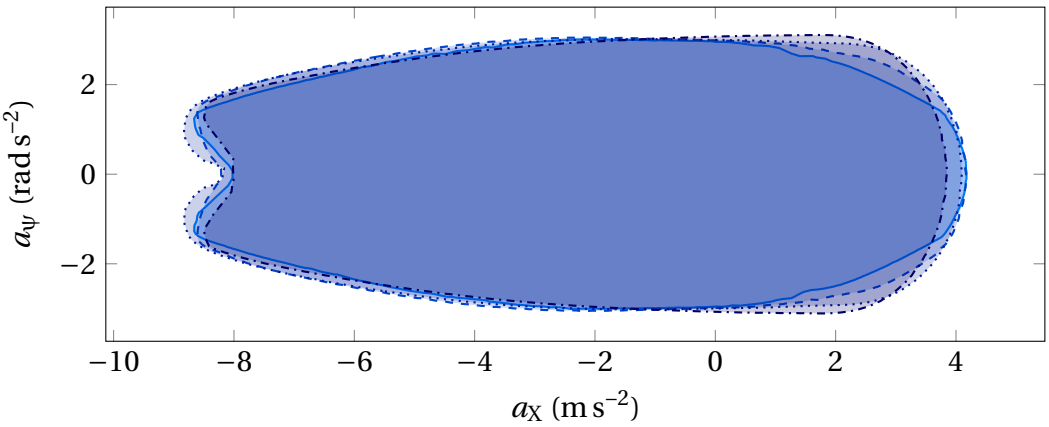
D.3.3 Friction coefficient

In the Pacejka combined slip tire model [99], the tire-road friction coefficient μ appears both as a multiplier and a nonlinear term in the tire-road forces. In Figure D.6, we show the variation of the envelope of feasible accelerations with μ. As for the study on initial longitudinal velocity, we observe that the envelopes keep a similar shape in the (a_X, a_Y) and (a_X, a_ψ) planes, despite the nonlinearity of the tire model. In the (a_X, a_ψ), and in spite of more important slip occurring, the reachable sets also remain aligned along the same line.

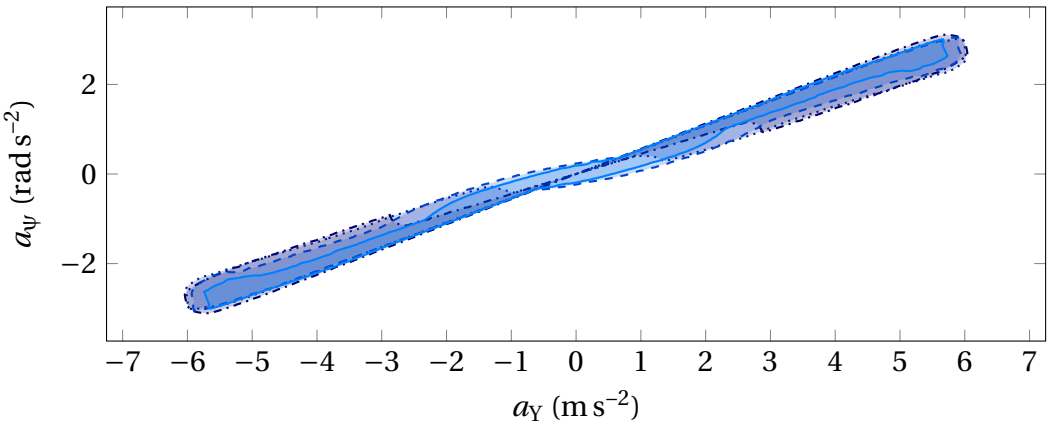
Moreover, our sampling-based method evidences an interesting pattern as the friction coefficient μ decreases. Figure D.7 compares the distribution of the sampled points for $\mu = 0.3$ (icy road) and $\mu = 1$ (dry road); different scales have been chosen for better readability. For $\mu = 1$, we observe that the sampled points are distributed almost uniformly inside the feasible envelope. However, for $\mu = 0.3$, the sampled points accumulate near several regions of attraction, with wide areas with relatively few sampled points. This observation suggests that caution should be exercised when using direct planning methods



(a) Lateral vs. longitudinal

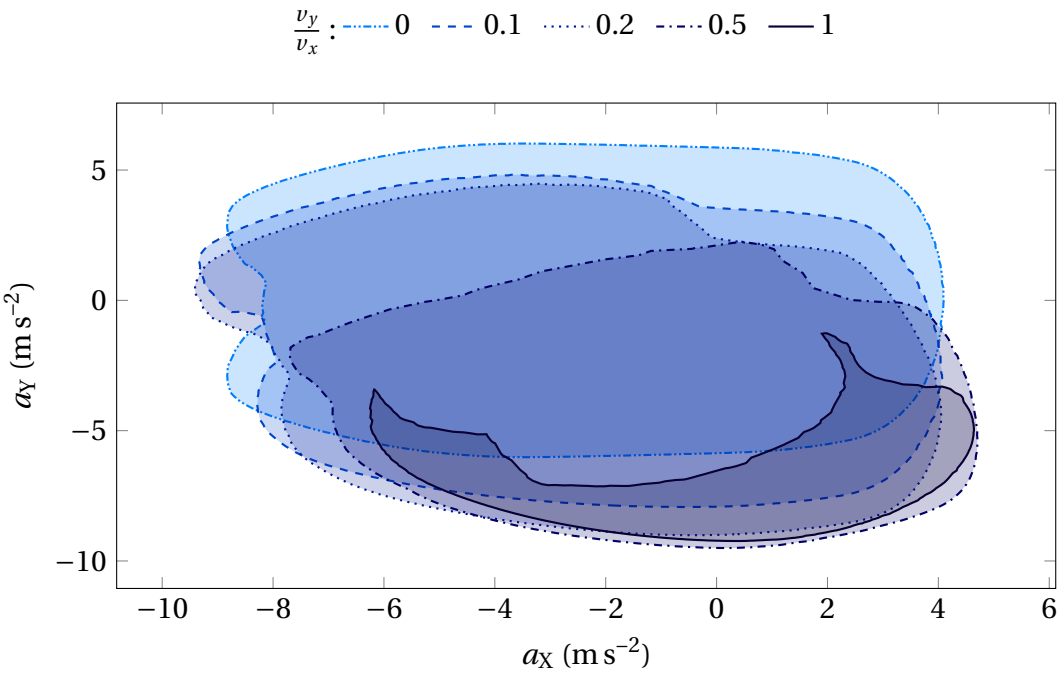


(b) Yaw vs. longitudinal

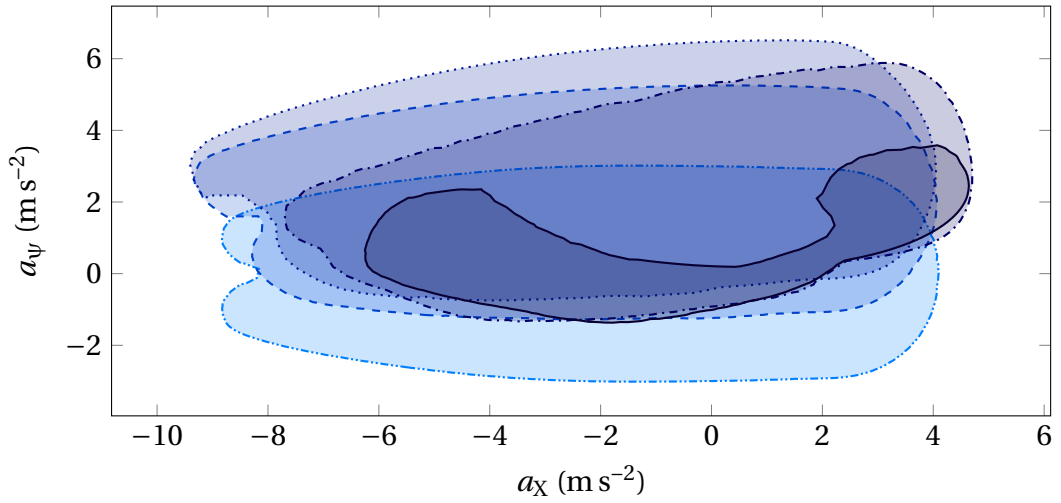


(c) Yaw vs. lateral

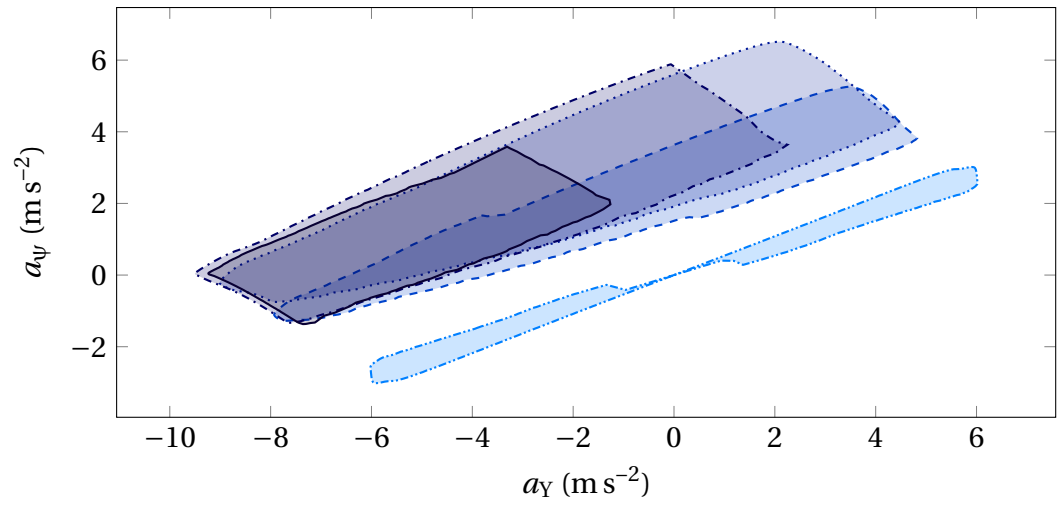
Figure D.4 – Envelope of the computed sets of feasible accelerations for various initial longitudinal velocities $v_{x,0}$ with $v_{y,0} = 0$ and 10^5 sampling points.



(a) Lateral vs. longitudinal

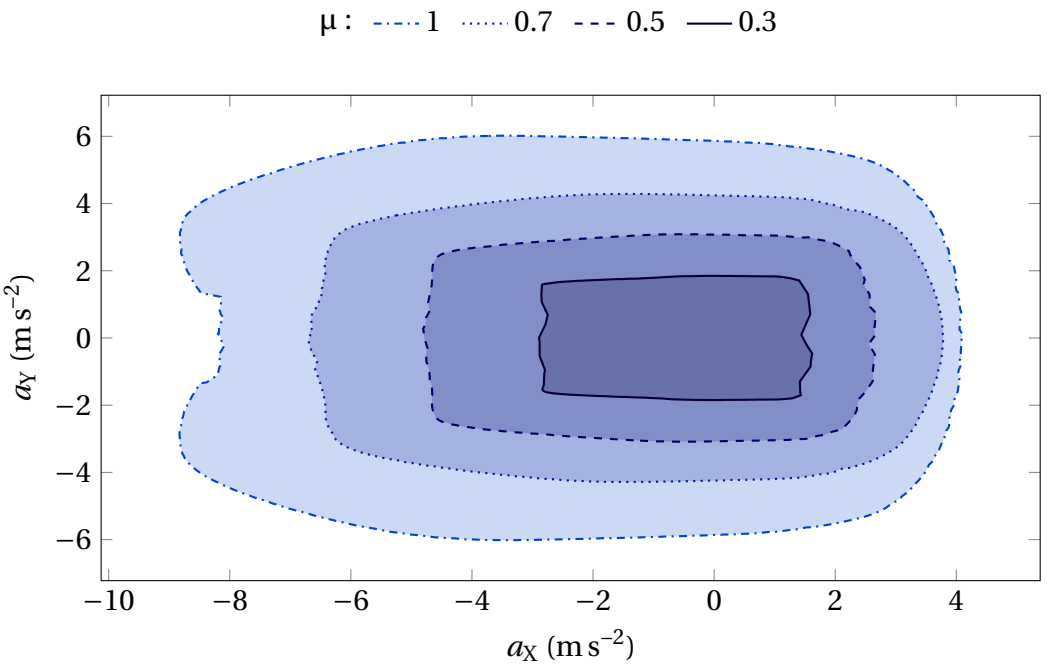


(b) Yaw vs. longitudinal

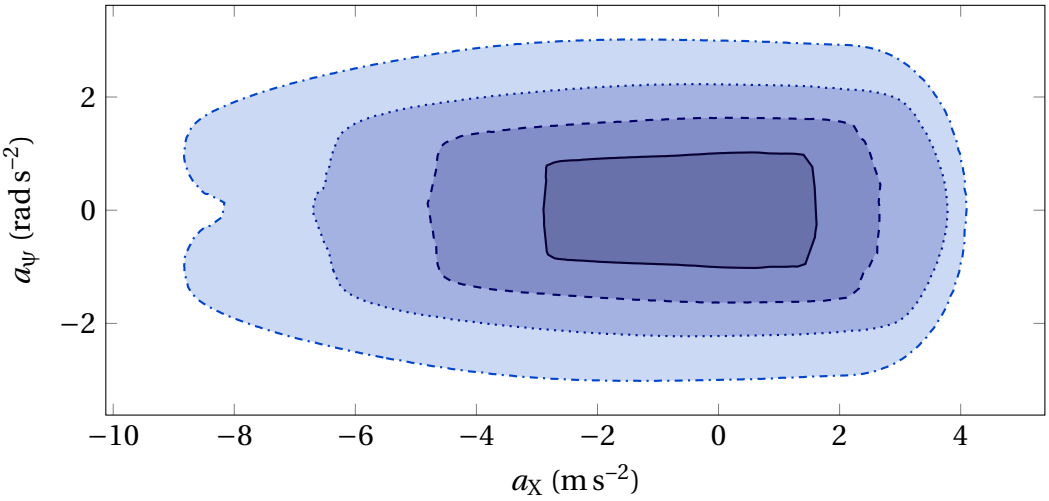


(c) Yaw vs. lateral

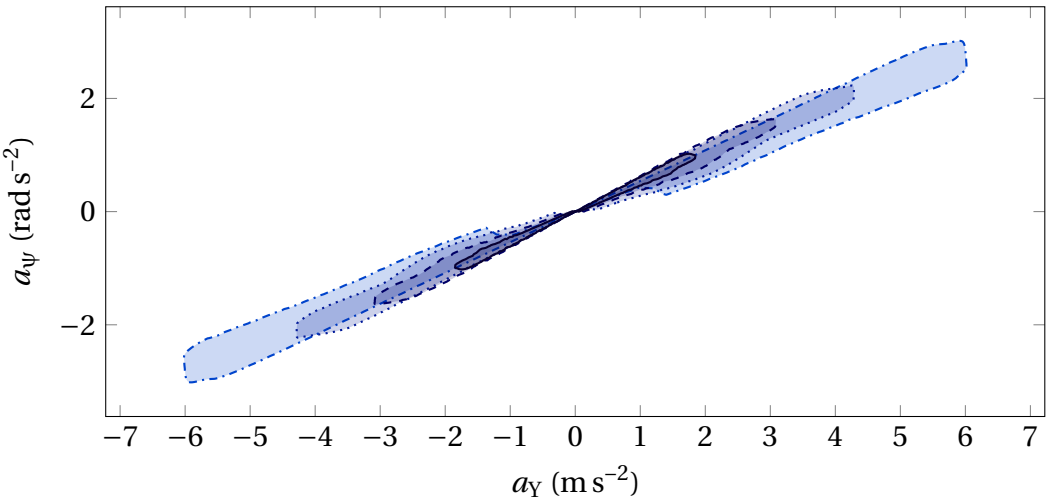
Figure D.5 – Envelope of the computed sets of feasible accelerations for various lateral velocities $v_{y,0}$ with $v_{x,0} = 20\,\mathrm{m\,s^{-1}}$ and 10^5 sampling points.



(a) Lateral vs. longitudinal

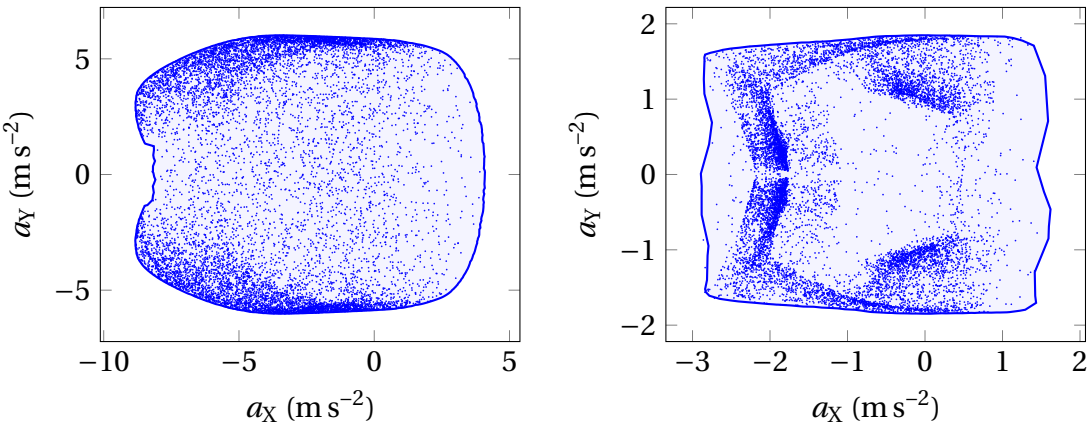


(b) Yaw vs. longitudinal



(c) Yaw vs. lateral

Figure D.6 – Envelope of the sets of feasible accelerations for $v_0 = 20\,\mathrm{m\,s^{-1}}$ and varying μ, with 10^5 sampling points



(a) $\mu = 1$ (b) $\mu = 0.3$

Figure D.7 – Accelerations corresponding to 10^4 points (uniformly) randomly sampled in the control space, shown in the (a_X, a_Y) plane for $\mu = 1$ (dry road) and $\mu = 0.3$ (icy road). Notice the accumulation of points in Figure D.7b.

based on sampling the control space, such as proposed in [172], since the planner would be heavily biased towards these attraction regions, especially in low adherence situations.

D.3.4 Initial rotation

Equations (D.1) and (D.2) show that the vehicle dynamics (except for the position in the ground coordinates) do not depend on the initial yaw angle ψ; therefore, the feasible regions presented above remain invariant (up to a rotation) with respect to the initial yaw angle. Moreover, we observe only very small variations of these regions for small initial values of the pitch and roll angle or rates, and variations of initial velocities of the wheels; these effects are neglected.

D.3.5 Model derivation

Using these results, we propose a constrained double integrator model for the vehicle dynamics. In general, such models are considered very rough approximations for the actual dynamics; however, using well-chosen constraints to couple the longitudinal, lateral and yaw accelerations, a relatively precise approximation can be obtained in this case. The proposed model considers a state vector $\mathbf{x} = [\mathrm{X}, \mathrm{Y}, \psi, v_x, v_y, v_\psi]^\mathrm{T}$ and a control $\mathbf{u} = [u_x, u_y, u_\psi]^\mathrm{T}$, with the same notations and reference frames as presented in Appendix D.2. The dynamic equation of the system is $\dot{\mathbf{x}} = f_{2di}(\mathbf{x}, \mathbf{u})$ with

$$f_{2di}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} v_x \cos\psi - v_y \sin\psi \\ v_x \sin\psi + v_y \cos\psi \\ [v_\psi, u_x, u_y, u_\psi]^\mathrm{T} \end{bmatrix}. \tag{D.6}$$

In theory, it is necessary to take into account the initial state of the vehicle at each time step, and use the sets shown in Figures 7.1 and D.5 to determine the feasible accelerations for the vehicle. However, the complex shape of these sets makes it impractical for trajectory planning. Instead, we propose to compute the "complete" set of feasible accelerations for the vehicle, *i.e.* the union of the sets shown in Figure D.5, which does not depend on the initial lateral velocity. These sets are shown in Figure 7.1; interestingly,

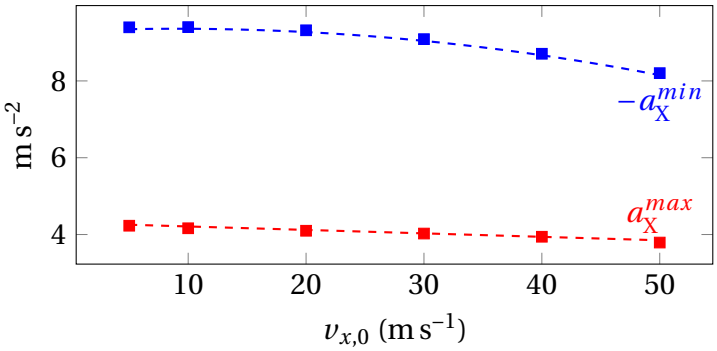


Figure D.8 – Variations of the a_X^{min} and a_X^{max} coefficients with the initial longitudinal velocity $v_{x,0}$; the dashed lines show the polynomial fit.

the boundary of Figure 7.1a can be reasonably well approximated as a truncated ellipse, which is very close to the "g-g diagram" presented in [173], although slightly smaller.

Using these results, we propose to approximate the sets shown in Figure D.4b as a cropped ellipse in the (a_X, a_Y) plane, and a parallelogram in the (a_Y, a_ψ) plane. The parallelogram is chosen constant with the initial velocity, whereas the lower and upper bound on a_X slightly var with the initial longitudinal speed. Note that a study of the 3D set of feasible accelerations (not displayed here) shows that this region is roughly convex, except for the lowest values of a_X, and that it is only necessary to consider constraints in two of these planes to ensure that a corresponding point exists in the 3D feasible region.

The resulting set of constraints on (a_X, a_Y, a_ψ) can be written as:

$$\left(\frac{a_X}{\alpha}\right)^2 + \left(\frac{a_Y}{\beta}\right)^2 \leq 1 \tag{D.7}$$

$$a_X^{min}(v_{x,0}) \leq a_X \leq a_X^{max}(v_{x,0}) \tag{D.8}$$

$$A[a_X, a_Y, a_\psi]^T \leq b \tag{D.9}$$

where A is a constant matrix, b a constant vector and a_X^{min}, a_x^{max} depend on $v_{x,0}$. For our model vehicle, the experimental data of Figure 7.1 yield $\alpha = 9.4\,\mathrm{m\,s^{-2}}$, $\beta = 9.0\,\mathrm{m\,s^{-2}}$, $A = \begin{pmatrix} 2.6 & 1 \\ 2.6 & -1 \end{pmatrix}$ and $b = \begin{pmatrix} 15.3 \\ 15.3 \end{pmatrix}\,\mathrm{m\,s^{-2}}$. Figure D.8 shows the variation of a_X^{min} and a_X^{max} with the initial longitudinal velocity; a polynomial fit yields

$$a_X^{min}(v_{x,0}) = -9.3 - 0.013\,v_{x,0} + 0.00072\,{v_{x,0}}^2$$

and

$$a_X^{max}(v_{x,0}) = 4.3 - 0.009\,v_{x,0}$$

(with $v_{x,0}$ expressed in $\mathrm{m\,s^{-1}}$). In the (a_Y, a_ψ) plane, we suppose a linear relation between a_ψ and a_Y in the form: $a_\psi = \gamma a_Y$, with $\gamma = 0.56\,\mathrm{rad\,m^{-1}}$. Although seemingly restrictive, this approximation provides better results than using a parallelogram acceptable region, and in practice corresponds to minimizing the vehicle slip angle.

Note that these constraints only guarantee the feasibility of a trajectory. To actually drive the vehicle, it is necessary to find a high-frequency low-level control loop capable of following this feasible trajectory. Moreover, the current set of constraints does not account for limitations on the actuator dynamics. Future research may focus on these limitations.

Appendix E

Complements on Chapter 8

E.1 Unicity of the Frenet coordinates

In Part II, we use the Frenet-Serret coordinates to encode the drivable portion of the road. This representation comes with two *caveats*: first, the Frenet frame may not always exist; second, the Frenet coordinates of a point outside of the curve may not be uniquely defined. In Part II, we briefly mentioned smoothness requirements and conditions on road width and curvature to allow using the Frenet-Serret coordinates, which we detail in this appendix.

We consider a simple curve γ in $\mathbb{R}^2$, *i.e.* a continuous and injective function $\gamma : \mathbb{R} \mapsto \mathbb{R}^2$. By choosing an arbitrary point of γ as a reference and a positive direction along γ, it is always possible to define the curvilinear abscissa of a point $X \in \gamma$, which we denote by s. If γ is continuously differentiable with respect to s (*i.e.*, $s \mapsto \gamma(s)$ is $\mathscr{C}^1$), then $\vec{T} = \gamma' = \frac{d\gamma}{ds}$ is the unit tangent vector to γ and we can define the unit normal vector $\vec{N}$ such that $(\vec{T}, \vec{N})$ is an orthonormal direct basis. As a result, ensuring that γ has $\mathscr{C}^1$ continuity ensures that the Frenet frame exists, thus covering the first caveat.

For given functions $r_{min} < r_{max}$, we denote by

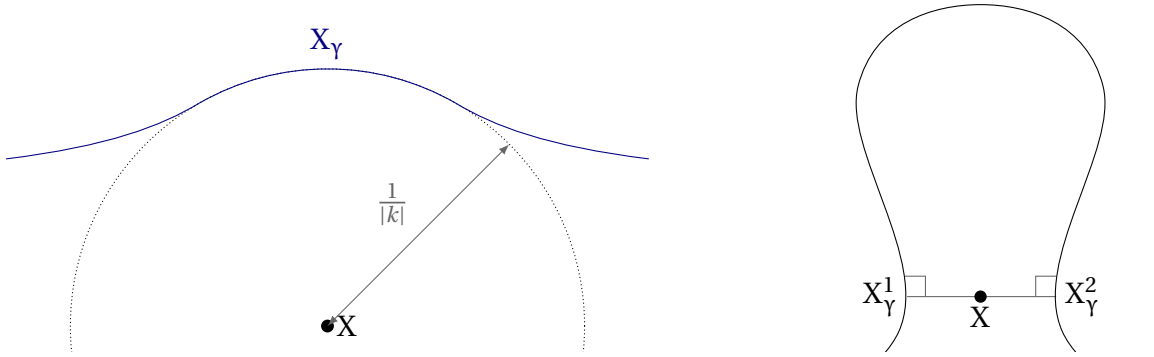
$$R = \left\{ \gamma(s) + r\vec{N}(s) \;\middle|\; s \in \mathbb{R},\ r_{min}(s) \leq r \leq r_{max}(s) \right\}$$

a subset of the plane. Determining whether curvilinear coordinates are uniquely defined for all points in R is tightly connected to the mathematical notion of *reach* of a manifold [174]. In particular, theoretical results ensure[1] that for any compact and $\mathscr{C}^2$ curve γ, there exists a non-empty region R such as defined above for which curvilinear coordinates are unique. Although a formal study is out of the scope of this thesis, we provide some elements of justification below.

If we assume that γ is $\mathscr{C}^2$ and has no multiple points, then we know that $\frac{d\vec{T}}{ds} = k\vec{N}$ where k is the signed curvature of γ at the considered point. Consider a point $X \in \mathbb{R}^2$: since γ is an injection, there are two kinds of situations where the point $X_\gamma \in \gamma$ is not uniquely defined: either γ contains a circular arc with center X (Figure E.1a), or multiple non-local arcs are equidistant to X (Figure E.1b).

In the case where γ has a locally circular arc which realizes the minimum distance from X to γ, Figure E.1a $d(X, \gamma) = \frac{1}{|k|}$ with k the signed curvature and that the rest of γ remains outside of the dotted circle. Therefore, a condition of the form $|r_{min}| < \left|\frac{1}{k}\right|$ and $|r_{max}| < \left|\frac{1}{k}\right|$ (with the convention that the right-hand term is $+\infty$ if $k = 0$) excludes such a case from occurring. The requirement on γ being compact is linked to the second case;

[1] See Proposition 14 in [175].



(a) Locally circular curve

(b) Non-local equidistant arcs

Figure E.1 – Non-unicity of X_γ

schematically, this condition ensures that the distance between "loop strands" (as shown in Figure E.1b) cannot become arbitrarily small.

In practice, as we usually consider "reasonable" road geometries, we consider that the unicity conditions of $\mathscr{C}^2$ continuity and compactness are always met, therefore justifying the use of curvilinear coordinates to represent the position of the ego-vehicle in most driving scenarios. However, some particular situations such as sharp turns in intersections wider than the turn radius may require additional work.

Appendix F

Complements on Chapter 9

F.1 Demonstration of Theorem 6

Proof of Theorem 6. We will first demonstrate the result for paths having only one transition; the generalization to paths with multiple transitions follows directly. We consider two trajectories x_1 and x_2, respectively corresponding to paths $\pi_1 = \big((\sigma_1, \sigma_2), t_1\big)$ and $\pi_2 = \big((\sigma_1, \sigma_2), t_2\big)$ with $t_1 \leq t_2$ and $[t_1, t_2] \subset \text{Adj}(\sigma_1, \sigma_2)$.

First, we will show that there exists a continuous, collision-free trajectory $\tilde{x}$ such that $\tilde{x} = x_1$ over $[t_0, t_1]$ and $\tilde{x} = x_2$ over $[t_2, t_0 + \text{T}]$. Using the convexity of cells $\sigma_t^{-1}(\sigma_1)$ and $\sigma_t^{-1}(\sigma_2)$ for all t, we will then show that x_1 and x_2 are both homotopic to $\tilde{x}$, thus proving the theorem by the transitivity of the homotopy relation.

For $t \in [t_1, t_2]$, we let $\text{I}(t) = \overline{\sigma_t^{-1}(\sigma_1)} \cap \overline{\sigma_t^{-1}(\sigma_2)}$ be the boundary between cells having signatures σ_1 and σ_2 at time t. Since these cells are convex and disjointed, $\text{I}(t)$ is therefore a convex, and non-empty as $[t_1, t_2] \subset \text{Adj}(\sigma_1, \sigma_2)$. Therefore, $\text{I}(t)$ is a segment that can be written in the form $\text{I}(t) = \{a(t) + \lambda u(t) \mid \lambda \in [0,1]\}$ where $a(t) \in \mathscr{R}$ and $u(t) \in \mathbb{R}^2$.

Assuming obstacles have a continuous motion, functions $t \mapsto a(t)$ and $t \mapsto u(t)$ are also continuous. We deduce that function $\phi : (t, \lambda) \mapsto (t, a(t) + \lambda u(t))$ is also continuous over the arc-connected set $[t_0, t_1] \times [0,1]$ and therefore the set $\text{I}_0 = \{(t, x) \mid t \in [t_1, t_2] \wedge x \in \text{I}(t)\}$ is arc-connected.

Since $x_1(t_1) \in \text{I}(t_1)$ and $x_2(t_2) \in \text{I}(t_2)$, we have $(t_1, x_1(t_1)) \in \text{I}_0$ and $(t_2, x_2(t_2)) \in \text{I}_0$ and the arc-continuity ensures that there exists a continuous function ψ defined over $[t_1, t_2]$ such that $\psi(t_1) = x_1(t_1)$, $\psi(t_2) = x_2(t_2)$ and for all $t \in [t_1, t_2]$, $\psi(t) \in \text{I}(t)$. As a result, we can build $\tilde{x}$ by stitching the restrictions of x_1, ψ and x_2.

We now proceed to show that x_1 and $\tilde{x}$ are homotopic: since $x_1 = \tilde{x}$ over $[t_0, t_1]$, we need only consider their restriction over $[t_1, t_2]$. Since x_1 corresponds to π_1, we know that for all $t \in [t_1, t_0 + \text{T}]$, $x_1(t) \in \overline{\sigma_t^{-1}(\sigma_2)}$; similarly, $\tilde{x}(t) \in \overline{\sigma_t^{-1}(\sigma_2)}$ by construction. Since $\sigma_t^{-1}(\sigma_2)$ is convex, we conclude that for all $\mu \in [0,1]$ and all $t \in [t_1, t_0 + \text{T}]$, $x_1(t) + \mu(\tilde{x}(t) - x_1(t)) \in \overline{\sigma_t^{-1}(\sigma_2)}$, thus proving that x_1 and $\tilde{x}$ are homotopic. A similar reasoning ensures that x_1 and $\tilde{x}$ are also homotopic, which proves the announced result. □

Appendix G

Résumés en français

G.1 Introduction

Cette section constitue l'introduction de la thèse.

G.1.1 Chapitre 1

Ce chapitre vise à présenter le contexte de la thèse, qui s'inscrit dans l'important effort de recherche et d'ingénierie visant à développer les véhicules autonomes. Pour ses défenseurs, l'automatisation de la conduite vise avant tout à améliorer la sécurité routière, puisqu'il est estimé que jusqu'à 90% des accidents routiers sont imputables à une erreur humaine. Un second avantage souvent attribué à cette automatisation est une réduction de la congestion routière, les véhicules autonomes étant supposés avoir des temps de réaction inférieurs aux conducteurs humains, et une capacité à communiquer entre eux et avec l'infrastructure pour se coordonner plus efficacement.

Il convient toutefois de garder à l'esprit que ces avantages supposés ne sont à ce jour que théoriques, et dépendent en grande partie des capacités des véhicules autonomes à planifier des trajectoires à la fois sûres et efficaces, deux critères *a priori* contradictoires qu'il est encore difficile de satisfaire simultanément.

L'objectif de cette thèse est de contribuer à la compréhension et la modélisation du processus de *prise de décision* nécessaire à la conduite, et d'utiliser cette compréhension pour améliorer la performance des algorithmes de planification de trajectoire actuellement utilisés pour la conduite autonome.

G.1.2 Chapitre 2

Ce chapitre présente un état de l'art du problème générique de *planification de mouvement*, bien connu du domaine de la robotique. Ce problème consiste, pour un système se trouvant dans un état donné, à calculer une "façon" de l'amener dans un état cible connu. En règle général, la planification cherche de plus à trouver une façon *efficace* d'atteindre l'état cible, selon des critères d'évaluation variés. Pour cette raison, la planification de mouvement est à son cœur un problème d'optimisation sous contrainte, dont une part de la complexité réside dans la non-convexité de l'espace de recherche en présence d'obstacles. Pour cette raison, les problèmes de planification sont généralement NP-difficiles.

Une des formes les plus simples du problème de planification de mouvement est la planification de *chemin*, qui consiste à trouver un chemin géométrique reliant l'origine et

la destination, par exemple une fonction continue à valeurs dans le plan dans le cas d'un espace à deux dimensions. Ce problème, déjà complexe en présence d'obstacles à éviter, est également rendu plus difficile si le système est soumis à des contraintes dites "non-holonomes"[1]. Cette thèse porte sur un problème plus général, dit de planification de *trajectoire*, dans lequel l'objet recherché n'est plus un simple chemin, mais une fonction du temps respectant de plus les contraintes dynamiques du système telles que l'impossibilité de s'arrêter ou de tourner instantanément.

La plupart des méthodes classiquement utilisés en planification de mouvement reposent sur deux familles de techniques. La première consiste à utiliser directement des algorithmes d'optimisation, qui doivent alors gérer explicitement la non-convexité du problème (par exemple *via* la programmation en nombres entiers, généralement coûteuse en temps), ou accepter que la solution trouvée ne soit que localement optimale (notamment dans les algorithmes de commande prédictive). La seconde utilise des méthodes de type Monte-Carlo, qui permettent notamment de maîtriser le temps de calcul au détriment de la qualité de la solution trouvée, ces algorithmes ne pouvant généralement garantir qu'une convergence en probabilité vers la solution optimale. Pour cette raison, la première famille d'approches est privilégiée dans le reste de la thèse.

Comme mentionné dans le paragraphe précédent, un inconvénient des méthodes d'optimisation est la multitude d'optima locaux vers lesquels les algorithmes variationnels sont susceptibles de converger. Dans le cas de la planification de chemin, une notion intéressante pour pallier ce problème repose sur l'utilisation de *classes d'homotopies* explicites*De façon imagée, une classe d'homotopie désigne un ensemble de chemins pouvant être déformés continûment l'un en l'autre.*.

G.1.3 Chapitre 3

Ce chapitre présente les principales contributions de la thèse et les publications associées.

La première contribution de cette thèse est de montrer, pour un système particulier et sous certaines conditions sur le critère d'optimisation, contraindre la solution à se trouver dans une classe d'homotopie donnée rend le problème convexe et garantit donc l'existence d'un unique optimum. Toutefois, la NP-complexité du problème est bien conservée puisque le nombre de classes d'homotopies croît exponentiellement avec celui des obstacles.

La seconde contribution de cette thèse est de généraliser cette notion d'homotopie, qui ne s'applique que difficilement dans le cadre de la planification de *trajectoire*, en une notion de *décision de conduite* pour des applications aux véhicules autonomes.

Enfin, une troisième partie de la thèse est consacrée au passage de la théorie à la pratique, discute des étapes visant à permettre l'intégration des éléments théoriques développés dans le reste de la thèse sur un véhicule réel. Une implémentation simplifiée sur un tel véhicule est également présentée.

G.2 Partie I

Cette première partie de la thèse porte sur la planification de trajectoire coopérative pour un ensemble de robots ou de véhicules autonomes. Elle démontre notamment

[1] C'est par exemple le cas du problème du créneau pour une voiture classique, celle-ci ne pouvant pas se déplacer latéralement (contrainte non-holonome)

l'intérêt de l'utilisation des classes d'homotopies en tant que variables de décision et, plus largement, d'incorporer cette prise de décision de façon explicite dans la planification de mouvement.

G.2.1 Chapitre 4

Ce chapitre étudie la coordination de multiples robots mobiles, par exemple de véhicules autonomes à une intersection, afin de permettre à l'ensemble des robots de rejoindre leur objectif en évitant les collisions entre robots. Sous plusieurs hypothèses simplificatrices, consistant notamment à ne considérer que la dynamique longitudinale des robots, ce problème particulier de planification de trajectoire peut être ramené à un problème, plus simple, de planification de chemin dans un espace dit *de coordination*.

Dans ce contexte, le reste du chapitre présente les principaux résultats concernant l'utilisation des classes d'homotopies pour aider à la résolution du problème. En particulier, il est montré dans ce chapitre qu'il est possible d'utiliser ces classes comme variables de décision entières. En supposant de plus que les robots suivent une dynamique (longitudinale) linéaire, il est alors possible de formuler les contraintes du problème de planification de trajectoire pour le système de robots – pour une décision donnée – comme un problème dont toutes les contraintes sont linéaires par morceaux, et qui peut donc être résolu efficacement de façon exacte pour toute fonction de coût quadratique convexe.

G.2.2 Chapitre 5

Ce chapitre applique les résultats précédents à la coordination *optimale* de robots mobiles, visant à ce que tous les robots atteignent leur objectif en temps minimal et sans collision. Pour ce faire, la planification de trajectoire est formulée comme un problème de programmation (linéaire) en nombres entiers. Cette formulation rend notamment possible l'utilisation de techniques de séparation et évaluation (*branch-and-bound*), qui permettent d'utiliser des informations sur ces sous-problèmes (*i.e.*, à classe d'homotopie fixée) pour accélérer la recherche de la classe d'homotopie optimale. Cette propriété permet de réduire fortement le temps de calcul malgré la complexité exponentielle du problème, et de coordonner jusqu'à une dizaine de robots en temps réel sur un ordinateur classique.

G.2.3 Chapitre 6

Ce chapitre utilise les résultats théoriques du Chapitre 4 pour proposer un système de conduite supervisée pour un ensemble de véhicules semi-autonomes, par exemple afin d'éviter les collisions aux intersections. Contrairement à l'approche entièrement autonome du Chapitre 5, les conducteurs de chaque véhicule sont ici en charge de la conduite. En cas d'erreur humaine, le système de supervision a en revanche la possibilité de se substituer au conducteur de l'un ou plusieurs de ces véhicules afin d'éviter une collision. Le système garantit en particulier de n'intervenir qu'en cas de nécessité et, dans ce cas, de dévier le moins possible les véhicules de l'intention originale de leurs conducteurs.

Ce problème de supervision est également formulé comme un programme (quadratique) en nombres entiers (PQNE), offrant les mêmes possibilités de séparation et évaluation. Il est notamment démontré que, sous certaines hypothèses sur le comportement des véhicules nouvellement entrés dans la zone supervisée, il est mathématiquement possible de garantir la sécurité du système de véhicules à horizon temporel infini en démon-

trant simplement l'existence d'une solution à un problème de type PQNE portant sur un horizon de temps relativement court – et donc soluble en temps réel pour des systèmes comportant jusqu'à une quinzaine de véhicules.

G.3 Partie II

Après une première partie considérant des problèmes de planification unidimensionnels, cette deuxième partie de la thèse considère la planification de trajectoire dans un espace à deux dimensions. En particulier, il est à nouveau démontré l'intérêt d'utiliser une formulation explicite de la prise de décision.

G.3.1 Chapitre 7

Ce chapitre présente une architecture de planification de trajectoire pour la conduite d'un véhicule à haute vitesse autour d'un circuit, en autorisant celui-ci à dépasser le régime de non-glissement mais en ajustant la vitesse automatiquement afin d'éviter les sorties de route. Cette architecture, basée sur la commande prédictive, fonctionne en temps réel grâce à l'utilisation d'enveloppes de contrôlabilité calculées hors-ligne afin de simplifier les calculs liés à la dynamique du véhicule durant l'exécution. L'algorithme développé permet également d'éviter des obstacles placés sur le circuit, à condition qu'une direction d'évitement (par la gauche ou la droite) soit explicitement fournie.

Au-delà des simples performances du système présenté, ce chapitre démontre que la planification de trajectoire, même à haute vitesse et en présence d'obstacles, est un problème relativement simple lorsque la décision de conduite est prise par ailleurs.

G.3.2 Chapitre 8

Ce chapitre étudie les classes de *trajectoires* dans le cas de la conduite autonome, en lien avec la notion de classes d'homotopies de *chemins*, et vise à montrer que la simple notion d'homotopie n'est dans ce cas pas suffisante pour décrire la prise de décision de façon utile à la planification. La raison avancée est double.

D'une part, la notion d'homotopie n'est correctement définie que lorsque les points d'origine et d'arrivée sont connus. Or, dans le cas de la conduite autonome[2], la planification est souvent effectuée avec un horizon temporel glissant (par exemple, les quelques prochaines secondes), sans qu'un point d'arrivée unique ne soit défini.

D'autre part, il est argué que la temporalité de la décision est un élément critique dans la conduite, si bien qu'il est aussi important de spécifier la décision de dépasser un véhicule que *l'instant* où le dépassement doit s'effectuer, or ce second élément est absent de la notion d'homotopie.

G.3.3 Chapitre 9

Sur la base du résultat négatif précédent, ce chapitre développe une nouvelle approche permettant de généraliser la notion de classe d'homotopie afin de surmonter les limitations établies au Chapitre 8.

[2] Et pour de nombreuses applications robotiques

Cette approche se base sur une segmentation sémantique de la portion libre de l'espace-temps dans l'horizon de planification choisi. Sous l'hypothèse de connaître le mouvement futur des autres véhicules, il est en effet possible de partitionner l'espace libre autour de ceux-ci, et de sémantiser chaque élément de la partition (" cellule" dans ce qui suit) selon sa position relative par rapport aux autres véhicules. Le même processus peut être appliqué dans un espace tri-dimensionnel incluant le temps.

Étant données deux cellules de cette partition 3D, il existe un intervalle[3] de temps pendant lequel il est possible de passer de l'une à l'autre sans collision. Il est argué que la donnée d'une séquence de cellules et des temps de transition d'une cellule à la suivante correspond, cette fois, à une décision de conduite. Par ailleurs, une représentation de ces décisions sous la forme d'un *graphe de navigation* est proposée, dans lequel tout chemin[4] correspond à une famille de trajectoires sans collisions. Il est par ailleurs démontré que les chemins dans ce graphe de navigation généralisent la notion de classe d'homotopie, *i.e.* deux trajectoires partageant la même représentation dans le graphe de navigation sont homotopiques.

G.3.4 Chapitre 10

Ce chapitre présente une utilisation possible du graphe de navigation pour la prise de décision appliquée à la planification de trajectoire, sur un scénario simple de dépassement. Il est notamment montré que l'utilisation du graphe de navigation permet effectivement la prise de décision pour la conduite. Toutefois, la principale difficulté de cette approche est la prise en compte des contraintes de faisabilité dynamique, qui n'est pas directement encodée dans le graphe de navigation. Pour cette raison, plusieurs heuristiques d'exploration sont proposées comme pistes de recherche ultérieures, notamment basées sur l'utilisation de techniques d'apprentissage automatisé.

G.4 Partie III

Cette dernière partie vise à discuter l'implémentation pratique des résultats plutôt théoriques établis dans les chapitres précédents. En particulier, de nombreuses hypothèses faites dans le cadre de la planification de trajectoire sont en réalité inaccessibles aux performances actuelles des systèmes de perception. En vue d'une implémentation sur un véhicule réel, un important effort est à fournir au niveau des interfaces entre les disciplines.

G.4.1 Chapitre 11

Ce chapitre étudie les méthodes permettant d'estimer les trajectoires futures des véhicules environnants, afin de pouvoir appliquer les techniques de planification discutées dans la seconde partie de la thèse. Deux méthodes sont ici proposées : une première, applicable en l'absence de données de trafic, est basée sur l'échantillonnage aléatoire d'une séquence réaliste de commandes pour chaque véhicule. La seconde, plus classique mais requérant des données collectées au préalable, consiste à utiliser des réseaux de neurones récurrents pour générer la trajectoire la plus probable d'un véhicule à partir des observations de son comportement passé.

[3] Ou, dans le cas général, une union d'intervalles potentiellement vides.
[4] La définition d'un chemin dans ce graphe incluant la dimension temporelle.

G.4.2 Chapitre 12

Ce dernier chapitre présente une implémentation simplifiée, sur véhicule réel, des algorithmes de prise de décision développés dans la thèse sur un scénario de conduite péri-urbain. Le système développé a présenté un fonctionnement satisfaisant en simulation *software-in-the-loop*, et aurait adopté un comportement qualitativement similaire à celui d'un conducteur humain sur un parcours effectué en conditions réelles (*hardware-in-the-loop*).

Résumé

Le déploiement des futurs véhicules autonomes promet d'avoir un impact socio-économique majeur, en raison de leur promesse d'être à la fois plus sûrs et plus efficaces que ceux conduits par des humains. Afin de satisfaire à ces attentes, la capacité des véhicules autonomes à planifier des trajectoires sûres et à manœuvrer efficacement dans le trafic sera capitale. Cependant, le problème de planification de trajectoire au milieu d'obstacles statiques ou mobiles a une combinatoire forte qui est encore aujourd'hui problématique pour les meilleurs algorithmes.
Cette thèse explore une nouvelle approche de la planification de mouvement, basée sur l'utilisation de la notion de *décision de conduite* comme guide pour structurer le problème de planification en vue de faciliter sa résolution. Cette approche peut trouver des applications pour la conduite coopérative, par exemple pour coordonner plusieurs véhicules dans une intersection non signalisée, ainsi que pour la conduite autonome où chaque véhicule planifie sa trajectoire.
Dans le cas de la conduite coopérative, les décisions correspondent au choix d'un ordonnancement des véhicules qui peut être avantageusement encodé comme un graphe. Cette thèse propose une représentation similaire pour la conduite autonome, où les décisions telles que dépasser ou non un véhicule sont nettement plus complexes. Une fois la décision prise, il devient aisé de déterminer la meilleure trajectoire y correspondant, en conduite coopérative comme autonome. Cette approche basée sur la prise de décision peut permettre d'améliorer la robustesse et l'efficacité de la planification de trajectoire, et ouvre d'intéressantes perspectives en permettant de combiner des approches mathématiques classiques avec des techniques plus modernes d'apprentissage automatisé.

Mots Clés

Planification de trajectoire, Prise de décision, Conduite autonome, Commande prédictive, Gestion d'ntersection autonome

Abstract

The deployment of future self-driving vehicles is expected to have a major socioeconomic impact due to their promise to be both safer and more traffic-efficient than human-driven vehicles. In order to live up to these expectations, the ability of autonomous vehicles to plan safe trajectories and maneuver efficiently around obstacles will be paramount. However, motion planning among static or moving objects such as other vehicles is known to be a highly combinatorial problem, that remains challenging even for state-of-the-art algorithms. Indeed, the presence of obstacles creates exponentially many discrete maneuver choices, which are difficult even to characterize in the context of autonomous driving.
This thesis explores a new approach to motion planning, based on using this notion of *driving decisions* as a guide to give structure to the planning problem, ultimately allowing easier resolution. This decision-based motion planning approach can find applications in cooperative driving, for instance to coordinate multiple vehicles through an unsignalized intersection, as well as in autonomous driving where a single vehicle plans its own trajectory.
In the case of cooperative driving, decisions are known to correspond to the choice of a relative ordering for conflicting vehicles, which can be conveniently encoded as a graph. This thesis introduces a similar graph representation in the case of autonomous driving, where possible decisions – such as overtaking the vehicle at a specific time – are much more complex. Once a decision is made, planning the best possible trajectory corresponding to this decision is a much simpler problem, both in cooperative and autonomous driving. This decision-aware approach may lead to more robust and efficient motion planning, and opens exciting perspectives for combining classical mathematic programming algorithms with more modern machine learning techniques.

Keywords

Motion planning, Decision-making, Autonomous driving, Model predictive control, Autonomous intersection management