TÉCNICO LISBOA

Contingency Planning for Automated Vehicles in Urban Traffic

João Pedro de Campos Salvado

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisor(s): Luis Manuel Marques Custódio

Chairperson:	Prof. João Manuel Lage de Miranda Lemos
Supervisor:	Prof. Luis Manuel Marques Custódio
Member of the Committee:	Prof. Maria Isabel Lobato de Faria Ribeiro

November 2015

”All change is hard at first,
messy in the middle
and gorgeous at the end.”
Robin Sharma

Acknowledgments

I would like to address a special thanks to Instituto Superior Técnico for the knowledge and expertise that were so useful during this work and I believe would also be in the future. More specifically a special greeting to the Professor Luis Custódio and his amazing course, which made me easy to choose such research project in the field of artificial intelligence. I also thank Professor Luis Custódio for being present during the master thesis process by exchanging emails and giving interesting feedback even 3000 kilometers away.

I can not forget giving a special thanks to: The Automotive Department of *Institut für Verkehrssystemtechnik* (Institute of Transportation Systems) of the *Deutsches Zentrum für Luft- und Raumfahrt* e.V. (DLR) Braunschweig, which gave me the necessary logistic supply and support to conduct my research. I am grateful for an amazing supervising work conducted by Daniel Hess, which was available all the time and would come to every meeting with a contagious passion on the topic and that was a source of inspiration to myself; and to the IT Felix Fleig who was always promptly available.

To my family specially my mother, father and sister that with long *Skype* conversations supported me all the way to here. And, of course, the effort that my father went through reviewing my thesis, which was truly helpful.

And last but not the least, to all my friends that are passing through the same process of writing the master thesis, Duarte Alves, Teresa Reis, Ricardo Diogo and my office colleague and friend Cagatay Erbateur which I spent long hours in constructive discussion. Thank you all.

Resumo

Imagine um mundo onde todos se poderiam deslocar de um ponto para o outro de uma forma fácil e segura. Onde mesmo, pessoas com deficiência motora e ou visual possam benificiar de uma condução autónoma.

O Departamento de Automação da DLR está a desenvolver investigação no campo da cooperação entre veículos autónomos em áreas urbanas. Com este projecto o instituto pretende desenvolver um planeador de contingência para tráfego urbano, que encontre uma solução em 0,1 [s] .

Algoritmos de *Anytime search* podem ser utilizados no planeador de contingência calculando uma trajectória segura e viável em ambiente livre, tirando partido de uma heurística inflacionada e das *motion primitivives* construidas *á priori*.

Na fase off-line do planeador de contingência pretende-se desenvolver um conjunto de *motion primitives*, seguindo um modelo com um sistema não-linear, limitado por condicões de fronteira, que retira o processo de verificação da viabilidade das trajectórias da fase on-line, onde o tempo tem um custo mais elevado. A fase online é a fase de procura onde é construido um *graph* composto por trajectórias tirando partido das *motion primitives*.

Uma melhoria no número de nós inválidos explorados e tempo de execução foi verificada nos testes efectuados. Relativamente aos nós foi conseguida uma redução de vinte a sessenta vezes e foi encontrada um solução entre seis a dez vezes mais rápida.

A stratégia implementada, na qual a parte inflacionada da heurística sobrestimada pode ser utilizada para realizar uma pesquisa mais informada, oferece uma *framework* para futuras adaptações e desenvolvimentos.

Palavras-chave: Keywords: *Motion Primitives*, *Anytime Search*, Heurística Sobrestimada , Planeador de Contingência

Abstract

Imagine a world where everyone could get from point to point in an easy and safely way. Where elderly or even visually impaired would greatly benefit from autonomous driving.

The Automotive Department of *Institut für Verkehrssystemtechnik* (Institute of Transportation Systems) of the *Deutsches Zentrum für Luft- und Raumfahrt* e.V. (DLR) Braunschweig is conducting research in the field of cooperative and automated vehicles driving in urban traffic. It is intended to develop a contingency planner for urban traffic that finds a solution in a 0.1[s] time frame.

Anytime Search algorithms can be utilized in constrained contingency planning to compute a safe and feasible path in a free environment by taking advantage of an inflated heuristic and pre-computed motion primitives. In the offline phase of the contingency planner it is intended to develop a set of motion primitives, following a nonlinear system model, bounded by physical constraints which takes the burden of feasibility verification from the online phase, where time has a higher cost. The online phase is the search phase in which a graph of maneuver segments is constructed using the motion primitives.

An improvement in the number of explored invalid nodes and runtime was verified in the experiments. In the first a reduction of twenty to sixty times and in the second it was accomplished a six to ten times faster search.

The strategy implemented, where the inflated part of the overestimated heuristic could be utilized to make a better informed search, gives a framework for future adaptations.

Keywords: Motion Primitives, Anytime Search, Overestimated heuristic, Contingency Planner

Contents

List of Tables

List of Figures

Nomenclature

$\bar{f}$ true cost of the solution found

$\dot{s}$ state first derivative.

ϵ bound factor

$\mathcal{O}$ the universe of all the objects detected by the ego vehicle.

$\mathcal{S}_{\sqcup}$ the group of all trajectory sets in a trajectory super set.

$\mathcal{S}_{FREE}$ state space that is collision free.

$\mathcal{U}$ control space.

μ friction coefficient.

$\overline{d_{os_t}}$ the average distance between the center of the obstacles (o) and the center of the trajectory set (s).

ϕ steering angle.

$\tau_u(s_I, t)$ trajectory function in order of s_I and t.

$\text{Inflated}_{\text{part}}$ the overestimation made by the ϵ bound in the evaluation function.

θ heading.

A_x longitudinal acceleration.

a_y lateral acceleration.

D distance between vehicles

f_{inflated} bounded evaluation function.

$f_{\text{uninflated}}$ unbounded evaluation function.

F_s safety factor.

F_x Llongitudinal force.

F_y lateral force.

F_z	normal force.
g	gravity constant.
K	curvature.
L	vehicle length.
s_F	final state.
s_I	initial state.
s_l	path length variable.
t_F	final time.
t_I	initial time.
TIV	time inter vehicle
TTC	time to collision
v_x	longitudinal velocity.
W	vehicle width.
F	transition function.
f(s)	evaluation function
g(s)	true cost of going from the initial state/node to the final state/node s
h(s)	heuristic function that predicts the true cost from state s to the goal
s	state.
s	state.
t	trajectory time.
u	control input.
x	longitudinal position.
y	lateral position.

Glossary

AMEIA*$_3$	Anytime Multi Set Environment Informed A^* with 3 sets
AD	Anytime D^*
ARA	Anytime Replanning A^*
BCP	Boundary Value Problem
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
DLR	*Deutsches Zentrum für Luft- und Raumfahrt*
ESC	Electronic stability control
ESP	Expansive Space Planner
Ep	Number expansions
IE	Number of invalid graph explorations
LPA	Lifelong Planning A^*
MHA	Multi-Heuristic A^*
M	Maximum number of nodes in Memory
OBB	Oriented Bounded Box
PRM	Probabilistic Road Map
RRT	Rapidly-Exploring Random Trees
SBL	Single Query Multi Directional Lazy Planner
SMHA	Shared Multi-Heuristic A^*
SOCS	International Symposium on Combinatorial Search
TIV	Time Inter Vehicle
TS1	Time spent till first solution
TTC	Time To Collision
T	Time spent till optimum solution
VE	Number of valid graph explorations

Chapter 1

Introduction

One of the greatest challenges is to formally prove the safety of a contingency planner. Since it is made of complex algorithms and strategies, further more the unpredictability and complexity of the outer environment plays a huge role on that behalf.

The Automotive Department of Institut für Verkehrs system technik (Institute of Transportation Systems) of the Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR) Braunschweig conducted research in the field of cooperative and automated vehicles driving in urban traffic. This is part of the UnCoVer CPS european project, which has a consortium of the following institutes and companies.


The overall DLR project approach falls upon two concepts: the first is in integrating several modules/planners which are optimized for a sort of situations (e.g emergency situations, nomimal driving, etc.). The second is developing those separate modules. For the case of the contingency planner it is also intended to subdivide it into an online and offline part.

In the offline phase, a set of motions primitives are generated, pursuing a nonlinear system model bounded by physical constraints, which takes the burden of feasibility verification from the online phase, where time has a higher cost.

The online phase is the search phase in which a graph of maneuver segments is constructed using the motion primitives. During this phase the collision tests upon moving obstacles and road boundaries

are conducted.

A maneuver planner for normal operation does not have to take into account *unreasonable* behavior of other traffic agents. And a contingency planner has to find, at all time, a totally safe maneuver. It is intended that the contingency planner and nominal planner work simultaneously and if one assures that the contingency planner has at least one safe maneuver then the combined system is also safe. In order to understand the scope of the module contingency planner in the overall DLR project the figure below can be consulted.

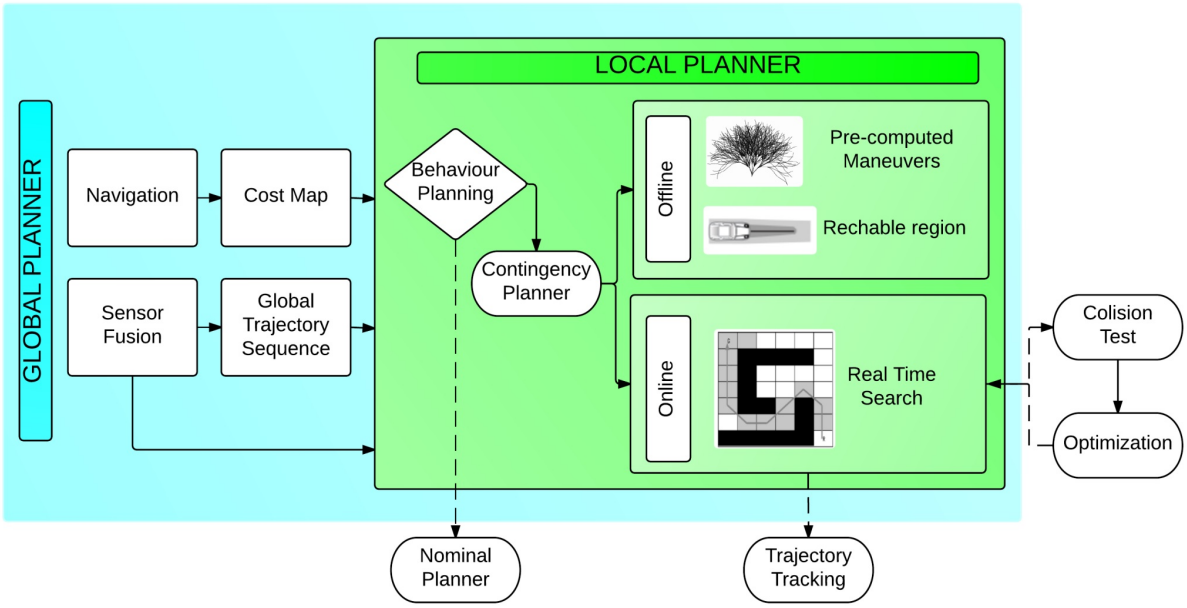


Figure 1.1: Overview of the hierarchical planner of the DLR project

The figure above is an hierarchical planner that represents the organization of the DLR overall project with all its modules. The hierarchical planner in the picture employs two distinct sub-planners which rely on two different paradigms. The global planner, which is out of the scope of this thesis, lies on the spectrum of long-range and low fidelity. On the other hand, the local planner lies on short-range and high-fidelity. A behavior module will be developed in the future in order to decide how all the modules will work together (nominal an contingency planner). An optimization phase could also be post-developed if the motion primitives do not fulfill certain criteria. The focus of this work relies on the contingency planner which is made of two distinct phases, an online and offline phase. It will also be created a collision testing strategy for the online phase. Both global planner features and trajectory tracking module are already implemented and will be utilized.

Hierarchical planners have many advantages such as: reactivity (fast local planner re-planning cycle), minimal startup latency (after local planner re-plan cycle, 10HZ, a set of control sequences is sent to the vehicle), scalability (inherited from the combination of both planners) and feasibility (the local planner creates trajectories contemplating motion physical constraints), Howard et al. [2008]. Those are the reasons why hierarchical planners present them-selfs as a good choice.

1.1 Motivation

Imagine a world where everyone could get from point to point in a easy and safely manner. And where elderly or even visually impaired would greatly benefit from autonomous driving.

Automotive industry has made important developments in ensuring the safety of the driver and its passengers by introducing technologies as airbags, anti-lock brakes, electronic stability control (ESC), crumple zones, automatic braking, adaptative cruise control, obstacle detection sensor systems and the list continues. Nevertheless, there is still a huge number of fatalities even in developed countries, as it can be seen below.

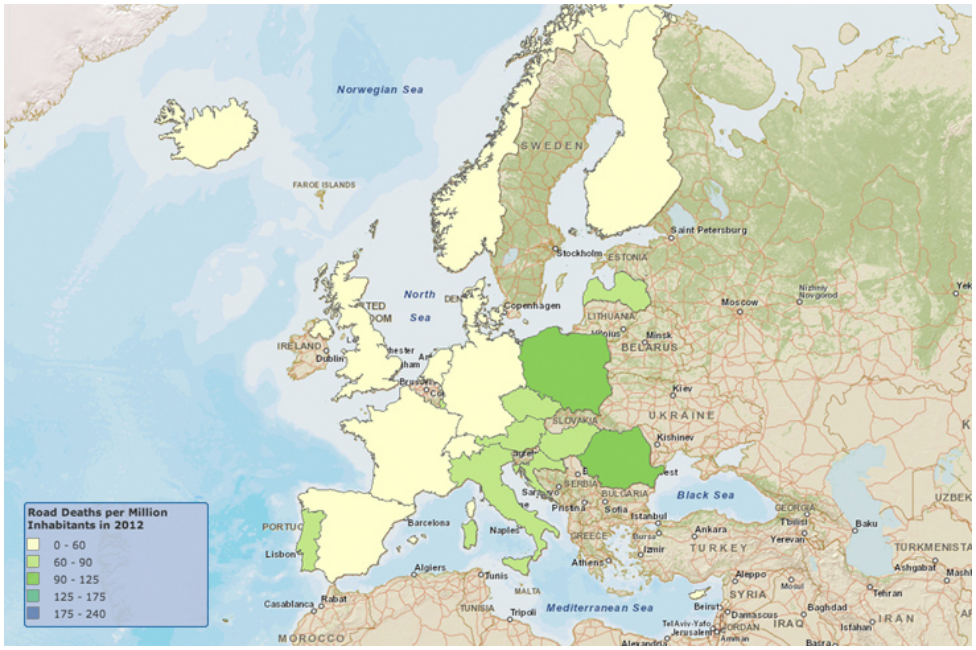


Figure 1.2: The road Safety Atlas provides accident statistics for each European country, `http://ec.europa.eu/transport/road_safety/specialist/statistics/index_en.htm`

Even though these systems had an important role in reducing the fatalities in the past decades, the number of accidents is not followed by this trend. In fact most of these systems have the aim to minimize the hazardous consequences of an accident. However, there is still a common denominator in all accidents that cannot be improved by this technologies. The driver.

The Geneva Convention of 1949 on Road Traffic as well as agreements in other areas like road rules and licensing stated in the article 8, that every vehicle have a driver who is *"at all times... able to control"* it. This does not represent an issue if one considers a system in which the driver belongs to the loop. However, one could argue that the real value of an autonomous driving system comes from being ready, alert at all the time even when the driver is not. Another solution to overcome this issue could rely on developing a system that could give a set of indications to the driver.

Nowadays, with the upcoming of the *google car* and advances in autonomous driving, new laws are being discussed. In United Kingdom it is predicted the implementation of 100 self-driving pods in Milton-Keynes between 2015 and 2017 which will allow a preliminary test on this technology.

One of the driverless vehicle competition of the *DARPA Grand Challenge* is known as *DARPA Urban Challenge*. It occurred in 2007 at the George Air Force Base. This was the first main event where autonomous vehicles where driving in a urban traffic situation. The winner was *BOSS* of the *Tartan Racing Team*, a collaborative effort between *Carnegie Mellon University* and *General Motors*. They choose a hierarchical planner. This event was a great source of inspiration. A video of the winning vehicle can be seen in here `https://www.youtube.com/watch?v=lULl63ERek0`

Google self-driving car project is good a example of the upcoming of this technology into society daily life. It already had self-driven over one million miles and the prospect of autonomous driving being a day reality is closer. An interesting link related with this project can be seen on Ted talks presented by Chris Urmson heads up of the project. `https://www.ted.com/talks/chris_urmson_how_a_driverless_car_sees_the_road`

Unmanned aerial vehicles (UAVs) are crossing into civil airspace, as a result it is necessary to develop new methods to ensure collision avoidance at all time. Moreover it requires an integration with the existent Traffic Alert and Collision Avoidance System (TCAS) and Air Traffic Control (ATC). This could be an area of interest where the planner here developed could be applied in the aeronautical field.

1.2 Objective

The Automotive Department of *Institut für Verkehrssystemtechnik* (Institute of Transportation Systems) of the *Deutsches Zentrum für Luft- und Raumfahrt* e.V. (DLR) Braunschweig is conducting research in the field of cooperative and automated vehicles driving in urban traffic. The focus of the research here developed is to create a contingency planner for urban traffic that finds a solution in a 0.1 [s] time frame.

1.3 Contributions

- The creation of an anytime search informed heuristic, with outer environment information, maintaining characteristics such as a sub-optimally bounded solution (bounded by the ϵ factor value of the last iteration), as well as completeness.

- The construction of a more computationally complex heuristic makes the search better informed which allows less invalid explored nodes making it possible to have a reduced runtime for the first solution.

- The construction of motion primitives that are suited for the emergency situations in the tested scenarios and that are not a burden in terms of memory to the online search. Several strategies where implemented in order to reduce the number of created trajectories explosion, when the grid resolution is increased. Such as a graph prune strategy, time ellipses bounds and physical bounds.

- An iterative method of making collision tests. Where simpler tests are made in the beginning and if a collision is detected (since it can be a false positive) the complexity of the tests increase in the next iteration. Tests such as first applying an overlapping test between a pie shape of the trajectory super set and the road boundaries, next an OBBtree test in which is in itself an iterative collision test, where in each depth of the tree the number OBB's enclosing the vehicle also increase. Resulting in an fast but also safe method to apply during the search phase.

1.4 Thesis Outline

Charpter 2 provides a review on the state of the art of this work. Also asserts what is the problem statement and inserts the contingency planner in the combinatorial search scope. It is also depicted the terminology and challenges of the contingency planner.

On Chapter 3 the methodologies developed are explained. A detailed description of how the contingency planner was implemended with two section, one for motion primitives and the other for the anytime search.

Chapter 4 consists on a presentation of the results for all the tests, and for several different scenarios. These were chosen aiming to be able to conclude and compare on the methods created in Chapter 2.

On Chapter 5, conclusions on the different scenarios presented and achievements reached with this work are listed. Suggestions for future work are also given. Appendix A contains trajectory path set examples.

Chapter 2

Problem Statement

Anytime Search algorithms can be utilized in constrained motion planning to compute a safe and feasible path in a free environment by taking advantage of an inflated informed heuristic and pre-computed motion primitives.

2.1 Combinatorial Search Scope

Combinatorial search is a field of artificial intelligence which has been applied to an increasing number of areas of knowledge and situations. E.g. finding the shortest round trips(TSP), finding models of propositional formulae (SAT), planning, scheduling, internet data packet routing, protein structure prediction and the list continues, Hoos and Stützle [2004].

Being so, it is important to define which type of combinatorial problem is approached in this work. In a simple manner, combinatorial problems are tackled by finding groups/combinations of objects which satisfy given conditions and afterwards evaluate if a given combination is in fact a solution.

Combinatorial problems can be divided in decision and optimization problems. Since, the aim of the planner is to find at least one solution, this is in the area of decision problems. Moreover, it fits the search variant, Hoos and Stützle [2004]. However, even though this is not an optimization problem, if a solution is found and there is still planning time remaining, the planner will attempt to optimize.

There are two main search paradigms. One is systematic search and the second is local search. In the first paradigm the search space is traversed in a systematic manner and in the second it starts from a certain position and continues to search in the neighborhood. Systematic search is complete and local search is typically incomplete. Even though this is an hierarchical planner, the focus of the motion planner developed is in local search. Which is often better suited when an acceptable solution is required in a short time frame, or when there is partial knowledge of the environment and usually parallel processing is necessary. Based in Hoos and Stützle [2004]

The assertion depicted above fits into the overall scope. Nevertheless, if one intends to define a more specific area of knowledge, according to *Frazzoli [2001]*, this is Constrained Motion Planning in a Free environment. In which, given a mechanical control system, and initial and final conditions it is

intended to find a control input sequence in a time frame, which produces a collision free motion of the system and ends in a desired goal state.

2.2 Terminology

This section is based upon these previously developed works, Howard et al. [2009], Knepper [2011a] and Pivtoraiko [2012], in which is defined the following terminology paradigms.

2.2.1 System

The system can be described by a nonlinear differential equation $\dot{s} = F(s(t), u(t))$, with $s \in \mathcal{S}$ the system state. $\mathcal{S} \subseteq \mathbb{R}^n$ in the state-space of the system. The control input is $u \in \mathcal{U}$, with $\mathcal{U} \subseteq \mathbb{R}^m$ the control space of the system.

2.2.2 Motion Primitve

A motion primitive can be considered as every control input $u \in \mathcal{U}_d$ with $\mathcal{U}_d$ being the discrete control space. Such control inputs are a solution of the discrete-time model $s_{k+1} = F_d(s_k, u_k)$, where F_d is an approximation of the original state transition function, LaValle [2006].

2.2.3 State Space

A set of all the possible states $\mathcal{S}$. The state space of the Motion Primitives has a dimensionality of six. In this application we are using the following state space to describe the motion of the vehicle.

$$s = [x, y, \theta, v_x, a_y, t] \tag{2.1}$$

where:

- s : state
- x : longitudinal position
- y : lateral position
- θ : heading
- v_x : longitudinal velocity
- a_y : lateral acceleration
- t : trajectory time

2.2.4 Control Space

A controlled vehicle moves upon certain inputs, the set of these inputs is called control space $\mathcal{U}$.

$$u = [a_y, v_x] \tag{2.2}$$

where:

u : control input
a_y : lateral acceleration
v_x : longitudinal velocity

2.2.5 Trajectory

It is defined as vector valued function of states (s) in order of time (t), or trajectory length (s_l), or other non-decreasing variable.

$$\dot{s} = F(s(t), u(t)) \tag{2.3}$$
$$\tau_u(s_I, t) = \int_{t_I}^{t_F} F(s(t), u(t))dt + s_I$$

where:

$\dot{s}$: state first derivative
s : state
u : control input
F : transition function
$\tau_u(s_I, t)$: trajectory function in order of s_I and t
s_I : initial state
t_I : initial time
t_F : final time

2.2.6 BVP - Boundary Value Problem

Problem of computing a control sequence in a manner that the resulting trajectory satisfies the boundary constraints.

$$\dot{s} = F(s(t), u(t)) \tag{2.4}$$
$$s.t. \qquad \tau_u(s_I, t) = s_I$$
$$\tau_u(s_I, t) = s_F$$

where:

$\dot{s}$: state first derivative

s : state

u : control input

F : transition function

$\tau_u(s_I, t)$: trajectory function in order of s_I and t

s_I : initial state

s_F : final state

t : time

2.2.7 Motion Planning

Problem of determining acceptable actions/inputs, which allows the vehicle to move from an initial to a final location, taking the presence of obstacles and physical constraints into consideration, while optimizing for an utility function (f).

$$u^* = \underset{u \in \mathcal{U}}{\operatorname{argmin}} \int_{t_I}^{t_F} f(\tau_u(s_I, t), u(t))dt \tag{2.5}$$

$$s.t. \quad \tau_u(s_I, t) = s_I$$

$$\tau_u(s_I, t) = s_F$$

$$\tau_u(s_I, t) \in \mathcal{S}_{Free} \ \ \forall_t \in [t_I, t_F]$$

where:

u^* : optimal input

$\mathcal{U}$: control space

$\dot{s}$: state first derivative

s : state

u : control input

$\tau_u(s_I, t)$: trajectory function in order to s_I and t

s_I : initial state

s_F : final state

t : time

t_I : initial time

t_F : final time

$\mathcal{S}_{FREE}$: state space that is collision free

2.3 Challenges

This section is based on the previously developed works, Howard et al. [2009],Knepper [2011a] and Pivtoraiko [2012].

In the context of **Feasibility**, this is the challenge of solving the motion planning problem by finding the correct control sequence that satisfies a state transition function that can be physically followed by the vehicle and also assure that the transition between trajectory segments is continuous . Depending on the application, the required degree of continuity C^1, C^2, etc. to ensure feasibility differs. As autonomous vehicles are tested upon increasingly difficult environments, travel at higher speeds and have the aim to perform harder tasks, the assurance of feasibility of generated trajectories gets more complicated. There exists two different approaches on ensuring feasibility. The first, when the trajectories are created during the planning cycle by a BVP-solver and the second when they are created offline where more complex algorithms can be implemented. It occurs that sometimes there is no closed form solution for a BVP-Problem being necessary to use a shooting method. Although in the online phase such methods are not recommended due to time constraints. Even though, one could agree that taking care of the feasibility of trajectories offline is a viable strategy, there are certain feasibility challenges which could not be solved offline (e.g. terrain variation, loose soil and air currents).

Optimality is a property that measures the quality of a solution. In this planner is a minimization problem where a certain evaluation function (f) is optimized. This function could have, for example, weighted members with the following variables: distance, time, risk, energy, lateral acceleration, etc. Since, the aim of the contingency planner is to find a solution which is feasible, optimality is a second order goal. So it is prioritized finding a solution as fast as possible rather than optimizing existing ones.

A crucial aspect of the planner is the **Runtime**, the time that the planner requires to find a solution. Computational Efficiency is of key importance even more in a contingency planner where it is necessary to have a quick reaction in case it occurs an unpredictable dangerous situation (e.g. right front vehicle overtaking). Parallel processing is a viable way to tackle this issue. The idea behind the system developed is to design several planners for clusters of situations and all of the planners would be a module working in parallel.

Completeness is a property of the motion planning algorithm that dictates whether a solution satisfies the motion problem conditions and determine if a solution exists. Usually, it is difficult to guarantee

such property, mainly because of unpredictable obstacles and differential constraints of automated systems. Nevertheless, another concept rises up, resolution completness, which satisfies the completness property till the degree of the environment representation. Another approach is reachability analyses where the extreme motions of the agents are considered and as result one is always moving in a safe environment. Other concept is probabilistic-completeness where the solution probability of being a safe maneuver tends to the unity.

Both **Uncertainty** and **Partial-Knowledge** of the environment have an intrinsic relation with the previous properties. Even though the present work does not have the objective to tackle this issues directly, one has to always bare in mind that they have an influence on the contingency planner.

2.4 State-of-the-art

2.4.1 Motion Planning

Grid Based Planner

A grid based approach is a deterministic search of motion primitives in a discretized state and/or control space. Some examples of this approach are an N-order grid, Barroque Latombe motion primitives, Barraquand and Latombe [1993], Mihail Pivtoraiko Lattices, Pivtoraiko et al. [2009]. As a result of the discretization of the state space, one can only assure resolution completeness. On the other hand, the quality of the path sets generated is high, since the computation of the trajectories is made off-line, which allows the resolution of a BVP Problem by using time demanding numerical methods, when there is no closed form solution and the state space is sampled instead of the control space. Even when the control space is sampled off-line, in case is not necessary to solve a BVP, the higher order of the search space can have an effect in the online search.

Since an increase in the number of the motion primitives also increases the number of expansions (branching factor) during the online phase and only a limited data file size can be accepted, several strategies were developed in order to reduce its amount by following different criteria. Some of the metrics used during the sampling of the state space are: discrepancy, dispersion or diversity, Erickson and LaValle [2009], Knepper et al. [2009]. In Knepper [2011b], a formulation concerning metrics to sample the state space, such metrics are: quasimetric area metric (related with the computation of the area between consecutive points of two paths), Green and Kelly [2007], Mutual Collision metric (measures the probability of two paths colliding into the same object, it does not know the object *a priori* so it is computed an average over all possible configurations), Exact Area metric, Hausdroff metric (similarly to Green Kelly metric it increases diversity, but instead of using an area uses the distance between two points along two different paths at the same path length), Hilbert Space L2 metric (is an increment of the Green Kelly metric by taking other dimensions into account, e.g. time, acceleration, etc.), Path Diversity Inner Product, Branicky et al. [2008] (deterministic approach that relies on the combination of two criteria, path cells occupation and path cell overlapping). Nevertheless, there are deterministic approaches

exclusive to grid based methods, e.g. pruning method for newly created trajectories by accessing previous lattices and comparing them within a certain adjustable boundary factor, this strategy is defined as primitive set decomposition, Pivtoraiko and Kelly [2011]. As is shown in figure 2.1, the paths generated by Pivtoraiko are lattices.

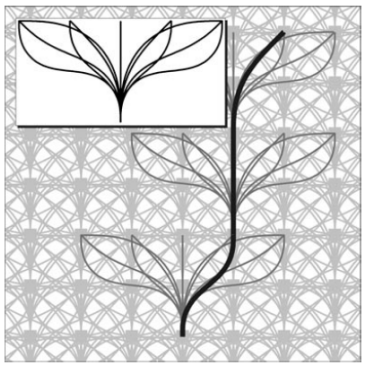


Figure 2.1: State lattice for general car like system, Pivtoraiko [2012]

Potential Fields Planner

This approach is related to an optimization process where an evaluation function is overlaid into an environment map or specific features of a map (e.g. obstacles). In terms of optimality this method has the problem of reaching a local minimum, as a result it can not assure an optimal solution in most of the approaches, Ram et al. [1996]. There are exceptions where a post search optimization is conducted. Covariant Hamiltonian Optimization for Motion Planning (CHOMP), is one example, where an initial guess is derived by an optimal search method and afterwards a post optimization is executed, Zucker et al. [2013]. It is shown in figure 2.2 the paths generated in the previous referred work. A similar approach was conducted in Hesse et al. [2010]. Concerning runtime the method is fast and the quality of the solutions are high. Usually potential fields are a good solution for nominal planning where it is intended to increase comfort and safety of the driver.

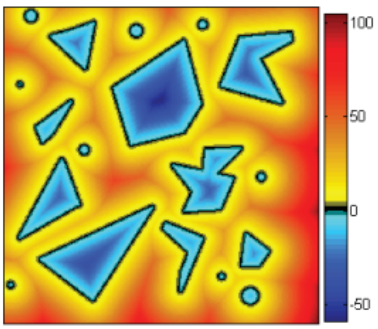


Figure 2.2: Signed distance field, Zucker, Ratliff, Dragan, Pivtoraiko, Klingensmith, Dellin, Bagnell, and Srinivasa [2013]

Discrete Optimization Hierarchical Planner

A hierarchical planner subdivide the overall system into several modules. One is local planner, also known as motion planner module; behavior module, which rules based on reasoning and a global planner that takes care of the mission goals. This type of planner have certain characteristics such as: reactivity (local planner), scalability (global planner), minimal start-up latency, and some examples can be found in the following literature: Howard et al. [2009](an example of the created trajectories can be consulted on the following picture 2.3), Knepper [2011b] and Kelly et al. [2006]. In those planners there were no motion primitives built offline. The search space is sampled online and then trajectories are generated (BVP problem). Nevertheless, it is a balance between quality and flexibility of the solution, which in this case, when compared with grid based methods, an hierarchical planner has a lower solution quality but a higher flexibility. A drawback of this kind of planners relies on the fact that is incomplete. Sampling the search space is an entire field of expertise, but generally, in the examples of hierarchical planners referenced above, the final states of a constrained graph are sampled online. Other formulation is related with a sampling strategy along the road centerline and width, Chu et al. [2012], Gu et al. [2013]. Since trajectory generation online is time demanding, a possible solution to surpass this issue is a decoupling of a trajectory generation into path and speed profiles, Xu et al. [2012].

An important topic concerning trajectory generation is the choice of the trajectory ”primitive”. Where one can encounter several examples in the literature, such as: trigonometric splines, Biagiotti and Melchiorri [2008], cubic and other order polynomials, Nagy and Kelly [2001]; linear paths with parabolic blends, Biagiotti and Melchiorri [2008]; quadratic and cubic Bezier curves, Yang et al. [2014]; cubic trigonometric polynomial B-spline Basis function with shaping parameter, Han [2004]; clothoids, Fraichard and Scheuer [2004], arc length parametrized curves ,Wang et al. [2002]. This choice is a counterbalance between runtime, feasibility and optimality. Other possible approach is to choose simple trajectories and in a post optimization phase smoothing can be applied. The vehicle model is also important in generating trajectories.There are two vehicle models that use Ackermann steering geometry in its definition. The Dubins [1957] and the Reeds-Shepp car model, Reeds and Shepp [1990]. The last is an extension of the Dubins car model where it is allowed to have trajectory curves with an initial heading greater than 90 degrees (going backwards).

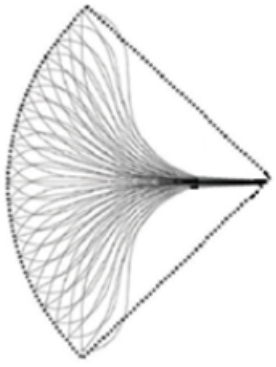


Figure 2.3: Uniform terminal state sampling, Howard, Green, Kelly, and Ferguson [2008]

Random Sampling Based Planner

Random Sampling Based Planners are probabilistic methods of motion planning. As a result, it is only possible to assure probabilistic completeness, because it can be proven that with the increase of the number of samples the probability of finding a solution converges to unity. Feasibility is assured during the online phase. In terms of runtime, it is a fast method when comparing with previous approaches. On the other hand, the quality of the solution is low, which results in the necessity of a post optimization/smoothing process. The Rapidly-Exploring Random Trees (RRT) from LaValle [1998] (an example of the trajectories created can be seen in the following picture 2.4), which has a single query tree, is a well known example in this type of planners. Further improvements on RRT were developed, in which the search is biased to enable a generation of complex maneuvers in complex environments, Kuwata et al. [2009], Ma et al.. Kavraki et al. [1996], introduced the Probabilistic Road Map (PRM) as a multi query graph planner where an high dimensional state space is sampled and afterwards attempts to connect neighbor states to form a graph. Expansive Space Planner (ESP) ,Hsu et al. [2002], is another road map alike planner where a graph is generated by first selecting a node randomly and secondly creating an edge by applying a random action into that node. Single Query Multi Directional Lazy Planner (SBL) uses characteristics of both methods PRM and RRT, but postpones collision tests because, in the random sampling planners depicted previously, most of the generated paths are not apart of the final solution.

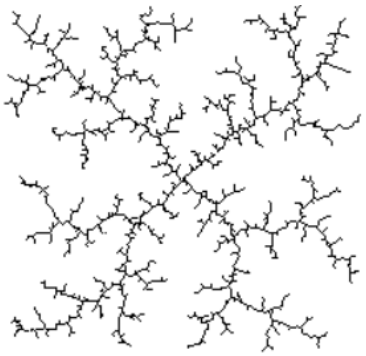


Figure 2.4: RRT tree exploring four corners rapidly, LaValle [1998]

It is necessary to consider that what is depicted above does not contemplate all methods in each motion planner approach, and further improvements were conducted in each field in order to overcome certain relative disadvantages. Nevertheless here is an overview below based upon the following work, Knepper [2011a].

Table 2.1: Comparison between motion planner approaches

	Optimality	Completeness	Runtime	Feasibility	Solution Quality
Potencial Fields	Local Minimum	No	Fast	Online	High
Random Sampling	Probabilistic	Probabilistic	Fast	Online	Low*
Grid Based	Sub-Otpimal Bound	Resolution	Slow	Offline	High
Discrete Optimization	Sub-Otpimal Bound	No	Slow	Offline/Online	High

*post optimization/smoothing can be utilized.

2.4.2 Search Methodologies

The following work Ferguson et al. [2005] gives an overview about several combinatorial methods going from the fundamentals with the classic A* to more recent methods. A* is a method used in static environments, Hart et al. [1968], where the environment does not change so it is known and certain. A first development to A* is the D*Lite which allows replanning cycles. ARA* gives anytime characteristics to the planner, Likhachev et al. [2003]. LPA* is able to repair the search tree with a changing environment, Likhachev and Koenig [2005]. Finally AD* combines the anytime characteristics of ARA* and the repairing features of the LPA* algorithm. Anytime search is a pragmatic approach for trading solution cost and solving time. It can also be used for solving problems within a time bound. Three frameworks for constructing anytime algorithms in a bounded suboptimal search have been proposed: continuing search, repairing search and restarting search, Thayer and Ruml [2010a]. Continuing is a bounded suboptimal search which after encountering a solution continues by iterating the process allowing ever improving solutions. The OPEN list is maintained in each cycle. This approach was initiated in the following work Hansen and Zhou [2007]. Repairing searches have a different manner of handling duplicate states by creating three list (OPEN, CLOSED and ICONS), where all inconsistent nodes are stacked, rather than immediately expanded, until the next iteration of the repairing cycle. This method is valuable in situations where an agent finds a solution initially and while moving through the goal perceives a changed environment, being easier to repair only the different part of the environment, Koenig et al. [2004]. And a second difference is related with the fact that in each cycle the search starts from the initial node instead of a node with the lower evaluation function value in the OPEN list. Restarting search is similar with continuing search where a suboptimal bound is tightened in each iteration improving the solution in each cycle. The main difference is that it restarts the search from the initial node. This approach gets rid of the low-h-bias in the initial phase of the search (since the effect of the inflated part of the heuristic is bigger at lower depths, final states would have lower evaluation function values, which would be responsible for having similar solutions in the following anytime cycles) and such is accomplished by recomputing the cost of each node in the OPEN list knowing already the true cost to the goal from the first iteration, Koenig and Likhachev [2006].

Concerning heuristics, exists methods which allow the usage of inadmissible heuristics with an anchor admissible heuristic (an anchor heuristic has properties such as admissibility which when used together with other inadmissible heuristics allows the maintenance of such properties), e.g.Thayer et al. [2008], in which a rather simple technique is used where the heuristic is the minimum between the anchor admissible and the inadmissible heuristic (*pathmax* strategy). The International Symposium on Combinatorial Search (SOCS) is great a source of such methods. Where one can also find strategies using multi-heuristic search with uncalibrated heuristics, e.g Aine et al. [2014] , several OPEN lists are used, each one with an uncalibrated heuristic and one of the OPEN lists has an anchor heuristic to

assure admissibility. With this strategy it is possible to create several heuristics for different situations which will make the overall planner more adaptable to the environment features. Approaches to the multi-heuristic strategy can go from MHA*++, Focal-MHA*, to Unconstrained-MHA* ,Narayanan et al. [2015]. In the previous method OPEN lists are completely separated, but it is possible to share information between lists with SMHA* which can be seen in one of its variations in this work Brown et al. [2014]. Nevertheless, in MHA* it is easier, to do parallel processing, than SMHA*. Other formulations intend to find acceptable solutions faster by using inadmissible heuristic by using the number of nodes to expand till the goal node, Thayer and Ruml [2010b]; or after finding the initial solution (suboptimal) by computing the probability of finding a solution with lower cost, Potential Search, Stern et al. [2010]. The following publication Holte [2010] presents a discussion upon common misconceptions concerning heuristic search such as: more accurate heuristics result in fewer node expansions, A* does fewer expansions than any other equally informed method which finds an optimal solution or bidirectional A* stops when both trees frontiers meet.

Chapter 3

Methodology

3.1 Motion Primitives

The Motion Planner has two distinct phases. One of them is Online and the other is Offline. The Offline phase of the planner aims to compute the set of motion primitives, which will be utilized during the Online phase where the search method is applied.

The motion primitives are well defined trajectories that result from solving a BVP problem between a starting and final state, which belong to a discretized state/action space referred as constrained graph.

3.1.1 Constraint Graph

The continuum sampled state space referred as constraint graph can be seen in the figure below.

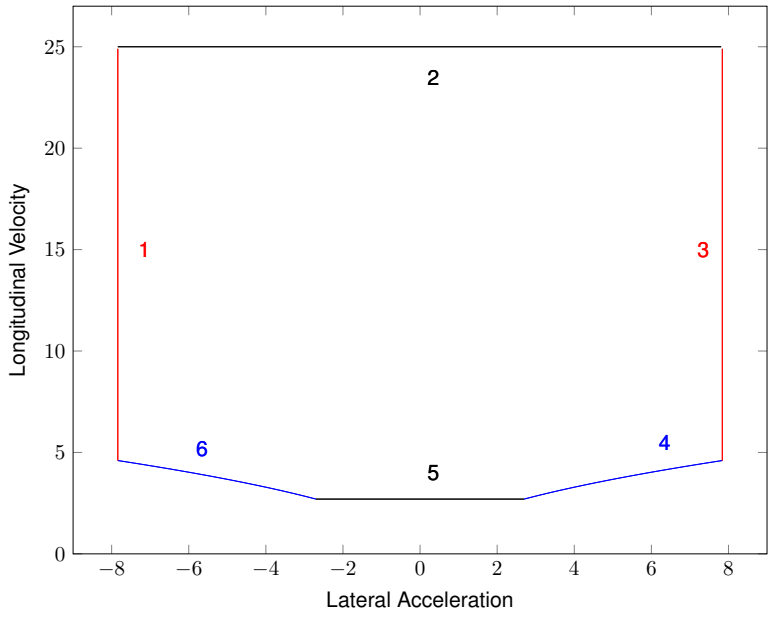


Figure 3.1: Constraint Graph representing the continuous state space, in 2D (A_y,V_x), defined by six constraints.

The three major constraints are related with the velocity, normal acceleration and curvature of the vehicle. Then, each of them is subdivided in another two constraints, the maximum and the minimum value allowed for each.

Concerning the velocity constraints, the assumptions made in previous motion planners will be here utilized. E.g. in the DARPA challenge the maximum allowed velocity of the vehicles was 50 [km/h] , Buehler et al. [2009]). The time constraint 10[Hz] of the planner to find a viable and safe solution represents a huge challenge, moreover if the vehicle is moving at high speed. So as a result, the maximum velocity was set to be 90 Km/h (note: the DARPA challenge was in 2007, in recent years happened several developments which could assure higher velocities and in the scope of this project it is intended to surpass previous accomplishments). It can also be considered that at 10[Km/h] the vehicle is almost static. As a result the velocity constraints are the following.

$$\text{Constraint 2 - } V_{x\text{max}} \leq 90 \text{ [Km/h]}$$

$$\text{Constraint 5 - } V_{x\text{min}} \geq 10 \text{ [Km/h]}$$

where:

V_x : longitudinal velocity

Relatively to the lateral acceleration the constraints chosen follow the well known dynamic constraint, the traction circle.

$$\begin{aligned} \sqrt{{F_x}^2 + {F_y}^2} &\leq \mu F_z \\ \sqrt{{A_x}^2 + {A_y}^2} &\leq \mu g \\ \text{if } \ A_x = 0, \ \text{ then} & \\ A_y &\leq \mu g \end{aligned} \tag{3.1}$$

where:

F_x : longitudinal force
F_y : lateral force
F_z : normal force
A_x : longitudinal acceleration
A_y : lateral acceleration
g : gravity constant
μ : friction coefficient

One has to take into account that the friction coefficient in both F_y and F_x usually are not the same, which would transform the traction circle into a ”traction ellipse”. In fact, usually the μ_y is smaller than μ_x, which would reduce the maximum possible force in y-direction. As a result, it was assumed a common low friction coefficient.

$$A_y \leq \mu \mid g \mid \quad : \mu = 0.8 \tag{3.2}$$

where:

A_y : lateral acceleration

g : gravity constant

μ : friction coefficient

Definition of the 1 and 3 constraint lines of the graph which can be viewed on figure 3.1.

$$\text{Constraint 1 - } A_{y\text{max}} \leq 7.8 \ [\text{m/s}^2]$$

$$\text{Constraint 3 - } A_{y\text{min}} \geq -7.8 \ [\text{m/s}^2]$$

where:

A_y : lateral acceleration

The last constraints are the maximum and minimum curvature. These boundaries have a direct relation with the model of the vehicle chosen.

Since the main objective of the contingency planner is not related with optimal trajectories, it was prioritized simplicity. So, the model is based on the following work, Dubins [1957].

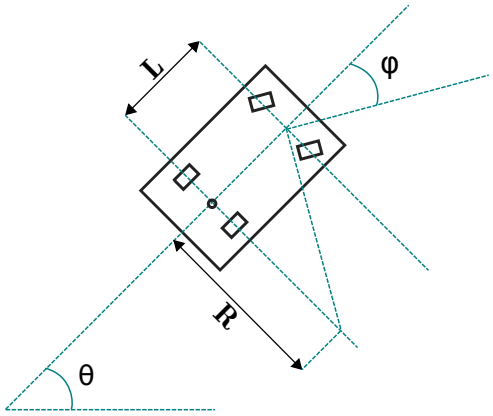


Figure 3.2: Dubins Car kinematic model - L represents the length of the vehicle, R the curvature radius, θ the angle between a inertial reference system and the vehicle reference system, the ϕ represents the steering angle

Being so, the curvature is constrained by the steering angle and the length of the vehicle. In literature usually the value for the maximum steering angle is assumed to be 45° . Note: the true size of the testing vehicle is used, in which Length$(L) = 2.7[m]$ and Width$(W) = 1.9[m]$ As a result, the maximum curvature for low velocities is computed as follows.

$$tan(\phi) = \frac{L}{R} = L \ K \tag{3.3}$$

$$K_{\text{max}} \leq \frac{tan(\phi_{\text{max}})}{L} = \frac{tan(\frac{\pi}{4})}{2.7} [\text{m}^{-1}]$$

where:

L : vehicle length

K : curvature

ϕ : steering angle

The equations representing the constraints of the maximum curvature in the constraint graph are:

$$\text{Constraint 4 - } A_y \leq |K_{\text{max}}| \ {V_x}^2$$

$$\text{Constraint 6 - } A_y \geq |K_{\text{min}}| \ {V_x}^2$$

where:

K : curvature

A_y : lateral acceleration

V_x : longitudinal velocity

3.1.2 Vehicle Model

As it can be seen on the picture 3.2 the model chosen is the *Dubins Car*, which has the following Kinematic equations.

$$\begin{aligned} \frac{dx}{ds_l} &= cos(\theta) \\ \frac{dy}{ds_l} &= sin(\theta) \\ \frac{d\theta}{ds_l} &= K \end{aligned} \tag{3.4}$$

where:

x : longitudinal position

y : lateral position

s_l : path length variable

K : curvature

θ : heading angle

Applying the chain rule:

$$\frac{ds}{dt} = \frac{ds_l}{dt}\frac{ds}{ds_l} = V\dot{s} \tag{3.5}$$

Transforming the kinematic equations depicted in 3.4 as it follows

$$\begin{aligned}\frac{dx}{dt} &= Vcos(\theta)\\ \frac{dy}{dt} &= Vsin(\theta)\\ \frac{d\theta}{dt} &= VK\end{aligned} \tag{3.6}$$

where:

x : longitudinal position
y : lateral position
t : time
K : curvature
V : vehicle velocity in s_l direction

This kinematic model allows the creations of clothoid trajectories. Nevertheless, even though clothoids are a starting point, several assumptions and changes were developed into its definition, in order to have trajectories that better suit the motion planner specific requirements. The general definition of clothoids comes from the Fourier series with only one degree of freedom.

$$K(s_l) = \sum_{n=0}^{1} p_n {s_l}^n = p_1 s_l + p_0 \tag{3.7}$$

where:

K : curvature
s_l : path length variable
p_1 : parameter 1
P_0 : parameter 0

If s_l = 0 then $K(0) = p_0$ (already known starting curvature), so the only parameter that is not defined is p_1.

$$p_1 = \frac{K(s_l) - p_0}{s_l} \tag{3.8}$$

where:

K : curvature
s_l : path length variable
p_1 : parameter 1
P_0 : parameter 0

As a result clothoids only have one degree of freedom and are computed by knowing the intended final curvature of the trajectory.

Since the curvature is defined by the lateral acceleration and the velocity, going from a set of coordinates in the constraint graph 3.1 is the same as going from an initial to a final curvature.

Note:

$$K = \frac{A_y}{V^2} \tag{3.9}$$

where:

K : curvature

V : vehicle velocity in s_l direction

A_y : lateral acceleration

Summing up, it is intended to create a trajectory from an initial set of coordinates in the constraint graph (A_{y_0},V_0) to a final set of coordinates (A_{y_1},V_1) using a clothoid trajectory and a vehicle model inspired in the *Dubins Car*. Next it will be presented several assumptions and choices that will be discuss afterwards.

Assumptions made:

$$\begin{aligned} A_x &= sign(V_1 - V_0)\sqrt{g^2 - max(|A_{y_0}|, |A_{y_1}|)} \\ \Delta t &= \frac{V_1 - V_0}{A_x} \\ \frac{dA_y}{dt} &= \frac{A_{y_1} - A_{y_0}}{\Delta t} \\ {A_y}^2 + \left(\frac{V_1 - V_0}{\Delta t}\right)^2 &\leq g^2 \ : \ 1 \leq \Delta t \leq 5[s] \end{aligned}$$

where:

V_0 : vehicle initial velocity in s_l direction

V_1 : vehicle final velocity in s_l direction

A_{y_0} : lateral initial acceleration

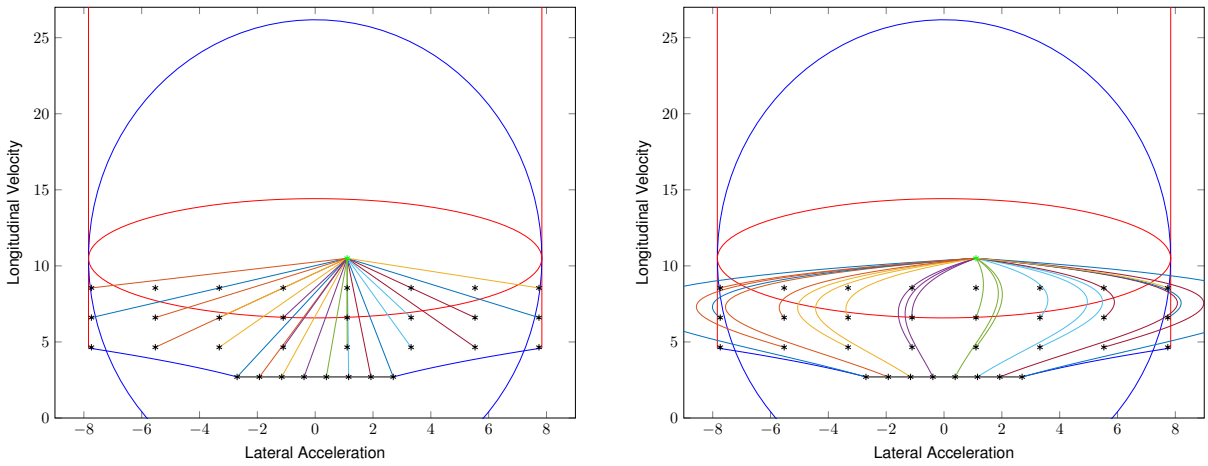
A_{y_1} : lateral final acceleration

g : gravity constant

Δt : trajectory time

The longitudinal acceleration (A_x) is computed using the traction circle equation, which can be found in 3.1. By choosing between the maximum value of the initial and final lateral acceleration ($max(|A_{y_0}|, |A_{y_1}|)$), as a result A_x is underestimated. In the clothoids definition, the velocity in order to time is decoupled from the path (X,Y), making it only dependent of s_l. As a result, it is possible to define any kind of velocity profile. In this case was chosen a linear profile. It was also assumed a constant lateral acceleration derivative. A plot with trajectories with a zero and non-zero constant derivative can be seen in this figure 3.3, an explanation concerning this choice is depicted after the figure. Furthermore, it is used the traction circle to define bounding ellipses, such that the trajectories created were between one and five seconds. The reason behind it has to do with the fact that: in the case the trajectory has an infinite time frame, every coordinate of the graph can be reached; nevertheless the longer the trajectory is, it is more probable that it can not be followed, because of the unpredictable nature of

the environment. The reason for a low bound is to assure that, on the other hand, the vehicle goes to a state which is consistently different from its predecessor.



(a) $\frac{dA_y}{dt} = const.$ (b) No contraint on A_y

Figure 3.3: Path set of trajectories starting from an initial set of coordinates (A_{y_0},V_{x_0}). The blue (5s) and red (1s) ellipses represent the time constraints depicted in 3.10

As it can be seen in the pictures above, when the lateral acceleration derivative is not assigned as a constant the trajectories overshoot. As a result it is not possible to assure that they remain within the constraint graph.

The complete model based upon Dubins model, that follows the assumptions and constraints depicted above, is the following.

$$\begin{aligned} \frac{dx}{dt} &= V_x cos(\theta) \\ \frac{dy}{dt} &= V_x sin(\theta) \\ \frac{d\theta}{dt} &= \frac{A_y}{V_x} \\ \frac{dV_x}{dt} &= A_x \\ \frac{dV_y}{dt} &= A_y \end{aligned} \tag{3.10}$$

where:

V_x : vehicle longitudinal velocity

V_y : vehicle lateral velocity

A_y : lateral acceleration

A_x : longitudinal acceleration

x : longitudinal position

y : lateral position

θ : vehicle heading

The reference frame of the coordinate system is the vehicle center of mass and the x axis is assumed to be parallel to the trajectory, as it can be seen in the figure bellow.

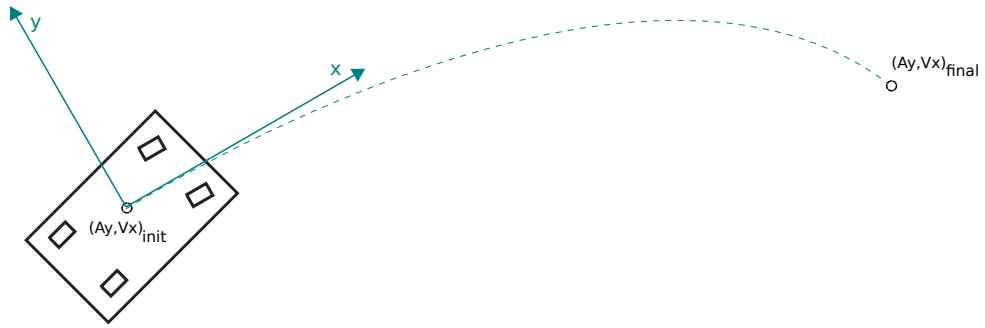


Figure 3.4: Vehicle following a clothoid trajectory with changing curvature

One has to notice that a geometric increase of the constraint graph resolution results in an exponential explosion of motion primitives. This fact generates two main issues. First the data file of motion primitives can quickly reach a size that is not acceptable. And second, more edges are expanded in each expansion step. In order to overcome these issues an empirical pruning strategy was utilized. Where instead of creating all the trajectories that have an end state inside the bounding ellipses, only trajectories within a grid cell distance from the bounding ellipses are used. The alternative constraint graph can be seen below.

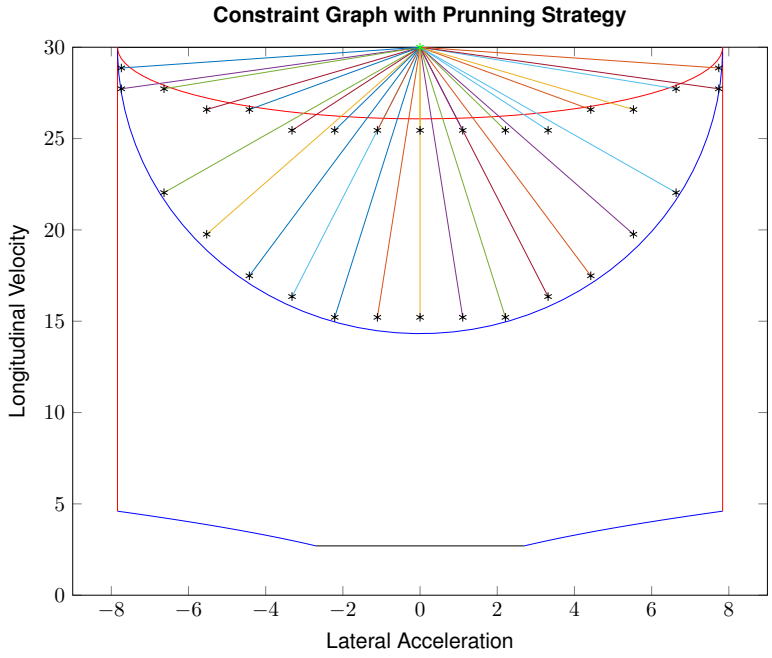


Figure 3.5: Constraint Graph representing the continuous state space, in 2D (A_y,V_x) with a prunning strategy applied. And the resulting trajectory set with a starting state with maximum velocity and zero lateral acceleration

This strategy allows the construction of both short and long trajectories and also contemplating all the lateral possible accelerations while not losing resolution. For example, if it is necessary to reach

a state inside the ”half moon” shape, in the figure above, then that means that a short maneuver can be explored and afterwards it is possible to reach such state in the new trajectory set. To the author knowledge it is not possible to agree upon what is preferable for the online search method applied. For instance, which one is better, having a low branching factor or a low graph tree depth. Both effects could compensate, but one has to bear in mind that with this strategy the data file is highly reduced while maintaining search resolution.

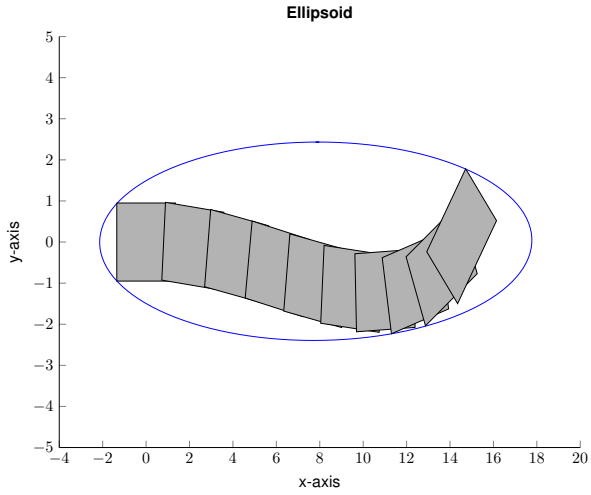
3.1.3 Collision Test Approaches

Several approaches were developed in order to test possible vehicle collisions. One has to consider that collision testing is a crucial part of a contingency planner. Since the outer environment is dynamic, unpredictable and could potentially contain many obstacles; which results in an runtime expensive part of the algorithm. In fact, during one search step, many collision tests should be done in order to increase the probability of finding a safe trajectory. Further more, for specific situations or agents, different methods are implemented. If the collision tests are relative to other dynamic objects, the trajectories swaths were enclosed by an ellipsoid, arc section or OBB tree. On the other hand, concerning road boundaries collisions it was defined a pie shape that encloses the path set of trajectories. The implementation of the collision approaches in the search online are depicted in the following section 9.

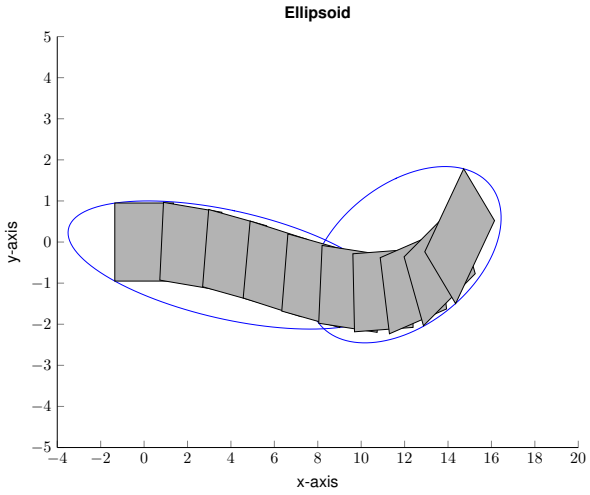
Ellipsoid

The Ellipsoid shape is created in such a way that all the trajectories are enclosed by itself. Since the variables of the trajectories were the longitudinal and lateral distances, the Ellipsoid is reduced in its simplest form, an Ellipse. Nevertheless, the strategy is implemented to allow future improvements on that behalf (e.g. increasing the dimensionality of the trajectory representation by adding the time variable). One must note that the vehicle dimensions are also considered.

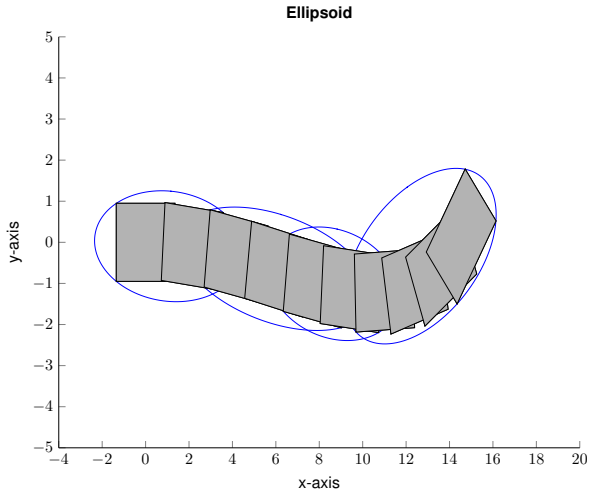
In order to accomplish such shape it was used a MATLAB toolbox created by *Nima Moshtagh*, *Minimum Volume Enclosing Ellipsoid*. A representation of the ellipsoid can be consulted in the figures below.



(a) One Ellipsoid



(b) Two Ellipsoids



(c) Fours Ellipsoids

Figure 3.6: Ellipsoid in a blue line enclosing the vehicle dimensions for a trajectory considering vehicle dimensions. Figure (b) has two ellipsoids and (c) has three, they represent depth one and two, respectivelly, of the ellipsoids tree

Note: In an hypothetical situation it can happen that it is detected a collision, but it is false positive. Because, has it can be seen in the figure 3.6 there is an area inside the ellipsoid that is considered as a collision but in fact it is not. In order, to reduce false positives an adaptive method is implemented where the enclosing volume is reduced by increasing the number of ellipses. Nevertheless, more ellipsoids also means more time spent in the collision test phase.

Based on the best knowledge of the author, there is not a fast method to evaluate if the obstacle is inside the shape when comparing with the following methods (note: test upon ellipsoid *versus* ellipsoid). Being so, further more when compared with the other formulations this is the strategy that suits worse the trajectory swath. Nevertheless, it is here for future reference.

Arc Section

The arc section is another strategy to enclose the trajectory swath. Next, follows up an explanation on how it was designed, but first a schematic of such geometry is presented in the following figure.

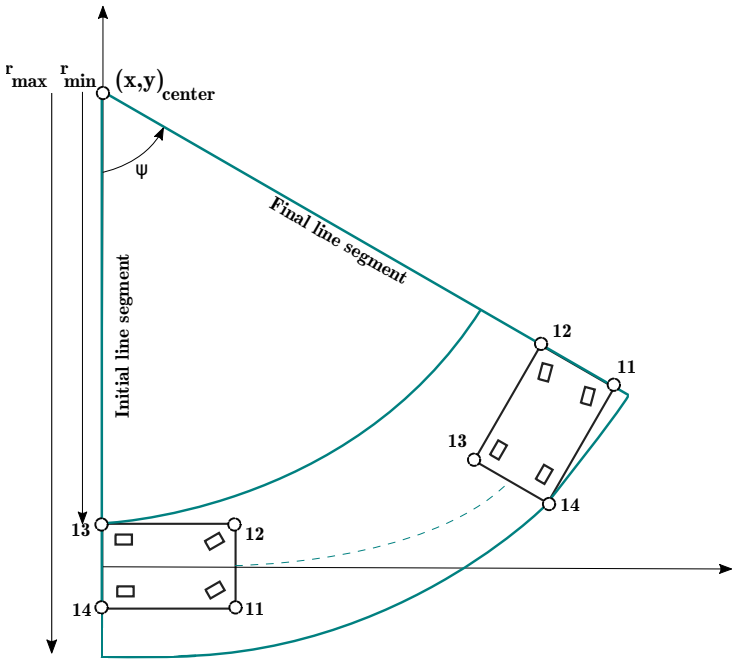


Figure 3.7: Arc Section visual representaion.

Initial line segment which can be seen above in figure 3.7 is defined next.

$$x_{\text{initial}} = x_{13} \tag{3.11}$$

where:

x_{13} : longitudinal position of bottom left corner of the vehicle at t=0 [s]

The final line segment present in the figure 3.7) is defined next.

$$m_{\text{final}} = \frac{y_{12_{\text{final}}} - y_{11_{\text{final}}}}{x_{12_{\text{final}}} - x_{11_{\text{final}}}} \tag{3.12}$$

$$b_{\text{final}} = y_{12_{\text{final}}} m_{\text{final}}\, x_{12_{\text{final}}}$$

where:

m : slope

b : point where line crosses y-axis

x_{11} : longitudinal position of top right corner of the vehicle at t=t_{Final} [s]

x_{12} : longitudinal position of top left corner of the vehicle at t=t_{Final} [s]

y_{11} : lateral position of top right corner of the vehicle at t=t_{Final} [s]

y_{12} : lateral position of top left corner of the vehicle at t=t_{Final} [s]

The coordinates of the center of the arc section are computed in the following manner.

$$x_{\text{center}} = X_{\text{initial}}$$
$$y_{\text{center}} = m_{\text{final}}\, x_{\text{center}} + b_{\text{final}}$$

m : slope

b : point where line crosses y-axis

x_{center} : longitudinal center of the arc section

y_{center} : lateral center of the arc section

The next step consists in computing the minimum and maximum radius of the center of the arc section.

$$\begin{aligned} r_{\text{max}_{13}} &= \sqrt{\left(x_{\text{center}} - X_{13}\right)^2 + \left(y_{\text{center}} - Y_{13}\right)^2} \\ r_{\text{max}_{11}} &= \sqrt{\left(x_{\text{center}} - X_{11}\right)^2 + \left(y_{\text{center}} - Y_{11}\right)^2} \\ r_{\text{max}} &= \max\left(r_{\text{max}_{13}}, r_{\text{max}_{11}}\right) \end{aligned} \tag{3.13}$$

$$\begin{aligned} r_{\text{min}_{14}} &= \sqrt{\left(x_{\text{center}} - X_{14}\right)^2 + \left(y_{\text{center}} - Y_{14}\right)^2} \\ r_{\text{min}_{12}} &= \sqrt{\left(x_{\text{center}} - X_{12}\right)^2 + \left(y_{\text{center}} - Y_{12}\right)^2} \\ r_{\text{min}} &= \min\left(r_{\text{min}_{14}}, r_{\text{min}_{12}}\right) \end{aligned}$$

x_{center} : longitudinal center of the arc section

y_{center} : lateral center of the arc section

x_{11} : longitudinal position of top right corner of the vehicle

x_{12} : longitudinal position of top left corner of the vehicle

y_{11} : lateral position of top right corner of the vehicle

y_{12} : lateral position of top left corner of the vehicle

x_{13} : longitudinal position of bottom left corner of the vehicle

x_{14} : longitudinal position of bottom right corner of the vehicle

y_{13} : lateral position of bottom left corner of the vehicle

y_{14} : lateral position of bottom right corner of the vehicle

r_{min} : minimum arc section radius

r_{max} : maximum arc section radius

Note: the x and y boundary points differ with time, as a result the maximum and minimum radius have to be computed taking that into consideration. The ψ angle is then computed in the following way.

$$\psi = atan\Big(\frac{Y_{11_{\text{final}}} - Y_{12_{\text{final}}}}{X_{11_{\text{final}}} - X_{12_{\text{final}}}}\Big) \tag{3.14}$$

where:

ψ : angle between the inital and final line

As a result, the parameters that define an arc section are: ψ, r_{max}, r_{min}, X_{center} and Y_{center}.

Note: In order to plot the arcs it was used a MATLAB toolbox created by *Matt Fig*, *plot_arc*.

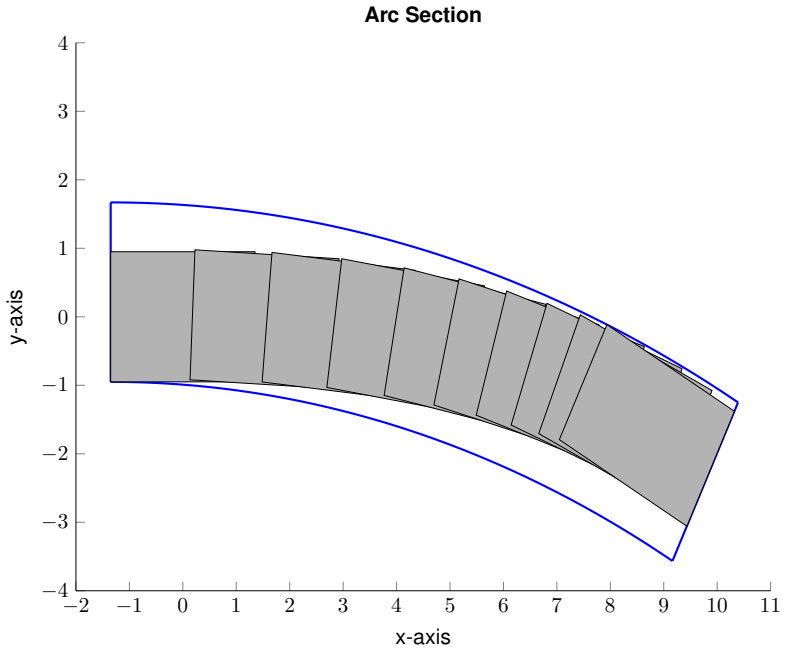


Figure 3.8: Arc Section in blue enclosing the trajectory swath of the vehicle with initial and final longitudinal velocity of 25[m/s] and 10.5[m/s] respectively, and with initial and final lateral acceleration of 0[m^2/s] and 1.9[m^2/s] respectively.

OBB Tree

The methodology for the creation and *a posteriori* collision detection follows the approach developed in the following work Gottschalk et al. [1996]. In which an hierarchical data structure using tight-fitting oriented bounding box trees (OBBTrees) for a trajectory and a fast overlapping collision test is presented. The depth of the OBBTree assigned by the author is two. Next is will be shown a representation of such tree constructed during the offline phase.

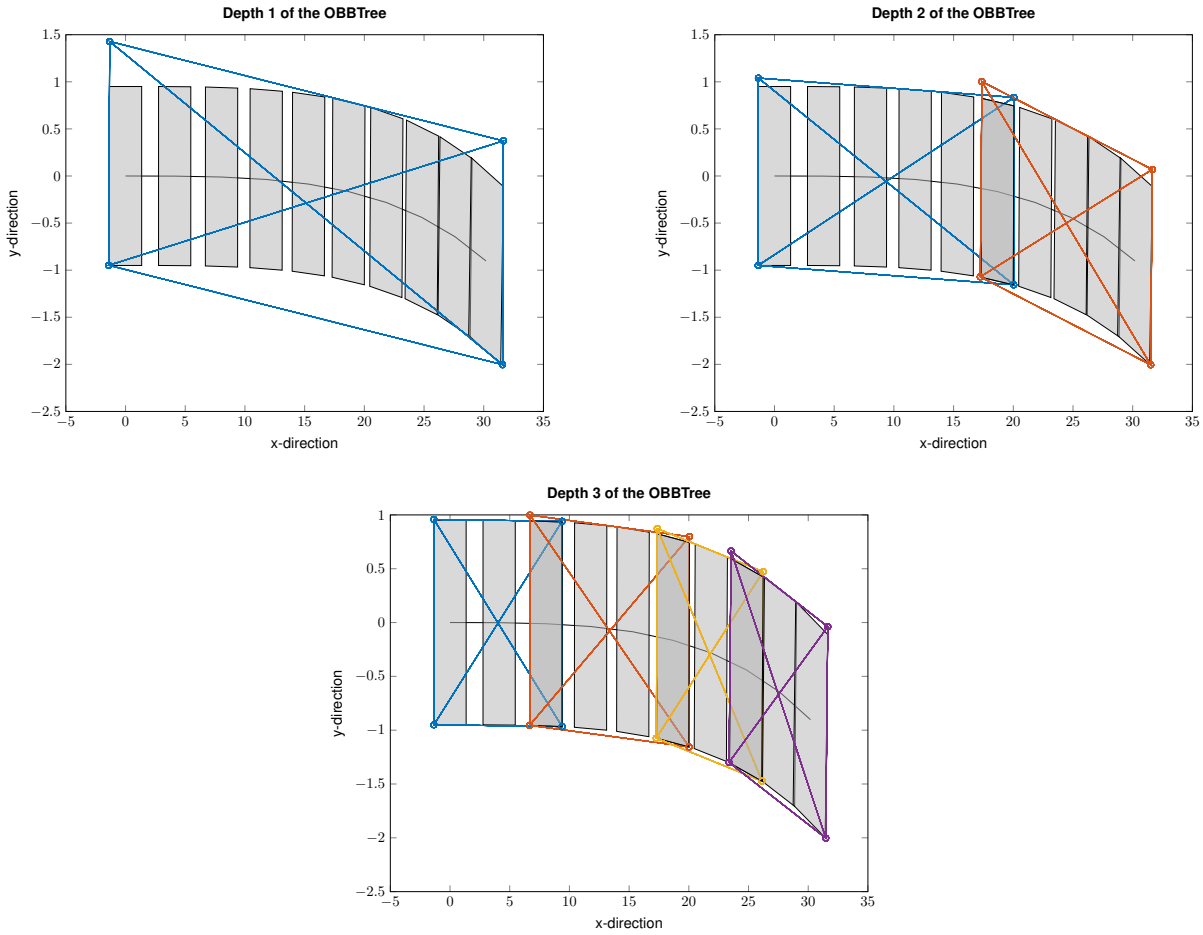


Figure 3.9: Representation of the bounding boxes present in the OBBTree structure of a trajectory

For a better vizualization the plotted OBB's are enclosing the trajectory in x and y. Nevertheless it is applied for x, y and t. A figure with the three variables an be consulted in the Appendix B.

Pie Shape

A pie Shape encloses several trajectories (trajectory set). This method is useful when testing a possible collision of a vehicle set of available actions against road boundaries. It is used as a preliminary test which can reduce the number of road boundary points necessary to test in a further phase of the online search, where tests are time demanding. The application can be seen on the following section 8. Each pie shape is defined by six variables $x_c, y_c, \theta_{max}, \theta_{max}, r, R$ that are computed as follows.

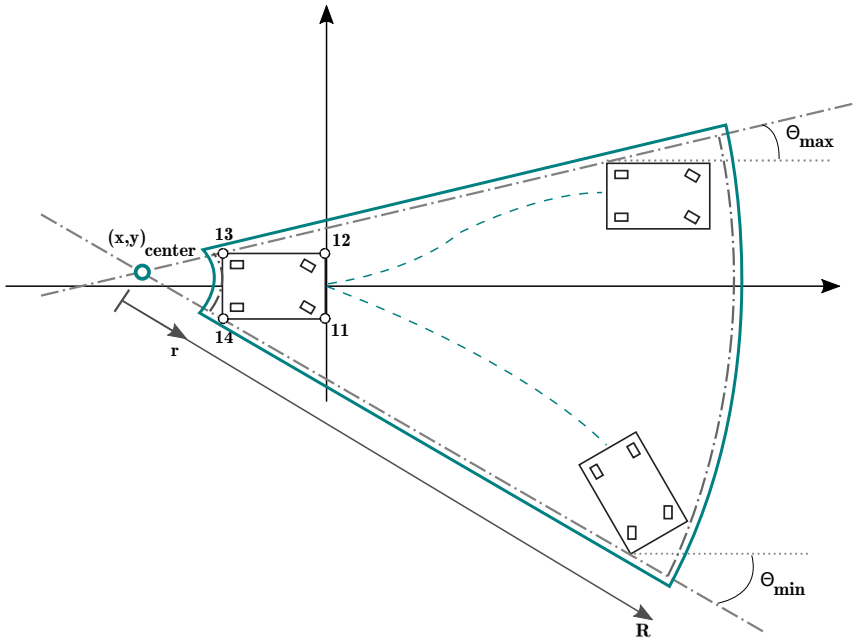


Figure 3.10: Pie shape enclosing two trajectories

$$\theta'_{\text{max}} = \max_{t\in]0,T]} \left(tan^{-1} \left(\frac{y_{2,4}(t) - y_4(t=0)}{x_{2,4}(t) - x_4 t = (0)} \right) \right)$$

$$\theta'_{\text{min}} = \min_{t\in]0,T]} \left(tan^{-1} \left(\frac{y_{1,3}(t) - y_3(t=0)}{x_{1,3}(t) - x_3 t = (0)} \right) \right)$$

$$b_{\text{max}} = y_4(t=0) - tan(\theta'_{\text{max}}) - x_4(t=0)$$

$$b_{\text{min}} = y_3(t=0) - tan(\theta'_{\text{min}}) - x_3(t=0)$$

$$x_c = \frac{b_{\text{min}} - b_{\text{max}}}{tan(\theta'_{\text{max}}) - tan(\theta'_{\text{min}})}$$

$$y_c = tan(\theta'_{\text{max}}) x_c + b_{\text{max}}$$

$$\theta_{\text{max}} = \max_{t\in]0,T]} \left(tan^{-1} \left(\frac{y_{2,4}(t) - y_c}{x_{2,4}(t) - x_c} \right) \right)$$

$$\theta_{\text{min}} = \min_{t\in]0,T]} \left(tan^{-1} \left(\frac{y_{1,3}(t) - y_c}{x_{1,3}(t) - x_c} \right) \right)$$

$$R = \max_{t\in]0,T]} \left(\sqrt{(x_{1,2}(t) - x_c)^2 + (y_{1,2}(t) - y_c)^2} \right)$$

$$r = \left| \frac{(y_2(t=0) - y_1(t=0))x_c - (x_2(t=0) - x_1(t=0))y_c + x_2(t=0)y_1(t=0) - y_2(t=0)x_1(t=0)}{\sqrt{(y_2(t=0) - y_2(t=0))^2 + (x_2(t=0) - x_1(t=0))^2}} \right|$$

max : variables with this subscript define the line segment in the bottom

min : variables with this subscript define the line segment in the top

θ' : angle between x-axis and the respective line segment

θ : angle between an horizontal line with x_c value and the respective line segment

c : this subscript stands for center

r : minimum radius

R : maximum radius

x_1 : longitudinal position of top right corner of the vehicle

x_2 : longitudinal position of top left corner of the vehicle

y_1 : lateral position of top right corner of the vehicle

y_2 : lateral position of top left corner of the vehicle

x_3 : longitudinal position of bottom left corner of the vehicle

x_4 : longitudinal position of bottom right corner of the vehicle

y_3 : lateral position of bottom left corner of the vehicle

y_4 : lateral position of bottom right corner of the vehicle

3.2 Anytime Search

The second phase of the motion planner is the anytime online search. During the search, a graph of nodes is generated, using the pre-computed motion primitives. Contrary to the offline part of the planner, in this phase, time is a crucial constraint. As a result, other considerations have to be taken into account. The anchor algorithm used is an epsilon bounded anytime search. There are several branches which could be followed in such a type of search and they can be seen in 2.4. A continuing anytime search was followed, nevertheless from now on it will be referred to as anytime search.
Epsilon bounded anytime search is an extension of the classical A for an unpredictable and partially unknown environment, which has the aim to find a solution in a short time period (10 Hz) and its evaluation function is the following

$$f(s) = g(s) + \epsilon h(s) \tag{3.15}$$

where:

$g(s)$: true cost of going from the initial state/node to the final state/node s
$h(s)$: heuristic function that predicts the true cost from state s to the goal
$f(s)$: evaluation function

This approach is fundamentally different in two points when comparing with the classical A^*. The first is the inflated heuristic, where the evaluation function (f) is depicted above. As it can be seen in 3.15, this inflation is due to the fact that the heuristic is multiplied by an epsilon factor. Increasing the relative value of the heuristic in the evaluation function makes the search more greedy. As a result, the algorithm goes more ”directly” (3.15) to the goal but looses optimality at the same time. Nevertheless, it can be proven that the cost of the solution found is at most sub-optimally bounded by ϵ, further more the main objective is to find at least one solution and optimality is a secondary criteria. The second point is related with the anytime steps. In each anytime step, ϵ is updated and nodes in the OPEN list are reordered. One of the strong points of A_ϵ is the fact that if there is enough planning time and knowing that in each anytime search step ϵ is updated, then ϵ will tend to 1 so A_ϵ tends to the classical A^*, so also to the optimal solution.

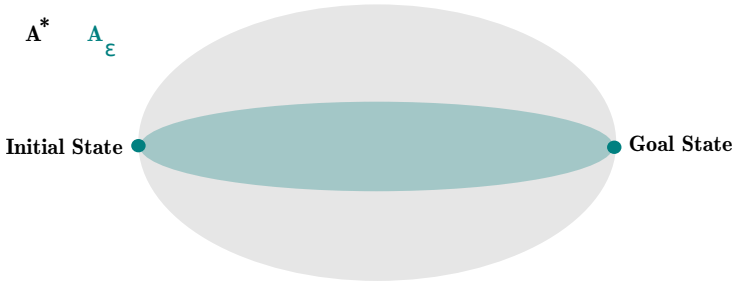


Figure 3.11: Representation of the nodes searched by each algorithm, A star and anytime A star

3.2.1 Algorithm Explanation

In this sub section it can be found a top-down explanation of the algorithm implemented.

Algorithm 1 Upload Motion Primitives Algorithm

procedure UPLOADMOTIONPRIMITIVES
  Define number of Trajectory sets $= n_{\text{sets}}$;
  while DATA file didn't end **do**
    Initialize n_{sets} Trajectory Sets with respective Pie Shape for a_{y_0} v_0
    Define $a_{y_{\text{end}\,1}}$,$a_{y_{\text{end}\,2}}$,$a_{y_{\text{end}\,3}}$,...,$a_{y_{\text{end}\,n_{sets}-1}}$
    for all Trajectories with same a_{y_0} v_0 **do**
      Trajectory = createTrajectory(x, y, θ, v, a_y, arc Section, OBB Tree);
      while setNumber $<$ n_{sets} **do**
        if $a_{y_{\text{end}_{\text{setNumber}}}} <= a_{y_{\text{end}}} < a_{y_{\text{end}_{\text{setNumber}+1}}}$ **then**
          Trajectory Set$_{\text{setNumber}}$$\leftarrow$ Trajectory;
        end if
        setNumber++;
      end while
    end for
    Initialize Trajectory Super Set
    for all Trajectory Set **do**
      Trajectory Super Set $\leftarrow$ Trajectory$_{\text{setNumber}}$;
      setNumber++;
    end for
    next(a_{y_0} v_0);
  end while
  for all Trajectory Super Sets **do**
    Trajectory Set Map = mapping(KEY(a_{y_0},v_0), Trajectory Super Set)
  end for
end procedure

Algorithm 1, demonstrates how the upload of the computed motion primitives offline is structured in the online algorithm. Each trajectory has information related with vehicle position, orientation, velocity and lateral acceleration; further more has an arc section and OBB tree that constraints the path taking vehicle boundaries 3.1.3 into account. Each trajectory set is a group of trajectories starting from the same (a_{y_0},v_0) and belonging between an $a_{y_{\text{end}}}$ interval. Next all the trajectory sets with the same (a_{y_0},v_0) make a trajectory superset. And finally it is computed a trajectory set map which is a mapping between a key made of (a_{y_0},v_0) and the trajectory super set. This structure allows an adaptive way for conducting collision tests.

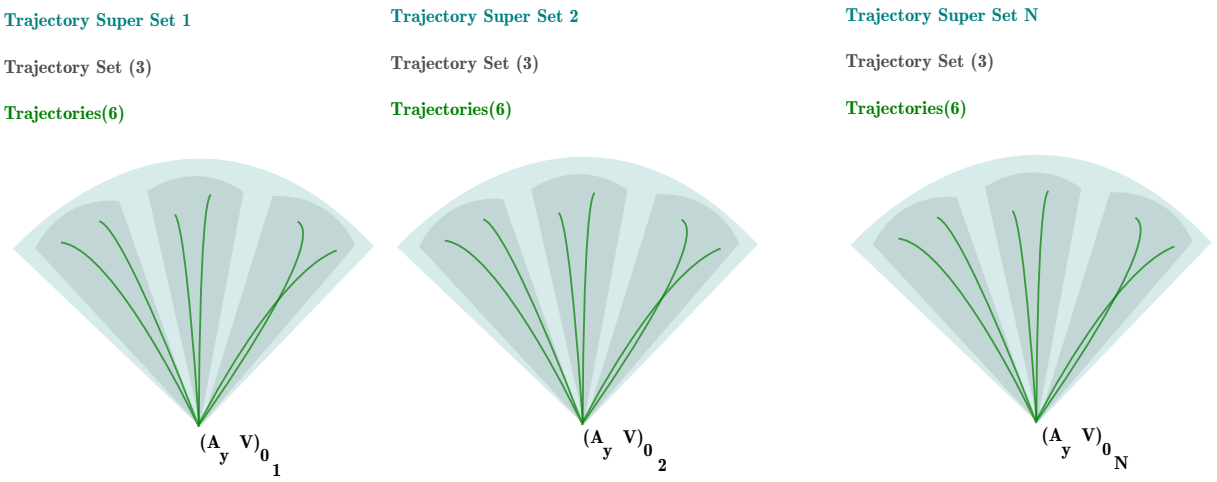


Figure 3.12: Representation of the data structure of motion primitives

Algorithm 2 Environment Representation

procedure CREATEENVREP

Ego Vehicle = sensorVEH(t, x, y, θ, v, a_y)

Map Origin = sensorMAP(x, y, θ, t)

for all Obstacles **do**

Obstacle = createOBS (t, x, y, v_x, v_y, a_x, width, length, roadID, laneID)

Obstacle.Path = obstacleModelPredictor(x, y, θ, v, a_x)

Obstacle.createOBBtree()

end for

end procedure

Environment is partially represented and, of course, it does not take information into account that is not perceived by vehicle sensors (unknown environment). Furthermore, the method implemented does not use all the information received by vehicle sensors. To the knowledge of the author, some of the information does not improve the specific planner and here it was applied the reverse engineer principle. Nevertheless, it is perceived position, orientation, velocity and lateral acceleration of the ego vehicle and the map origin defined by center position and orientation. The map is assigned to be the reference system. In a next step, obstacles are sensed and the algorithm uses its position, orientation, velocity, length, width, road ID and lane ID. For each obstacle detected a prediction of its trajectory is created using a simple predictor, in which it is assumed the vehicle maintains its initial acceleration, while moving along the lane, to which it was matched [3.2.1].

$$s_l(t) = \frac{1}{2}at^2 + v_0t \tag{3.16}$$

s_l : path length variable

v_0 : initial velocity.

a : initial acceleration

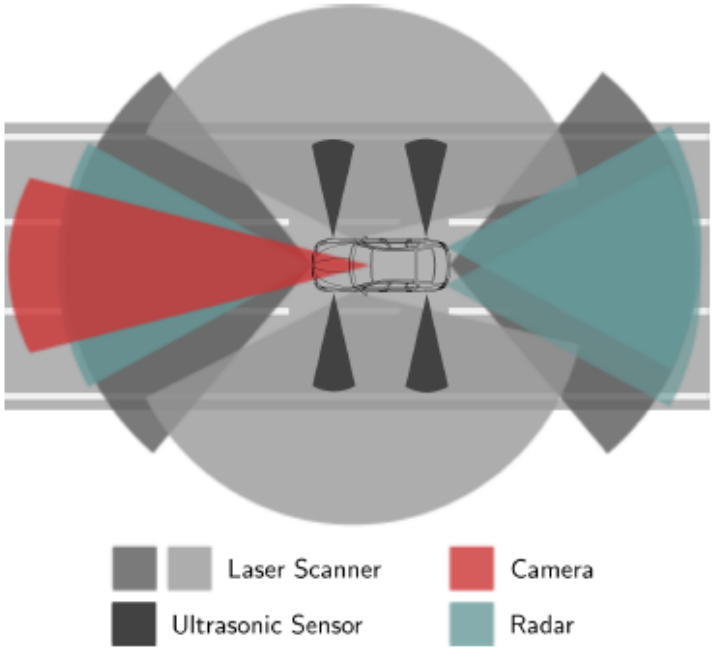


Figure 3.13: Sensor range of integrated sensors for vehicle and lane detection, Aeberhard, Rauch, Bahram, Tanzmeister, Thomas, Pilat, Homm, Huber, and Kaempchen [2015]

Algorithm 3 Prepare Search

```
procedure PREPARESEARCH(Trajectory Set Map, Environment Representation)
    root = getRootNode(Trajectory Set Map, Environment Representation)
    root.assessCombinatorial()
    OPEN ← root
end procedure
```

Environment representation and the trajectory set map (it can be seen as a look-up table) are the necessary inputs to start the search. This first step is accomplished by Prepare Search algorithm [3], which is responsible for computing the root node or nodes (e.g forest search, not used). Function *getRootNode()* finds the best trajectory super set taking initial velocity and lateral acceleration into account. In the end of prepare search the root node is pushed to the OPEN list.

Algorithm 4 Search Step

```
procedure SEARCHSTEP
    if currentExpansion = 0 then
        explore()
    else
        expand()
    end if
end procedure
```

This is how expansion and exploration steps are combined in the search process. Current expansion has the current expanded trajectory super set data, in order to proceed to an expansion. When it is zero it means all the nodes in the trajectory super set were expanded.

Algorithm 5 Exploration Search Step

```
procedure EXPLORE
    parent = OPEN.pop()
    feasible = assesssColision(Parent)
    if !feasible then
        unreserve Parent Memory
    else
        if Goal Node then
            bestResult = Parent
            CLOSED ← parent
            if useAdaptativeStrategy then
                updateEpsilon()
                for all node ∈ OPEN do
                    node.reAssessCombinatorial()
                end for
                OPEN.reOrder()
            end if
        else
            CLOSED ← parent
            currentExpansion = parent.getChildren()
        end if
    end if
end procedure
```

Algorithm 5 follows the classical A^* with some problem instance adaptations and modifications. First of all, after a node with the least $f(s)$ has been popped from the OPEN list a collision assess is made. Where the path swath of the vehicle is tested upon road boundaries and obstacles. If the parent is feasi-

ble and it is not a goal state then exploration step ends by computing its children. Function *getChildren* is responsible for finding the best match in the trajectory set map which fits better the parent current state. A better discretization would mean an increase in the number of computed motion primitives, which would result in a more time demanding search and in fact it can happen that the size of the offline motion primitives can not be upload due to the fact of its dimensions. One have to take notice that the increase of discretization in the trajectory set map is exponential (e.g. doubling the discretization in both axis of the constraint graph 3.1 means 16x more trajectories). Further more if a solution is found then ϵ is updated and afterwards all the nodes in the OPEN list are re-accessed such as in algorithm [7] and then OPEN list nodes are reorder taking the new computed evaluation function $f(s)$ into consideration.

$$\epsilon_{new} = \min_{s \in OPEN} \left(\epsilon_{old}, \frac{\bar{f}}{g(s) + h(s)} \right) \tag{3.17}$$

where:

ϵ : bound factor
$\bar{f}$: true cost of the solution found
$g(s)$: true cost of going from the initial state/node to the final state/node s
$f(s)$: evaluation function

Updading ϵ 3.17 is an adaptation of the method followed in ARA* (repairing anytime Likhachev et al. [2003]) for a continuing anytime search.

Algorithm 6 Edge Expansion Search Step

```
procedure EXPAND
    if currentExpansion.hasNextChild() then
        Child. reserveMemory()
        currentExpansion.createNext(Child)
        child.assessCombinatorial()
        OPEN ← child
    else
        currentExpansion = 0
    end if
end procedure
```

Other important step in a graph search is the expansion. *currentExpansion* is a variable that has the information relative to the trajectory super set of the parent node. As a result, one can compute the children and after every edge expansion a combinatorial access [7] is made.

The overall planner has three access algorithms. This specific one is made for every node. The other two, which will be further explained, will only be used in certain situations. This layer strategy was utilized for reducing the number of computations and as a result save computational time.

Algorithm 7 Assess Combinatorial Search

```
procedure ASSESSCOMBINATORIAL(node)
    node.setC()
    node.setG()
    node.setH()
    node.setF_uninflated()
    node.setF_inflated()
    node.isgoalNode()
end procedure
```

$$
\begin{aligned}
c(s) &= \Delta t_{\text{incoming trajectory}} \\
g(s) &= c(s) + g(\text{parent}) \\
h(s) &= \frac{V_s}{a_{max}} \\
f_{\text{uninflated}}(s) &= g(s) + h(s) \\
f_{\text{inflated}}(s) &= g(s) + \epsilon h(s)
\end{aligned}
\tag{3.18}
$$

where:

- $c(s)$: cost between the node and its parent, cost of the edge
- $g(s)$: true cost of going from the initial state/node to the final state/node s
- $h(s)$: heuristic function that predicts the true cost from state s to the goal
- $f_{\text{uninflated}}(s)$: unbounded evaluation function
- $f_{\text{inflated}}(s)$: bounded evaluation function

Node cost is defined as the total trajectory time, since it is thoroughly accepted that a faster vehicle in the case a collision happens the damages are minimized.

For the anchor heuristic it was chosen an admissible heuristic by setting a_{max} (9.81 [m/**s**^{2}]), which in fact always under estimates the true cost to the goal. It is also computed an inflated and uninflated heuristic for the adaptive strategy.

Since the heuristic does not have any information about the environment, it would be interesting to create an heuristic better informed and test if it improves the search performance. In order to fit the heuristic to the current environment an heuristic was created as follows:

$$
\begin{aligned}
f_{\text{inflated}} &= g(s) + \epsilon h(s) \\
\equiv f_{\text{inflated}}(s) &= g(node) + (\epsilon + 1 - 1)h(s) \\
\equiv f_{\text{inflated}}(s) &= f_{\text{uninflated}}(s) + (\epsilon - 1)h(s) \\
\equiv f_{\text{inflated}}(s) &= f_{\text{uninflated}}(s) + \text{Inflated}_{\text{Part}}
\end{aligned}
\tag{3.19}
$$

where:

$g(s)$: true cost of going from the initial state/node to the final state/node s

$h(s)$: heuristic function that predicts the true cost from state s to the goal

$f_{\text{uninflated}}(s)$: unbounded evaluation function

$f_{\text{inflated}}(s)$: bounded evaluation function

ϵ : bound factor

$\text{Inflated}_{\text{part}}$: the overestimation made by the ϵ bound in the evaluation function

As a result one can change the inflated part between 0 and $(\epsilon - 1)h(node)$ to make the search more informed and at the same time not loose suboptimality. The approach for better inform the heuristic is related with the inclusion of the obstacles information into the inflated part of the evaluation function. The inflated part is computed as follows.

$$\text{Inflated}_{\text{Part}} = \alpha(\epsilon - 1)h(s) : \alpha \in [0, 1] \tag{3.20}$$
$$\alpha_{s_t=i} = \frac{\max_{s_t \in \mathcal{S}_t}(\overline{d_{os_t}}) - \overline{d_{oi}}}{\max_{s_t \in \mathcal{S}_t}(\overline{d_{os_t}})} : o \in \mathcal{O}$$

where:

α : inflated part factor

ϵ : bound factor

$\text{Inflated}_{\text{part}}$: the overestimation made by the ϵ bound in the evaluation function

$\mathcal{S}_t$: the group of all trajectory sets in a trajectory super set

$\mathcal{O}$: the universe of all the objects detected by the ego vehicle

$\overline{d_{os_t}}$: the average distance between the center of the obstacles (o) and the center of the trajectory set (s)

It is important to address that by getting the search better informed, the sub optimality bounds and completeness was not lost in between. Next a scheme of the created evaluation function is presented.

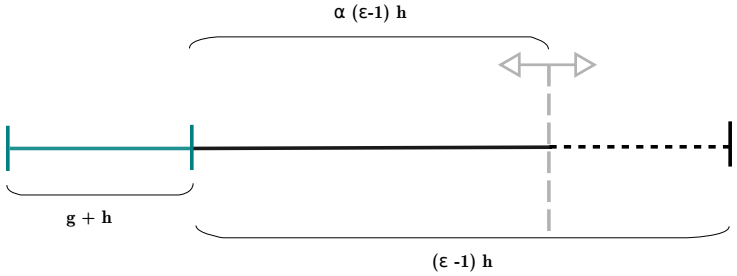


Figure 3.14: Representation of the use of the inflated heuristic part by changing α parameter

One could argue that the number of computations increases geometrically the more obstacles are detected. Further more instead of choosing the center of each set it could be used other sampling strategy with more samples rather then three *per* trajectory super set, which would make the heuristic more informed but on the other hand slower to compute. This could be tested in a steady benchmark.

The succeeding figure demonstrates an example where the usage of this strategy would speed up the search.

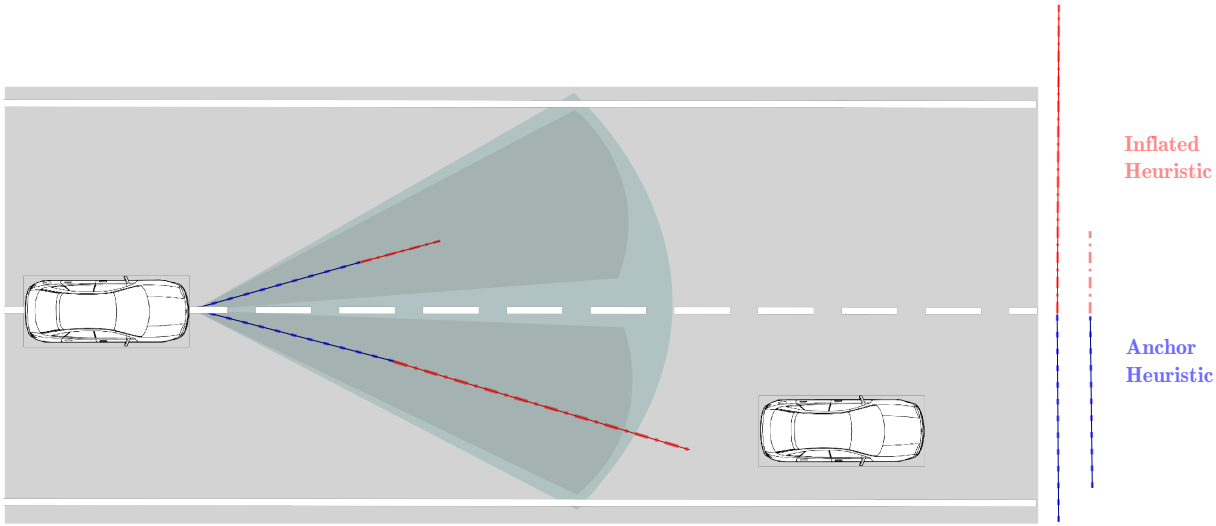


Figure 3.15: Ego vehicle detected the obstacle in front. The semi circles are the representations of the trajectory sets. Trajectory set closer to the obstacle has an higher inflated heuristic than the furthest one

Note: The inflated heuristic is equal for every trajectory inside the same trajectory set. Although, the uninflated heuristic differs by trajectory [3.15].

Algorithm 8 Assess Collisions

```
procedure ASSESSCOLLISION
    trajectory = getIncomingTrajectory(node)
    parent = getParent(node)
    OBB_EGO_Vehicle = getOBBTree(trajectory)
    [Position_parent,Heading_parent] = getPOS&Heading(parent)
    OBB.transform(Position_parent,Heading_parent)
    for all point i ∈ Preselected Road Boundaries Points do
        Free = isCollisionRoadFree(i,OBB_EGO_Vehicle)
        if !Free then
            node.setInvalid()
        end if
    end for
    for all Obstacles OBBTrees i do
        Free = OBB_EGO_Vehicle.iscollisionObstacleFree(i)
        if !Free then
            node.setInvalid()
        end if
    end for
end procedure
```

Testing for collisions is one of the most time demanding algorithms of the planner. Being so it is intended to use it only when necessary (exploration step), in order to counter this disadvantage several methods were implemented ,which allow an iterative process of testing. This means that simple methods are applied first and if there it is collision free than the more precise methods do not have to be utilized (note: simpler methods generate more false positives). So in order to reduce false positives other methods with increasing order of complexity are applied. In this algorithm the upcoming trajectory and the parent is taken from the node in test. Afterwards it is necessary to change the trajectory origin for the parent state due to the fact that the trajectory is taken from the trajectory super set where all trajectories start from a reference point ($x = 0, y = 0, t = 0$), such is accomplished by a translation to the parent position and heading rotation. It is necessary to take into consideration that OBB tree also goes through a similar process but for its corner point and three edges. After the OBB tree of the ego vehicle is created, collision tests start. The OBB tree is an approximation of the trajectory swath. So, first it is tested against road boundaries and next against other obstacle OBB trees.

Each time a node is explored and it is found to be feasible (look algorithm 5) then *getChildren()* function is applied. In this function a pre-collision test is made for road boundaries. These are represented as a matrix with number of road boundary and coordinates spaced by one meter. Since a trajectory super set has a pie shape ,that means it is impossible that any of the trajectories will collide with points closer to the ego vehicle. So only the points between the intersection of the pie shaped trajectory sets with the road boundaries are tested. A figure representing what was explained can be seen below.

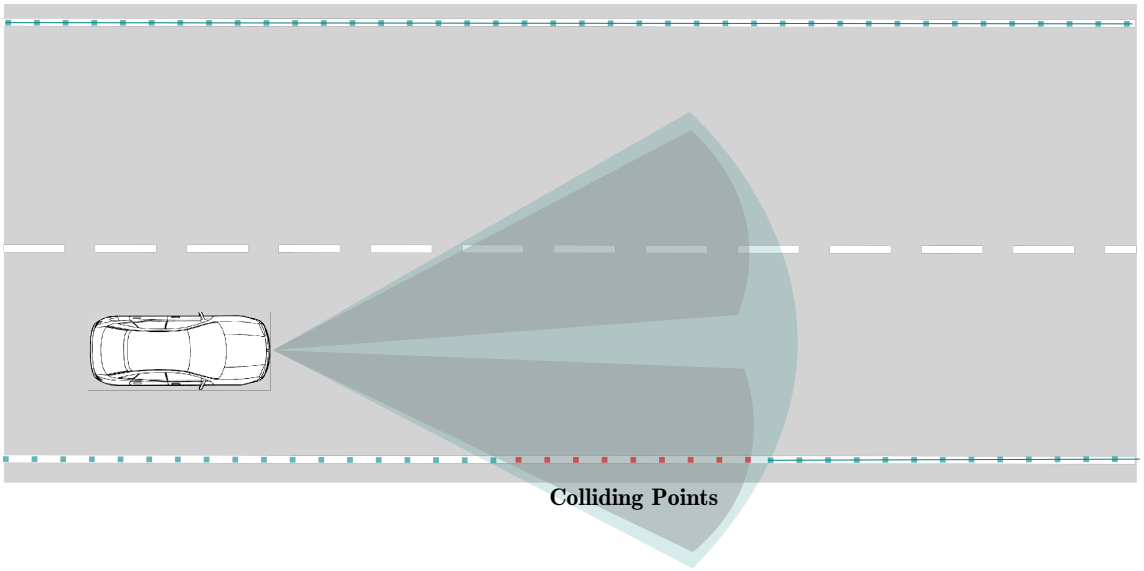


Figure 3.16: Each side of the road has a road boundary which is represented by coordinates separated by a meter. Red dots represent the points of the road boundary in which occured an overlap with the trajectory set pie shape

For every trajectory set there is a variable that has the *id* of the road boundary and the initial and final point (red dots on figure 3.16) in which it is possible that a trajectory from that set is colliding with.

Algorithm 9 OBB Tree Collision Tests

```
procedure isCollisionFree(OBBTree)
    if exists EGO vehicle obb and obstacle obb then
        test = Ego obb.isCollisionFreeComputation(Obstacle obb)
    else
        test = false
    end if
    if test is True then
        return True
    else
        if left or right EGO obb do not exist and left and right Obstacle obb exists then
            a = Obstacle left obb.isCollisionFree(Ego obb)
            b = Obstacle right obb.isCollisionFree(Ego obb)
        else
            if Both left and right obb of the Obstacle and Ego vehicle exist then
                a = Ego left.isCollisionFree(Obstacle left)
                b = Ego left.isCollisionFree(Obstacle right)
                c = Ego right.isCollisionFree(Obstacle left)
                d = Ego right.isCollisionFree(Obstacle right)
            else
                return False
            end if
        end if
    end if
end procedure
```

OBB tree is a group of obb's where each father has two children assigned as left and right. The tree ends when right and left are equal to zero (NULL). The algorithm 9 is responsible for testing collisions between obstacle and ego vehicle obb's recursively. First it starts by testing the bigger obb's in *Ego obb.isCollisionFreeComputation(Obstacle obb)* then if there is a collision it tests more precise inner obb's in the OBB tree. Also it takes into consideration if for example the Ego vehicle tree has a lower depth than Obstacle OBB tree (third if clause). Further more when testing two obb's for collision it is also done in a iterative way, which will be depicted in the next algorithm.

Algorithm 10 Iterative method for checking collision between two obb's

```
procedure ISCOLLISIONFREECOMPUTATION(obb)
    for i = x,y,t do        % checking axis
        if Ego obb.pointMax(i) ≤ Obstacle obb.pointMin(i) || Obstacle obb.pointMax(i) ≤ Ego
obb.pointMin(i) then
            return True
        end if
    end for
    for ( doi=x,y,t)        % checking normals
        if isFreeProjection(Obstacle obb.normals, Ego obb.normals) then
            return True
        end if
    end for
    for ( doi=x,y,t)        % checking edges
        if isFreeProjection(Obstacle obb.edges, Ego obb.edges) then
            return True
        end if
    end for
end procedure
```

Both pointMax and pointMin are assigned by finding the minimum values for x, y, and t for the eight corners of the OBB. Three tests are made in a sequence of increasing time complexity. The second and the third collision test follows the work of Gottschalk et al. [1996] which is represented in the figure bellow. Where a projection of vectors onto an axis L (r_A and r_B) are summed and tested if they are smaller or bigger than the inner product between T and L.

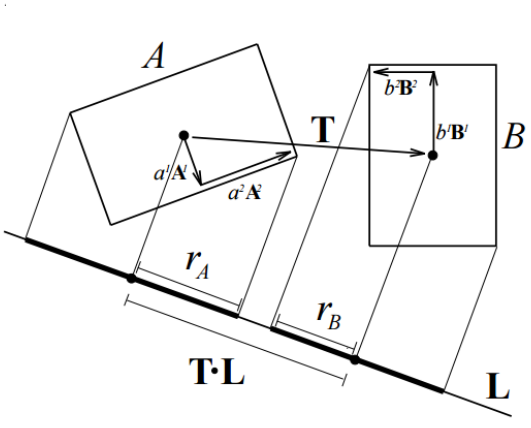


Figure 3.17: L is a separating axis for the OBB's A and B since A and B become disjoint intervals under projection onto L. If the projection overlaps than a collision is detected, Gottschalk et al. [1996]

Next it is presented a flow chart representing and overview of the online search algorithm implemented and described above.

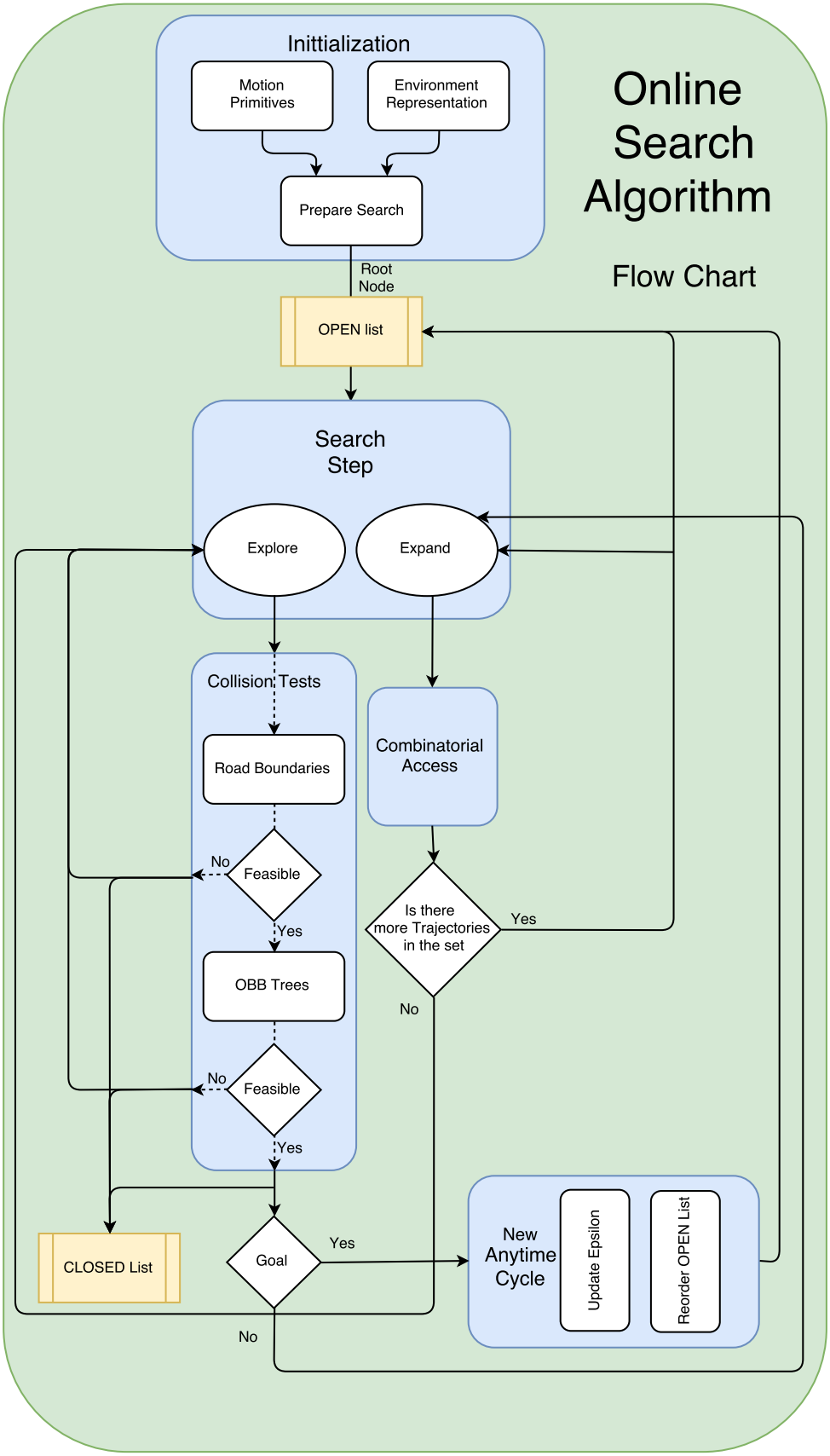


Figure 3.18: Online Search Algoritm flow diagram

Chapter 4

Results

The integration of the motion primitives into the anytime online search depicted in the previous chapters allowed the construction of two strategies that will be compared in this section. One is an anytime sub optimally bounded search using a triple set division of the trajectory super set and an environment informed heuristic (AMEIA$^*{}_3$ - Anytime Multi set Environment Informed A^* with 3 sets) and the second one an anytime sub optimally bounded search using one set and an environment uninformed inflated heuristic (Standard Search).

Even though both use the same anchor heuristic one has to consider that the strategy (AMEIA$^*{}_3$) has an inadmissible heuristic sensitive to environment features such as moving obstacles. This informed heuristic has a higher computation time, nevertheless it also means that less collision tests are made. As a result, it is intended to create tests and evaluate the results such that it is possible to answer if one effect compensates the other and what are the advantages and disadvantages of each method.

The criteria used to compare both methods are the following: number of valid and invalid graph explorations, number of node expansions, number of solutions, time spent till the first solution, maximum number of nodes in memory, time spent till the optimum solution is found.

- Number of valid graph explorations, **VE**
- Number of invalid graph explorations, **IE**
- Number of graph expansions, **Ep**
- Time spent till first solution, **TS1**
- Maximum number of nodes in Memory, **M**
- Time spent till optimum solution, **T**

Several scenarios were developed in *.xml* files and they will be presented in order of complexity. Since this is a contingency planner, the scenarios should reflect potentially dangerous driving situations with an imminent risk of collision. To assure such scenario characteristics, two measurements (TIV-Time Inter Vehicle and TTC- Time To Collision) were combined such that a risk of collision is high and

can be predicted.

Projects such as ARCOS(www.arcos2014.fr) and PREVENT(www.prevent-ip.org) have analised TTC and defined a set of boundaries for its value. Where at TTC greater than ten seconds, an obstacle vehicle is not supposed to have an interaction with the ego vehicle and if TTC becomes lower than one second then the control of the vehicle could be switched to an automated planner. According to french law TIV must be lower than two seconds and the lower boundary is assigned by the same reason as TTC, Glaser et al. [2010] and Bahram et al. [2014].

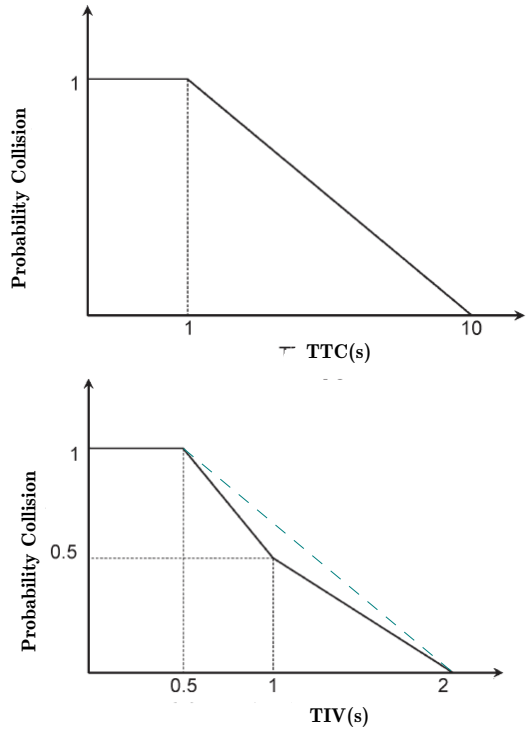


Figure 4.1: Representation of the probability of collision for both parameters TTC and TIV in order of time. Dotted line in blue represent an author approximation. Glaser, Vanholme, Mammar, Gruyer, and Nouveliere [2010]

$$t_{TTC} = \frac{D_i}{V - V_i}$$
$$t_{TIV} = \frac{D_i}{V - V_i} \tag{4.1}$$

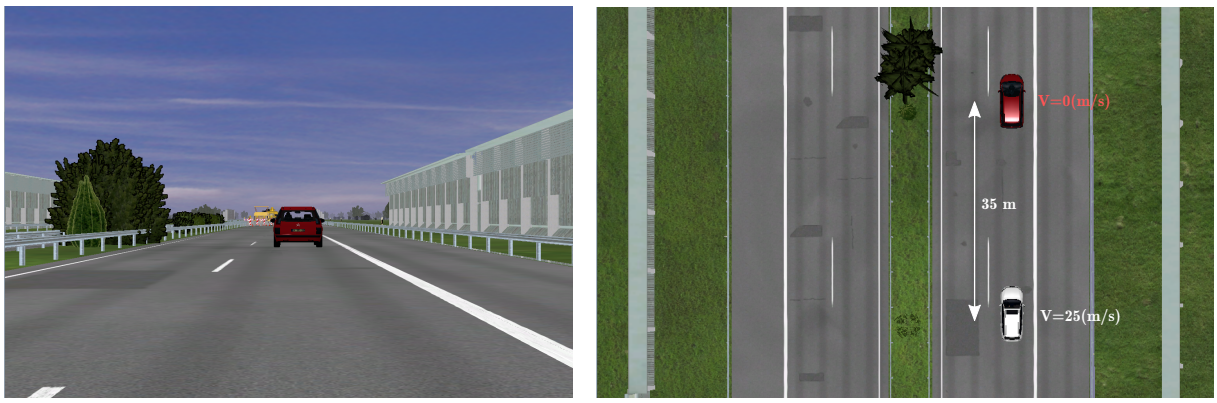
$$\text{Risk}_{\ TTC/TIV} = \begin{cases} 1 & ,if \ \ t_{\ TTC/TIV} <= t_{\ TTC/TIV_{\text{min}}} \\ 0 & ,if \ \ t_{\ TTC/TIV} >= t_{\ TTC/TIV_{\text{max}}} \\ 1 - \frac{t_{\ TTC/TIV} - t_{\ TTC/TIV_{\text{min}}}}{t_{\ TTC/TIV_{\text{max}}} - t_{\ TTC/TIV_{\text{min}}}} & otherwise \end{cases} \tag{4.2}$$

$$\text{Total Risk} = 1 - (1 - Risk_{\ TTC})(1 - Risk_{\ TIV})$$

where:

D : distance between vehicles
TTC : time to collision
TIV : time inter vehicle
$risk$: measures the possibility of a collision
V : Velocity

4.1 Scenario 1 Analysis



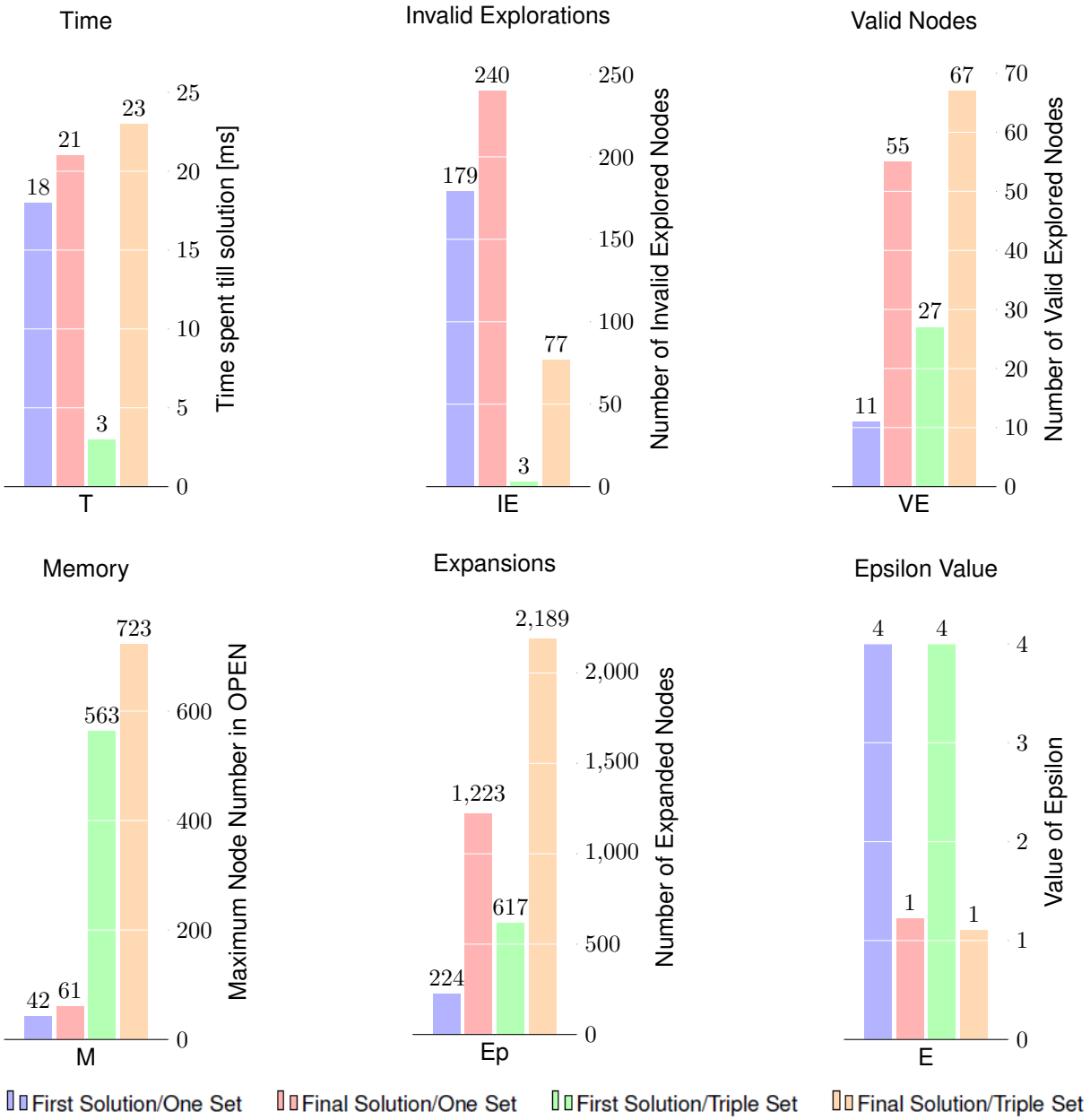
(a) Front View (b) Top View

Figure 4.2: Scenario 1 comtemplating a stationary obtacle vehicle in red and the ego vehicle with a velocity of 25 (m/s)

Following the method in 5.1, which allows a computation of the first scenario risk of collision, one reaches the conclusion that the total risk of collision for this scenario is 97%. Being so, one could agree that this scenario has an high risk of collision and as a result could be considered a scenario where an emergency maneuver is necessary. Further developed scenarios will increment on the first scenario,

assuring that all of them have an higher collision risk than 97% . Since the measurement of the risk of a specific scenario is not the aim of this work, one will use an empirical criterion. The criteria relies on the idea that by maintaining the first scenario characteristics and adding other objects, the resultant scenario has a higher risk of collision. Nevertheless, one has to bear in mind that it does not necessarily means that it is more difficult to find a solution in such scenarios.

Next it will be presented a bar chart with the results of each search method based upon criteria depicted above.



4.1.1 Time Analysis

The results show a major improvement when comparing the first solution of both methods. In fact with the standard search strategy the time to get the first solution is 600% bigger than AMEIA$^*{}_3$. Comparing the time to find a final solution both methods are similar with the standard search being slightly better.

N.B. the aim of the planner is to find a solution as soon as possible.

The approximation of the final solution is similar, which is expected, since both methods converge to an A^* by reducing ϵ value to almost unity.

4.1.2 Valid and Invalid Explorations Analysis

In this criteria the superiority of the AMEIA$^*{}_3$ method is straight forward. Where the number of invalid explorations is bigger in both first and final solution found. Concerning valid edge explorations, AMEIA$^*{}_3$ method explores two to three times more than the other method.

In fact since the AMEIA$^*{}_3$ is environment sensitive, one expects that less invalid explorations are made. Nevertheless, it is exploring valid edges that are not necessarily going to the solution. One could argue that since there were more invalid explorations in the standard search, only on a later phase of the standard search it starts getting valid nodes, because most of the nodes were searched previously. Other possible reason for an higher value of valid explored nodes has to do with the fact that inside each set there is no *better* informed heuristic to distinguish between nodes, since all the nodes follow the anchor heuristic.

The overall result is expected in this scenario since the heuristic here used prioritize *going in front* and deviates from that objective when continuing invalid explorations are being taken.

4.1.3 Expansions and Memory Analysis

The number of expanded nodes and nodes in memory in AMEIA$^*{}_3$ strategy is more than ten times bigger than the standard search in both final and first solution.

Looking to the time interval till a solution is found, it is possible to conclude that expansions don't have a major role on that behalf. Nevertheless, one could point out that with an increased scenario complexity that could represent an issue. Note: the assigned nodes limit of the program is 100.000, where the memory is statically allocated.

The elevated number of expansions can also be explained due to the fact that with a AMEIA$^*{}_3$ strategy more jumps between sets are expected, and even though a superset is divided in three sets all of the nodes inside a set are expanded.

4.1.4 ϵ value Analysis

An important characteristic of both methods is the following: AMEIA$^*{}_3$ strategy made two anytime cycles which also means two solutions were found, the other method realized three anytime cycles finding three solutions. The final value of ϵ in AMEIA$^*{}_3$ method is closer to 1 than in the standard search and just two cycles were needed which means that AMEIA$^*{}_3$ has a better convergence to A^*. Nevertheless, even with three cycles the standard search was faster finding the optimal solution. The reason has to due with the fact that the second solution of the standard search is similar to the first and few search steps were necessary.

Since the final solution was found in a time lower than 0.1[s] this means the search stopped in both cases because it found a resolution optimum solution.

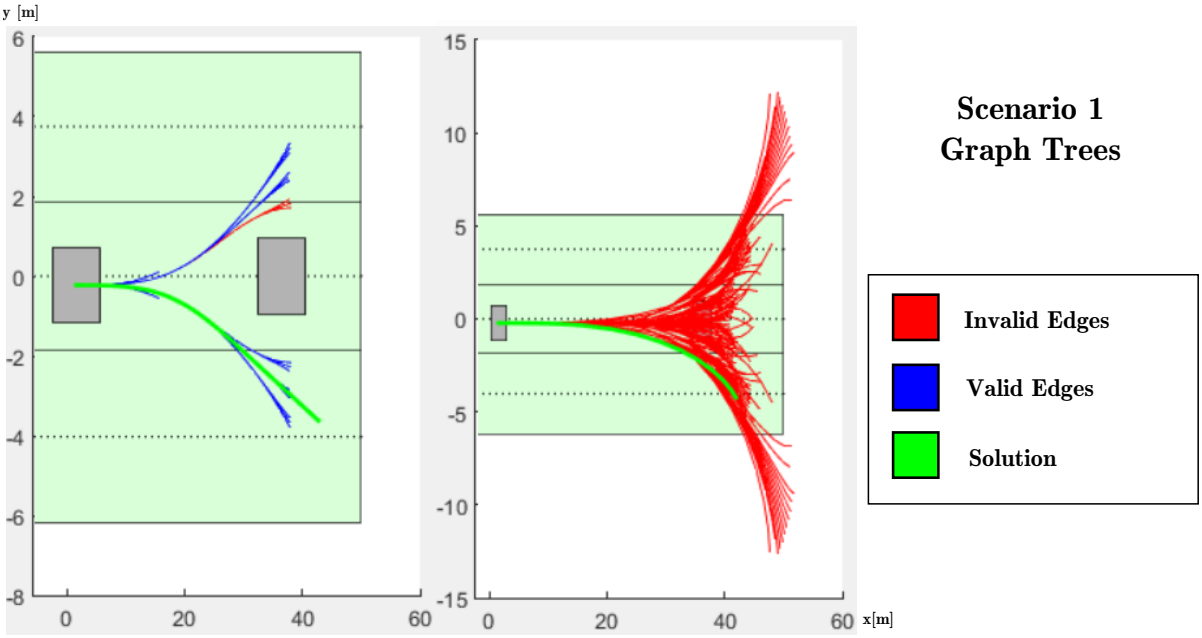
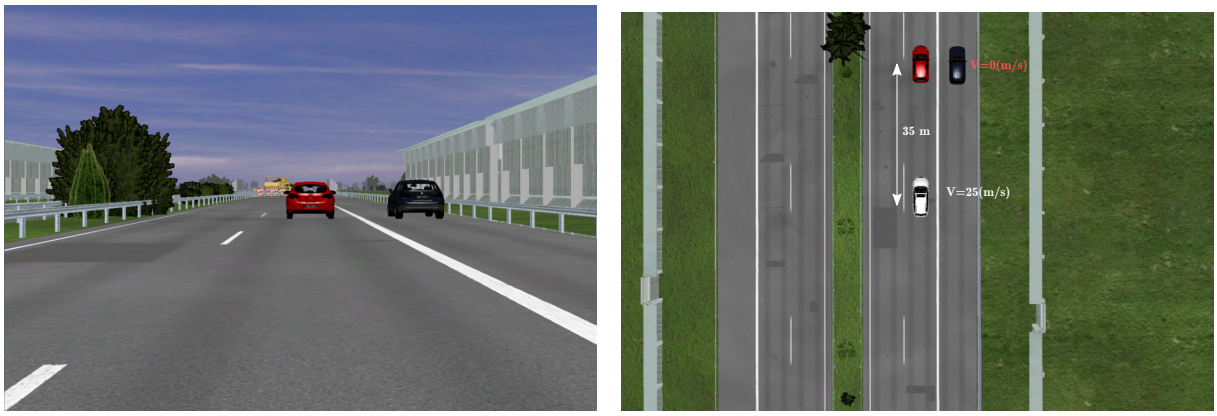


Figure 4.3: Graph trees generated using both strategies in scenario 1. Figure on the left has represented the tree of AMEIA$^*{}_3$ method and the right one the standard search method.

4.2 Scenario 2 Analysis

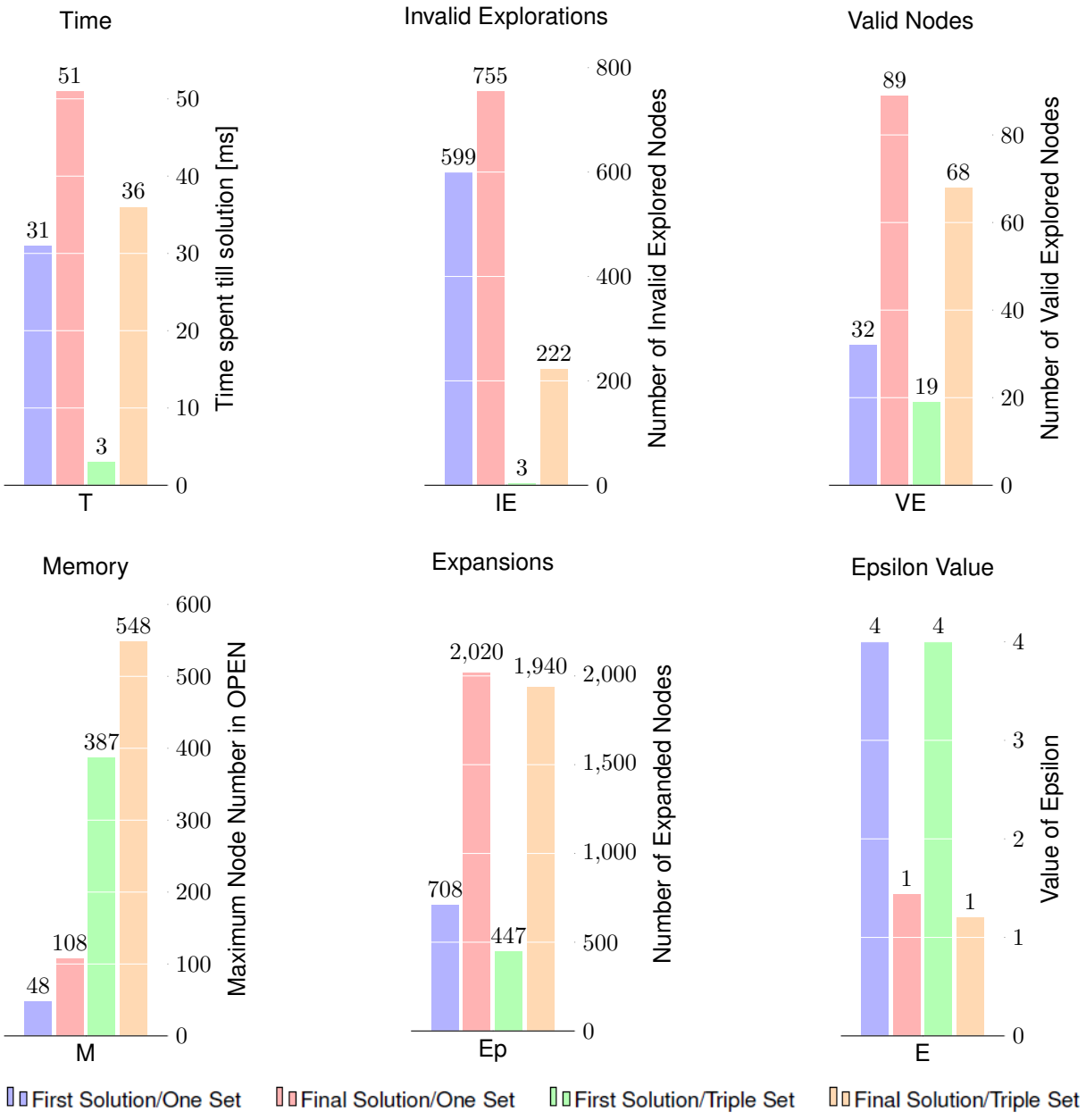


(a) Front View (b) Top View

Figure 4.4: Scenario 2 comtemplating two stationary obtacle vehicles, one in red and other in black, and the ego vehicle with a velocity of 25 (m/s)

In this scenario an extra object is added and a *cul-de-sac* is formed increasing the complexity of the maneuver. *Cul-de-sac* situations are extremely difficult to solve for a standard combinatorial search method because of the creation of a local minimum, Likhachev and Stentz [2008]. Next it will be made an

evaluation of the performance of both methods in this scenario taking into account the criteria depicted in the section 4.



4.2.1 Time Analysis

The trend of finding a solution faster in AMEIA$^*{}_3$ method maintains. It is interesting to verify that the time to find a solution in the standard search increased in both final and initial solutions, on the other hand for AMEIA$^*{}_3$ actually it decreased in the final solution and maintained for the first.

The increase in the standard search is expected because the scenario is also more complex and the heuristic is uninformed about the environment. On the other hand, for AMEIA$^*{}_3$ this scenario is actually less complex, because a great majority of the search space (center lane and right lane graph sub-trees) have an higher cost due to the strategy implemented, where the heuristic is informed about the outer environment taking the obstacles into consideration. On the first scenario for AMEIA$^*{}_3$ both right and

left lane were prioritized and for this scenario the existence of more obstacles constrained the search to the left lane, making the search faster.

4.2.2 Valid and Invalid Explorations Analysis

In this point, one can verify that in all situations the invalid explorations increased, but in the case of the AMEIA$^*{}_3$ the number of explored invalid nodes for the first solution was maintained (three). Concerning valid explorations it occurred a similar trend as the one seen for the time criteria, in which for the standard search in both final and first solution it increased the VE and in the case of AMEIA$^*{}_3$ it maintained or reduced the number of valid explorations.

It is interesting to verify that the VE for the AMEIA$^*{}_3$ reduced its relative difference to the standard search. Nevertheless, it is expected that for the final/optimum solution both methods behave similarly relatively to VE, because they are both converging to the same solution, since ϵ is converging to one, which also means both methods are converging to A^* if there is enough time.

4.2.3 Expansions and Memory Analysis

A similar trend, as the one seen for the first scenario in terms of memory, is verified in the second scenario. The memory of AMEIA$^*{}_3$ strategy continues to be bigger than in the standard search (five to ten times). Although, looking to the expansions of the anchor search they almost doubled and in AMEIA$^*{}_3$ search it slightly reduced.

Nevertheless, it is expected that the number of nodes in memory would be greater for the AMEIA$^*{}_3$, and that happened, it is also intriguing to verify that in the second scenario the memory reduced when compared with the first scenario. A possible explanation for such occurrence might be due to the fact that it occurred an effective prunning in the beginning of the search by highly prioritizing one trajectory set (left lane).

4.2.4 ϵ value Analysis

The standard search realized five anytime cycles and AMEIA$^*{}_3$ only two, exactly the same as in the first scenario. A same trend as depicted in the previous scenario ϵ analysis is verified in this scenario. Where the final value of ϵ in AMEIA$^*{}_3$ method is closer to 1 than in the standard search and just two cycles were needed which means that AMEIA$^*{}_3$ has a better ϵ convergence to A^*. Contrary to the first scenario it occurred a better convergence of AMEIA$^*{}_3$ both for the ϵ value and for the resolution optimum solution, finding it in 36 [ms].

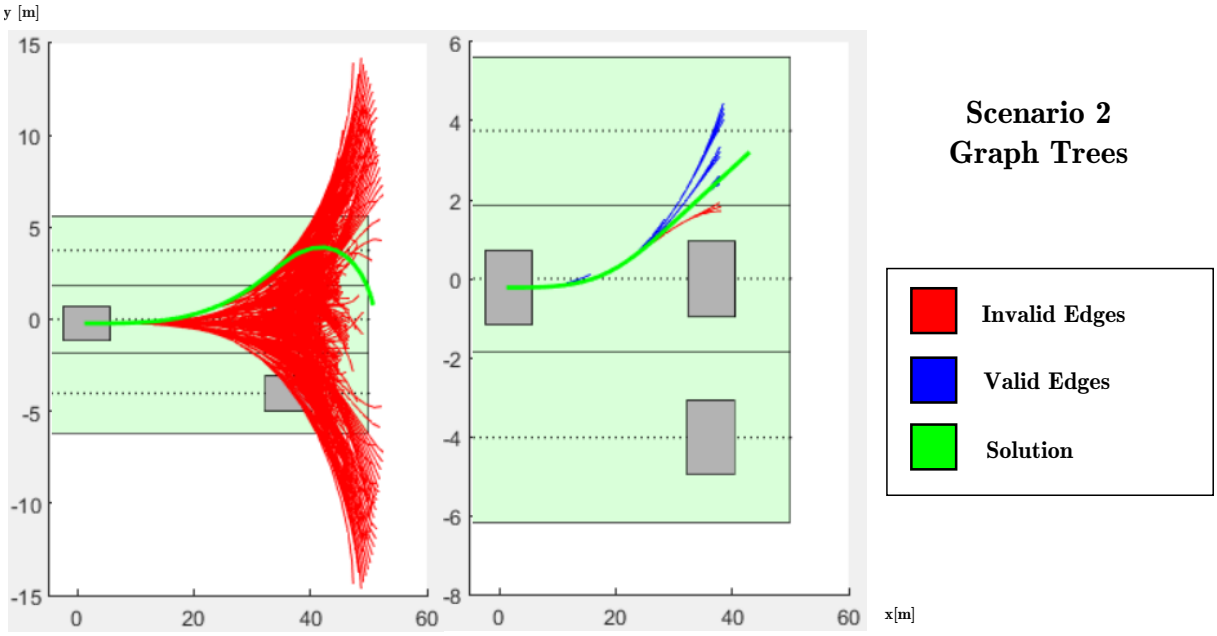


Figure 4.5: Graph trees generated using both strategies in scenario 2. Picture on the right has represented the graph tree of AMEIA$^*{}_3$ method and left one standard search method

Chapter 5

Conclusions

The proposed contingency planner with an anytime sub optimally bounded search using a triple set organization and an environment informed heuristic with the help of the created motion primitives can generate reasonable and fast solutions for the tested scenarios, improving upon existing anytime methods. It also fulfills the project objective of finding a solution in a time frame of 0.1 [s], further more it finds a resolution optimal solution for the tested scenarios. The work developed presents a bottom to top solution for an emergency maneuver planner focusing on two distinct phases.

In the offline phase, where the motion primitives were created, it was constructed a constraint graph in order to lateral acceleration and longitudinal speed, also a vehicle model taking the Dubins Car model into consideration and finally several collision tests were developed such as: ellipsoid, arc section, pie shape and OBBtrees. Even though not all the approaches were tested it was built a solid framework in order to pursue on that path and reach a conclusion about which method better suits what kind of situations.

In the online phase, it was utilized an anytime search method with an environment informed heuristic taking obstacles into account, which allowed a decrease in the number of invalid explorations and also time interval till the first solution, which is the main objective of the planner. Since collision tests are time demanding an adaptive collision test strategy was implemented, in which an iterative method of executing first simple tests and in each iteration increase its complexity. This allows a reduction in the runtime of the search and at the same time safety is not compromised.

5.1 Achievements

Following next it will be presented a table comparing the overall results for the first solution depicted in both scenarios of the section 4 in order of the invalid edges exploration and runtime.

Table 5.1: Comparing AMEIA$^*{}_3$ with the standard search in terms of time spent till the first solution found and invalid edges explorations

	Time	Invalid Edges
Scenario 1	+ 6x	+ 60x
Scenario 2	+ 10x	+ 20x

The major achievements of the contingency planner developed are listed below:

- The creation of an anytime search informed heuristic, with outer environment information, maintaining characteristics such as a sub-optimally bounded solution (bounded by the ϵ factor value of the last iteration), as well as completeness.
- The construction of a more computationally complex heuristic makes the search better informed which allows less invalid explored nodes making it possible to have a reduced runtime for the first solution.
- The construction of motion primitives that are suited for the emergency situations in the tested scenarios and that are not a burden in terms of memory to the online search. Several strategies where implemented in order to reduce the number of created trajectories explosion, when the grid resolution is increased. Such as a graph prune strategy, time ellipses bounds and physical bounds.
- An iterative method of making collision tests. Where simpler tests are made in the beginning and if a collision is detected (since it can be a false positive) the complexity of the tests increase in the next iteration. Tests such as first applying an overlapping test between a pie shape of the trajectory super set and the road boundaries, next an OBBtree test in which is in itself an iterative collision test, where in each depth of the tree the number OBB's enclosing the vehicle also increase. Resulting in an fast but also safe method to apply during the search phase.

5.2 Future Work

In this section several considerations of the work conducted will be made in order to prospect future possible modifications and improvements.

The heuristic used could be changed since the $\text{Inflated}_{\text{Part}}$ constructed allows a great freedom of choice in this behalf. Instead of using the average distance between the objects and the trajectory set test point, it could be changed to the minimum distance. These would give an higher degree of importance to closer objects. For example:

$$\text{Inflated}_{\text{Part}} = \alpha(\epsilon - 1)h(node) : \alpha \in [0, 1] \tag{5.1}$$

$$\alpha_{s_t=i} = \frac{\max_{s \in \mathcal{S}}(\min_o \in \mathcal{O}(d_{os_l})) - \min_{o \in \mathcal{O}}(d_{oi})}{\max_{s_l \in \mathcal{S}}(\min_{o \in \mathcal{O}}(d_{os_l}))}$$

Other possibility could be following the path of refining the existent heuristic by varying the number of points that are tested in each set. The work here developed uses d_{oi} as the distance between an obstacle and the center of the set. And based on that all the trajectories inside that set would have the same inflated part, as a result when comparing trajectories from the same set they will have a different evaluation function because of the anchor heuristic. So one could interpolate between each testing point of each set in order to differentiate the inflated part of the heuristic between trajectories inside the same set.

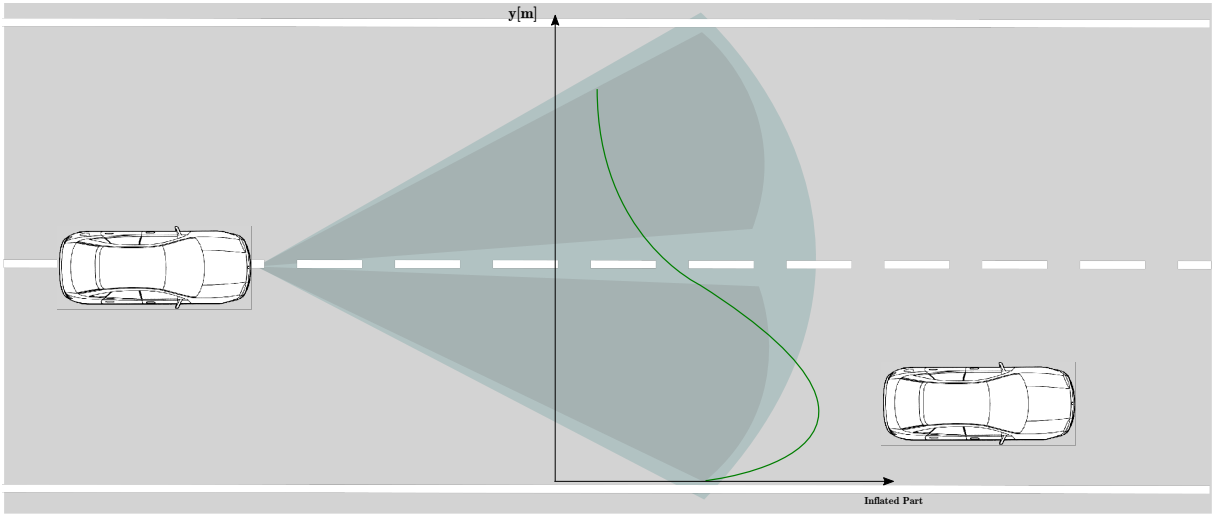


Figure 5.1: Representation of a possible inflated part of an evaluation function when an interpolation between testing points of each set is done

It would be interesting to test if an increase of testing points would accelerate the search, and at which point does the computation of the inflated part does not compensate a reduction in the number of invalid explorations and runtime, if any. Because more testing points, to measure the distance to the obstacles, also means more computations are executed and, further more, if the scenario has more objects the number of computations would increase even more.

Since the objective of the search is to find a solution as soon as possible then one could argue that increasing the diversity of the motion primitives would help in such manner. Considering that the anchor heuristic would search faster trajectories this also means that it would prioritize trajectories with less lateral acceleration (”going in front”). But one could create an heuristic that could prioritize lateral acceleration which would double the initial diversity of the search.

Analyzing the results it is straightforward that even though the number of expansions do not have a major influence on terms of time it has an influence in terms of memory and with the increase of the scenario complexity this could be a problem. In order to surpass this fact it could be applied a model prediction method before any expansion and exploit that information in order to take a decision whether to expand or not.

Motion primitives were created in a way that they could be used in other planners. Some of the methods created were not tested. For example, which one is a faster test, an ellipsoid, OBBtree or an

arc section. Concerning the vehicle model, it would be important to actually use the full power of the vehicle by following the traction circle.

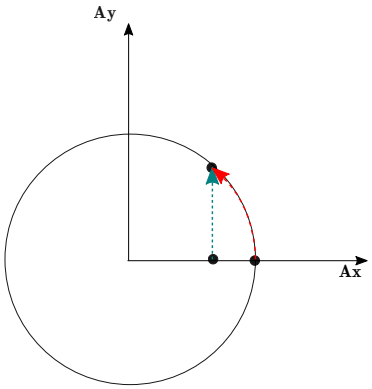


Figure 5.2: Vehicle model implemented follows the line in green and what is intended with a an improved vehicle model is to follow the traction circle in red

Bibliography

M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen. Experience, results and lessons learned from automated driving on germany's highways. *Intelligent Transportation Systems Magazine, IEEE*, 7(1):42–57, 2015.

S. Aine, S. Swaminathan, V. Narayanan, V. Hwang, and M. Likhachev. Multi-heuristic a*. In *Seventh Annual Symposium on Combinatorial Search*, 2014.

M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr. A prediction-based reactive driving strategy for highly automated driving function on freeways. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 400–406. IEEE, 2014.

J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10(2-4):121–155, 1993.

L. Biagiotti and C. Melchiorri. *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

M. S. Branicky, R. Knepper, J. J. Kuffner, et al. Path and trajectory diversity: Theory and algorithms. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1359–1364. IEEE, 2008.

N. Brown, A. Ebtekar, and Y. Nakamura. Distributed multi-heuristic a*(hamstar). 2014.

M. Buehler, K. Iagnemma, and S. Singh. *The DARPA Urban Challenge: Autonomous vehicles in city traffic*, volume 56. springer, 2009.

K. Chu, M. Lee, and M. Sunwoo. Local path planning for off-road autonomous driving with avoidance of static obstacles. *Intelligent Transportation Systems, IEEE Transactions on*, 13(4):1599–1616, 2012.

L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, pages 497–516, 1957.

L. H. Erickson and S. M. LaValle. Survivability: Measuring and ensuring path diversity. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2068–2073. IEEE, 2009.

D. Ferguson, M. Likhachev, and A. Stentz. A guide to heuristic-based path planning. In *Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS)*, pages 9–18, 2005.

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

E. Frazzoli. *Robust hybrid control for autonomous vehicle motion planning*. PhD thesis, Massachusetts Institute of Technology, 2001.

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

S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM, 1996.

C. Green and A. Kelly. Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*, 2007.

T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee. Focused trajectory planning for autonomous on-road driving. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 547–552. IEEE, 2013.

X. Han. Cubic trigonometric polynomial curves with a shape parameter. *Computer Aided Geometric Design*, 21(6):535–548, 2004.

E. A. Hansen and R. Zhou. Anytime heuristic search. *J. Artif. Intell. Res.(JAIR)*, 28:267–297, 2007.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

T. Hesse, D. Hess, and T. Sattel. Motion planning for passenger vehicles-force field trajectory optimization for automated driving. In *The 15th IASTED International Conference on Robotics and Applications*, 2010.

R. C. Holte. Common misconceptions concerning heuristic search. In *SOCS*, 2010.

H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004.

T. M. Howard, C. J. Green, A. Kelly, and D. Ferguson. State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. *Journal of Field Robotics*, 25 (6-7):325–345, 2008.

T. M. Howard et al. *Adaptive model-predictive motion planning for navigation in complex environments*. PhD thesis, 2009.

D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4): 566–580, 1996.

A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, et al. Toward reliable off road autonomous vehicles operating in challenging environments. *The International Journal of Robotics Research*, 25(5-6):449–483, 2006.

R. Knepper, M. T. Mason, et al. Path diversity is only part of the problem. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3224–3229. IEEE, 2009.

R. A. Knepper. *On the fundamental relationships among path planning alternatives*. PhD thesis, Citeseer, 2011a.

R. A. Knepper. *On the fundamental relationships among path planning alternatives*. PhD thesis, Citeseer, 2011b.

S. Koenig and M. Likhachev. Real-time adaptive a*. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 281–288. ACM, 2006.

S. Koenig, M. Likhachev, Y. Liu, and D. Furcy. Incremental heuristic search in ai. *AI Magazine*, 25(2):99, 2004.

Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. P. How, and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *Control Systems Technology, IEEE Transactions on*, 17(5): 1105–1118, 2009.

S. M. LaValle. Rapidly-exploring random trees a ?ew tool for path planning. 1998.

S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

M. Likhachev and S. Koenig. A generalized framework for lifelong planning a* search. In *ICAPS*, pages 99–108, 2005.

M. Likhachev and A. Stentz. R* search. *Lab Papers (GRASP)*, page 23, 2008.

M. Likhachev, G. J. Gordon, and S. Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, page None, 2003.

L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng. Efficient sampling-based motion planning for on-road autonomous driving.

B. Nagy and A. Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. *Field and Service Robots*, 11, 2001.

V. Narayanan, S. Aine, and M. Likhachev. Improved multi-heuristic a* for searching with uncalibrated heuristics. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2172–2179. IEEE, 2011.

M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

M. N. Pivtoraiko. Differentially constrained motion planning with state lattice motion primitives. Technical report, DTIC Document, 2012.

D. J. Ram, T. Sreenivas, and K. G. Subramaniam. Parallel simulated annealing algorithms. *Journal of parallel and distributed computing*, 37(2):207–212, 1996.

J. Reeds and L. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

R. T. Stern, R. Puzis, and A. Felner. Potential search: a new greedy anytime heuristic search. In *Third Annual Symposium on Combinatorial Search*, 2010.

J. T. Thayer and W. Ruml. Anytime heuristic search: Frameworks and algorithms. In *Third Annual Symposium on Combinatorial Search*, 2010a.

J. T. Thayer and W. Ruml. Finding acceptable solutions faster using inadmissible information. In *Third Annual Symposium on Combinatorial Search*, 2010b.

J. T. Thayer, W. Ruml, and E. Bitton. Fast and loose in bounded suboptimal heuristic search. In *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*, 2008.

H. Wang, J. Kearney, and K. Atkinson. Arc-length parameterized spline curves for real-time simulation. In *5th international conference on Curves and Surfaces*, 2002.

W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2061–2067. IEEE, 2012.

K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim, and S. Joo. Spline-based rrt path planner for non-holonomic robots. *Journal of Intelligent & Robotic Systems*, 73(1-4):763–782, 2014.

M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

Appendix A

Trajectory Path Set - Motion Primitives

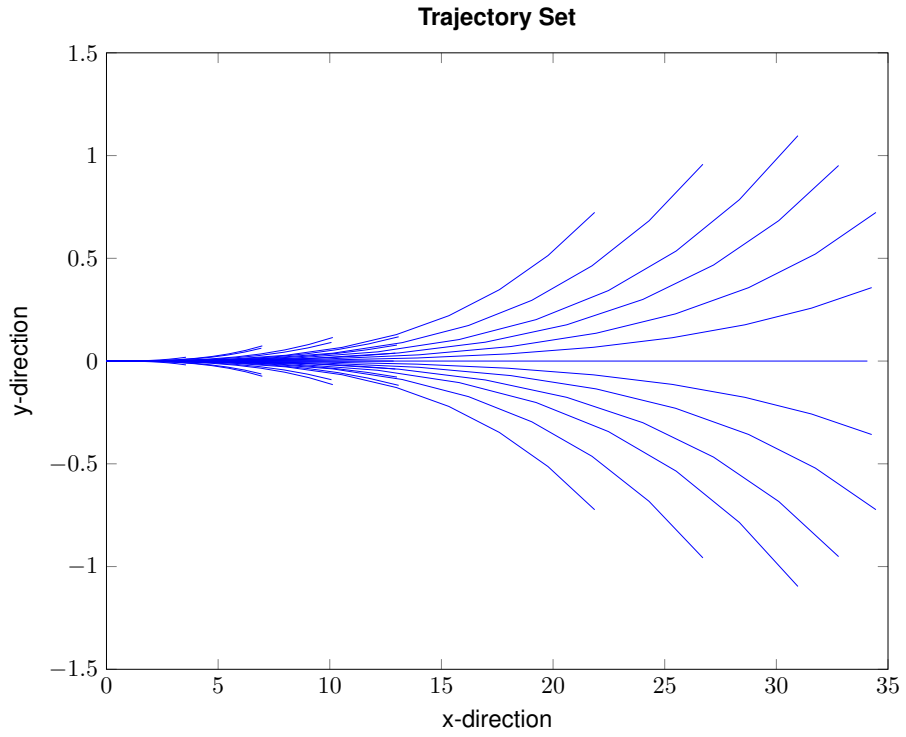


Figure A.1: Trajectory path set with trajectories in blue the initial state has maximum velocitiy and zero lateral acceleration

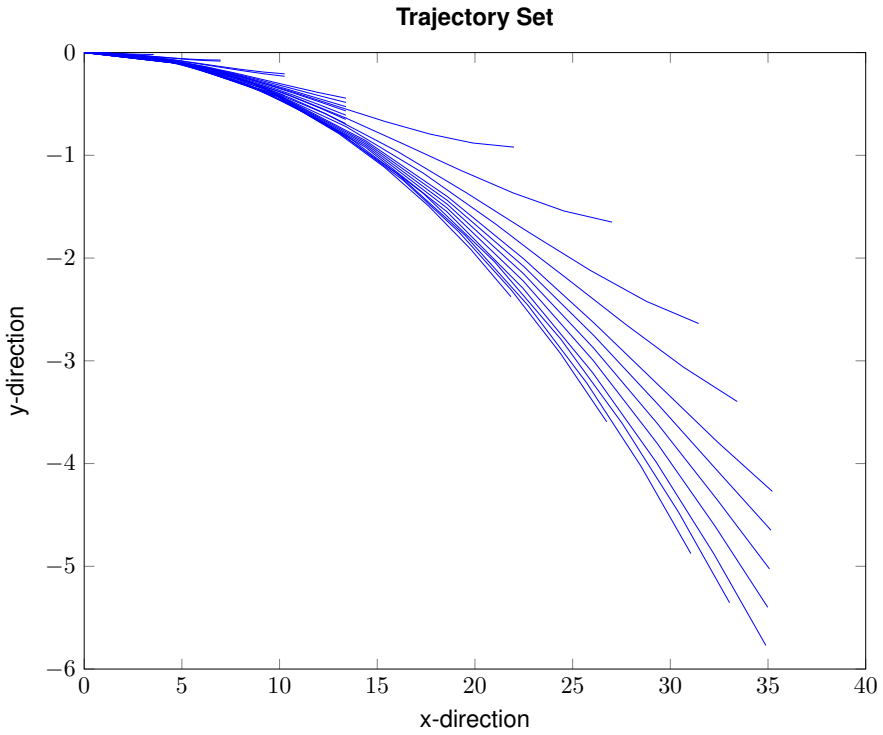


Figure A.2: Trajectory path set with trajectories in blue the initial state has maximum veloctiy and minimum lateral acceleration

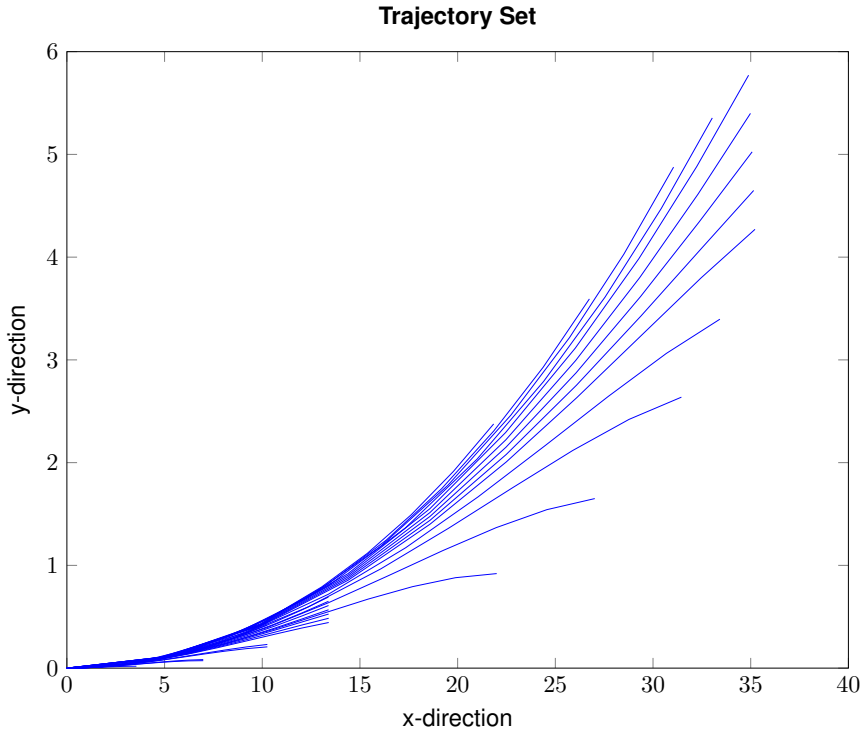


Figure A.3: Trajectory path set with trajectories in blue the initial state has maximum veloctiy and maximum lateral acceleration

Appendix B

3D OBB tree

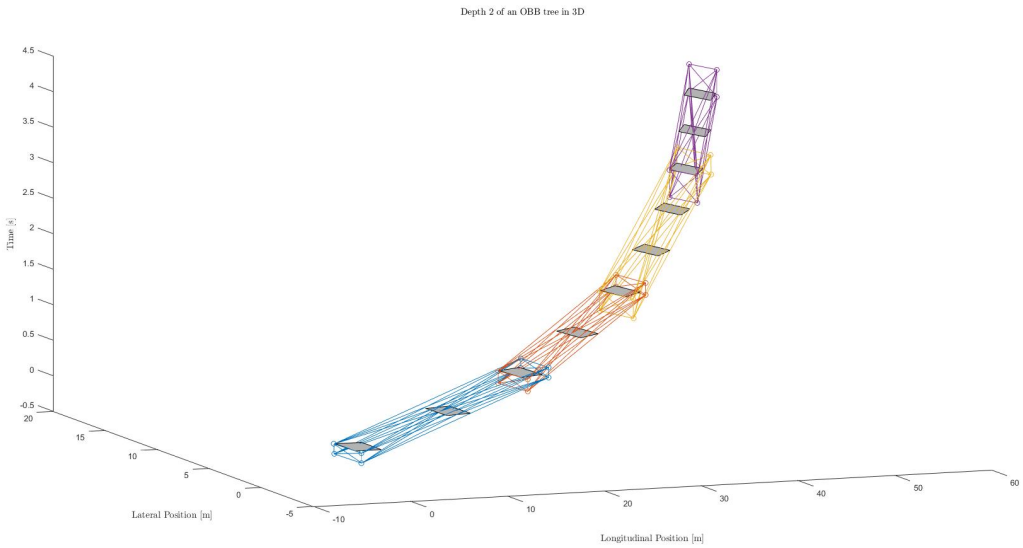


Figure B.1: 3D representation of the depth 2 of an OBB tree