TÉCNICO LISBOA

Hybrid Modeling and Control for an Autonomous Passenger Car

André Miguel Guerreiro Carvalho

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Dr. João Fernando Cardoso Silva Sequeira

Examination Committee

Chairperson: Prof. Dr. Maria Eduarda De Sampaio Pinto de Almeida Pedro
Supervisor: Prof. Dr. João Fernando Cardoso Silva Sequeira
Members of the Committee: Prof. Dr. Fernando Manuel Ferreira Lobo Pereira

June 2018

Agradecimentos

Esta dissertação, além do seu carácter cientifico, tem para mim um peso manifestamente simbólico. Neste documento, assentam todas as minhas angústias e crises existenciais acumuladas durante os últimos anos. E devido a isso mesmo, assenta também uma visão de uma vida autêntica e com significado. Foi com as dificuldades e com as responsabilidades, que encontrei motivação para acordar todos os dias e trabalhar, trazendo um pouco de mim ao mundo.

Seria impossível dissociar toda esta experiência das pessoas com quem a partilhei. Em primeiro lugar os meus queridos pais e familiares. Aos meus pais, que sempre se sacrificaram para que eu pudesse ter as ferramentas necessárias para construir o meu próprio caminho, juntando aos valores da honestidade, de assumir os erros e da perseverança perante os obstáculos que sempre me transmitiram. Aos meus familiares pelo apoio incondicional e o carinho que sempre me fizeram chegar ao longo dos anos. Aos meus queridos amigos, cujo suporte foi fulcral em toda este caminho. Ao Filipe, Tiago, Pedro e ao Vítor, um obrigado especial por toda a paciência nos momentos atribulados. Aos colegas, ao pessoal com quem partilhei muitas canecas de chá no quinto piso da torre norte, à equipa do projeto TLMoto, aos professores e aos funcionários do Técnico, um obrigado também. A todas as pessoas com quem me cruzei durante este tempo, porque, de certo modo todas tiveram impacto na minha vida.

Referente a este trabalho especificamente, agradeço ao meu orientador, Prof. João Sequeira, pela liberdade que sempre me concedeu para definir o meu próprio percurso na tese, pelas horas de discussão e conversa que me permitiu aprender mais do que eu alguma vez julgaria conseguir aprender. Ao Prof. Paulo Branco, ao seu humor e boa disposição e à confiança que sempre depositou em mim que acabou por me levar até a este tema dos carros autónomos. Ao Sérgio e ao David pelas suas revisões cuidadas deste documento.

Hat man sein warum? des Lebens, so verträgt man sich fast mit jedem wie?
(Tendo o seu *porquê* da vida, o indivíduo tolera quase todo o *como*.)
Friedrich Wilhelm Nietzsche
em *Götzen-Dämmerung* (1889)

Acknowledgments

This dissertation, besides its scientific purpose, has a strong symbolic weight to me. This document lies down all the anguishes and existential crisis accumulated in the last few years. And, due to that, also lies a vision of the authentic and meaningful life. It was with the daily struggles and with the responsibilities that I found the motivation necessary to wake up every day and work, bringing something of me to the world.

It would be impossible to disassociate this whole experience from the people I shared it with. Firstly, my dear parents and family. To my parents, that made sacrifices so I could have all the necessary tools to build my own path, also the values of honesty, to assume my own mistakes and the perseverance when facing obstacles, that they always teached me. To my family for all the unconditional support and care they always managed to reach me. To my dear friends, whose support was vital along the way. To Filipe, Tiago, Pedro and Vítor, a special thank you for all the patience during the rough patches. To my colleagues, to the people I spent many tea cups with in the fifth floor of the north tower, to the team of the TLMoto project, to the professors and IST staff. To all the people I have come across with, because, in some way, all of them had an impact in my life.

Referring to this work specifically, I acknowledge my supervisor, Prof. João Sequeira, for all the freedom he always gave me to define my own route in the thesis, for all the hours of discussion and conversation that allowed me to learn more than I would ever imagine to be able to learn. To Prof. Paulo Branco, to his wittiness and for the trust he always had in me that eventually led me to this topic of autonomous vehicles. To Sérgio and David for their careful revisions of this document.

Hat man sein warum? des Lebens, so verträgt man sich fast mit jedem wie?
(Having his *why*, the individual can bear almost any *how*.)
Friedrich Wilhelm Nietzsche
in *Götzen-Dämmerung* (1889)

Declaration

I, André Miguel Guerreiro Carvalho, declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Abstract

The popularity of autonomous vehicles has been gaining momentum in the last few years. Safety assessment of such systems is of utmost importance for a proper introduction among society. In this work, we will be modeling the vehicle as an Hybrid System - combining discrete states where in each, the system displays different continuous dynamics. This thesis presents an architecture to control such system,slightly based on how humans drive and intuitively embed uncertainty. We will employ ideas from Viability Theory, such as viability kernels, to restrict the system's inputs to the safe ones. The viability kernel is the largest subset of the constraint set where at least one trajectory starting there remains inside the constraint set for some time. We have improved an existing algorithm to compute finite-time viability kernels, based on sampling the boundary of the constraint set. The adaptation resulted in a faster algorithm and is intuitively extended to non-linear systems described by a first order, continuous, differential equation, convex in the input variables. Proofs on the correctness of such adaptations are presented. We claim that our algorithm, for non-linear systems and in some conditions, has a better asymptotic complexity than the usual griding methods. Results on the convexity of the infinite-time viability kernel for some class of differential inclusions are also given.

Keywords

Autonomous Vehicles; Hybrid Systems; Viability Kernel; Reachability; Set-Valued Analysis; Non-Linear Systems.

Resumo

A popularidade dos veículos autonomos tem vindo a ser cada vez maior nos últimos anos. Garantir a segurança destes sistemas é de extrema importância para uma introdução adequada na sociedade. Neste trabalho, iremos modelar o veículo como um sistema híbrido - combinando estados discretos em que cada um, o sistema evolui de acordo com uma dinâmica contínua diferente. Esta tese apresenta uma arquitetura para controlar tais sistemas, ligeiramente inspirada em como as pessoas conduzem e intuitivamente assimilam as incertezas dessa tarefa. Iremos aplicar ideias da Teoria da Viabilidade, tais como *kernels* de viabilidade, de modo a restringir as entradas do sistema a apenas aquelas consideradas seguras. O *kernel* de viabilidade é o maior subconjunto das restrições no qual existe pelo menos uma trajetoria que comece aí e aí permaneça durante um certo tempo. Melhoramos um algoritmo existente para calcular o *kernel* de viabilidade, baseado em amostras da fronteira do conjunto de restrições. A adaptação resultou num algoritmo mais rapido e que é intuitivamente extendido para sistemas não-lineares descritos por equações diferenciais de primeira ordem, contínuas e convexas nas variáveis de entrada. A exatidão dos resultados após a adaptação é demonstrada. Alegamos que o nosso algoritmo, para sistemas não lineares e em certas condições, tem melhor complexidade assimptotica do que os usais metodos de discretização do espaço de estados. Resultados sobre a convexidade de *kernels* de viabilidade para certas classes de inclusões diferenciais são também apresentados.

Palavras Chave

Veículos Autonomos; Sistemas Híbridos; Kernel de Viabilidade; Atingibilidade; Analise de funções multi-valoradas; Non-Linear Systems.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

ABS	Anti-Lock Braking System
LSC	Lower Semi-Continuous
USC	Upper Semi-Continuous
MPC	Model-Predictive Controller

1

Introduction

1.1 Motivation

The topic of autonomous mobile robots is a well covered subject in control and robotics. Their purposes range from surveillance of coastal zones [**?**], warehouse management systems [**?**] and, the one approached in this work, autonomous passenger cars [**?**].

The popularity of this field, among general audience has raised dramatically since 2012, (see figure **??** below), making this a trendy topic since then.

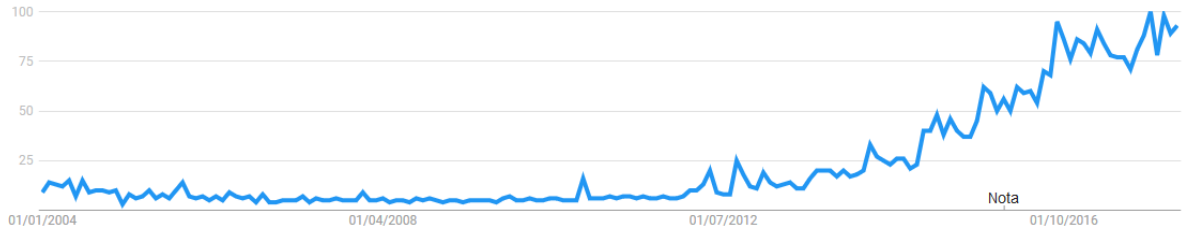


Figure 1.1: Search popularity for the topic "Autonomous Vehicles" in Google search engine. 100 being the relative peak in popularity.

Autonomous vehicles are already well established in our day to day life. Trains and subways are mostly autonomous, with its driver having smaller and smaller influence [**?**], also, in an airplane flight, most of the flight time is done in autopilot mode [**?**].

Fully autonomous road vehicles could be the next major breakthrough in urban transportation. Buses may be more often on schedule and delays due to traffic are mitigated since the time of arrivals can be more accurately predicted. Fuel efficiency and increasing road capacity are also two big advantages of these vehicles [**?**], along with the possibility of reducing car crashes trough elimination of human errors [**?**], since they cause 90% of car crashes [**?**], is also a major driver in the development of this technology.

Safety assessment of such self-driving vehicles becomes of extreme importance for bigger acceptance of such technologies. Although most people feel moderately safe with autonomous vehicles [**?**], the guaranteeing that the vehicle will avoid most accidents - and not making them happen when a human driver would not - plays a major role in the spreading of usage of these cars. In this work we present an alternative approach to the safety assessment and propose a control architecture for a self-driving car.

1.2 Contributions

This thesis characterizes an autonomous car more closely to what a competent human driver would do. This description consists in looking at the problem of driving a car in a more broader sense - we are interested in a sequence of sets of actions possible to safely drive the car instead of a single sequence

of inputs. Human drivers typically have a mental model of a set of positions of the steering wheel and of the accelerator pedal that are somewhat safe for their level of skill or willingness to risk [**?**]. This mental process is what will inspire the architecture for the intelligent part of the vehicle and is related to the scope of *Viability Theory*.

The main focus of the thesis is on the safety assessment of the controls employed in the vehicle making use of Viability Theory [**?**]. This theory is based on the notion of viable solutions of the differential inclusion describing the system - solutions that stay inside a safe constraint set for a finite or infinite time horizon. The hybridization of the system comes from the fact that at each sampling time we are restricting the set of viable inputs, therefore, the differential inclusion changes at each instant. Our interest will be mainly on systems described by differential inclusions that are guaranteed to have solutions in convex constraint sets.

The *viability kernel* is the largest subset contained in the constraint set in which at least one trajectory starting in the former set remains in the latter for some time. We've adapted existing algorithms to approximate viability kernels for linear systems, making them faster up to 100-dimensional systems, and extendable to non-linear systems described by first-order, continuously differentiable, differential equations. The algorithm is based on sampling points in the boundary of the constraint set and assessing the points closest to that boundary that belong to the viability kernel by forward simulation. We proved the algorithm works with our adaptation in the linear case and established the conditions for it to work in non-linear cases. Proofs on the convexity of the viability kernel are also presented.

1.3 Structure

The thesis is structured as follows:

- The remaining of this chapter is dedicated to a brief review of the state of the art in the context of autonomous vehicles and mobile robots.
- In chapter **??** we discuss the mental process of a human driving a vehicle and how he regulates his actions. Based on this and on a few more autonomous vehicles architectures, we propose one that is regulated the same way a human driver is, by comparing the perceived risk of the task ahead with his willingness to take the risks. We also explain the several blocks that pertain to the architecture and suggest how each one could be implemented.
- A vehicle model is proposed in Chapter **??**, following the above conditions, this system model must be described by a first order, continuous differential equation, convex in the input variable.
- The largest part of this work is in Chapter **??**. There we establish the necessary mathematical background, the adaptations made to the original work and the necessary proofs for correctness.

- Finally, in Chapter **??**, we suggest how this work should be carried on by mentioning some ideas that popped out during the thesis as future work.

1.4 Literature Review

1.4.1 Classical Autonomous Mobile Robots

The natural action is to bring the before-used concepts in mobile robotics and autonomous systems into the realm of autonomous passenger cars. The DARPA Urban Challenge [**?**] brought those exact concepts into the field of urban autonomous driving. For example, the winning car in the 2007 edition, 'Boss', which was equipped with GPS, laser sensors, cameras and radars (see figure **??**, source[1]), had an architecture based on generating dynamically feasible trajectories and choosing among those the one that was optimal with some criteria. This approach is, more or less, what one would do with an average mobile robot that ought to do some task.

Figure 1.2: 'Boss', the winning vehicle in 2007

The autonomous architecture of the vehicle consisted in three layers. The mission planning that solved the problem on how to get where it is supposed to (goal state). The behavior layer dealt with the decision part of, for example, changing lanes, reducing speed and avoiding obstacles. The final layer, named motion planning, was in charge of selecting the adequate, actionable, inputs to place the vehicle closer to its goal. The vehicle completed the 85km course at an average speed of 22.5 km/h. However, due to the high cost of the sensor equipment, it would be hard to commercialize such a vehicle, making this solely a prototype.

[1] http://www.etc.cmu.edu

1.4.2 Hybrid Reachability Analysis

A more broader class of systems is the Hybrid Systems. An hybrid system is a system that consists in several discrete states that in each of those the system has a different (in general) continuous dynamics, the *flow* function. When the state of the system fills some conditions, called *guards*, a 'jump' might occur, and the discrete state change. An easy to understand hybrid system is an automatic transmission car. Each gear makes the car have certain dynamics, when the engine reaches a certain angular velocity, the system changes gears accordingly, jumping to another dynamics.

Much of the safety assessment of autonomous cars is done in the sense of reachability analysis. Starting the vehicle at some point, is it possible to ensure it does not collide with obstacles - pedestrians or other vehicles - within all the possible trajectories the vehicle might unravel? In the thesis [**?**], the author offers a number of ways how to infer about the safety of the vehicle. Considering the vehicle as an hybrid system, the author verifies safety through over-approximation of the reachable set; the exact reachable set is usually hard to compute.

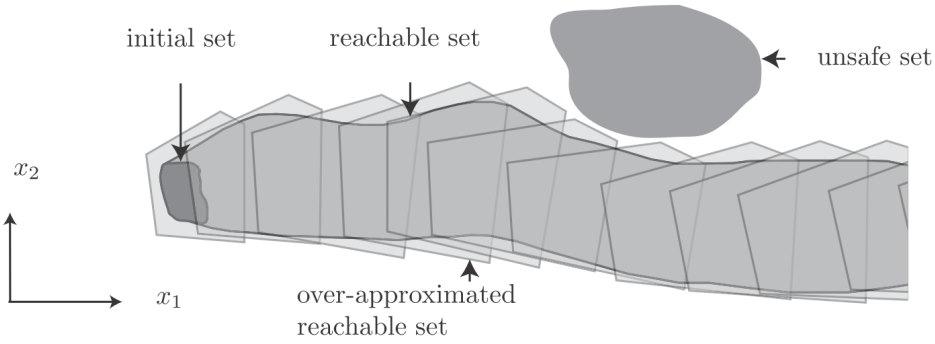


Figure 1.3: Verification using over-approximating sets (source [**?**])

However, if the reachable set is too much over-approximated, in case the error bound is much larger than its true value, it may yield unnecessarily over-conservative safety margins. For that matter, it is very important to chose a suitable mathematical representation for sets.

Another way to look at it, is to consider the vehicle as a stochastic system, yielding in this case a Stochastic Hybrid System [**?**]. In the present work, this way of modeling uncertainties will not be considered and instead one will consider the vehicle and its interactions with the world, as a deterministic process. This is a reasonable assumption given we work with adequate tolerances.

1.4.3 Viability Theory

As mentioned in the introduction, this thesis intends to use some aspects of the viability theory, such as viability kernels, in self-driving cars. The concepts of viability theory applied to robotics are not entirely novel. Aubin in [**?**], illustrated some practical applications where this theory can be used.

1.4.3.A Background

If we consider $x(t)$ as the state of the system at time t, this function is defined with $t \in [0, T]$ and is called viable in a set K if,

$$\forall t \in [0, T], \;\; x(t) \in K$$

The state of the system can be formulated in terms of a differential inclusion. This allow us to have a more general framework for systems' mathematical models.

$$\dot{x}(t) \in F(x(t)), \quad x(0) = x_0, \quad \text{for almost all } t \geq 0 \tag{1.1}$$

In this context, a set K is a viability domain of F if and only if,

$$\forall x \in K, \quad F(x) \cap T_K(x) \neq \emptyset,$$

where, $T_K(x)$ is the Bouligand contingent cone to the set K, at the point x [**?**]. We will not discuss, for now, whether this domains exist or not. From the previous formulation, we could synthesize a controller choosing the inputs as the feedback map, $R(x)$, such that,

$$\forall x \in K, \quad R(x) := F(x) \cap T_K(x) \neq \emptyset.$$

This feedback map guarantees that the system has atleast one trajectory that stays inside the set K. The *viability kernel*, $\text{Viab}_F(K)$, is defined as the *largest closed viability domain contained in* K.

1.4.3.B Light Autonomous Underwater Vehicle

A team from Porto's University dealt with the issue of finding minimum time trajectories for the specific case of a Light Autonomous Underwater Vehicle (LAUV) [**?**]. Their approach was to hybridize the LAUV model in order to better deal with environment characteristics such as the water currents present in the river, and to compute the viability kernel and capture basin by griding the state space and discretazing the system, adapting Saint-Pierre's algorithm [**?**]. The *capture basin* is the set of initial states such that at least one evolution starting there, is viable inside some set K, until it reaches some other set $C \subset K$ in finite time. Their algorithm consisted in assessing if each grid point is a viable point or not, checking also if it provided a better estimate of time to goal than the current estimate. While the viability kernel was computed, the set of inputs (possibly empty) that achieved minimum time was also returned by the algorithm.

1.4.3.C Pioneer 3AT Navigation

Using viability theory concepts, a navigation problem was implemented with a small robot, a Pioneer 3AT, equipped with odometers and GPS. When computing the viability kernels, it is possible to retrieve also the feedback map that one should apply in order to drive the vehicle from A to B, safely, in minimum time. The mobile robot was modeled with the simple non linear model,

$$\begin{cases} \dot{x}_1 = u\cos(\theta) \\ \dot{x}_2 = u\sin(\theta) \\ \dot{\theta} = \omega \\ u \in \mathcal{U}, \omega \in \Omega \end{cases}$$

For each position (x_1, x_2, θ), estimated by the sensors, the software returned a subset of inputs, $\mathcal{U} \times \Omega$, in order to keep the robot inside the constraint set.

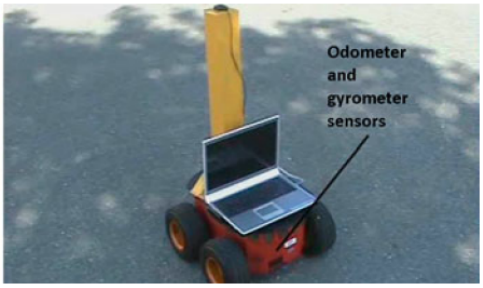


Figure 1.4: Equipment used for navigation problem (source [**?**])

1.4.3.D Miniature Racing Cars

More closely related to our problem, work in computing viability kernels for miniature race cars was done in [**?**]. Combining Model-Predictive Control (MPC) and viability theory, the authors were able to generate real-time trajectories to safely control miniature cars. A non-linear bicycle model was used for the car. Since the scope of the author's work was for racing cars, traveling at relatively high speeds, it makes perfect sense to take into account the non-linear dynamics of the tires. In their work, they also reduced the model to one with less state variables by 'slicing' the state space in null acceleration points. This points were computed for a certain longitudinal velocity and steering angle and changing, appropriately, the remaining variables, would make the acceleration to vanish.

It was also introduced a different kind of kernel to deal with discretization errors inspired in Game Theory, the *discriminating kernel*. This kernel is a result of considering the discretization error as an additive noise, involved in a dynamic game with the control input.

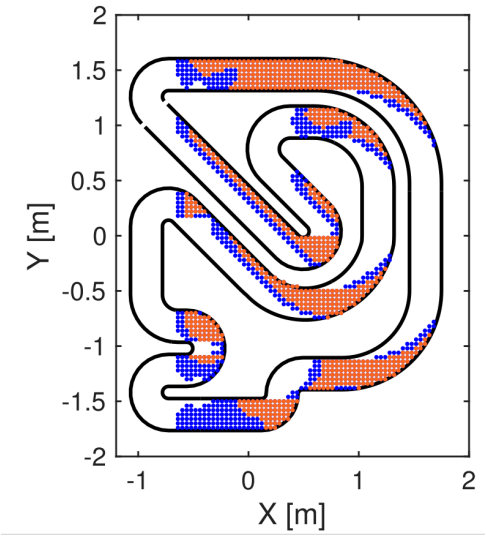


Figure 1.5: Blue dots represent exclusively the discrete viability kernel, the red ones represent both the viability and discriminating kernel for a particular orientation angle (source [**?**])

1.4.4 Approximating Viability Kernels

As told before, the main focus of this thesis is to improve existing algorithms to approximate the viability kernel of some systems and constraint sets.

1.4.4.A Linear Systems

The class of linear systems, described by,

$$\dot{x}(t) = Ax(t) + Bu(t),$$

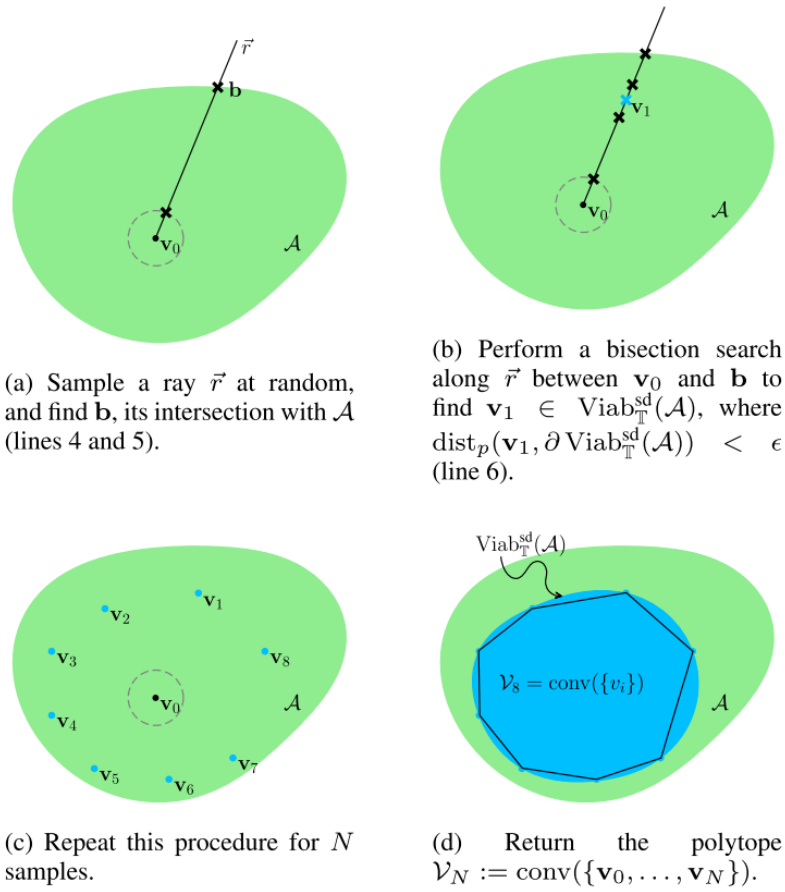
in the continuous time case, or,

$$x_{n+1} = A_d x_n + B_d u_n, \tag{1.2}$$

for discrete time systems, are provided with several algorithms to approximate (or in some cases, compute exactly) the viability kernel. Approximations of the viability kernel have to be always in the form of under-approximations to guarantee the safety of the system.

Our work is based on algorithms for linear systems developed in [**?**]. This algorithm under-approximates the viability kernel for sampled-data system, meaning, for this systems, the inputs must be piece-wise constant, since one can only change them once at each sampling interval. The method consists in sampling the boundary of the constraint set, making a line segment from a point known to be part of the viability kernel. From this line segment, the procedure extracts the point that is closest to the viability kernel boundary but is contained in it, up to some user-defined accuracy. An illustration of the algorithm

is in figure **??**.



(a) Sample a ray $\vec{r}$ at random, and find $\mathbf{b}$, its intersection with $\mathcal{A}$ (lines 4 and 5).

(b) Perform a bisection search along $\vec{r}$ between $\mathbf{v}_0$ and $\mathbf{b}$ to find $\mathbf{v}_1 \in \text{Viab}_{\mathbb{T}}^{\text{sd}}(\mathcal{A})$, where $\text{dist}_p(\mathbf{v}_1, \partial\,\text{Viab}_{\mathbb{T}}^{\text{sd}}(\mathcal{A})) < \epsilon$ (line 6).

(c) Repeat this procedure for N samples.

(d) Return the polytope $\mathcal{V}_N := \text{conv}(\{\mathbf{v}_0, \ldots, \mathbf{v}_N\})$.

Figure 1.6: Illustration of the Gillula's et al. algorithm (source [**?**]), where $\mathcal{A}$ is the constraint set being considered.

To decide whether some point is or not inside the viability kernel, the algorithm recurs to forward simulation of the system starting from that point. If the point belongs to the kernel, there exists at least one sequence of inputs, $u_0, u_1, ..., u_N$, such that all points from the trajectory stay inside the constraint set.

In [**?**] the authors review some methods for the approximation of such sets. First they present a method (*Exact Polytopic Method*) based on iteratively applying the backward reachable set - the set of all the initial states such that the trajectory of the system is in a given set in the next time step. For linear, discrete-time systems (**??**), the finite-time viability kernel, $\text{Viab}_n(\mathcal{K})$, for n steps, is computed as[2],

$$\begin{cases} \text{Viab}_0(\mathcal{K}) = \mathcal{K}, \\ \text{Viab}_{n+1}(\mathcal{K}) = \text{Viab}_0(\mathcal{K}) \cap A_d^{-1}(\text{Viab}_n(\mathcal{K}) \oplus B_d\mathcal{U}). \end{cases}$$

This methods rely upon having a convex polytope representation for sets since this representation

[2] Assuming an invertable matrix A_d.

is is closed under the operations used, linear maps and Minkowski sums ($\oplus$). However, due to successive Minkowski sums , this method is unfeasible for large number of variables, since it has exponential asymptotic complexity.

Afterwards they propose a method based on representing convex sets with supporting direction, since it is known that the support function $h_A : \mathbb{R}^n \mapsto \mathbb{R}$ of a non-empty convex set $A \subseteq \mathbb{R}^n$, is given by,

$$h_A(x) := \sup_{a \in A} x \cdot a.$$

Results in [**?**] state that Minkowski sums with support functions have an asymptotic complexity of $\mathcal{O}(1)$, since, for two convex sets A and B, $h_{A \oplus B}(l) = h_A(l) + h_B(l)$, for some direction $l \in \mathbb{R}^n$. This allows the authors to compute the viability kernels rapidly, with the desired accuracy, just by increasing the number of supporting directions.

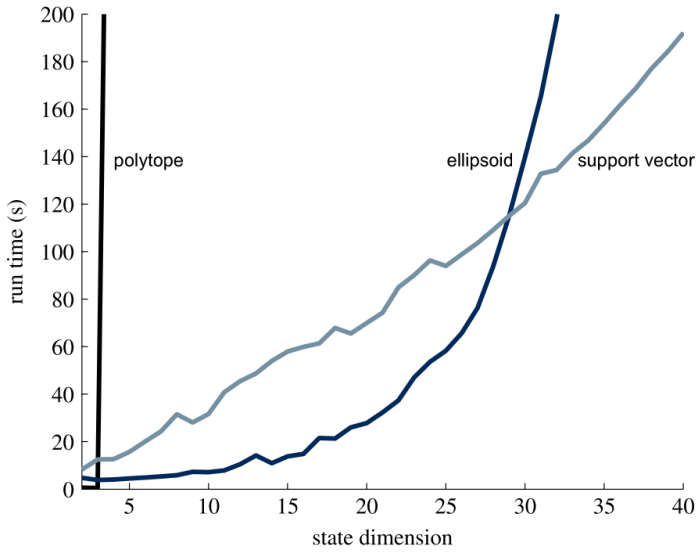


Figure 1.7: Performance comparison between a few methods (source [**?**])

1.4.4.B Non-Linear Systems

This class of systems generally lacks scalable algorithms. The ones that exist often have exponential complexity on the number of variables, making it virtual impossible to compute the reachable set for $n > 5$. One of the most known algorithms, based on griding the state space, was discussed in [**?**] by Saint-Pierre. This algorithm can be applied to systems described by differential inclusions (**??**) with set-valued maps having some properties such as convex images and upper semi continuity.

The first step consisted in approximating the differential inclusion with a difference inclusion, Γ_ρ, with

some time step, ρ, where,

$$\limsup_{\rho \to 0} \mathsf{Viab}_{\Gamma_\rho}(K) = \mathsf{Viab}_F(K).$$

Afterwards, it is needed to discretize the state-space with some step h. The griding consist in reducing the initial state space X to a subset of finite elements, X_h, where,

$$\forall x \in X, \qquad \exists x_h \in X_h, \quad \text{such that,} \quad ||x - x_h|| \leq h.$$

For the system,

$$\begin{cases} \dot{x}(t) = ax(t) - y(t), \\ \dot{y}(t) \in [-c, c], \\ x(t) \in [0, b], \\ y(t) \geq 0, \end{cases}$$

the algorithm produces sets like the one in the following figure **??**.

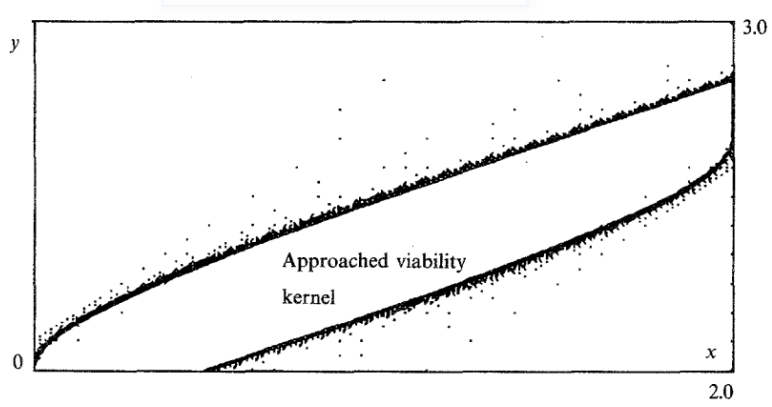


Figure 1.8: Viability Kernel computed with griding methods (source [**?**])

2

Autonomous System Architecture

This chapter reviews some architectures currently used in autonomous mobile robots. Such architectures, combining with the knowledge of how a human driver mentally processes the task of driving, inspired the novel architecture one presents here.

The main idea behind the intelligent system is to have a pre-computed set of 'safe' states, in the sense of viability theory, and at each sampling instant, one reduces the set inputs to just the ones that keep the system inside the viability kernel. The hybrid characterization of the system comes, precisely, from having the architecture and planning methods we propose. Since at each instant, the input set changes, its dynamics, described by a differential(difference) inclusion will also change, that would make it a variable structure hybrid system. It will be relevant throughout the thesis since it motivates the algorithms proposed in chapter **??** to compute viability kernels.

2.1 Human Driver Behavioral Model

The Theory of Risk Homeostasis, applied to vehicle drivers' behavior by Wilde in [**?**] states that a person while making decisions engages, in general, in a self-regulatory process. The experienced level of risk for some decision or behavior is compared to the tolerated level of risk the person is willing to undergo, eventually in a subconscious level, and its behaviors are adjusted accordingly. Basically, the author states that we all have a comfortable level of stress, which may need to change throughout the driving experience, at this point, the driver regulates his actions in order to ensure the experienced risk level stays around that mental state. This can be perceived as the feedback loop provided in figure **??**,

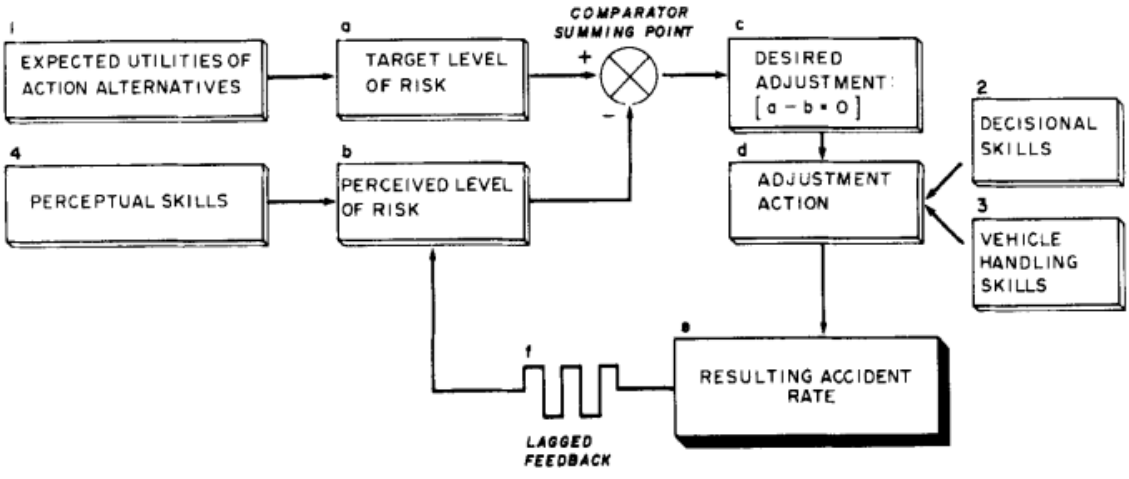


Figure 2.1: Homeostatic Risk Model of human driving behavior, source: [**?**]

Roughly speaking, we all have an internal target level of risk we are willing to take in order to perform some task, wanting to arrive sooner or later or to avoid speed-tickets. The regulation in driving occurs

when there is a mismatch between that target level and the risk currently perceived from the environment's conditions. That discrepancy yields an adjustment that must be made in the driving, making the perceived level of risk closer to the affordable level of risk.

In a different perspective, the author in [**?**] states that the risk of collision or harm is not what really matters in the decision making. The homeostases actually occurs at the level of the driving difficulty for driver. In fact, the author proposes that at the start of a journey or during it, the driver "determines a range of task difficulty that he/she is prepared to accept, a kind of target margin or envelope of the task's challenge" [**?**]. The most important variable considered to regulate the task difficulty is the vehicle's speed. The speed is modified by controlling the accelerator and/or brake pedal in order to match that perceived level. Another important variable is the distance to obstacles for example, distance to the gutter or to the front vehicle.

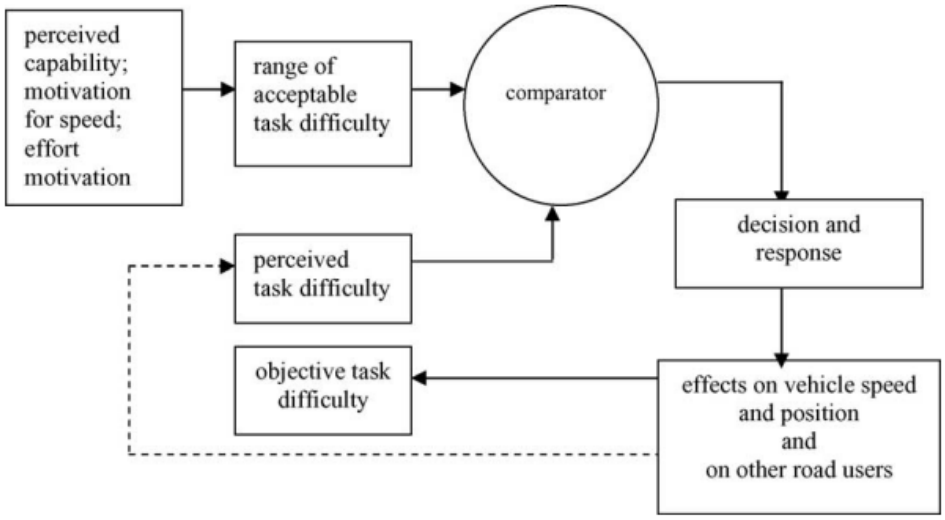


Figure 2.2: Task Difficulty homeostasis, source: [**?**]

The driver's self perceived capability of driving is among of what influences its acceptable task difficulty the most. Being in a rush or in a Sunday cruise along the city are motivational factors that are also taken into account for that acceptable level. The feedback loop closes when the driver perceives the new task difficulty upon actuating on the vehicle. It is the difference between the perceived task difficulty and the objective difficulty that is highly related to the probability of collision. That might explain why young drivers with poor experience in driving may wrongly assess the hardness of the driving task ahead, such as dangerous takeovers.

In a practical sense, both these models are equivalent. The only subtlety between them is that one says it is risk acceptance that matters in the decision making and the other one says it is the difficulty of the task that matters. Our work will enclose these two concepts in a single one, the uncertainty present in the task. The uncertainty is both linked to risk and driving difficulty and is easily adapted to a context

with mathematical formalism, whether in terms of probabilities or in terms of tolerances.

2.2 Autonomous Vehicle's Architectures

For a well functioning autonomous system, the top level architecture must be congruent and it must show how each subsystem should interact with each other. This architecture will strongly influence the mode of functioning of the lower level implementation.

In [**?**] the author proposes an hierarchical architecture. The aim of of such structure, depicted below (figure **??**), is to let the non-urgent planning be performed at the top-level, where it depends much less on the data from the perception.

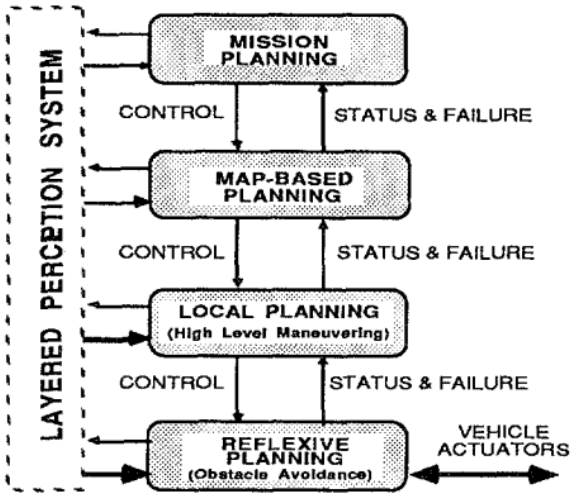


Figure 2.3: Hierarchical Architecture proposed in [**?**]

The lower level layers are the ones that must have lower delays between perception and action. The first one, *mission planning*, is responsible for transforming broader type of system inputs, such as where should the vehicle go, into more concrete ones, the constraints of the system while performing the task. Below that, the *Map-Based Planning* makes the decision part with the information given from the previous layer. More specific routes are delivered to the subsequent module. The *Local Planning* ensures that the plan received from *Map-Based Planning* is correctly applied. This layer decides which high level maneuvers are needed to be employed for the vehicle to perform the plan, such as, turning left, turning right, changing lanes, etc. Finally, *Reflexive Planning* is responsible for computing (relatively fast) the exact inputs of the vehicle actuators designed to match higher level instructions. There is one layer that communicates with all previous modules, therefore, being a common factor between them, the *Perception*. It receives raw inputs from the world and assigns meaning to it that can be used to plan and decide the tasks ahead.

In [**?**], the authors suggest an architecture similar to what is currently used in modern autonomous vehicles navigation problems. Figure **??** illustrates the relationship between each subsystem of the architecture, with the relevant inputs outside the system.

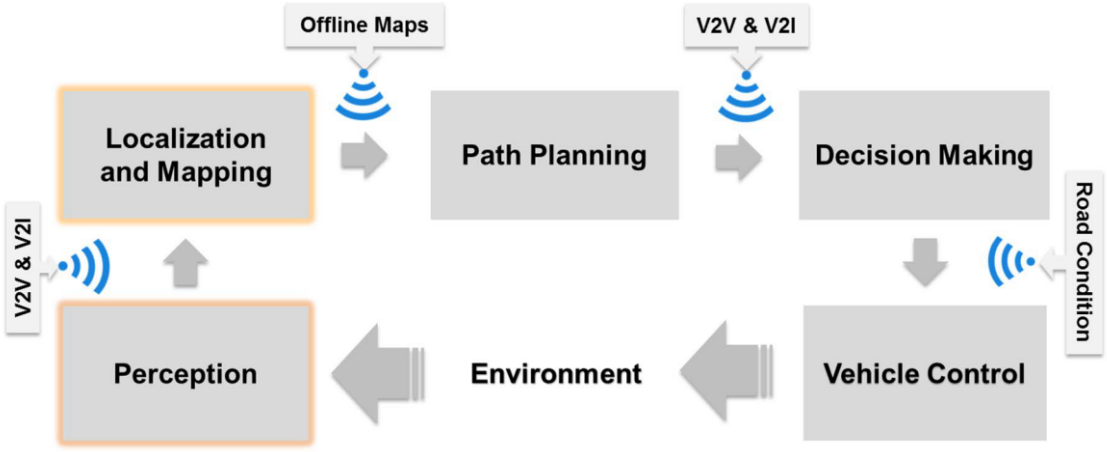


Figure 2.4: Architecture proposed in [**?**]

Here *Perception* is similar to the one discussed in the previous architecture. The *Localization and Mapping* block is responsible for localizing the vehicle in the world while mapping the environment, in real-time, combining this with the information provided from offline maps. *Path planning* uses the data provided from the previous module and determines the safe routes for the vehicle to take. Based on the safe routes allowed to the vehicle, the *Decision Making* block decides, based on the vehicle state, the current weather conditions and road attributes - road signs, traffic, road-works - the optimal path, with some criteria. Similarly to the *Reflexive Planning*, the *Vehicle Control* block is responsible for computing the exact inputs, motor torque, steering speed and such.

Although this architecture is not explicitly hierarchical, it is evident that the information from sensors is carefully distributed between blocks in some kind of layers - the ones requiring faster computations are the ones that compute the least amount information and make the fewest decisions.

2.3 Proposed Model

Taking into account how a driver behaves and how the flow of information must occur in a autonomous vehicle, we propose a top-level architecture for the intelligent part of the vehicle (Figure **??**). This model encompasses the main idea of how the human brain works while performing these kind of tasks. In this chapter we will explain how each module should be implemented, however, since this thesis will be focused essentially on the Local Planning module, only that module will have a detailed explanation - such as how to approximate the viability kernels - in Chapter **??**.

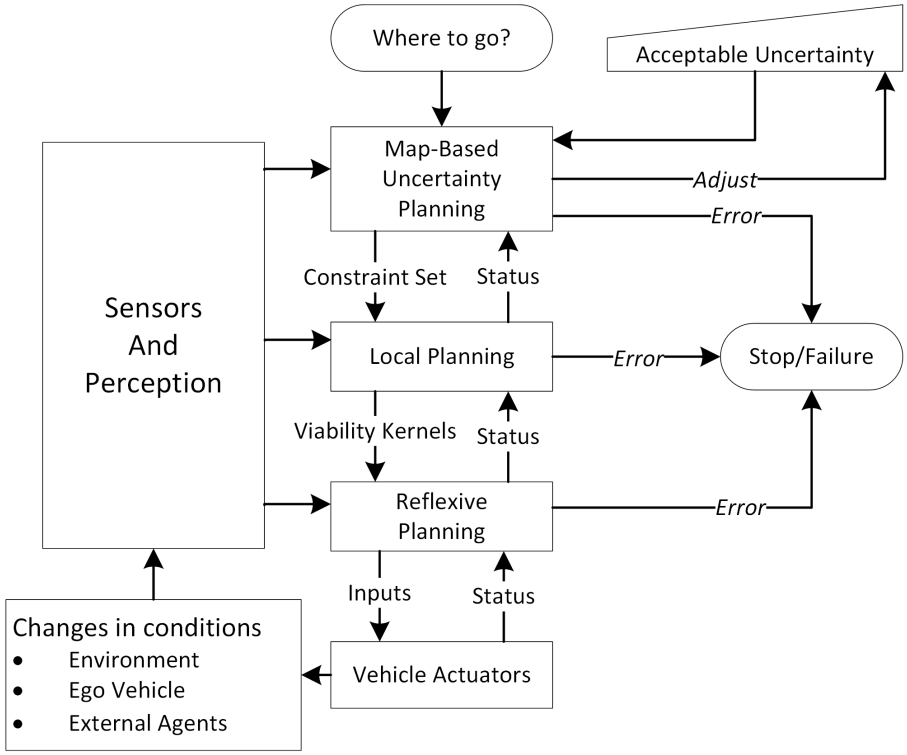


Figure 2.5: Top-Level Architecture of the cognitive part of the vehicle

The system starts by determining the vehicle's state, where the passenger would like to go and its acceptable uncertainty. It acts upon a tuning knob, adjusted by the user, selecting the acceptable uncertainty for the trip. Every system has some probability of occurring an error. Faulty data and too much loose tolerances are examples of things responsible for making the vehicle misbehave. Increasing the acceptable uncertainty 'tells' the system to lower the accuracy of computations and thus make the input set 'bigger', allowing higher velocities. It can happen that the system, for some route, hits its lower bound of uncertainty and enforces to be chosen a 'safer' uncertainty. This feedback may occur at any stage and if the system is already performing the mission, it should alert the passenger and initiate a safe stop procedure.

2.3.1 Map-Based Planning

The first layer of the vehicle architecture involves high-level decisions. In Map-Based Planning, the system accepts the destination and the acceptable uncertainty for the task. This uncertainty is a measure that reflects the accuracy and tolerances used for approximating the viability kernel and the controller specifications.

This task is solved by means of considering the world map as a directed graph, where streets are edges and intersections are nodes. For that, the system employs a shortest path search algorithm, such

as Dijkstra [**?**], to chose the best path to the destination following some criteria specified by the user, least time, least distance, least expected energy expended or a weighted combination of those.

With the route established, the system extracts the state-space constraints of each path segment with the adequate tolerances from the information available in the map. The scheme in figure **??** illustrates the process involved at this layer.

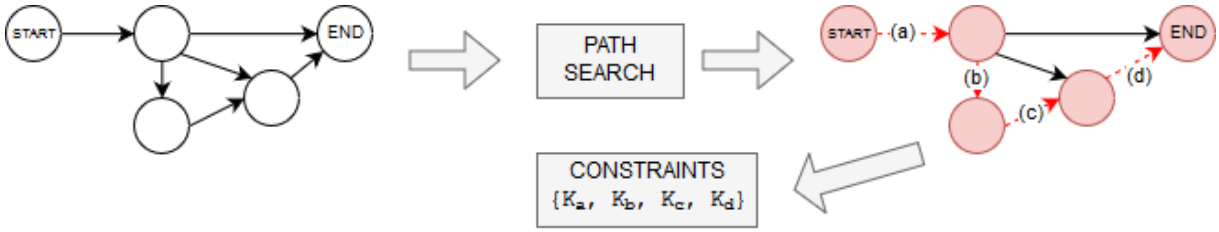


Figure 2.6: Computations involved in Map-Based planning

These constraints are, for example, the maximum allowed speed in that route segment, the maximum allowed steering angle or the lane dimensions. As stated, the uncertainty the user defines for the start up of the car greatly influences the constraint sets. This uncertainty intends to mimic the behavior of a human driver (section **??**) that adjusts its driving based on its perceived level of risk. The same would happen with the autonomous vehicle. Before-hand, the user defines his willingness to risk. For example, a more tolerant user would be able to get to its destination faster at the expense of having looser constraints that would increase the probability of having an accident (even slightly).

The system designed so, would impede the car from starting its mission if minimum conditions of safety were not met - too large tolerances and adverse road conditions would play a major role in this decision.

2.3.2 Local Planning

At this layer, the exact set of inputs allowed for safely driving the car, based on the viability kernel, is computed. Such inputs are defined as a feedback map,

$$R(\xi_k) := \{u_k \in \mathcal{U} \mid \xi_{k+1} \in \text{Viab}(K)\}.$$

This means that, for some trajectory with the discrete state ξ_k, one will reduce the original input set, $\mathcal{U}$ to $R(\xi_k)$. This reduced set is the set of all inputs that allow the state of the system to be inside the viability kernel, Viab(K), of the constraint set K, in the next sampling instant, ξ_{k+1}. The computations in this layer can and should be done before the vehicle starts its mission, since these are complex and will eventually take some time. An illustration of this process of selecting viable trajectories can be seen in figure **??**.

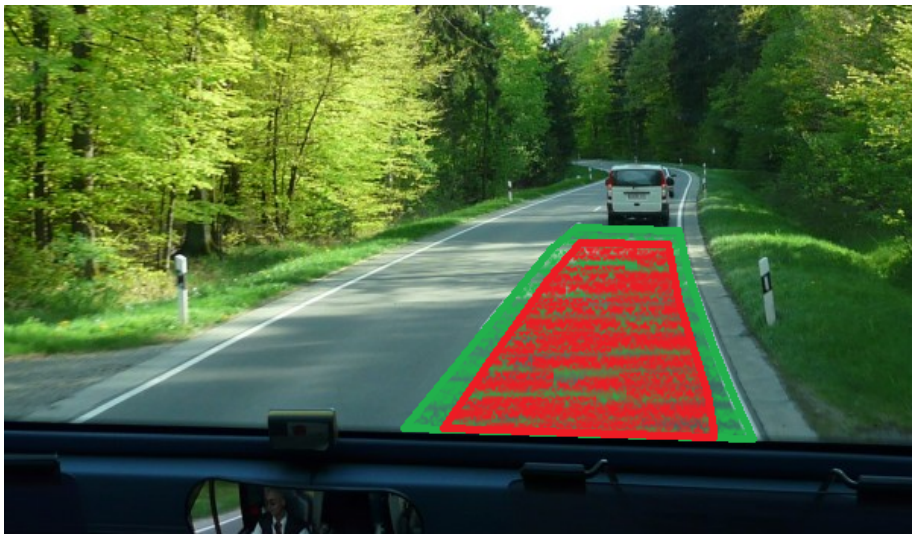


Figure 2.7: Illustration of the viability kernel in red, and the outermost set in green, the constraint set

One aspect of having an architecture lied out like this is that if there are only a finite number of uncertainty levels for the user to choose, all this information up to this point, Map-Based Planning and Local Planning, can be stored in memory and downloaded after the previous layer decided the route the vehicle should take. Online change of routes is still possible, since it suffices to extract this information from the newly included map segments.

The hybrid aspect of the vehicle comes from the fact that at each sampling time the dynamics of the vehicle change due to a change in the input space. The differential inclusion that represents the vehicle at each time is modified in order to keep the vehicle inside the viability kernel. At this stage, it is important to provide a discretization scheme for the differential inclusion, with the set-valued map with domain and images in subsets of metric spaces, $F : \mathcal{X} \rightsquigarrow \mathcal{Y}$,

$$\dot{x}(t) \in F(x(t)), t \geq 0,$$

assuming existence of solutions (more on this on Chapter **??**). The most common approach is the Euler explicit scheme [**?**],

$$\xi_{n+1} \in \xi_n + hF(\xi_n) := F_n(\xi_n).$$

Reducing this set-valued map to yield only viable trajectories,

$$\bar{F}_n(\xi_n) := \{F_n(\xi_n) \mid u_k \in R(\xi_n)\}.$$

Non-emptiness of $R(\xi_k)$ has to be guaranteed by the program itself, meaning, as soon as the set of usable inputs is the empty set the program sends a failure signal to the layer below and starts a safe

stopping procedure. The user is then advised to choose a more tolerant uncertainty level.

2.3.3 Reflexive Planning

The reflexive layer computes the exact input commands - steering velocity and engine torque - that put the vehicle closer to its goal safely. At this layer, some contingency measures should be present for the cases where unexpected dangers are detected by the Sensors and Perception block or to do maneuvers such as lane changing or parking.

For example, if the priority of the user is minimum time, then a controller that penalizes excessive time should be used. One that intends to minimize the probability of accident would penalize excessive speed and set greater safety distances from other vehicles and from the road edges.

A common controller that can be employed in this architecture is a Model Predictive Controller (MPC) for non-linear systems. This type of controller evaluates a receding horizon of sequences of controls/states that optimize some function. Mathematically, it involves minimizing,

$$J(x_k, u_k) := \sum_{i=k}^{k+T} l(x_i, u_i, i) + S(x_T, T).$$

Subject to,

$$u_k \in R(x_k), \quad x_{k+1} \in F_k(x_k).$$

The function $l(\cdot)$ is often called the stage cost function, while the function $S(\cdot)$ is the terminal cost. The criteria of choice for the driving task is embedded in both these functions. The inputs are restricted to the ones that guarantee safety, recalling that $R(x_k)$ is the subset of inputs that allows the system to stay inside the viability kernel. This approach is similar to previous ones such as in [**?**].

One important issue is how to deal with traffic. The data, from perception, for example, the distance to the front vehicle, should be included in the stage cost function in order to achieve some safe distance from other objects. Potential functions are a good way to model these objects and can be included in the stage cost function, $l(\cdot)$ [**?**].

2.3.4 Sensors and Perception

For a proper functioning of an autonomous system, there is the need for sensors. The sensing of the system extracts physical information about the environment and transforms it in a stream of data. The perception part uses that data and tries to assign meaning to it, which will be used to make decisions. The perception part, although simply explained here, is no easy task for a computer.

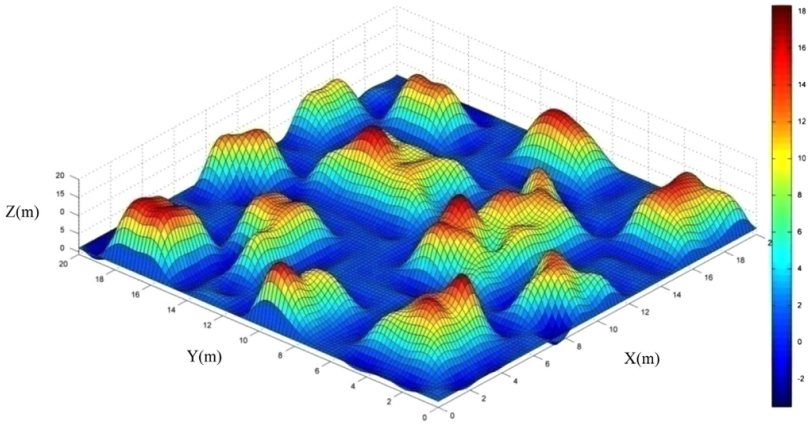


Figure 2.8: Artificial Potential Field (source [**?**])

This portion of the architecture is aimed at detecting the differences between the map and the real world. Traffic is an information that is often missing or fairly inaccurate in a map. Lanes may also differ from the maps due to road-works.

Sensors play an important role in assessing the state of the vehicle. Only with a well known state of the vehicle it will be possible for the Reflexive Planning block to correctly decide what set of actions are safe to be used. Information about the lane width, distance from other vehicles and obstacles is crucial. It is the data from this block that is used to model the environment and to usefully define the parameters of the cost function for the controller.

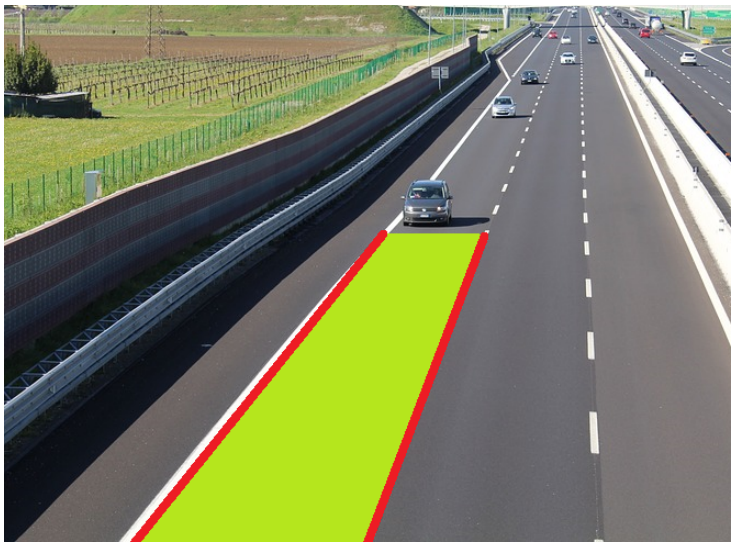


Figure 2.9: Lane Detection for Autonomous Vehicles

3

Vehicle Model

The model presented in this chapter is simplified enough to be easily implemented in our architecture but realistic enough to be used in practical applications. Since the aim of this model is to be used to compute viability kernels, it must be described by a first order differential equation, continuous in some compact convex domain and to be a convex function in each input variable.

3.1 Other Available Models

There are several vehicle models available in the literature, with a wide range of complexity. Some of them rely on several parameters that are usually hard to obtain in practice and have complex non-linear behaviors, such as when modeling tire stiffness. Many models take into account the existence of slipage between the road and the tire [**?**] [**?**], however, including this slip would make the dynamics equations lose some smoothness and continuity properties, which will be important in the context of computing viability kernels.

In [**?**] [**?**] for instance, the authors use a full four-wheel model of the vehicle. For any usual vehicle that does not have independent motors for each wheel, it is, most of the time, only necessary to account for the front and rear wheels as sets. In section **??**, that in some conditions the side-slip angle in the vehicle and in the tires can be negligible, supported by [**?**] [**?**].

3.2 Kinematics

The bicycle model is used. This means that both front and rear wheel set are collapsed in a single wheel, equivalent, in their respective axle. Vector variables are represented in bold, $\boldsymbol{x}$.

3.2.1 Cornering

When the steering angle, δ, is different from zero, i.e., the vehicle is cornering, we assume, in this steering model, that the front tire is traveling along a circumference of a certain radius. The center of gravity of the vehicle also travels along a circumference, concentric to the previous one but with smaller radius. The same happens with the rear tire but with an even smaller radius [**?**]. The center of these three circles is given by the intersection of the line that is perpendicular to the direction of the front wheel with the line that passes trough the rear axle (figure **??**). Since points *F*, *C* and *R* each travel their respective circumferences at the same angular rate (if one of them had a different rate, the vehicle would tear apart), one concludes that each point is traveling at a different absolute linear velocity, since the radius of each circumference is different.

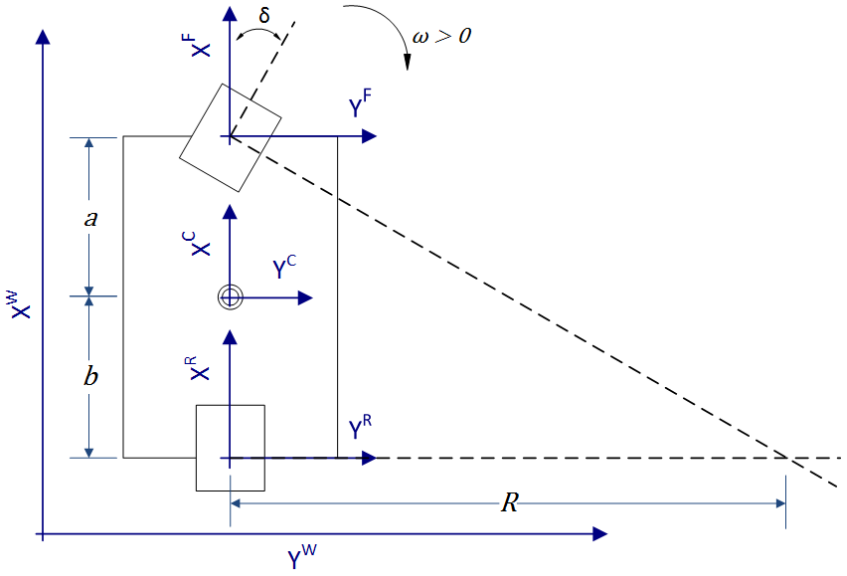


Figure 3.1: Vehicle Body Diagram

3.2.2 Translational and Rotational Motion

With the previous ideas one is capable of finding the positions and velocities of points F, C and R. The vectors from each point to the center of the curving circumferences are given by [**?**],

$$\boldsymbol{p_F} = \begin{bmatrix} -L \\ L\cot\delta \end{bmatrix}, \quad \boldsymbol{p_C} = \begin{bmatrix} -b \\ L\cot\delta \end{bmatrix}, \quad \boldsymbol{p_R} = \begin{bmatrix} 0 \\ L\cot\delta \end{bmatrix}.$$

Where L is the total distance between front and rear axles, with, $L = a+b$, the distance between the front axle and the center of mass is a and b is the equivalent to the rear axle. The vehicle body is assumed to rotate around its rear wheel, an explanation on why is found in [**?**]. The velocities of each point[1] are,

$$\boldsymbol{v_F} = \begin{bmatrix} v_x \\ L\omega \end{bmatrix}, \quad \boldsymbol{v_C} = \begin{bmatrix} v_x \\ b\omega \end{bmatrix}, \quad \boldsymbol{v_R} = \begin{bmatrix} v_x \\ 0 \end{bmatrix}.$$

The angular speed of this rotation motion, relates to the steering angle [**?**] [**?**],

$$\omega = \frac{v_x \tan\delta}{L} \tag{3.1}$$

3.3 Dynamics

In this section we will denote the canonical unit vectors as $\boldsymbol{e_i}$ in the direction i.

[1] The axes of the points considered here are always aligned with the orientation of the rear wheel (therefore, aligned with the Center Of Gravity), despite the steering angle.

3.3.1 Centripetal Force

When the vehicle is cornering, as said, it is traveling along a circumference, for that, it must exist a centripetal force, pulling the vehicle inwards, in the direction of the center of the circumference. This force can be decomposed as the sum of the forces pulling the front wheel and the rear wheel towards the center of curvature [**?**], $\boldsymbol{F_c} = \boldsymbol{F_c^{(R)}} + \boldsymbol{F_c^{(F)}}$.

$$\boldsymbol{F_c^{(R)}} = \frac{aM_v}{(a+b)}\boldsymbol{a_c^{(R)}} = \frac{aM_v}{L}\frac{|\boldsymbol{v_R}|^2}{|\boldsymbol{p_R}|^2}\boldsymbol{p_R} = \frac{aM_v v^2}{L^2}\tan(\delta)\boldsymbol{e_y}.$$

Likewise,

$$\boldsymbol{F_c^{(F)}} = \frac{bM_v v_x^2}{L^2}\tan^2(\delta)(-\boldsymbol{e_x} + \cot(\delta)\boldsymbol{e_y}).$$

The sum of both results in,

$$\boldsymbol{F_c} = \frac{M_v v_x^2}{L}\tan\delta\Big(-\frac{b}{L}\tan\delta\boldsymbol{e_x} + \boldsymbol{e_y}\Big) \tag{3.2}$$

Please note that $\boldsymbol{e_i}$ is the unit norm vector (versor) in the i direction.

3.3.2 Centrifugal and Tire Lateral Forces

When describing a curvilinear motion, there is another force, named centrifugal, that has equal magnitude but pushes in the opposite direction of the centripetal force, $\boldsymbol{F_{cf}} = -\boldsymbol{F_c}$. This centrifugal force is what passengers of a vehicle, that is curving, feel when they are pushed to the opposite direction of where the car is going.

$$\frac{1}{M_v}\big(F_{t,y}^{(F)}(\alpha_F) + F_{t,y}^{(R)}(\alpha_R) - F_{c,y}^{(C)}\big) = \dot{v_y} \tag{3.3}$$

The dynamics of the lateral speed of the vehicle [**?**], in its own referential, v_y, are related to the difference of the centrifugal force and the tires' lateral force generated by their deformation, as a function of the tire slip angle, α [**?**] [**?**] [**?**]. In this computations one considers a flat road in the direction transversal to the vehicle direction. If it is not the case, another term due to the gravity should be properly added.

$$\alpha_F \approx \delta - \arctan\Big(\frac{v_y + a\omega}{v_x}\Big) = \delta - \arctan\Big(\frac{v_y}{v_x} + \frac{a}{L}\tan\delta\Big) \tag{3.4}$$

$$\alpha_R \approx -\arctan\Big(\frac{v_y}{v_x} - \frac{b}{L}\tan\delta\Big) \tag{3.5}$$

Considering the y direction, pointed to the center of the curve. By inspecting eq. (**??**), if the tire stiffness, which is responsible for practically all tire lateral force, is not sufficient to compensate the

centrifugal force, v_y starts decreasing ($\dot{v_y} < 0$) and, eventually when $v_y < 0$, the vehicle starts having a lateral speed in the direction opposite to the center of the curve, making the car drift away from its path. Also, from eqs. (**??**)-(**??**),for a fixed δ and v_x, the decrease of v_y makes both α_F and α_R to increase as seen in the following figure.

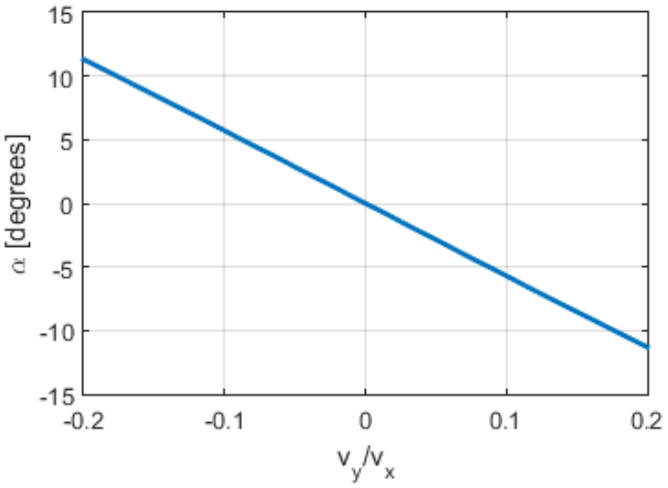


Figure 3.2: Relationship between v_y and α

These lateral tire forces are approximately proportional to the tire slip angle for small angles, $\alpha < 5^o$, and then, tend to get slightly smaller as α further increases (Fig. **??**). In the case of linear regime, when $|v_y|$ increases in the direction opposed to the center of curvature, the tires produce an even bigger lateral force, due to the increase of the tire slip, leading to a negative feedback loop, up to some point, stabilizing the car, while the slip is in the linear region.

As a result of having a lateral speed, the car, in general, does not follow the curved path one would want, it has some sort of drift. In what conditions is the assumption, in which there is no vehicle side slip, still be useful? In section **??** the model will be instantiated and the conditions for this will be assessed.

3.3.3 Wheels' Dynamics

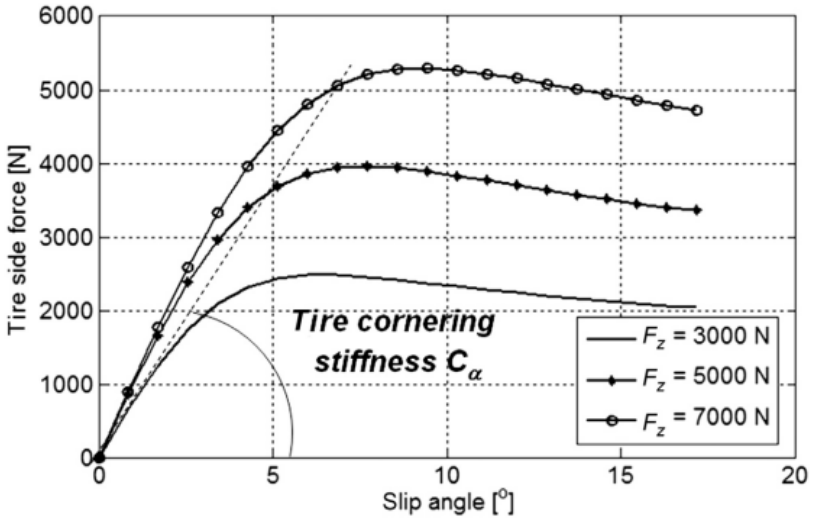
The diagram in fig. **??** shows the forces applied by the wheel and on the wheel.

From the moment and forces balance, one has the following equations for the front and rear axles,

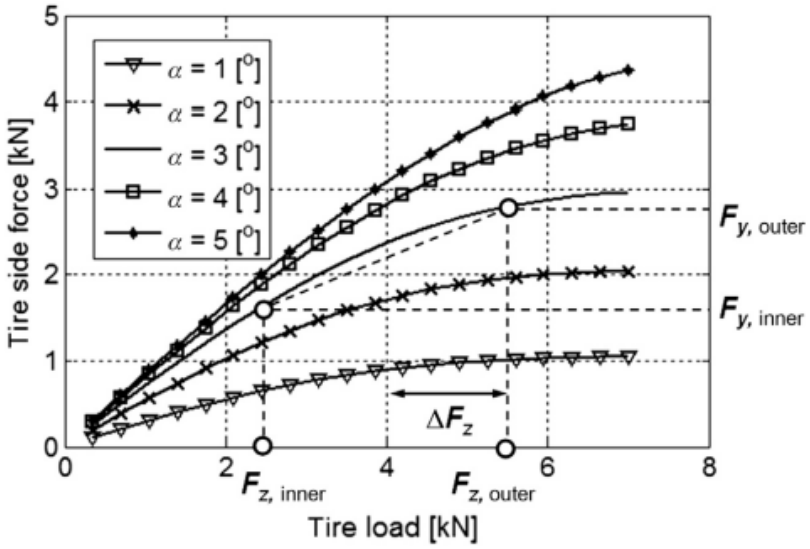
$$T^{(F)} = I_w \dot{\omega_F} = -F_{A,x}^{(F)} r_w - T_b/2 \tag{3.6}$$

$$T^{(R)} = I_w \dot{\omega_R} = T_e - T_b/2 - F_{A,x}^{(R)} r_w \tag{3.7}$$

Where T_e is the torque transfered from the engine to the wheels. The torque T_b is the total braking



(a) Tire side forces as a function of α



(b) Tire side forces as a function of tire load

Figure 3.3: Relationship between slip angle, tire load and side forces for small angles, source: [**?**]

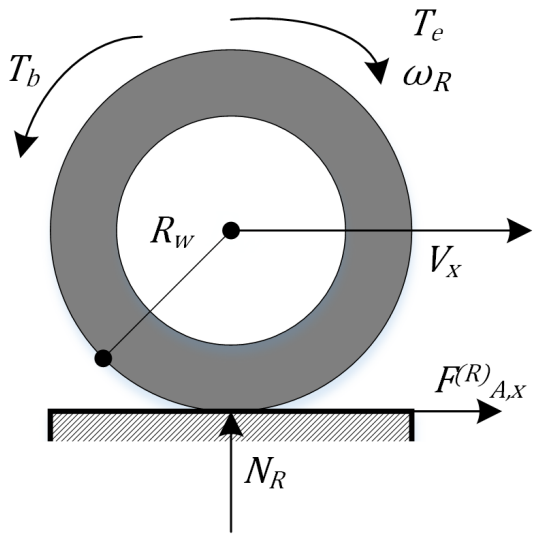


Figure 3.4: Forces, moments and velocities on the rear equivalent wheel

torque, here it is assumed that equal breaking is applied in both wheel sets. The vehicle has rear wheel traction. $\dot{\omega}_i$ is the angular acceleration of the $i \in \{F, R\}$ wheel set.

In the context of this work, the adherence between the ground and the tire is such that all the torque delivered to the wheels is used to propel the vehicle forward. This is valid both in acceleration and in braking. Traction control, for instance, does exactly this, it prevents excessive torque to be delivered to the wheels when adherence between surfaces is low. When breaking, the Anti-Lock Braking System (ABS) takes action in order to prevent the wheels from locking, which would induce also loss in steering [**?**].

3.3.4 Rotation Forces

To determine the angular acceleration around the point of rotation, which one determined to be the rear axle (Subsection **??**), taking the time derivative of **??**,

$$\dot{\omega} = \frac{\tan\delta}{L}\dot{v_x} + \frac{v_x}{L\cos^2\delta}\dot{\delta}. \tag{3.8}$$

The inertia around point R, I_R, is computed with the aid of the parallel axes theorem, find the inertia of an, approximated, cuboid around an axis trough its center of mass, I_C, and add a term that amounts for the inertia around another point, trough an axis parallel to the previous one,

$$I_R = I_C + M_v L^2 = \frac{M_v}{12}(4d^2 + L^2) + M_v L^2$$

.

Deriving the $\boldsymbol{e_y}$ component of the force using the torque produced by it at the point F,

$$\boldsymbol{T_r^{(F)}} = I_R \dot{\boldsymbol{\Omega}} \rightarrow F_{r,y}^{(F)} = \frac{I_R}{L}\dot{\omega}$$

Moreover, as of [**?**], the longitudinal component of the force acting on the front wheel is given by,

$$F_{r,x}^{(F)} = -F_{r,y}^{(F)} \tan\delta \tag{3.9}$$

3.3.5 Vehicle Longitudinal Dynamics

The main forces actuating on the vehicle are the ones between the engine and the wheels and each wheel between the ground. The traction force of each wheel are described as,

$$\boldsymbol{F_T^{(F)}} = \begin{bmatrix} \cos\delta & -\sin\delta \\ \sin\delta & \cos\delta \end{bmatrix} \boldsymbol{F_A^{(F)}} + \boldsymbol{F_r^{(F)}} + \boldsymbol{F_c^{(F)}} \tag{3.10}$$

$$\boldsymbol{F_T^{(R)}} = \boldsymbol{F_A^{(R)}} \tag{3.11}$$

Note that $\boldsymbol{F_A}$ is the force that is used to actually accelerate the vehicle, therefore, only the longitudinal component exists.

Since the movement is in the x direction, only this component from the traction force matters for the actual longitudinal vehicle motion, so,

$$F_T = F_{T,x}^{(F)} + F_{T,x}^{(R)} - F_d \tag{3.12}$$

Where F_d is the sum of the dissipative forces actuating in the car body, aerodynamic drag [**?**] and rolling resistance. The term due to gravity is omitted for simplicity,

$$F_d = \frac{1}{2}\rho C_d A v_x^2 + K_f(v_x)$$

With ρ being the air density, C_d the car drag coefficient and A the frontal area of the vehicle. K_f is a friction term, depending on the longitudinal velocity, this term reflects the rolling resistance and all the shaft and engine frictions, it is going to be modeled as $K_f(v_x) = K_{sq}\sqrt{v_x} + K_{li}v_x$. From [**?**],

$$\frac{F_T}{M_v} = \frac{\cos\delta r_w}{I_w} T^{(F)} = \frac{r_w}{I_w} T^{(R)} \tag{3.13}$$

Relating **??** with equation **??**,

$$F_{A,x}^{(F)} = -\frac{I_w}{M_v r_w^2 \cos\delta} F_T - \frac{T_b/2}{r_w}, \tag{3.14}$$

and for the rear axle,

$$F_{A,x}^{(R)} = -\frac{I_w}{M_v r_w^2} F_T - \frac{T_b/2}{r_w} + T_e/r_w. \tag{3.15}$$

Joining equations **??**-**??**, yields,

$$\frac{\dot{v_x}}{K_I} = T_e - (1+\cos\delta)T_b/2 - -r_w \frac{\tan\delta}{L}\left(\frac{bM_v \tan\delta}{L} v_x^2 + I_R\dot{\omega}\right) - r_w F_d. \tag{3.16}$$

With,

$$K_I = \frac{r_w}{M_v r_w^2 + 2I_w}$$

Equation **??**, gives the rotation motion dynamics, if the lateral dynamics are necessary, one should include equation **??**. Note that in the world coordinate system the transformation,

$$\begin{bmatrix} V_x^W \\ V_y^W \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix},$$

between the vehicles velocity and the velocities in the world frame has to be made. With $\dot{\theta} = \omega$.

Rearranging the terms and defining,

$$h(T_e, T_b, \delta, v_x) := K_I\Big(T_e - (1+\cos\delta)T_b/2 - r_w \frac{\tan^2\delta}{L^2} bM_v v_x^2 - r_w F_d(v_x)\Big),$$

$$g(\delta) := 1 + K_I I_R r_w \frac{\tan^2\delta}{L^2}.$$

We get the state-space formulation,

$$\begin{cases} \dot{v_x} = \frac{1}{g(\delta)}\big(h(T_e, T_b, \delta, v_x) - r_w I_R \frac{\tan\delta}{L^2\cos^2\delta}\dot{\delta}\big) \\ \dot{\theta} = \omega = (v_x \tan\delta)/L \\ \dot{\omega} = \frac{1}{g(\delta)}\big(\frac{\tan\delta}{L} h(T_e, T_b, \delta, v_x) + \frac{v_x}{L\cos^2\delta}\dot{\delta}\big) \\ \dot{x} = v_x\cos\theta \\ \dot{y} = v_x\sin\theta \\ \dot{\delta} = u_\delta. \end{cases} \tag{3.17}$$

In this setting, the input variables are, T_e, T_b and u_δ. Clearly this is a non-linear model, and the domain of validity of this model, (the domain of the state-space variables) should be carefully selected so as $g(\delta)$, for instance, does not equal zero.

3.4 Simulations and Results

We simulated the system to fine tune some of the model parameters, more specifically to extract the friction related coefficients, in order to have a similar acceleration performance to the real vehicle being modeled.

3.4.1 Vehicle Parameters

The vehicle being modeled in this work is the FIAT Seicento ELETTRA, an electric variation of the typical FIAT Seicento, depicted in the figure **??** down below. From the manufacturer manual, table **??**

Figure 3.5: Scheme of the electric vehicle being modeled, source: [**?**]

encompasses the parameters used in this model.

Table 3.1: Parameters Used

Parameter	Value	Description
r_w	$0.285m$	Wheel radius
I_w	$0.51kg.m^2$	Wheel inertia along rotation axis
M_w	$12.5kg$	Wheel Mass
M_v	$1200kg$	Vehicle total mass (No passengers)
L	$2.18m$	Total vehicle length
I_R	$6337kg.m^2$	Inertia around the rear axle midpoint
d	$0.63m$	Front/Rear Axle mid-distance
a, b	$1.09m$	Distance from the center of gravity to the front/rear axle
T_p	$1124N.m$	Motor peak torque
T_c	$562N.m$	Steady state motor torque
A	$1.84m^2$	Vehicle Frontal Area
ρ	$1.2kg.m^{-3}$	Air Density (20^oC)
C_d	0.3	Drag Coefficient

3.4.2 Friction Model

Before going deep in the simulations one has to confirm if the assumption of the torque being totally transfered from the wheel to the road, whether it is accelerating or breaking, in normal road conditions (dry asphalt flat road). In this model, the normal force in the traction set of wheels is, $N_R = M_v \frac{b}{a+b} g = 5886N$. The peak torque will be defined as the torque available at the motor for a relatively small amount of time (usually limited by the temperature of the copper windings). From the table **??**, $T_p = 1124$, implies that the minimum static friction coefficient for the tire not to slip is $\mu_s^{min} \approx 0.48$. This value is similar to the coefficient in a lightly wet road [**?**], this gives plenty of margin for the validity of this hypothesis, since a typical dry road has a coefficient of around 0.9. As discussed previously, the vehicles' manufacturer assures that traction control is being done in the vehicle, so, road-wheel traction is always guaranteed.

To find a suitable friction model, we've simulated the system varying its parameters. The friction model tries to encompass the losses from the rolling resistance, the transference of torque between the motor and the wheels and the ones in the wheels, tires, bearings and so on.

The series of simulations consisted in delivering the motor's peak torque to the wheels and adjust the friction model to match the car performance (Table **??**). After several trials the model is,

$$K_f(v_x) = 500\sqrt{v_x} + 30v_x, \quad v_x \geq 0 \tag{3.18}$$

This model resulted in the following speed performance with full torque, plotted with the vehicle specifications,

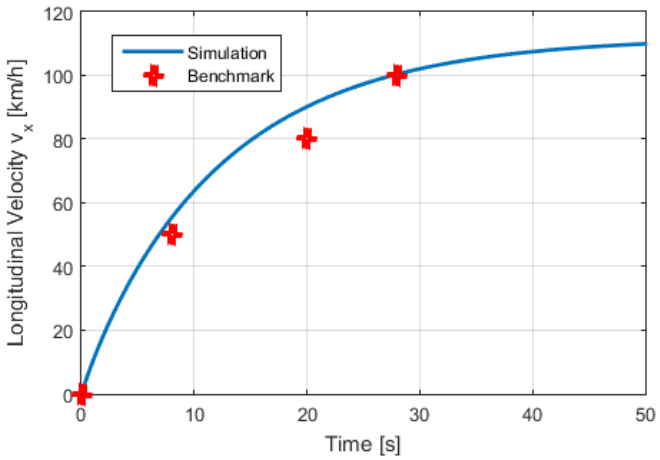


Figure 3.6: Simulation with peak torque

The simulation has a similar performance to the one specified in the vehicle model. Note that the exact test conditions for the benchmark, given by the manufacturer are unknown. Therefore, overfiting to this points is something undesirable for our model.

Table 3.2: Acceleration Performance

Velocity Range	Specification Time	Simulation Time
0-50km/h	8 s	7 s
0-80km/h	20 s	15 s
0-100km/h	28 s	28 s

3.4.3 Cornering Motion

Since we don't have any data about the real vehicle performance when cornering, we have to assume that any feasible and real-situation plausible result in the simulations is a proof of this model reliability.

The data provided in [**?**], which is shown in figure **??**, are used. Assuming $v_y = 0$, meaning, the car is currently on track and it is not drifting sideways, and with equations **??**-**??**, one can interpolate from the empirical data provided by figure **??**, these data was extracted to table **??**, for some particular steering angles.

Table 3.3: Data interpolated from figure **??**

δ	α_F	α_R	Total Side Force	Norm. Side Force
10^o	5.0^o	5.0^o	8000N	14.5
15^o	7.4^o	7.6^o	9200N	16.7
20^o	9.7^o	10.3^o	9400N	17.1

In figure **??** it is shown the conditions at which the vehicle starts having a lateral acceleration towards the exterior of the curve. The values for the centripetal force are normalized, meaning, the curves shown are valid for any vehicle since the normalized centripetal force was defined, based on eq. (**??**) as, $\hat{F_c} = \frac{F_c}{M_v/L} = v_x^2 \tan\delta$. In order to be able to compare forces, we normalized also the total tire side forces, $\hat{F_{t,y}} = \frac{F_{t,y}^{(F)} + F_{t,y}^{(R)}}{M_v/L}$.

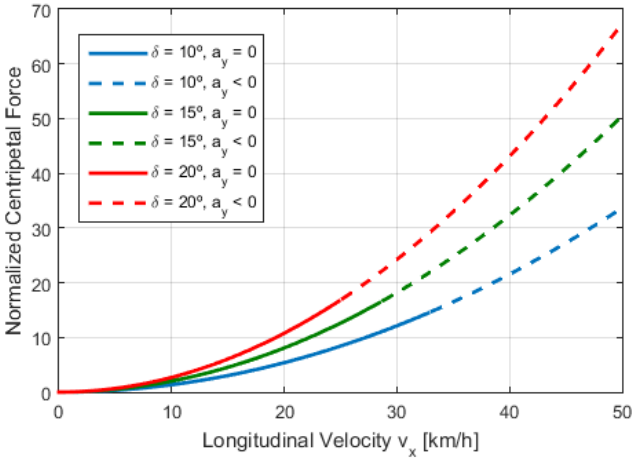
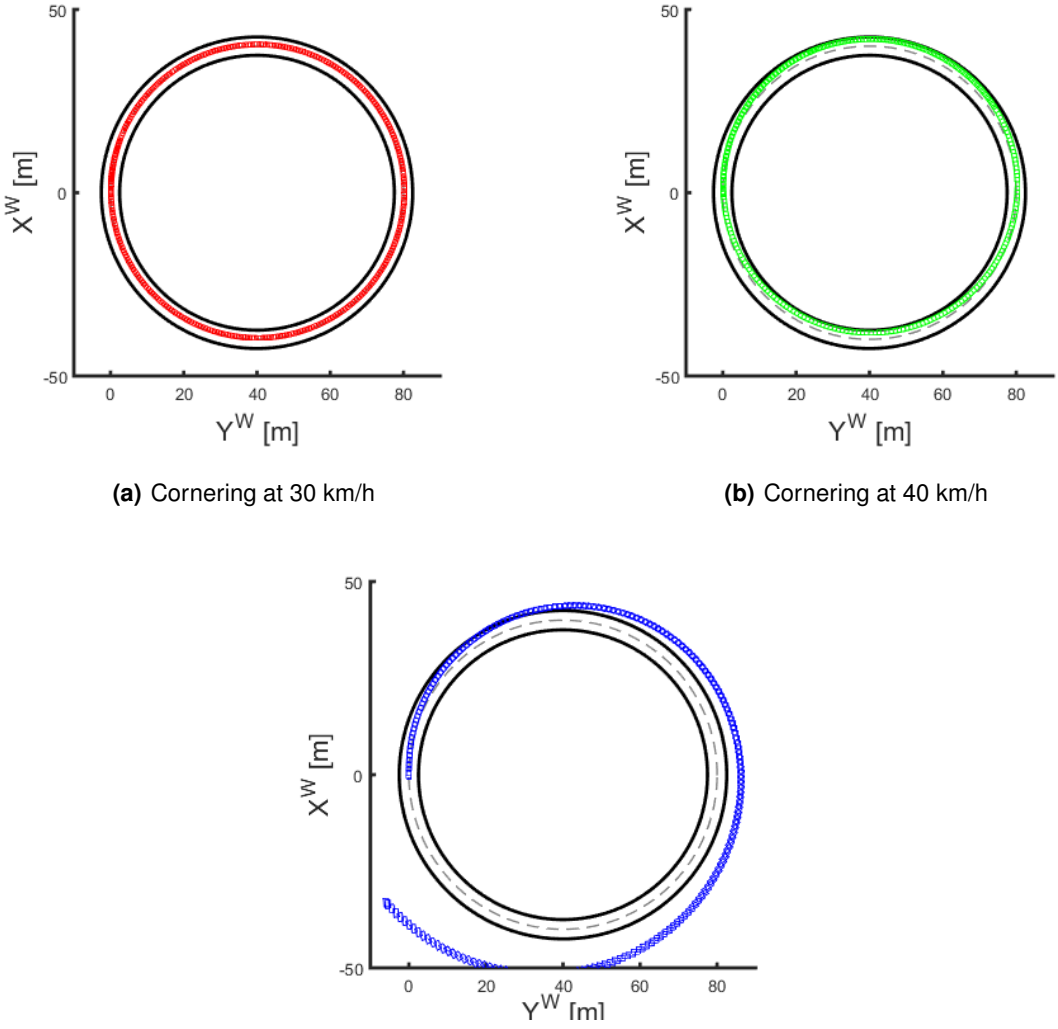


Figure 3.7: Limits on the lateral drift when cornering - Top curve is for $\delta = 20^o$, below, 15^o, and at the bottom, 10^o.

One sees that the typical limiting longitudinal speed for a correct following of the track is around 25

to $35km/h$ for a significant steering angle.

In order to illustrate this issue in a more concrete situation, we run a few simulations where the lateral dynamics (**??**) are taken into account. The simulation consisted in assessing the behavior of the vehicle at several longitudinal velocities when forced to drive in a curve of radius of $40m$ (for instance a roundabout). This radius of curvature corresponds to, $\delta = \arctan(L/R) \approx 3.1^o$. The simulations are presented in figure **??**.



(a) Cornering at 30 km/h

(b) Cornering at 40 km/h

(c) Cornering at 45 km/h

Figure 3.8: Performance in an average size roundabout of radius = 40m ($\delta \approx 3.1^o$)

The first simulation (the vehicle in red) is performed at a constant speed of $30km/h$. The lateral velocity is not substantial, being the longitudinal speed a safe one for this curve. The green car travels at a longitudinal velocity of $40km/h$. This performance is near the limit, in some points the vehicle touches

the boundary of the roundabout. For $45km/h$ there is no hope for the driver, it is an excessive speed for this curve and the vehicle is projected outwards the center of curvature.

3.4.4 Parameters Sensitivity

In order to determine the sensitivity to the parameters of the model, one has to define some metrics. Define the length of the graph of a discrete time function $y(t_i)$ with support N as,

$$L(y) := \sum_{i=1}^{N-1} \sqrt{(t_{i+1} - t_i)^2 + (y(t_{i+1}) - y(t_i)^2}$$

To determine the sensitivity of the model to parameters, we changed some of the parameters, one by one, and compared the performance with the one using nominal values. That comparison is done recurring to the root-mean-square difference of both function, defined as,

$$\text{diff}(y, f) := \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y(t_i) - f(t_i))^2}$$

Assuming the support of both y and f is N. Making this a relative difference, it requires the normalization of such measures,

$$\overline{\text{diff}}(y, f) := \frac{\text{diff}(y, f)}{L(y)}$$

Each parameter was changed in $\pm 10\%$ of its base value. The absolute difference of the velocity profile when changing plus or minus 10% is present in the second and third columns respectively. The average relative difference between the two values is computed in the fourth column,

Table 3.4: Parameter sensitivity in longitudinal acceleration

Parameter	x1.1	x0.9	Avg[%]
r_w	10.75	13.24	9%
M_v	2.57	2.71	2%
C_d	0.61	0.63	0.5%
m_w	0.03	0.03	0%
K_{sq}	6.81	7.34	5%
K_{li}	2.08	2.17	1.6%

3.5 Conclusions

Regarding the lateral dynamics, one should conclude that this aspect should not be totally neglected. The cases where there exists significant lateral acceleration, the Ackermann Steering [**?**] approach, as some authors make use of, cannot be freely used and the effects of the lateral acceleration should be

taken into account in the model. In this particular aspect of the work we neglect the lateral dynamics and simply constraint the range of acceptable values for the longitudinal velocity and steering angle. This information could be extracted automatically from a map which for each segment of road, computes the radius of curvature and the maximum allowed safe speed for that segment. More said, if even so, some lateral speed appears, one can always consider it as a perturbation in the orientation of the vehicle and control the steering angle via feedback to make the perturbations vanish.

Analyzing the sensitivity of the model parameters, in subsection **??**, it is evident that the most impacting parameters are the wheel radius and the friction's square root coefficient. The effective wheel radius, although no explicitly mentioned in the model, depends on the real tire radius, the weight of the vehicle, the tire vertical stiffness [**?**] and the tire pressure. As its impact in the performance of the vehicle model is significantly high, any change, or uncertainty in these variables that compose it will be greatly amplified in the total performance of the model. This should be an important aspect to take into account when modeling uncertainties. The same goes for the square-root friction coefficient used in (**??**). One should determine, empirically, a more reliable value for a more reliable model, since the one here determined was based on simulation and data sheets with no clear statement on the conditions their values were obtained.

4

Approximating Viability Kernels

As mentioned in previous chapters, one important step in our approach to the safety assessment and safe control of autonomous vehicles is to compute the viability kernel, the set of initial states that yield viable trajectories of the system inside some constraint set. In this chapter, one will be dealing with how to compute approximations of the viability kernel.

Using the framework of differential inclusions, Viability Theory [**?**], [**?**], provides a generalization to the differential equations, commonly used to model dynamics of vehicles that (i) has similarities to human-like strategies to model navigation problems and (ii) embeds uncertainties in a very intuitive way.

With differential inclusions, instead of the rate of change being at each instant a single value, in general, there is a set of possible values. Some conditions on the set-valued map that define those values ensure the existence of solutions, although not always unique, to the corresponding differential inclusion. Viability theory also states the conditions under which there exists at least one viable trajectory for the system - it is kept inside a given constraint set - given the inputs available.

Viability theory models the navigation of autonomous systems very similar to the way a human driver processes information and acts. There exists a set of trajectories that are viable - keep the car on track, safely - instead of the usual tracking of a specific parameterized curve in the state space. The viability kernel is a central piece in this theory, it is the largest subset of the constraint set, that starting in it, there exists at least one trajectory that is kept inside the constraint set.

The classical numerical method to compute the viability kernel for systems described by differential inclusions with Marchaud and Lipschitz set valued maps [**?**] is based on griding the state space, by Saint-Pierre in [**?**]. However, griding methods are known to have exponential complexity in the number of dimensions of the system. Several methods for high dimensional linear systems have been developed in the late years. Since the computation of viability kernels can be related to reachability [**?**] [**?**], the majority of them bring algorithms already used in reachability analysis [**?**] [**?**] to the spectrum of the viability theory, e.g., Lagrangian methods as presented in [**?**]. For example, the *Exact Polytopic Method* takes advantage of representing sets as *Polytopes*. This strategy is also used in this work. Methods based on support vectors, also presented in [**?**] win over the ellipsoidal methods [**?**] in terms of accuracy, since the former can be as accurate as one wishes, just by considering more supporting directions, and the latter is usually over-conservative.

The main procedure used in this work is based on the random sampling of supporting directions used in [**?**] for continuous linear systems. In this approach, one discretizes the model and obtains an under-approximation of the the true viability kernel for the continuous time system. The complexity of the original algorithm comes essentially from running the linear feasibility program, that represents the forward simulation of the system.

We modified the original feasibility program into a reachable set computation followed by testing if the reachable and the constraint set intersect each other. We prove that this adaptation yields the same,

correct, result. Unlike the original work, our adaptation can also be used in non-linear systems as long as the reachable set for that system exists and whether the overlapping test between two sets can be determined or not.

The most common representation for convex sets are polytopes and zonotopes [**?**]. Both objects are closed under the operations used in the computation of reachable sets of linear systems, such as Minkowski sums and affine-maps. However, polytopes have a major disadvantage when used in high dimensional systems. Due to its combinatorial nature, the Minkowski sum complexity grows exponentially in terms of space and in time complexity [**?**]. Zonotopes are more appealing for this kind of operations than polytopes [**?**], since the complexity is linear in the state space dimension.

This chapter presents, firstly, some mathematical background. Proper notation and definitions will be established. Then, we introduce the adaptation to high-dimensional, linear systems. The improvement here presented is later shown to be faster, although its asymptotic complexity remains similar. Afterwards, the algorithm is generalized to approximate also the viability kernel for some non-linear systems using some ideas in [**?**]. The overlapping test will be implemented as a simple Linear Programming optimization problem. At this stage we determine if the reachable set represented as a zonotope intersects the constraint set represented as a polytope without computing explicitly the intersection. The non-linear systems here approached are the ones described by first order differential equations with continuous derivatives and convex with respect to the inputs, which, as one will see, are guaranteed to have solutions.

4.1 Linear Systems

In this section, the mathematical background concerning linear systems will be revisited. First we consider linear systems without perturbations, meaning, the differential equation modeling the system is a linear combination of state variables and inputs only.

4.1.1 Background

Consider the linear system described by the differential equation,

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t), \quad \forall t > 0 \\ x(t) \in \mathcal{X}, u(t) \in \mathcal{U}, x(0) = x_0, \end{cases} \tag{4.1}$$

with $\mathcal{X} \subset \mathbb{R}^n$, $\mathcal{U} \subset \mathbb{R}^d$ compact, convex subsets.

Define the sampling time as ρ. The time horizon considered is denoted as $\bar{T}$, with the number of time steps being, $N_\rho = \lfloor \bar{T}/\rho \rfloor$. The discrete version, considering constant input between sampling intervals of the system,

$$x_{k+1} = e^{A\rho}x_k + \Big(\int_0^{\rho} e^{Ar}drB\Big)u_k, \quad \forall k \in \{0, ..., N_\rho - 1\}, \tag{4.2}$$

with,

$$A_\rho := e^{A\rho}, \quad B_\rho := \Big(\int_0^{\rho} e^{Ar}drB\Big).$$

The state of the system at some time step s, with some initial state x_0, is given by,

$$x_s = A_\rho^s x_0 + \sum_{i=0}^{s-1} A_\rho^i B_\rho u_{s-i-1}. \tag{4.3}$$

The viability kernel, mentioned a few times, but not thoroughly defined, consists in,

Definition 4.1. The finite-horizon sampled-data viability kernel of $\mathcal{K}$, is the set of all the initial states in $\mathcal{K}$ for which there exists at least one trajectory $x_k := x(k\rho)$, with piecewise-constant input, that stays inside the constraint set $\mathcal{K}$ for the time horizon $T = \rho N_\rho$.

$$\text{Viab}_T^{sd}(\mathcal{K}) := \{x_0 \in \mathcal{K} \mid \forall i \in \{0, ..., N_\rho - 1\}, \exists u_i \in \mathcal{U} : x(t) \in \mathcal{K}, \forall t \in [0, T]\}.$$

A key concept is also the reachable set. This is the set of all the final states possible for the system to be in a certain amount of time starting in some initial condition set.

Definition 4.2. The finite-horizon, reachable set of the system (**??**) is defined as,

$$\text{Reach}_{N_\rho}(\mathcal{K}) := \{x \in \mathcal{X} \mid x = x_{N_\rho}, x_0 \in \mathcal{K}, \forall j \in \{0, ..., N_\rho - 1\}, u_j \in \mathcal{U}\}. \tag{4.4}$$

4.1.2 Set Representations and Operations

The main type of representation of sets being used here are Polytopes and Zonotopes, they are common choices when one wants to represent convex sets. The choice on how to represent sets massively affects the kind of operations one can perform and how computationally efficient they will be.

Define the following set operations,

Definition 4.3. The Minkowski Sum and Pontryagrin Difference between two sets are defined, respectively as,

$$A \oplus B := \{a + b \mid a \in A, b \in B\},$$
$$A \ominus B := \{a \mid a \oplus B \subseteq A\}.$$

A ball is a set in some metric space $\mathcal{M}$, with some norm p, defined as,

$$\mathcal{B}_p(x, r) := \{y \in \mathcal{M} \mid ||y - x||_p \leq r\}.$$

4.1.2.A Polytopes

Since the constraint sets are often described as linear inequalities and equalities, a preferred type of representation is the polytopic. There exists two main types of polytopes, the $\mathcal{V}$-polytopes, represented by its vertex and $\mathcal{H}$-polytopes, represented by its facets.

A convex $\mathcal{V}$-polytope, represented by a finite set of points, $\{v_1, v_2, ..., v_k\}$, is the convex hull of such set of points.

$$P = \text{Conv}(\{v_1, v_2, ..., v_k\}).$$

If we stack every linear inequality that defines the polytope we have,

$$P = \{x \in \mathcal{X} \mid Hx \leq b\},$$

where the stacked matrix H and the vector b represent the intersection of all the half spaces (linear inequalities) that define the polytope.

4.1.2.B Zonotopes

A special class of convex polytopes are the zonotopes. They will be the main type of representation for reachable sets in this work. In [**?**] this representation is fully explained.

Definition 4.4. (Zonotope)

A zonotope is a set defined in an Euclidean space $\mathbb{R}^n$,

$$Z := \Big\{x \in \mathbb{R}^n \mid x = c + \sum_{i=1}^{p} \alpha_i g_i,\ -1 \leq \alpha_i \leq 1\Big\},$$

where, c is the center of the zonotope, and, $g_1, ..., g_p$ are its generators, all of them are vectors in $\mathbb{R}^n$. Denote the zonotope, Z, as the tuple,

$$Z := (c, < g_1, ..., g_p >).$$

The space of these objects is closed under affine transformations, $\mathcal{A}$, and Minkowski sums.

$$\mathcal{A}Z = (\mathcal{A}c, < \mathcal{A}g_1, ..., \mathcal{A}g_p >,$$

$$Z_1 \oplus Z_2 = (c^{(1)} + c^{(2)}, < g_1^{(1)}, ..., g_p^{(1)}, g_1^{(2)}, ..., g_s^{(2)} >).$$

The fact that these operations are simple and efficient to implement, they will be our object of election to represent reachable sets.

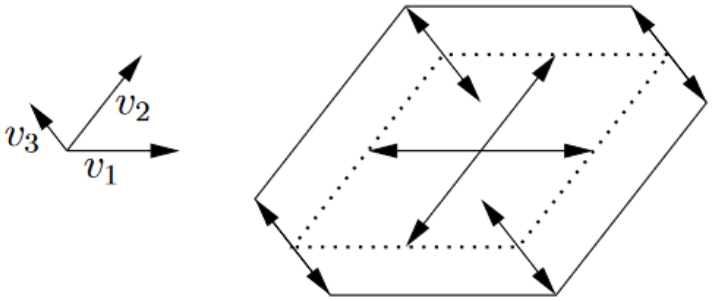


Figure 4.1: A zonotope with three generators (source [**?**])

4.2 Approximating Viability Kernels of Linear Systems

4.2.1 Basic Algorithm

The following algorithm is extracted from the work in [**?**] and will be the common part in our adaptation. The main idea behind it is to sample the boundary of the constraint set, $\mathcal{K}$, with N_r random directions originated at v_0, an initial point known to belong in the viability kernel. The line segment formed between v_0 and the point at the boundary of $\mathcal{K}$ is sequentially bisected, to find the outermost point that belongs in the viability kernel, up to some tolerance.

Algorithm 1 [**?**] Computes a polytopic under-approximation of $\text{Viab}_T^{sd}(\mathcal{K})$

function POLYTOPIC-APPROX($\mathcal{K}, \boldsymbol{v}_0, N_r$)
  $\mathcal{V} \leftarrow \{\boldsymbol{v}_0\}$
  for i from 1 to N_r **do**
    $\boldsymbol{r} \leftarrow$ SampleRay(v_0)
    $\boldsymbol{b} \leftarrow$ IntersectionOnBoundary$(\mathcal{K}, \boldsymbol{r})$
    $\boldsymbol{v}_i \leftarrow$ BisectionFeasibilitySearch$(\mathcal{K}, \boldsymbol{b}, \mathcal{K})$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{\boldsymbol{v}_i\}$
  end for
  return Conv$(\mathcal{V})$
end function

Functions SampleRay(v_0) returns a random vector sampled uniformly from the unit box in n dimension starting at v_0. IntersectionOnBoundary$(\mathcal{K}, \boldsymbol{r})$ outputs the point of intersection of the boundary of $\mathcal{K}$ in the direction $\boldsymbol{r}$. The BisectionFeasibilitySearch$(\cdot)$ function returns the point closest to the boundary of the viability kernel, with user-defined accuracy ϵ, that is contained in it.

The output of the function described in Algorithm **??** is contained in the viability kernel as shown in Theorem **??**.

Theorem 4.5. *(From [?]) For the linear system defined in* (**??**)*. Given convex sets $\mathcal{K}$, $\mathcal{U}$, a number of random directions, N_r and an initial point $\boldsymbol{v}_0 \in$ Viab$_T^{sd}(\mathcal{K})$, and let, $\mathcal{S} =$ Polytopic-Approx$(\mathcal{K}, \boldsymbol{v}_0, N_r)$, then,*

$$\mathcal{S} \subseteq \text{Viab}_T^{sd}(\mathcal{K}).$$

4.2.2 Sampled-Data System

Since one is dealing with a sampled-data system, to ensure that the original continuous trajectory is inside the constraint set, the original set is eroded, and the algorithm applied to the eroded set,

Admit that the velocity, $\dot{x}(t)$, of some system is bounded for any t, in the subset $\mathcal{X}$. Using the ∞-norm,

$$||\dot{x}(t)||_\infty \leq M, \forall t > 0.$$

Lemma 4.6. (From [**?**]), consider the system (**??**) with M being the bound, in the ∞-norm of $\dot{x}(t)$. The trajectory $x(t)$ is being sampled, with a sampling interval ρ. Define the set

$$\mathcal{K}^\flat := \mathcal{K} \ominus \mathcal{B}_\infty(0, M\rho). \tag{4.5}$$

For some initial value x_0, if there exists a sequence of input controls $\{u_0, u_1, ..., u_{N_\rho}\}$ such that, $x(k\rho) \in \mathcal{K}^\flat$, $\forall k \in \{0, ..., N_\rho\}$, then,

$$x_0 \in \text{Viab}_T^{\text{sd}}(\mathcal{K}).$$

4.2.3 Error Bounding

Gillula's *et al.* algorithm and ours discretize the system. That procedure will induce errors in the computation of the feasibility of the desired point, more precisely, truncation errors caused by the Taylor expansion of the matrix exponential computed up to a finite order of the power series. The authors provide a bound of such error and its propagation along the iterations. The following lemma shows how to account for any bounded error.

Lemma 4.7. ([**?**]), define $e_k = x_k - \hat{x}_k$ as the error between the the real system state x_k and the sampled model used $\hat{x}_k$ at the sampling instant k. Admit that $||e_k|| \leq \gamma_k$.

$$\mathcal{K}_k^\flat := \mathcal{K}^\flat \ominus \mathcal{B}_\infty(0, \gamma_k) \neq \emptyset.$$

Then,

$$\hat{x}_k \in \mathcal{K}_k^\flat \implies x_k \in \mathcal{K}^\flat.$$

Although this error is of utter importance, one will not provide here a precise bound for such error, truncation error bound is given in [**?**]. Any bounded perturbation of the system can be included as the error. This feature will be later used to encompass linearization errors from the non-linear algorithm.

4.2.4 Feasibility Program

The procedure modified in this work is the point feasibility assessment. The original algorithm in [**?**] checks if the samples from the trajectory of the system, starting from x_0, are feasible in the constraint set, applying the formula (**??**) for each sampling instant. An initial point x_0 of the system is feasible in some set if there exists a sequence of inputs that keep the system inside the set for some (possibly infinite) time.

Let d_i be the dimension of the input space. The original linear program solves a $d_i N_\rho$ dimensional problem subject to $2d_i + nN_\rho$ inequality constraints, recalling that n is the state-space dimension.

4.2.5 Modified Algorithm

From the original algorithm, the feasible point assessment is replaced by Algorithm **??**.

Algorithm 2 Determines if the evolution of the system with initial set X_0 is feasible at each iteration k in the set $\mathcal{K}_k^\flat$, using reachability.

```
function FEASIBLE(N_ρ,ρ,S,X_0,{K_1^♭, ..., K_{N_ρ}^♭})
    for k from 1 to N_ρ do
        X_k ← Reach_S(ρ[k − 1, k], X_{k−1})
        if not Overlaps(X_k, K_k^♭) then
            return False
        end if
    end for
    return True
end function
```

Proposition 4.8. Consider a point $x_0 \in \mathcal{X}$, if FEASIBLE$(N_\rho, \rho, \mathcal{S}, x_0, \{\mathcal{K}_1^\flat, ..., \mathcal{K}_{N_\rho}^\flat\})$ returns *True*, then $x_0 \in \text{Viab}_T^{\text{sd}}(\mathcal{K})$, with $T = \rho N_\rho$.

Proof: From the algorithm returning *True* we have,

$$\forall k \in \{1, ..., N_\rho\}, X_k \cap \mathcal{K}_k^\flat \neq \emptyset \Rightarrow \forall s \in \{0, ..., N_\rho - 1\}, \exists u_s \in \mathcal{U} : \hat{x}_{s+1} \in \mathcal{K}_{s+1}^\flat. \tag{4.6}$$

Using Lemma **??**, and rearranging the quantifiers, (**??**) implies that,

$$\forall s \in \{1, ..., N_\rho\}, \exists \{u_0, ..., u_{N_\rho - 1}\} : x_s \in \mathcal{K}^\flat.$$

By Lemma **??**,

$$x_0 \in \mathsf{Viab}_T^{sd}(\mathcal{K}).$$

□

The complexity of this modified function depends essentially on how $\mathsf{Reach}(\cdot)$ and $\mathsf{Overlaps}(\cdot)$ are implemented. When the exact reachable set is not possible to compute, one may under-approximate it and still use the algorithm. The viability kernel computed this way is contained in the one we might got using the exact reachable set computation.

Proposition 4.9. Consider Algorithm **??**, with line 3 replaced by the under-approximation of the reachable set,

$$\overline{X}_k \leftarrow \mathcal{P}_k \subseteq \mathsf{Reach}_{\mathcal{S}}(\rho[k-1, k], X_{k-1}), \quad \forall k = \{0, 1, ..., N_\rho\}$$

If $\overline{X}_k \cap \mathcal{K}_k^\flat \neq \emptyset, \forall k \in \{0, 1, ..., N_\rho\}$, then, we are able to construct an under-approximation of the originally intended viability kernel,

$$x_0 \in \mathsf{Viab}_T^{sd,\flat}(\mathcal{K}) \subseteq \mathsf{Viab}_T^{sd}(\mathcal{K}).$$

Proof: By proposition **??**,

$$\forall k \in \{1, ..., N_\rho\}, \overline{X}_k \cap \mathcal{K}_k^\flat \neq \emptyset \Rightarrow x_0 \in \mathsf{Viab}_T^{sd,\flat}(\mathcal{K}).$$

Here $\mathsf{Viab}_T^{sd,\flat}(\mathcal{K})$ represents the viability kernel of the system whose step reachability is $\overline{X}_k$. Also, from the original reachability, $X_k \leftarrow \mathsf{Reach}_{\mathcal{S}}(\rho[k-1, k], X_{k-1})$, with $\overline{X}_k \subseteq X_k$, it implies that,

$$x_0 \in \mathsf{Viab}_T^{sd}(\mathcal{K}).$$

Consider now the case where,

$$X_k \cap \mathcal{K}_k^\flat \neq \emptyset, \ \overline{X}_k \cap \mathcal{K}_k^\flat = \emptyset.$$

By prop. **??**, $x_0 \in \mathsf{Viab}_T^{sd}(\mathcal{K})$, but, $x_0 \notin \mathsf{Viab}_T^{sd,\flat}(\mathcal{K})$.

By the definition of set inclusions,

$$\mathsf{Viab}_T^{sd,\flat}(\mathcal{K}) \subseteq \mathsf{Viab}_T^{sd}(\mathcal{K}).$$

□

4.2.6 Computing Reachable Sets for Linear Systems

An efficient algorithm (Alg. **??**) to compute the reachable sets of linear discrete time systems is found in [**?**]. This method intends to avoid the wrapping effect due to successive Minkowski sums. Doing the Minkowski sum operation in high-dimensional spaces is very demanding computationally, its time complexity is exponential in the dimension of the set when using polytopes, [**?**]. The use of zonotopes becomes obvious, since the algorithm recurs a lot to this operation and has linear complexity.

Algorithm 3 Feasibility with linear time-invariant systems using [**?**]

function FEASIBLE(N_ρ,ρ,x_0,A_ρ,W,$\{\mathcal{K}_1^\flat, ..., \mathcal{K}_{N_\rho}^\flat\}$)
  $V_0 \leftarrow W$
  $S_0 \leftarrow \{0\}$
  for k from 0 to $N_\rho - 1$ **do**
    $X_{k+1} \leftarrow A_\rho(X_k)$
    $S_{k+1} \leftarrow S_k \oplus V_k$
    $V_{k+1} \leftarrow A_\rho(V_k)$
    $\Omega_{k+1} \leftarrow X_{k+1} \oplus S_{k+1}$
    if not Overlaps($\Omega_{k+1}, \mathcal{K}_{k+1}^\flat$) **then**
      return False
    end if
  end for
  return True
end function

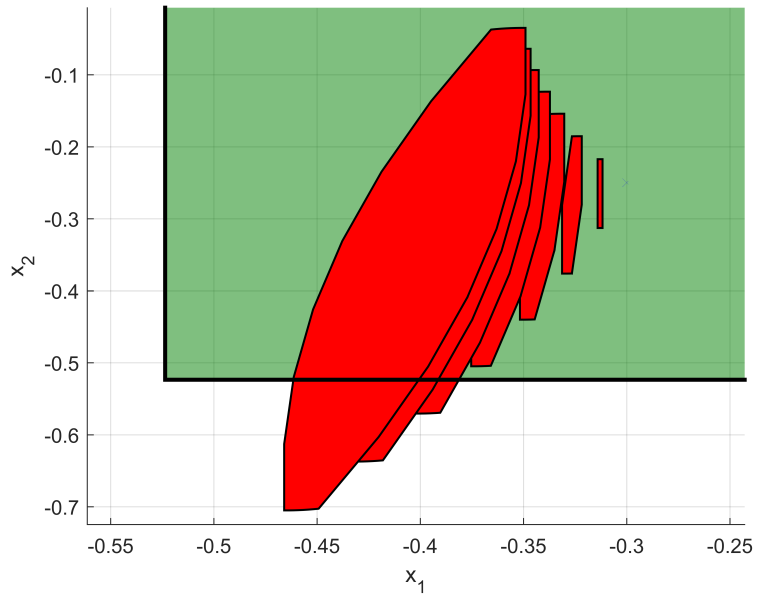
For the sake of simplicity, define $W := B_\rho \mathcal{U}$. From [**?**] and adding the complexity at each iteration to assess if two sets overlap, the bound for this algorithm is $\mathcal{O}(N_\rho(\mathcal{M} + \mathcal{A} + \text{Ov}(N_\rho, n)))$. The complexity of the Minkovski Sum is denoted by $\mathcal{M}$ and $\mathcal{A}$ is the bound for applying affine maps to a zonotope. In our case, $\mathcal{M} = \mathcal{O}(n)$ and $\mathcal{A} = \mathcal{O}(n^2)$. The complexity of computing the overlap between a zonotope and a polyhedron in $\mathcal{H}$-representation is denoted by $\text{Ov}(N_\rho, n)$, depending on the number of time steps and in the state space dimension (see ahead).

4.2.7 Overlap between a zonotope and a $\mathcal{H}$-polytope

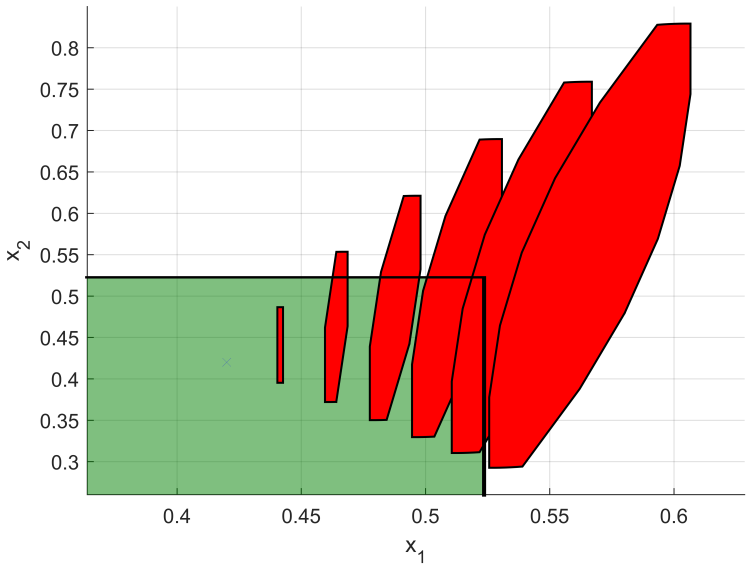
In order to determine if there are trajectories that keep the system inside the constraint set, the test for overlap between the reachable set and the constraint set must be performed. Using the aforementioned definition of zonotope, one will pose this problem as a simple feasibility program. The program will consist in determining the existence of the coefficients α_i such that the zonotope generated respects the inequality constraints derived from the facet representation of the constraint set. The inequalities from the constraint set are in the form of,

$$A\boldsymbol{x} \leq b. \tag{4.7}$$

Defining the matrix where the generators of the zonotope defining the reachable set are its columns



(a) Reachable sets from initial state, $(-0.3, -0.25)$.



(b) Reachable sets from initial state, $(0.42, 0.42)$.

Figure 4.2: Illustration of the algorithm, the light colored (green) region is the constraint set, the darker (red) ones are the reachable sets at each time instant. In (a), the initial state is viable for eight time steps. No viable trajectories exists for more than six steps in (b).

and the vector of the corresponding coefficients,

$$G = \begin{bmatrix} \boldsymbol{g_1} & \cdots & \boldsymbol{g_p} \end{bmatrix}, \qquad \boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 & \cdots & \alpha_p \end{bmatrix}^T.$$

Substituting x in (**??**) by the definition of zonotope, with column-vectors, yields,

$$A(\boldsymbol{c} + G\boldsymbol{\alpha}) \leq b.$$

This is equivalent to the standard notation of inequalities for optimization with $\boldsymbol{\alpha}$ as the variable to optimize,

$$AG\boldsymbol{\alpha} \leq b - A\boldsymbol{c} \Leftrightarrow A^*\boldsymbol{\alpha} \leq b^*.$$

Other restrictions have to be added since each α_i only assumes values in $[-1, 1]$, so, the optimization problem becomes,

$$\begin{array}{lrl} \min\limits_{\boldsymbol{\alpha}} & 0 & \\ \text{s.t.} & A^*\boldsymbol{\alpha} & \leq b^*, \\ & I\boldsymbol{\alpha} & \leq 1, \\ & -I\boldsymbol{\alpha} & \leq 1. \end{array} \tag{4.8}$$

The Overlaps$(\cdot)$ function, returning whether a zonotope Z and a polytope P intersect each other or not, is then defined as,

$$\text{Overlaps}(Z, P) = \begin{cases} \text{True}, & \exists \boldsymbol{\alpha} \text{ in (??)} \\ \text{False}, & \text{o.w.} \end{cases}$$

For each feasible point computation, it is solved *at most* a $N_\rho p$ dimensional problem with $(h+2)N_\rho p$ constraints, where h are the number of facets of the polytope representing the constraint set and p, the number of generators of the zonotope. The number of computations, for a point x_0 that is not feasible, is largely reduced, for instance, if the first time steps computing each of the reachability set do not overlap with the constraint set, the algorithm terminates.

4.2.8 Results

This section presents simulation results that point to the correctness of the algorithm, and, as well, its time performance.

For visualization purposes, we opted to compute first the viability kernel for a double-integrator in two dimensions, using similar parameters to the original algorithm. For the sake of simplicity the erosion process of the constraint set is omitted. To assess the scalability and the asymptotic performance of the aforementioned improvement, the algorithm was also used to determine the viability kernel of a chain of n integrators.

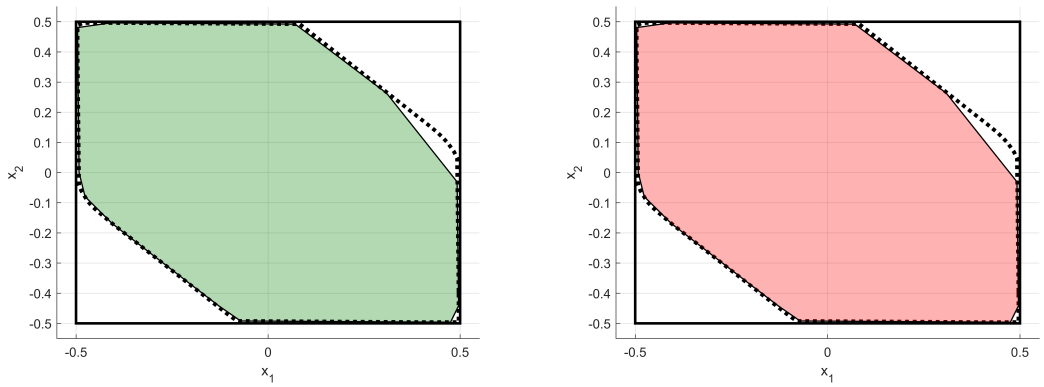


Figure 4.3: Double Integrator: The outer line is the limit of the constraint set, the dotted line is the true viability kernel. The light color in the left is the kernel computed with our adaptation and on the right with the original one

The parameters kept constant in both sets of simulations were (i) the sampling time $\rho = 0.05$, (ii) the number of time steps, $N_\rho = 20$, (iii) the range of the input command, $\mathcal{U} = [-0.15, 0.15]$ and, (iv) the bisection procedure accuracy, $\epsilon = 0.01$.

The simulations were performed using MATLAB R2017^{TM} software, on a machine provided with a Intel Core i7 3.6GHz with 16GB RAM running in a Windows 10 Professional. The optimization problems were solved using MATLAB function *fmincon*. The polytope manipulation and plot were done using MPT toolbox [**?**].

The system is described by (**??**), considering a constraint set $\mathcal{K} = \mathcal{B}_\infty^{(2)}(0, 0.5)$.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \tag{4.9}$$

The viability kernel was under-approximated with $N_r = 44$, the number of vertex. Clearly from Figure **??**, both algorithms achieve the same result, using the same directions. They are in fact, under-approximations of the true viability kernel shown as the dotted line. Our adaptation took 20 seconds to compute and the original one, 13 seconds. For lower dimensions ($n \leq 5$) our algorithm was slightly slower that the original one.

4.2.8.A Scalability

Once more, in order to be able to compare both algorithms, the same kind of testing done in [**?**], was repeated for the improved algorithm, for a chain of n integrators. With $\frac{\partial^n x}{\partial t^n} = u$ and $\mathcal{K} = \mathcal{B}_\infty^{(n)}(0, 0.5)$. In this series of simulations, the number of rays chosen was $N_r = 2n$.

As the system dimension gets higher, our algorithm outperforms the original one. Although the performances of the original algorithm are slower in our implementation than what is referred in [**?**], the

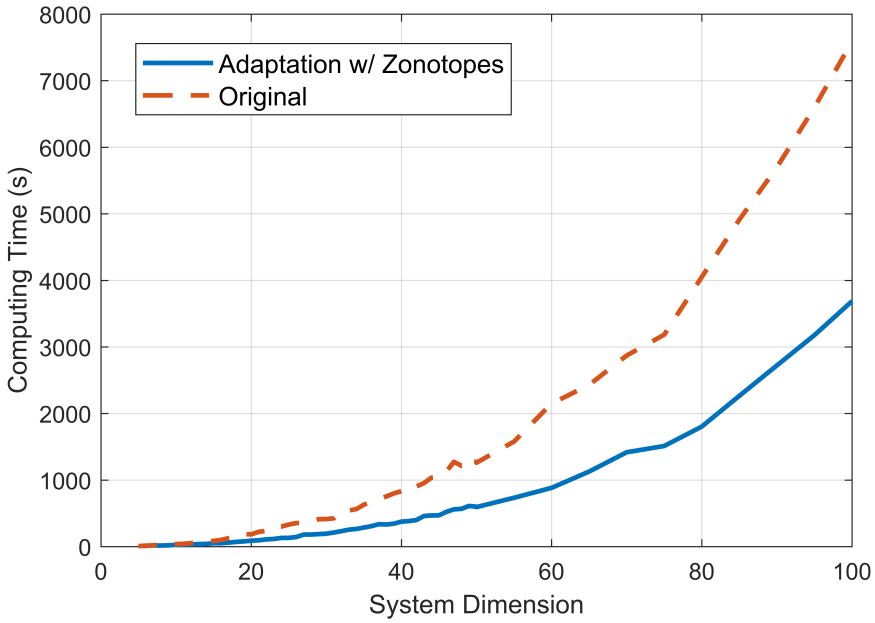


Figure 4.4: Comparison between the original algorithm and the adapted with zonotopes for a chain of n-integrators

rate at which both of them grow (the one in the original paper and implemented here) is the same so it is plausible to assume that the factor separating them is probably due to using different solvers for the linear programming problems.

4.2.8.B Asymptotic Complexity

A complexity bounded by $\mathcal{O}(N_r \log(d)\Phi(n))$ for a fixed time horizon is claimed by [**?**], with $\Phi(n)$ the complexity of the original feasibility program, bounded between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$. Roughly speaking, the 'diameter' of the constraint set is d.

Since our algorithm contains a simple loop, each iteration has a bounded complexity from the reachable set computation of $\mathcal{O}(hn^2)$ plus the computation of the overlap which we will call Ov(n).

The estimate of the average case complexity uses the experimental data shown in figure **??** fitted to a power function of the type $f(n) = an^b$. Using MATLAB *fit* function, the best fit to the data showed that the original algorithm has a complexity bound of $\mathcal{O}(n^{2.472})$ while ours is bounded in $\mathcal{O}(n^{2.557})$. However, at a confidence level of 95% the hypothesis that our bound is lower than the original should not be discarded. The difference in the power is rather small, and for a sufficiently wide range of n, our algorithm is actually faster. This data also supports the original claim of the algorithm being supra quadratic but sub cubic. The data used in the fitting is available online[1].

[1] http://web.ist.utl.pt/~andre.miguel.c/kernel_zonotope/data

4.2.9 Comments

The adaptation of the algorithm in Gillula's *et al.* proved to be a valid alternative approach to the same problem. The approach presented here reduces the viability kernel computing problem to an iterative reachability problem. With the appropriate choice of representation of sets, such as zonotopes and polytopes, we were able to have a slightly faster algorithm. Further improvements can be expected if better representations are found or better ways to compute the overlap between sets were used.

Since the scope of this work is in the field of autonomous vehicles, it is a rather over-simplified assumption to consider such a system to be linear. The next step is to be able to use this improved algorithm in more reliable models of an autonomous car, such as non-linear.

4.3 Approximating Viability Kernels of Non-Linear Systems

Before explaining thoroughly the adaptation to the non-linear class of systems, some properties and proofs will be given in order to ensure the proper functioning and existence of solutions to our algorithm.

The systems considered in the following part of this work can be represented by the differential equation,

$$\begin{cases} \dot{x}(t) = f(x(t), u(t)), \quad \forall t \\ u(t) \in \mathcal{U}, \quad x(0) = x_0 \in \mathcal{X}. \end{cases} \tag{4.10}$$

Where, $\mathcal{X}, \mathcal{U}$ are sets in a metric space standing, respectively, for a state space and a compact control space. Both are considered non-empty.

The conditions for existence and uniqueness of the solutions of the differential inclusion (**??**) are discussed below. Consider a set-valued map $F : \mathcal{X} \rightsquigarrow \mathcal{Y}$, with $\mathcal{Y}$ a set in a metric space, defined as,

$$\forall x \in \mathcal{X}, \quad F(x) := \{f(x, u)\}_{u \in \mathcal{U}} = \bigcup_{u \in \mathcal{U}} f(x, u). \tag{4.11}$$

This is a common extension of a wide class of differential equation models to account for uncertainties in control (see for instance [**?**], pp. *xiv*). We can reformulate the system (**??**) as,

$$\dot{x}(t) \in F(x(t)), \qquad \text{for almost all } t > 0. \tag{4.12}$$

Continuity properties play a fundamental role in the existence of solutions of systems in the form (**??**). These are briefly reviewed below.

Definition 4.10. ([**?**]) F is *upper semicontinuous* (USC) at $\bar{x} \in X$ if for any open N containing $F(\bar{x})$ there exists a neighborhood M of $\bar{x}$ such that $F(M) \subset N$. F is an upper semicontinous set-valued map if it is USC for every $\bar{x} \in X$.

Definition 4.11. ([**?**]) F is *lower semicontinuous* (LSC) at $\bar{x} \in X$ if for any $\bar{y} \in F(\bar{x})$ and any neighborhood $N(\bar{y})$, there exists a neighborhood $M(\bar{x})$ of $\bar{x}$ such that

$$\forall x \in M(\bar{x}), \qquad F(x) \cap N(\bar{y}) \neq \emptyset$$

F is a lower semi-continuous set-valued map if it is LSC for every $\bar{x} \in X$.

The following result summarizes the continuity properties of the dynamics map that are key to the existence of solutions of (**??**).

Theorem 4.12. *Consider (**??**), with $\mathcal{X}, \mathcal{Y}, \mathcal{U}$ subsets of metric spaces. If (i) $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{Y}$ is C^1 function in both variables, (ii) for each $x \in \mathcal{X}$, f is convex relative to the second variable, u, and (iii) $\mathcal{U}$ is a non-empty, compact convex set, then $\forall x \in \mathcal{X}, F(x)$ is a continuous (LSC and USC) and Lipschitz set-valued map, with non-empty, compact and convex images.*

Proof: The demonstration follows by showing, in sequence, non-emptiness, compactness, convexity, Lipschitziness[2] and each semi-continuity.

Non-emptiness:

Since $\mathcal{U} \neq \emptyset$ and $f(\cdot)$ is a continuous function in both variables, the set $F(x)$ has always at least one element, thus, it is non-empty.

Compactness:

The subset $\mathcal{U}$ being compact means that every sequence, u_n, in $\mathcal{U}$ has a convergent subsequence. By definition of uniform continuity, (see for instance [**?**]), with n a sucession index,

$$\forall (x, u_n)_{n \in \mathbb{N}} \subset \mathcal{X} \times \mathcal{U},$$
$$\lim_{n \to \infty} (x, u_n) = (x, \bar{u}) \implies \lim_{n \to \infty} f(x, u_n) = f(x, \bar{u}).$$

This means that, for every convergent subsequence in $\mathcal{U}$, we have a corresponding convergent subsequence $f(x, u_n) \in F(x)$. Therefore, every sequence of elements in $F(x)$ has at least one convergent subsequence, the set is then sequentially compact (by definition of compactness). Every sequentially compact metric space is compact[3], so, $F(x)$ is compact $\forall x \in \mathcal{X}$.

Convexity:

The convexity follows from the fact that $\forall x \in \mathcal{X}$, $f(x, u)$ is a convex, continuous function of $u \in \mathcal{U}$ and $\mathcal{U}$ a convex set. The range of such function, for each x, is a convex set (Prop. 2.32 [**?**]).

[2] The property of being Lipschitz.
[3] Bolzano-Weierstrass Theorem.

Lipschitz:

The set-valued map is globally Lipschitz at x if there exists a positive constant L such that,

$$\forall x_1, x_2 \in \mathcal{X},\ F(x_1) \subset F(x_2) + \mathcal{B}_2(0, L||x_1 - x_2||),$$

which implies that[4],

$$\begin{aligned} \forall x_1, x_2 \in \mathcal{X}, \forall u_1, u_2 \in \mathcal{U},\ ||f(x_1, u_1) - f(x_2, u_2)|| &\leq L||x_1 - x_2|| \Rightarrow \\ \Rightarrow ||f(x_1, u_1) - f(x_2, u_2)||^2 &\leq L^2||x_1 - x_2||^2 \\ &\leq L^2(||x_1 - x_2||^2 + ||u_1 - u_2||^2) \Rightarrow \\ ||f(x_1, u_1) - f(x_2, u_2)|| &\leq L||(x_1, u_1) - (x_2, u_2)|| \end{aligned} \tag{4.13}$$

Since f is C^1, it is a Lipschitz function, $||f(x_1, u_1) - f(x_2, u_2)|| \leq M||(x_1, u_1) - (x_2, u_2)||, \forall x_1, x_2 \in \mathcal{X}, \forall u_1, u_2 \in \mathcal{U}$, the constant L, in (**??**) identifies with M, so there exists a constant such that F is Lipschitz.

Upper Semi-Continuity:

Denote $\mathring{\mathcal{B}}(a, r)$ as the open ball of center a and radius r with some metric. To show USC, assume that $F()$ is not USC everywhere.

Therefore,

$$\exists \bar{x} \in \mathcal{X}, \exists N \supset F(\bar{x}), \forall M(\bar{x}), F(M(\bar{x})) \not\subset N \implies \forall x \in M(\bar{x}), \forall u \in \mathcal{U}, f(x, u) \notin N \tag{4.14}$$

Knowing that x is in a neighborhood of $\bar{x}$,

$$\forall u \in \mathcal{U}, \exists \rho > 0, |(x, u) - (\bar{x}, u)| < \rho \implies \forall \epsilon > 0, |f(x, u) - f(\bar{x}, u)| < \epsilon,$$

that is equivalent to

$$\forall u \in \mathcal{U}, \forall \epsilon_u > 0, f(x, u) \in \mathring{\mathcal{B}}(f(\bar{x}, u), \epsilon_u) \implies \forall \epsilon_u > 0, F(x) \subset \bigcup_{u \in \mathcal{U}} \mathring{\mathcal{B}}(f(\bar{x}, u), \epsilon_u). \tag{4.15}$$

To exist a neighborhood N, for $F(\bar{x})$ means that,

$$\exists \epsilon > 0, \forall y \in F(\bar{x}), \mathring{\mathcal{B}}(y, \epsilon) \subset N \Leftrightarrow \exists \epsilon > 0, \bigcup_{u \in \mathcal{U}} \mathring{\mathcal{B}}(f(\bar{x}, u), \epsilon) \subset N, \tag{4.16}$$

[4] $||(x, y)||^2 = x_1^2 + ... + + x_p^2 + y_1^2 + ... + y_n^2 = ||x||^2 + ||y||^2$

by the transitive property of inclusions using (**??**) and (**??**),

$$\forall x \in M(\bar{x}), F(x) \subset N.$$

This contradicts the proposition in (**??**), therefore, F must be USC.

Lower Semi-Continuity:

By one hand, admitting that F is not LSC everywhere implies,

$$\begin{aligned} \exists \bar{x} \in \mathcal{X}, \exists \bar{y} \in F(\bar{x}), \exists N(\bar{y}), \forall M(\bar{x}), \exists x \in M(\bar{x}), F(x) \cap N(\bar{y}) = \emptyset \implies \\ \forall u \in \mathcal{U}, f(x,u) \notin N(\bar{y}) \implies \nexists \epsilon > 0, |f(x,u) - \bar{y}| < \epsilon. \end{aligned} \tag{4.17}$$

By the other, using the definition of continuity,

$$\forall \bar{x} \in \mathcal{X}, \forall x \in M(\bar{x}), \forall u \in \mathcal{U}, \exists \rho > 0, |(x,u) - (\bar{x},u)| < \rho \implies \forall \epsilon > 0, |f(x,u) - f(\bar{x},u)| < \epsilon,$$

and, as it is valid for any u and, $\bar{y} = f(\bar{x},u)$,

$$\forall \bar{y} \in F(\bar{x}), \exists \epsilon > 0, |f(x,u) - \bar{y}| < \epsilon,$$

this contradicts (**??**), therefore, F is also lower semi-continuous.
□

Definition 4.13. (Selection [**?**]) Consider a set-valued map $F : \mathcal{X} \rightsquigarrow \mathcal{Y}$ with non empty images. A single valued map $f : \mathcal{X} \mapsto \mathcal{Y}$ is called a selection of F if for every $x \in \mathcal{X}$, $f(x) \in F(x)$.

The following theorem will also be useful ahead in this chapter to ensure existence of solutions to out algorithm.

Theorem 4.14. *([?]) Let F be a LSC set-valued map, from some open set $\Omega \subset \mathbb{R} \times \mathcal{X}$, into some non-empty, closed and convex subset of $\mathcal{Y}$. Let $(0, x_0) \in \Omega$. Then there exists some interval $I = (\omega_-, \omega_+)$, $\omega_- < 0 < \omega_+$ and at least one continuously differentiable function $x : I \mapsto \mathcal{Y}$, where $x(t)$ is a solution to the Cauchy problem*[5] *for the differential inclusion,*

$$\dot{x}(t) \in F(t, x(t)), \qquad x(0) = x_0.$$

[5] Also known as initial value problem.

4.3.1 Linearizing Systems

A linear system is described by the following differential equation,

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) + l(t), \quad \forall t > 0 \\ x(t) \in \mathcal{X}, u(t) \in \mathcal{U}, l(t) \in \mathcal{L}, x(0) = x_0. \end{cases} \tag{4.18}$$

With $\mathcal{X} \subset \mathbb{R}^n, \mathcal{U} \subset \mathbb{R}^d$ and $\mathcal{L} \subset \mathbb{R}^n$ compact, convex subsets. In the system above, $l(t)$ is a perturbation of the system, in our case, the linearization error. Define the sampling time as ρ. The time horizon considered is denoted as $\bar{T}$, with the number of time steps being, $N_\rho = \lfloor \bar{T}/\rho \rfloor$. Define the following integral,

$$I_\rho := \int_0^\rho e^{Ar} dr.$$

The sampled-time discrete system becomes,

$$x_{k+1} = A_\rho x_k + B_\rho u_k + I_\rho l_k, \quad \forall k \in \{0, ..., N_\rho - 1\}, \tag{4.19}$$

with,

$$A_\rho := e^{A\rho}, \; B_\rho := I_\rho B.$$

The state of the system at some time step s, with some initial state x_0, is given by,

$$x_s = A_\rho^s x_0 + \sum_{i=0}^{s-1} A_\rho^i (B_\rho u_{s-i-1} + I_\rho l_{s-i-1}). \tag{4.20}$$

4.3.2 Linearization

The linearization of the system (**??**) is done using some results in [**?**], in the following manner,

$$\dot{x} \in f(x^*, u^*) + \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x^*, u=u^*} (x - x^*) + \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x^*, u=u^*} (u - u^*) + \mathcal{L}. \tag{4.21}$$

Matrices $A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x=x^*, u=u^*}$ and $B = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x=x^*, u=u^*}$ are the jacobians of the respective linearizations, in terms of state and input variables, around the point (x^*, u^*). The term $\mathcal{L}$ is the Lagrangian Remainder. Let i be the index of the i-th state variable, defining $z = \begin{bmatrix} x \\ u \end{bmatrix}$, the jacobian of the i-th state variable is,

$$J_i(\xi) := \frac{\partial^2 f_i(\xi)}{\partial z^2},$$

the i-th component of the remainder,

$$\mathcal{L}_i = \frac{1}{2}(z - z^*)^T J_i(\xi)(z - z^*).$$

with, $\xi \in \{z^* + \alpha(z - z^*) \mid \alpha \in [0, 1]\}$.

This is the continuous time, linearized system. This system must now be converted to a discrete-time version. Combining(**??**) with (**??**),

$$\begin{cases} \hat{x}_{k+1} = A_\rho^{(k)} \hat{x}_k + B_\rho^{(k)} u_k + I_\rho(z^*)(f(z^*) - A^{(k)} x_k^* - B^{(k)} u_k^*), \\ \hat{x}_k \in \mathcal{X}, u_k \in \mathcal{U}, z^* \in \mathcal{X} \times \mathcal{U}, \forall k \in \mathbb{N}^0. \end{cases} \tag{4.22}$$

The step reachable set of such linearized and discretized system (**??**), from the set $\mathcal{E}$, around the point z^* is computed as in [**?**],

$$\text{Reach}_\rho^{lin}(\mathcal{E}) := A_\rho[\mathcal{E}] \oplus B_\rho[\mathcal{U}] + I_\rho(f(z^*) - Ax^* - Bu^*). \tag{4.23}$$

Usually, it is hard to compute the exact set generated by the Lagrangian remainder. For that matter, an over-approximation is required for it. In this work we will address on how to compute the approximation in a further section, using partially results from [**?**].

4.3.3 Over-Approximating the Lagrangian Remainder

The set in which the Lagrangian remainder is computed, $\bar{\mathcal{Z}}$, is defined as,

$$\bar{\mathcal{Z}} := \text{Conv}(\mathcal{E}, \text{Reach}_\rho^{lin}(\mathcal{E})) \oplus \mathcal{B}_\infty(0, M\rho), \tag{4.24}$$

where, Conv$(\cdot)$ is the convex hull operator and $\mathcal{E}$ is the initial set.

If we make use of the zonotope notation for our sets, and also of the convex hull properties one has,

$$\mathcal{E} \oplus \mathcal{B}_\infty(0, M\rho) = (c_E, < g_1, ..., g_p, (M\rho)e_1, ..., (M\rho)e_n >),$$

with e_i the vectors of the canonical basis of $\mathbb{R}^n \supset \mathcal{E}$.

$$\mathcal{R} := \text{Reach}_\rho^{lin}(\mathcal{E}) \oplus \mathcal{B}_\infty(0, M\rho) = (c_R, < h_1, ..., h_s, (M\rho)e_1, ..., (M\rho)e_n >).$$

For simplicity we aggregate the terms with $(M\rho)e_i$ as generators of each zonotope. The convex hull of both sets is,

$$\bar{\mathcal{Z}} = \{x \in \mathcal{X} \mid x = \lambda\mathcal{E} + (1 - \lambda)\mathcal{R}, \lambda \in [0, 1]\}. \tag{4.25}$$

Proposition 4.15. (Over approximation of the set of linearization errors):
Consider $\bar{\mathcal{Z}}$ the zonotope defined as in (**??**), the Lagrangian remainder in such set can be over approxi-

mated as,

$$|\mathcal{L}_i| \subseteq [0, l_i],\ l_i = \frac{1}{2}\gamma^T \max(|J_i(\xi)|)\gamma,\ z \in \bar{\mathcal{Z}}, \tag{4.26}$$
$$\text{with, } \gamma = \max(|c_E - z^*|, |c_R - z^*|) + \max\Big(\sum_{i=1}^{p}|g_i|, \sum_{j=1}^{s}|h_j|\Big).$$

The $\max()$ function is element-wise so as the absolute values, $|\cdot|$.

Proof: Using partially the proof in [**?**],

$$|\mathcal{L}_i| \subseteq \frac{1}{2}\Big[0, \max_{z\in\bar{\mathcal{Z}}}(z - z^*|)^T \max_{z\in\bar{\mathcal{Z}}}(|J_i(\xi)|) \max_{z\in\bar{\mathcal{Z}}}(z - z^*|)\Big].$$

Noticing that by the definition of the set $\bar{\mathcal{Z}}$,

$$\begin{aligned}
\max_{z\in\bar{\mathcal{Z}}}(|z - z^*|) &= \max_{\substack{\lambda\in[0,1]\\ \alpha_i\in[-1,1]\\ \beta_i\in[-1,1]}} \Big(|\lambda(c_E - z^*) + (1-\lambda)(c_R - z^*) + \lambda\sum_{i=1}^{p}\alpha_i g_i + (1-\lambda)\sum_{j=1}^{s}\beta_j h_j|\Big) \\
&\leq \max_{\lambda\in[0,1]}\Big(|\lambda(c_E - z^*) + (1-\lambda)(c_R - z^*)|\Big) + \max_{\substack{\lambda\in[0,1]\\ \alpha_i\in[-1,1]\\ \beta_i\in[-1,1]}}\Big(|\lambda\sum_{i=1}^{p}\alpha_i g_i + (1-\lambda)\sum_{j=1}^{s}\beta_j h_j|\Big) \\
&\leq \max(|c_E - z^*|, |c_R - z^*|) + \max\Big(\sum_{i=1}^{p}|g_i|, \sum_{j=1}^{s}|h_j|\Big) = \gamma.
\end{aligned}$$

□

It remains to find an optimal value of z^* that minimizes the over-approximation being made.

Remark: The choice of z^* that minimizes l_i, eq. (**??**), is given by,

$$z^* = \frac{1}{2}(c_E + c_R) = c_E + \frac{\rho}{4}f(c_E).$$

Since,

$$c_R \approx c_E + \frac{\rho}{2}f(c_E).$$

Clearly, by the first term of γ, if z^* is equally distant from both points, c_E and c_R, both functions of the $\max$ operator have the same value and any change would make either $|c_E - z^*|$ or $|c_R - z^*|$ to increase, inscreasing the $\max$ function. Since the maximum of two convex functions is convex aswell, this is a global minimum. The choice of c_R comes from [**?**]. We don't known the center of the reachable set before needing to compute the linearization point, therefore it has to be approximated in some way.

4.3.4 Existence of Reachable Sets

For the proper functioning of the algorithm in this work, we have to ensure that the reachable set is possible to compute. First we will state the conditions in which there exists finite-time reachable sets, independently of the method of discretization or linearization. Afterwards, we do the same, except using the linearization and discretization procedures stated in subsection **??**.

Definition 4.16. The finite time horizon, $T = N_\rho \rho$, reachable set of the continuous non-linear system (**??**) with piecewise constant input is defined as,

$$\text{Reach}_T(\mathcal{K}) := \{x \in \mathcal{X} \mid x = x(T), x_0 \in \mathcal{K} : \exists\{u_0, ..., u_{N_\rho - 1}\} \in \mathcal{U}\} \tag{4.27}$$

Proposition 4.17. Let a function $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{Y}$ be defined as in (**??**), and such that the conditions of Theorem **??** hold. Then, the finite-horizon reachable set, (**??**) can always be computed, for any $T \in \mathbb{R}^+$.

Proof: From Theorem **??**, $F(x)$ is non-empty, with compact, convex values and LSC, therefore we are in the conditions of Theorem **??**. This implies the existence of at least one continuously differentiable function, $x(t)$ for some tight interval. Since the properties of $F(x)$ are verified everywhere in its domain, the local existence of solutions becomes global. Therefore, $\forall T \in \mathbb{R}^+$, there exists $x(t)$, $\forall t \in [0, T]$.
□

Proposition 4.18. Given a function $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{Y}$ defined as in (**??**). If f is once differentiable in both variables in all its domain, then, for any $x_k \in \mathcal{X}, u_k \in \mathcal{U}, (x^*, u^*) \in \mathcal{X} \times \mathcal{U}$, the step reachable set (**??**) can be computed.

Proof: Since f is everywhere differentiable, the matrices A and B (defined in (**??**)) always exist. It is known that the power series defining e^{tA}, converges uniformly on compact time intervals, therefore A_ρ can be computed. Since the integral $\int_0^\rho e^{\lambda A} d\lambda$ also converges, B_ρ and I_ρ exist and can be computed. Since differentiability implies continuity, $f(x^*, u^*)$ also exists. Therefore, the step reachable set can always be computed.
□

4.3.5 Convexity of Viability Kernels and Algorithm Correctness

If $F(x)$ is a non-empty, compact, convex set-valued map and x taking values in a compact convex set $\mathcal{X}$, we assure that the output of Algorithm **??** still works for this system and it is in fact an under-approximation of the viability kernel of the non-linear system.

Also we prove in what conditions does the infinite time viability kernel of a continuous time system

(not-sampled), $\dot{x}(t) \in F(x(t))$,

$$\text{Viab}_F(K) := \{x_0 \in K \mid \forall t > 0, \exists u(t) \in \mathcal{U} : x(t) \in \mathcal{K}\},$$

is a convex set. Define the following,

Definition 4.19. (Kuratowski Set Convergence [**?**])

Given a sequence of non-empty subsets, $\{A_n\}_{n=1}^{\infty}$, of a Banach Space X, t converges to A in the sense of *Kuratowski* if,

$$\liminf_{n\to\infty} A_n = \limsup_{n\to\infty} A_n = A,$$

where,

$$\begin{aligned}\limsup_{n\to\infty} A_n = \{x \in X : &\text{ there exists a subsequence } \{A_{n_k}\}_{k=1}^{\infty} \text{ of } \{A_n\}_{n=1}^{\infty},\\ &\text{and a sequence } \{x_k\}_{k=1}^{\infty}, x_k \in A_{n_k}, \forall k \in \mathbb{N},\\ &\text{such that } \lim_{k\to\infty} x_k = x\}.\end{aligned} \tag{4.28}$$

And,

$$\begin{aligned}\liminf_{n\to\infty} A_n = \{x \in X : &\text{ there exists a sequence} \{x_n\}_{n=1}^{\infty},\\ &x_n \in A_n, \forall n \in \mathbb{N} \text{ such that } \lim_{n\to\infty} x_n = x\}.\end{aligned} \tag{4.29}$$

The following lemma will be used in a theorem ahead,

Lemma 4.20. Let $\{A_n\}_{n=1}^{\infty}$ be a sequence of sets, in which $A_{n+1} \subseteq A_n$. Consider the upper limit of the sequence, $\limsup_{n\to\infty} A_n = A$. If A_n is convex, $\forall n \in \mathbb{N}$, then, A is convex.

Proof: Consider $x, y \in A$ and $\lambda \in [0, 1]$. Then, there exist two subsequences $\{A_{n_k}\}_{k=1}^{\infty}$, $\{A_{\bar{n}_k}\}_{k=1}^{\infty}$ and two, respective, sequences, $\{x_k\}_{k=1}^{\infty}$ and $\{y_k\}_{k=1}^{\infty}$, where, $x_k \in A_{n_k}$, $y_k \in A_{\bar{n}_k}, \forall k \in \mathbb{N}$. With,

$$\lim_{k\to\infty} x_k = x, \quad \lim_{k\to\infty} y_k = y.$$

Since both sequences of elements are convergent, the limit of the convex combination of the sequences is the convex combination of the limits of each sequence,

$$\begin{aligned}\lambda x + (1-\lambda)y &= \lambda \lim_{k\to\infty} x_k + (1-\lambda) \lim_{k\to\infty} y_k\\ &= \lim_{k\to\infty} \lambda x_k + (1-\lambda) y_k.\end{aligned}$$

Choose a new subsequence of sets based on $\hat{n}_k := \min(n_k, \bar{n}_k)$.[6] Since $A_{n+1} \subseteq A_n$, if $\bar{n}_k > n_k$, then, $A_{\bar{n}_k} \subseteq A_{n_k}$, which means that, $y_k \in A_{n_k}$. Remember that by the definition, $x_{n_k} \in A_{n_k}$, therefore, $y_k, x_k \in A_{\hat{n}_k}$. The same applies in the cases where $n_k > \bar{n}_k$. Due to convexity of $A_n, \forall n \in \mathbb{N}$,

$$\lambda x_k + (1-\lambda) y_k \in A_{\hat{n}_k}, \forall k \in \mathbb{N}.$$

By Definition **??**, $\lambda x + (1-\lambda)y \in A$, and since x and y are arbitrary, A is convex.

□

Definition 4.21. Marchaud Set-Valued Map ([**?**])

Consider the set valued map, $F : X \rightsquigarrow Y$. Where X, Y are finite dimensional vector spaces. F is Marchaud if,

$$\begin{cases} \text{Dom}(F) \neq \emptyset \\ F \text{ is USC with convex, compact values} \\ \forall x \in \text{Dom}(F), ||F(x)|| := \max_{y \in F(x)} ||y|| < c(||x|| + 1), \end{cases}$$

with some $c \in \mathbb{R}$.

Theorem 4.22. *Let $F : X \rightsquigarrow Y$ be a set-valued map with X, Y finite dimensional vector spaces. If F is a Marchaud, L-Lipschitz set-valued map, satisfying the boundedness condition* (**??**)*, and $K \subset X$ a compact convex subset, then, the infinite time viability kernel $\text{Viab}_F(K)$ is convex.*

$$M := \sup_{x \in K} \sup_{y \in F(x)} ||y|| < \infty \tag{4.30}$$

Proof: Consider the difference inclusion,

$$x_{n+1} \in \Gamma_\rho(x_n),$$

with, $\Gamma_\rho(x) := x \oplus \rho F(x) \oplus \mathcal{B}(0, \frac{ML}{2}\rho^2)$, $\rho \in \mathbb{R}^+$. The infinite time viability kernel, $\text{Viab}_{\Gamma_\rho}(K)$ can be constructed as [**?**] [**?**],

$$\begin{cases} K_0 = K, \\ K_{n+1} = K_n \cap \text{Back}_{\Gamma_\rho}(K_n), \end{cases}$$

where, $\text{Back}_{\Gamma_\rho}(K_n)$ is the backward reachable set from K_n - the set of states that end in K_n the next time step. The backward reachable set of Γ_ρ coincides with the reachable set of the system with the inverse dynamics $-F(x)$, $x_{n+1} \in \hat{\Gamma}_\rho(x_n) = x_n \oplus (-\rho F(x)) \oplus \mathcal{B}(0, \frac{ML}{2}\rho^2)$ [**?**] [**?**]. Since it is a difference inclusion, its reachable set is simply, $\text{Back}_{\Gamma_\rho}(K) = \text{Reach}_{\hat{\Gamma}_\rho} = \hat{\Gamma}_\rho(K)$. This is a convex set, since K is

[6] This is still a subsequence of $\{A_n\}_{n=1}^{\infty}$. If $n_k \leq \bar{n}_k$, then, $n_k < \bar{n}_{k+1}$, and reminding that, $n_k < n_{k+1}$, results in, $\hat{n}_k < \hat{n}_{k+1}, \forall k$.

convex and F has convex images. In the limit $n \to \infty$,

$$\text{Viab}_{\Gamma_\rho}(K) = K_\infty.$$

By construction, $\text{Viab}_{\Gamma_\rho}(K)$ is a convex set.

Define now a sequence of strictly decreasing values of ρ_n, where, $\lim_{n\to\infty} \rho_n \to 0$. Since $\forall n \in \mathbb{N}, \rho_{n+1} < \rho_n, \forall x \in K, \Gamma_{\rho_{n+1}}(x) \subseteq \Gamma_{\rho_n}(x)$, then, $\text{Viab}_{\Gamma_{\rho_{n+1}}}(K) \subseteq \text{Viab}_{\Gamma_{\rho_n}}(K)$.[7] From Theorem 3.2 of [**?**] and since F is Marchaud, L-Lipschitz bounded by M,

$$\limsup_{\rho\to 0} \text{Viab}_{\Gamma_\rho}(K) = \text{Viab}_F(K).$$

From Lemma **??**, $\text{Viab}_F(K)$ is convex.

□

Theorem 4.23. *Consider the system defined by a differential equation such as* (**??**)*. Admitting that the state space, $\mathcal{X}$, constraint set, $\mathcal{K}$, and the input set, $\mathcal{U}$, are non-empty, compact convex sets, and $f : \mathcal{X} \times \mathcal{U} \mapsto \mathcal{Y}$ is a C^1 function, convex in the second variable for each fixed x. Given a number of random directions, N_r, and an initial point $\boldsymbol{v}_0 \in$ Viab$_T^{sd}(\mathcal{K})$, and let, $\mathcal{S} =$ Polytopic-Approx$(\mathcal{K}, \boldsymbol{v}_0, N_r)$, then,*

$$\mathcal{S} \subseteq \text{Viab}_T^{sd}(\mathcal{K}).$$

Proof:

Recall that, $\mathcal{S} = \text{Conv}(v_0, v_1, ..., v_{N_r})$. We want to verify that,

$$v_i \in \text{Viab}_T^{sd}(\mathcal{K}), \quad i \in \{0, 1, ..., N_r\} \implies \text{Conv}(v_0, v_1, ..., v_{N_r}) \subseteq \text{Viab}_T^{sd}(\mathcal{K}). \tag{4.31}$$

A sufficient condition for this to hold is the viability kernel to be a convex set. Since we are in the conditions of Theorem **??**, we can describe the system by a differential inclusion $\dot{x}(t) \in F(x(t))$, where $F(x)$ has the property (among others) of having non-empty, compact, convex values and being Lipschitz. As in [**?**] and by the definition of finite time horizon viability kernel of the sampled data system (Def. **??**), the finite-time viability kernel for the sampled-data system can be built recursively as,

$$\begin{cases} K_0 = \mathcal{K} \\ K_{n+1} = \{x \in K_n \mid K_n \cap \text{Reach}_\rho(x) \neq \emptyset\}. \end{cases}$$

[7] Let A, B, S be subsets in the same space, $A \subseteq B \implies A \cap S \subseteq B \cap S$.

Where, K_n is the the n step viability kernel. Using a proof from [**?**],

$$\begin{aligned} x \in K_{n+1} &\Leftrightarrow x \in K_n \wedge (\text{Reach}_\rho(x) \cap K_n \neq \emptyset) \\ &\Leftrightarrow x \in K_n \cap \text{Back}_\rho(K_n) \end{aligned}$$

Notice that the set $\text{Back}_\rho(K_n)$ is the set of initial states in which there are trajectories that end in K_n the next instant, this is the backwards reachability set. This set coincides with the reachable set of the system similar to (**??**) but with, $\dot{x}(t) \in -F(x(t))$, $x_0 \in K_n$, instead [**?**] [**?**].

If the step reachable set of the inverse dynamics is convex, then, by construction, $\text{Viab}_T^{sd}(\mathcal{K}) = K_{N_\rho}$ is a convex set.

Definition 4.24. (Set-Valued Map Composition [**?**]) Consider two set valued-maps G and H with domains and values in a linear vector space. Define the composition $(G \circ H)(x) := \{z \mid \exists y \in H(x) \text{ with } z \in G(y)$. If a set-valued map $G(x)$ is composed with itself N many times, we denote the resulting set-valued map as $G^N(x)$.

Introducing the following theorem,

Theorem 4.25. *(Reachable set exponential formula [**?**])*

Suppose $X \subseteq \mathcal{D} \subseteq \mathbb{R}^n$ is an open set, and $F : \mathcal{D} \rightsquigarrow \mathcal{Y} \subseteq \mathbb{R}^m$ a set valued function. Assume F has non-empty, compact, convex values on X and is locally Lipschitz on X. Then, for all $x \in X$ and $T > 0$, such that all solutions starting at x remain in the set X by time T, we have the reachable set,

$$\textit{Reach}_T(x) = \lim_{N \to \infty} \left(I + \frac{T}{N} F \right)^N (x),$$

where the limit above defined is in the Kuratowski limit of sequence of sets sense.

Define $G(x) := -F(x)$. Since it was just applied a linear map to the set-valued function, $G(x)$ is still non-empty, compact, convex valued, Lipschitz in $\mathcal{X}$. From the formula of the above Theorem **??**, notice that, $\forall x \in \mathcal{K}, \forall N \in \mathbb{N}$, the set $\left(I + \frac{T}{N} G\right)^N (x)$ is convex , because the set valued map $S(x) = I + \frac{T}{N} G(x)$ is convex (linear maps and adding 1 to every element in every dimension of $G(x)$) and the composition of convex imaged set-valued maps on convex-imaged set-valued maps yield convex sets.

From [**?**], the limit of a sequence of convex sets is convex, therefore, K_n is convex $\forall n \in \mathbb{N}$, and therefore, $\text{Viab}_T^{sd}(\mathcal{K})$ is also convex. That means that $\mathcal{S} \subseteq \text{Viab}_T^{sd}(\mathcal{K})$

□

4.3.6 Error Bounding

In our case, we would like to take the linearization errors, such as present in (**??**), into account. Consider the later equation with,

$$e_k = I_\rho(l_k + d_k).$$

The total error of the linearized and discretized system at the instant k, where, l_k is the linearization error and d_k the discretization error. Lemma **??** still applies to this.

4.3.7 Computing reachable sets of non-linear systems

Following the main ideas in [**?**], we will recall how to implement the improvement presented in this paper in the realm of non-linear systems.

Propositions **??** and **??**, establish the condition in which it is possible to compute the reachable set. The assurance of these conditions is very useful if one wants to guarantee that the algorithm terminates with a definite answer and does not end with an ill-posed problem.

The reachable set computation using linearizations (Algorithm **??**) receives as inputs the discretization step ρ, the system's non-linear model, $\mathcal{S}$, the input space, $\mathcal{U}$, the initial condition set of the system, X_0, and the admissible Lagrangian remainder $\bar{\mathcal{L}}$.

Algorithm 4 Computes the step reachability set of non-linear systems [**?**]

1: **function** REACH(ρ,$\mathcal{S}$,$\mathcal{U}$,X_0,$\overline{\mathcal{L}}$)
2: $(A, B, \mathcal{L}) \leftarrow$ Linearize($\mathcal{S}, X_0$)
3: **if** $\mathcal{L} \not\subseteq \overline{\mathcal{L}}$ **then**
4: $(X_0^{(1)}, X_0^{(2)}) \leftarrow$ Split(X_0)
5: $R^{(1)} \leftarrow$ Reach($\rho, \mathcal{S}, \mathcal{U}, X_0^{(1)}, \overline{\mathcal{L}}$)
6: $R^{(2)} \leftarrow$ Reach($\rho, \mathcal{S}, \mathcal{U}, X_0^{(2)}, \overline{\mathcal{L}}$)
7: **return** $\{R^{(1)}, R^{(2)}\}$
8: **else**
9: $R \leftarrow AX_0 \oplus B\mathcal{U}$
10: **return** $\{R\}$
11: **end if**
12: **end function**

As previously stated, the zonotopes are the preferred type of set representation, since linear maps and Minkowski Sums are efficiently computed. The union of zonotopes is not necessarily a zonotope, but it is not necessary the explicit computation (although it might be rendered useful for tight admissible linearization errors) one avoids doing it, and simply add the several splits in a list and deal with each of them independently.

The user must provide the limit for the admissible Lagrangian remainder, $\bar{\mathcal{L}}$, before the splitting takes

place. The admissible bound for the remainder is used in eroding the constraint set as in Lemma **??**.

The splitting of zonotopes is done in the following way,

Proposition 4.26. ([**?**]) (Splitting of Zonotopes) A zonotope, $Z = (c, < g_1, ..., g_p >)$, can be split into two zonotopes Z_1 and Z_2 such that $Z_1 \cup Z_2 = Z$, where,

$$Z_1 = (c - \frac{1}{2}g_j, < g_1, ..., g_{j-1}, \frac{1}{2}g_j, g_{j+1}, ..., g_p >),$$
$$Z_2 = (c + \frac{1}{2}g_j, < g_1, ..., g_{j-1}, \frac{1}{2}g_j, g_{j+1}, ..., g_p >).$$

4.3.8 Results

This section presents simulation results that point to the correctness of the algorithm.

For the sake of simplicity, we will restrict our experiments to two-dimensional systems. The system being tested is a simple pendulum on a cart, with the adequate dimensions, where x_1 is the pendulum angle from the upright position and x_2 its angular velocity. The system is described by the following non-linear differential equation,

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \sin x_1 - u \cos x_1. \end{cases} \tag{4.32}$$

Here $x = (x_1, x_2) \in \mathcal{X} \subset \mathbb{R}^2$ and $u \in \mathcal{U} = [-1, 1]$. The goal here is to determine the set of admissible pairs of angular position and angular velocity for the pendulum be kept inside the constraint set, $\mathcal{K} = [-\pi/6, \pi/6] \times [-\pi/6, \pi/6]$, using the adapted algorithm for non-linear systems. It was set $\rho = 0.05$, the number of time steps, $N_d = 10$, the number of sampling rays was set to $N_r = 100$, the bisection accuracy $\epsilon = 0.03$ and the linearization tolerance, $\max_{x \in \mathcal{L}} |x| = 0.03$.

Figure **??** shows two viability kernels computed using two different methods. The one in lighter color is computed using Saint-Pierre's algorithm. As in [**?**], this algorithm can also return the finite-time viability kernel of the system. For that algorithm, $\rho = 0.05$, the space discretization step $h = 0.0033$, and the Lipschitz constant was set to $L = 2$, and it stopped at the $10th$ iteration so as to compute the 10-step viability kernel. The viable states computed by our algorithm are strictly contained in the one computed with the griding method. Since the accuracy of our method is user defined by the bisection accuracy, ϵ, and the acceptable Lagrangian remainder, $\overline{\mathcal{L}}$, our solution can approximate the true Viab$^{sd}(\mathcal{K})$ as close as one would want, at the expense of more computations. Since the outer one is not constrained by having piecewise constant inputs, it includes, in general, the one that is constrained.

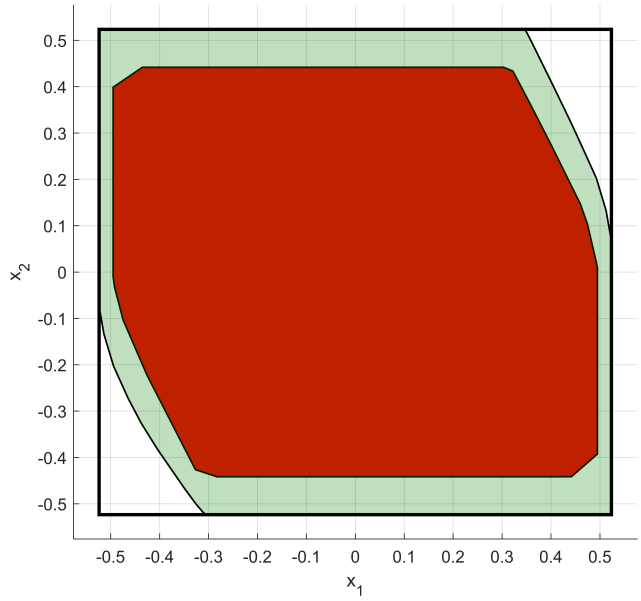


Figure 4.5: The outer line is the limit of the constraint set, the light colored (green) set is the finite horizon viability kernel with Saint-Pierre algorithm and the dark color (red) is the kernel computed with our adaptation.

4.3.9 Comments

This section of the present thesis described an algorithm that adapts Gillula's *et al.* work on computing viability kernels for non-linear, sampled systems by linearizing and discretizing. We also prove the algorithm's correctness. The approach presented here reduces the finite horizon viability kernel computation problem to an iterative reachability one. With the appropriate choice of representation for sets, such as zonotopes and polytopes, we used the same algorithms from linear systems to non-linear ones.

As noted in section **??**, our algorithm under-approximates the true finite horizon viability kernel. In Figure **??** we can compare the infinite horizon viability kernel produced by Saint-Pierre's algorithm with ours. As it is known, for every time step, ρ, the viability kernel computed with Saint-Pierre algorithm is an under-approximation of the true viability kernel for the continuous time system. A system with piece-wise constant input will have less viable trajectories than one that can change inputs between sampling intervals, so, $\text{Viab}_T^{sd}(K) \subseteq \text{Viab}_T(K)$, where the latter set is the finite-time viability kernel for the continuous time system (not sampled). The results confirm this hypothesis, allowing us to confirm the correctness of the algorithm by simulation.

For a sufficiently large linearization error tolerance - does not require many set splitings - the proposed algorithm has a lower asymptotic complexity than that of strategies using state space griding, as in Saint-Pierre's type of methods, which is $\mathcal{O}(a^n)$, for some constant a. In these cases, our algorithm has a complexity similar to the one proposed in [**?**], greater than $\mathcal{O}(n^2)$ but smaller than $\mathcal{O}(n^3)$, since the

most complex operations involve running feasibility programs (the overlap test between sets).

There are essentially two points to be careful when employing our algorithm. One is that methods similar to ours only yield finite horizon viability kernels. It is not possible to guarantee viable trajectories in infinite time, since for that the algorithm would have to run forever. This is not a limitation from an application perspective. The other one is on the linearization errors. If one is largely tolerant on the linearization error, our under-approximation might be too conservative or even result in the empty-set due to the erosion process. If the tolerance is excessively strict, this may result in an enormous amount of splittings of sets, making the algorithm much slower.

Further improvements can be expected if better representations are found or better ways to compute the overlap between sets were used. Work on how to compute tight bounds for the Lagrangian Remainder is of great importance as well. The performance of the algorithm would be greatly improved if clever ways of splitting and storing the splitted sets were introduced.

Since the scope of this work is in the field of autonomous vehicles, this algorithm can be directly applied for sufficiently complex models. The computation of the viability kernels will be the basis in the synthesis of viable controls that keep the vehicle in safe states.

5

Conclusion

5.1 Conclusions

In this thesis we've presented an architecture of control for an autonomous vehicle, which can be used with any autonomous mobile robot. This architecture was inspired in how humans drive and introduced the sense of feedback on the uncertainty of the task. This type of feedback, adjusting the planning based on viability theory represents a novelty in this field. This controller makes the whole system hybrid, since that at each time step, the dynamics of the vehicle change.

In Chapter **??**, we derived the vehicle dynamics producing a sufficiently reliable model. The assumptions and domains of validity of the model are also stated in that chapter. This model is intended to be used to compute the viability kernels addressed in Chapter **??**, so it had to have some properties, such as Lipschitz continuity in its domain, convexity in the input variables and convexity also in the input set, so the algorithm could give a correct answer.

The main focus of this work was in the development of the local planning module, which is based in computing the viability kernel. We've adapted an existing algorithm for the linear system, recurring to reachability, represented with zonotopes. Our algorithm was faster, for a large range of variables, than the original one. Although we didn't prove it was asymptotically faster, it was not disproven either - based on fitting the computational time as a function of the system dimension $n = \{5, 6, ..., 100\}$ with a confidence level of 95%. We've extended the algorithm to be used in some non-linear systems (the majority of the ones with practical relevance), proved its correctness and provided formulas to over-approximate the linearization errors. We claim that in some conditions, sufficiently large admissible linearization errors, our algorithm outperforms most methods based on griding the state-space. The case of the convexity of the viability kernel for continuous time and sampled time systems was also addressed with the adequate proofs.

5.2 Future Work

Along this project, there were many uncharted paths one could pursue with the many ideas that occurred. Here are a few of them with a brief explanation,

5.2.1 Develop the remaining Architecture

The most obvious next step is to develop the conceptual blocks of the architecture here proposed.

- Sensor and Perception block would greatly benefit from an intense study of the minimal optimal amount of sensors and what type they need to be, while aiming also at the lowest cost possible. One interesting idea to implement in the perception layer is to detect the pedestrian awareness of the ego-vehicle, see a related case [**?**]. For example, the ego-vehicle is near a crosswalk and a

pedestrian is not facing the vehicle, he/she might go straight to the crosswalk without even looking for cars. If trajectories are chosen to take into account this occurrences, a few more accidents could be avoided.

- As told in the respective chapter, methods to find optimal paths in graphs are exhaustively present in the literature, picking one to implement Map-Based Planning block should not be troublesome.
- Properly define the stage cost functions and the respective minimization algorithms for the controller present in the Reflexive Planning.
- Contingency plans in case of failures in each block. What measures should be employed in case the viable set of inputs is the empty-set. What to do in case an unexpected obstacle appears.

5.2.2 Simulation the full model and implementation in a real system

After developing the remaining blocks of the architecture it is necessary to simulate the system, for the sake of having an idea of the performance of the overall architecture. Simulation will allow to fine tune some parameters and hierarchical relations among the several modules of the system.

The final step is to implement in a real system. This is almost always the most cumbersome step when developing a theoretical algorithm. Sampling times, parameters of the actual controllers of the actuators are some examples of important variables to take into account in the real system.

5.2.3 Non-Convex Constraint Sets

One interesting problem is in the case of non-convex constraint sets. The obvious next step is to apply convex decomposition algorithms to that set, and convert them to a list of convex polyhedron, for example. However, there is the subtlety on how to deal with the boundaries between divisions of the set. We propose the idea of hard and soft boundaries when applying the algorithms presented here (Figure **??**) Hard boundary is the one that separates the constraint set from the rest of the space. The soft ones separate the convex divisions of the original constraint set. This adds complexity on is currently done, since the algorithm has to make the overlap test with all the remaining sub-parts of the constraint set.

5.2.4 Dynamic Viability Kernels

What if the constraint sets have their own dynamics, despite of the system behavior, how would that influence the viability kernel. It would be reasonable to assume that if one knows how the constraint set changes in time, we would be able to modify the viability kernel accordingly, without having to compute it all over again. This particular question could introduce a novelty since, to the best of our knowledge, there are no theoretical results to this problem.

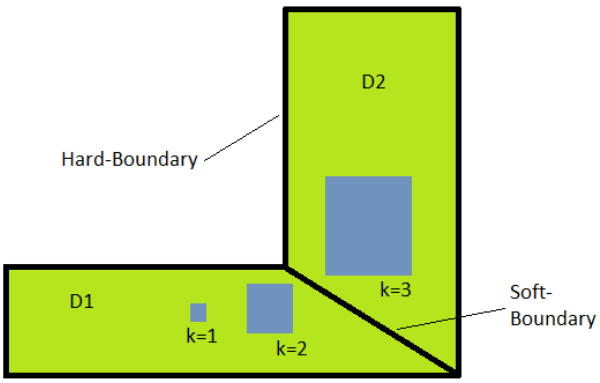


Figure 5.1: The reachable sets in dark (blue) color, the non-convex constraint set in light (green) color and its divisions