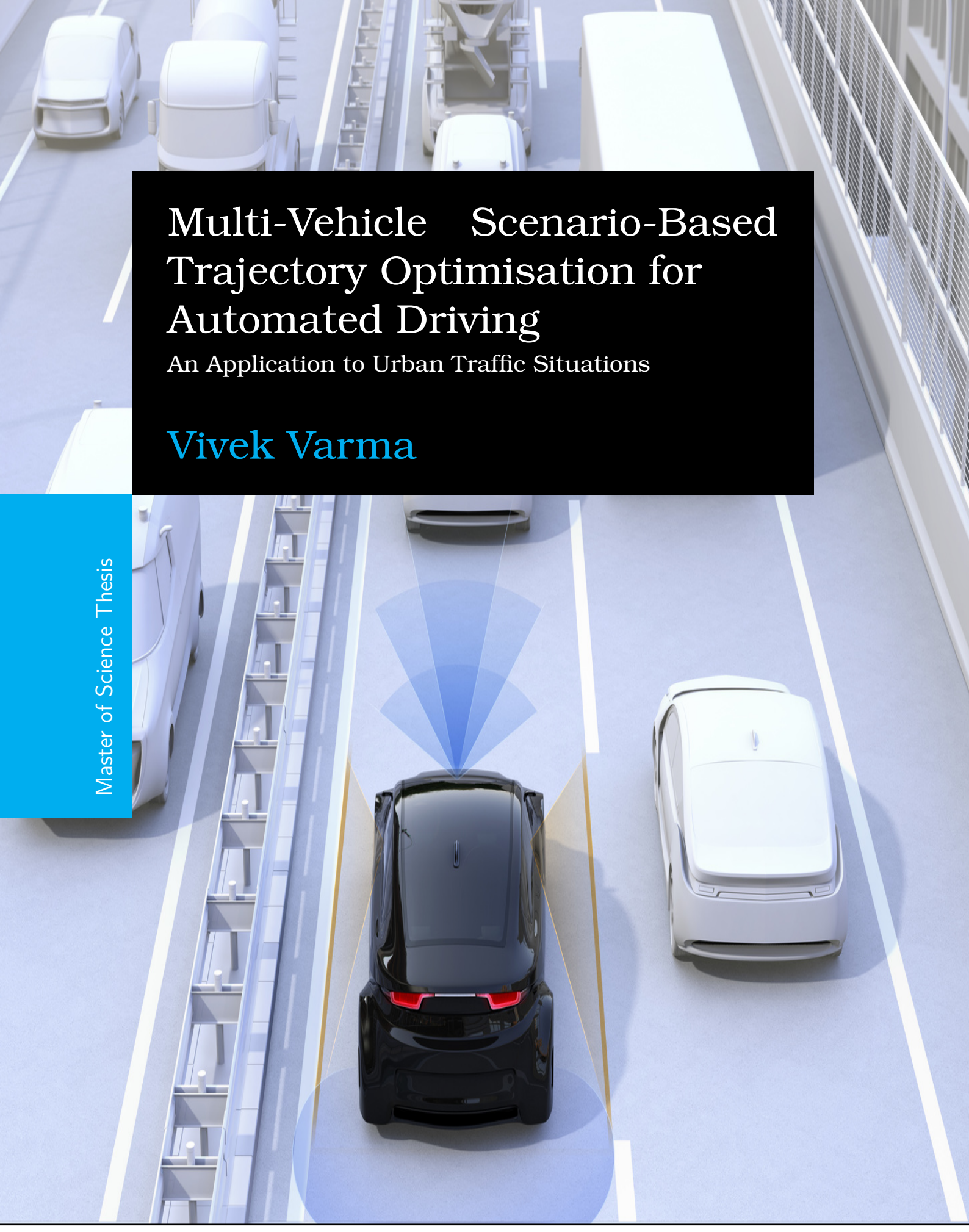


Multi-Vehicle Scenario-Based Trajectory Optimisation for Automated Driving

An Application to Urban Traffic Situations

Vivek Varma

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Multi-Vehicle Scenario-Based Trajectory Optimisation for Automated Driving

An Application to Urban Traffic Situations

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Vivek Varma

November 4, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Multi-Vehicle Scenario-Based Trajectory Optimisation for Automated Driving

by

Vivek Varma

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 4, 2022

Supervisor(s):

Dr.Ing. Sergio Grammatico

Ir. Luyao Zhang

Reader(s):

Dr. Azita Dabiri

Dr. Barys Shyrokau

Abstract

Automated driving is where automobiles meet robotics. With the recent advances in intelligence, sensor technology, wireless technology, and computation power, we are inching ever closer to realising full autonomy in a vehicle. We are nowhere near the end of the line, however. Automated vehicles will have to interact with static obstacles like pavements, dividers, and poles and dynamic elements like pedestrians and other vehicles. This opens up a range of sub-topics on privacy, safety, and decision-making. While driving on the road in the presence of other agents(human or autonomous), safety constraints and collision avoidance are of paramount importance. Even with this achieved, we need a good performance in the latency of processing each iteration; we need constraints to be followed and, of course, ensure minimal errors in our control.

Through this thesis, we introduce automated driving, its general pipeline stack, and industry standards for autonomy. We then get ourselves up-to-date with the latest trends and advances in motion planning, decision-making, and control of automated vehicles. Once that is covered, we narrow our focus towards using a scenario-based approach to safe trajectory planning. We delve in-depth into safety constraints guaranteed by this approach and discuss previous results obtained by using this method with pedestrians in an urban setting.

This thesis aims to extend this previously used scenario-based planning method to a multi-vehicle implementation. Two methods of modelling scenario distributions(assumed to be Gaussian) are proposed, implemented, and compared using evaluation metrics like computation times, safety, and time taken to reach the goal. The first method models each obstacle vehicle as a series of obstacles linked together by the vehicle constraints. Thus, each vehicle is represented by multiple collision regions corresponding to each obstacle. The second method models the vehicle as having a single collision region with multiple scenario distributions. This extension's safety/risk guarantee is shown both theoretically and experimentally. Experiments are conducted in simulation on an urban straight road and at a T-Junction with multiple obstacle vehicles, and performances are compared, not only between these two methods but also with each obstacle modelled as a single disc, which is the baseline implementation. Conclusions are then

made based on the performance metrics, and further improvements are proposed. It is shown that modelling the vehicle as a series of linked scenarios improves over the baseline method and the multiple obstacle discs implementation in terms of safety and computation times respectively.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This document is a part of my Master of Science graduation thesis. The idea of doing my thesis on Scenario-Based Trajectory Optimization came after a discussion with my daily supervisor, Ir. Luyao Zhang, who suggested that the interactions between an ego vehicle and crossing pedestrians could be modelled as a multi-vehicle interaction at a junction. Thank you to Luyao and Dr. Ing. Sergio Grammatico for their constant help and support throughout this period.

I would like to thank Ir. Oscar de Groot and Ir. Anish Sridharan for their help, starting from providing me with the code base, to answering many of the doubts and questions I posed.

Thank you to all those at the Formula Student Team Delft, DSA Kalman, and Loop Robots, who pushed me beyond my comfort zone and helped me realise different dimensions to my professional and personal self.

Immense gratitude goes out to all my friends in the Netherlands, and in India, for being pillars of strength and trust, even when I doubted myself. I will forever cherish the moments and memories that I have made in the Netherlands over the course of my studies. A special shout-out to my better half, Aparajita Karmakar, who has firmly stood by my side through thick and thin.

Lastly, my biggest source of inspiration, my incredible parents, who have made everything that I am today, a possibility. Nothing I ever do in this lifetime does justice to what they deserve.

Delft, University of Technology
November 4, 2022

Vivek Varma

"Do not go gentle into that good night,
Old age should burn and rave at close of day;
Rage, rage against the dying of the light.
Though wise men at their end know dark is right,
Because their words had forked no lightning they
Do not go gentle into that good night."
— *Dylan Thomas*

Chapter 1

Introduction

This introductory chapter talks about the origins of self-driving vehicles and their progress over time and raises a few questions, which will be answered in subsequent chapters.

1-1 History of Automated Driving

Figure 1-1 depicts one of the first ideas of a self-driving car. This was designed by Leonardo da Vinci around five centuries ago! He proposed using high-tension strings as the torque source to steer the cart along a predetermined path. There was not much-known progress since then, until the automobile revolution.

The 1920s bore the first witness to this revolution when a radio-controlled car was driven through New York City without a human presence. In the 1950s and 1960s, sensors like current sensors and cameras were integrated into cars, and these cars were run on their own using sensor feedback.

Figure 1-1: Leonardo da Vinci's Self-Propelled Cart Design [2]

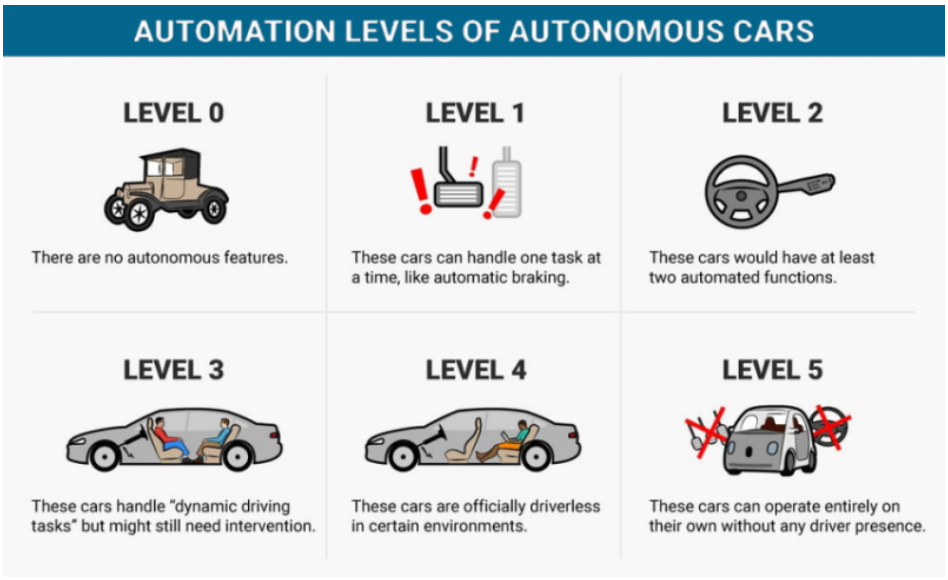


Figure 1-2: Visual Chart for "Levels of Driving Automation" Standard for Self-Driving Vehicles

By the 1990s, neural networks were already being integrated for image processing and steering control of self-driving cars, especially at Carnegie Mellon University. The 2000s witnessed the most significant progress in automated driving with the introduction of the DARPA challenge to navigate the desert and urban routes.

Witnessing the potential of self-driving cars started a giant race amongst companies and institutions worldwide to accomplish automated driving fully. Automobile companies like Ford, Mercedes-Benz and BMW were significant players but were also joined by non-automobile giants like Google, Amazon and Uber. No one has realised a fully automated car yet. Many companies have realised partial autonomy(the levels of autonomy will be covered in the next section), but we are still far from a 100% automated vehicle. Tesla has commercially deployed a semi-automated vehicle that can navigate highways entirely on their own. Cruise and Waymo are currently testing automated shuttle services for their employees in San Francisco, Seattle, and most recently, Los Angeles. Baidu has also recently obtained permission to run their automated shuttles in Wuhan and Chongqing in China.

Automated vehicles are more widely used in mines, reactors, and other applications successfully, but navigating urban cities along with other vehicles is a different ball game altogether. Researchers worldwide are working tirelessly to achieve this mammoth engineering task.

1-2 5 Levels of Automated Driving

The Society of Automotive Engineers(SAE) released a standard for vehicle autonomy known as J3016. This can be summarized in Figure 1-2. The 5 levels very roughly summarized[1] are:

- Level 0: No Driving Automation- Completely manual vehicle
- Level 1: Driving Assistance- Support the driver, increase safety and convenience
- Level 2: Partial Driving Automation- Can independently perform certain maneuvers
- Level 3: Conditional Driving Automation- Drives in an automated under certain conditions
- Level 4: High Driving Automation- Vehicle can drive in an automated fashion in local conditions
- Level 5: Full Driving Automation- Fully automated, can drive anywhere, whatever the condition

With this definition, we begin to assess where we stand concerning this standard. With Automotive Cyber-Security and stringent rules being a significant factor, we are still at Level 2 on this autonomy scale. With rapid research, development and funds poured into automated driving, many different ideas and methodologies come forth to reach the same ultimate objective. The technology stack of an automated vehicle is similar to that of a wheeled mobile robot in terms of the modules but at a much higher level due to more complex requirements.

The modules that an automated vehicle has are:

- Sensor Inputs
- Stack Hardware Infrastructure:
- Computation and Processing(Hardware/Software Interfacing)
- Applications like planning, decision making and control

The bulk of the focus of this thesis lies in the motion planning, decision making and control module. In the next chapter, we individually address the progress in this field concerning automated driving.

1-3 What lies ahead?

Every question we currently answer opens up more questions to be answered. With every passing software and hardware research iteration, we can access better infrastructure, which leads to us being able to perform more tasks of higher complexity.

Safe and optimal planning and decision-making are essential for any automated vehicle on the road. This needs to account not only for the road and lane boundaries but also for the presence of other road components like vehicles, pedestrians, and signals. A planning and decision-making module's task is to take sensor data from surroundings as input and accordingly lay out a set of waypoints that the car must follow.

This thesis investigates scenario-based trajectory optimisation and poses some interesting research questions:

- How does one transition from a pedestrian to a vehicular scenario model, and what factors should be considered when making this transition?
- What is the most computationally efficient manner to model obstacle(vehicle) uncertainties without compromising safety?
- What are the implications on the safety guarantees caused by such a representation?

To answer these questions, the rest of this thesis is structured as follows:

- Chapter 2- Provides a recap of the literature research, which lays down an overview of the different layers of a planning and decision making module and reviews some state-of-the-art algorithms.
- Chapter 3- Establishment of some mathematical background and terminologies prerequisite to understanding the thesis.
- Chapter 4- Introduction to the proposed approach and methodology followed to implement this approach.
- Chapter 5- A coverage of the results obtained and their explanations.
- Chapter 6- Draws conclusions based on the results and provides direction for improvement and future work.

Chapter 2

Summary of Literature Research

2-1 Planning and Decision Making

The decision making module for an automated vehicle can be hierarchically classified into four layers (Figure 2-1).

- The highest layer is the route planning layer, which takes the desired destination of the user and the map of the area(which it may or may not have) and returns the waypoints that need to be followed to reach the desired destination.
- The behavioural layer takes the global waypoints and some data from the surroundings. It selects the kind of behaviour the car must follow on the road(for example, to read a stop sign and stop, to decide to change lanes or to cross an intersection).

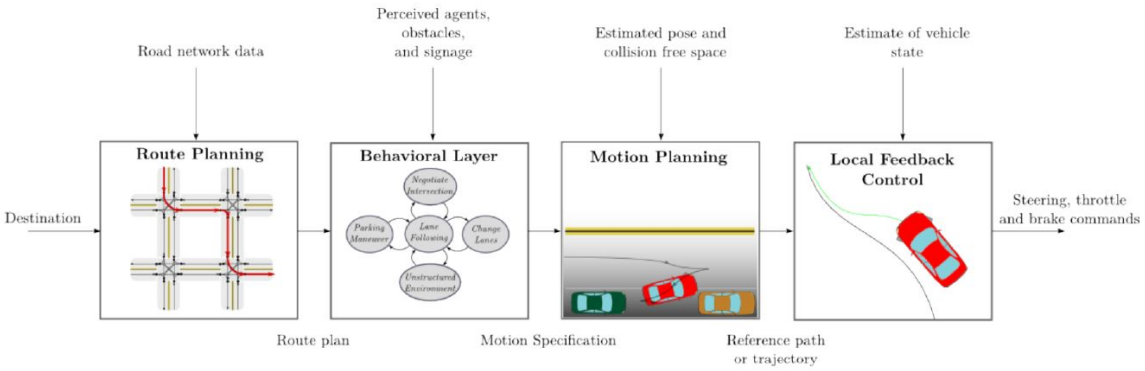
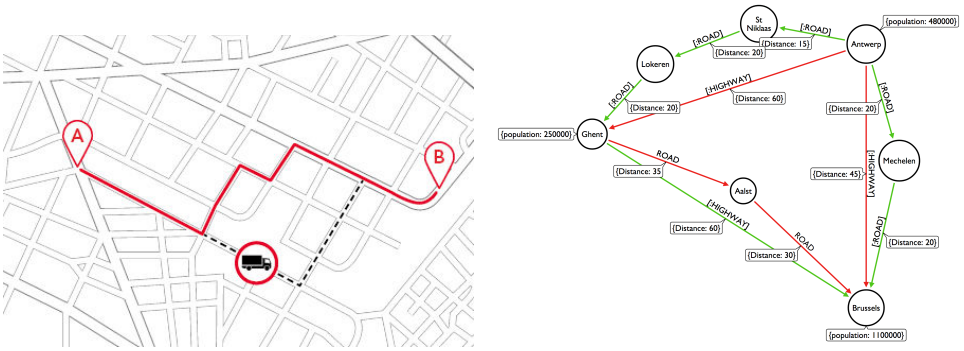


Figure 2-1: Overview of the Planning and Decision Making Module [60]



(a) Route Planning Layer

(b) Graphical Representation of a route with Edges and Vertices

Figure 2-2: Route Planning as a Graph

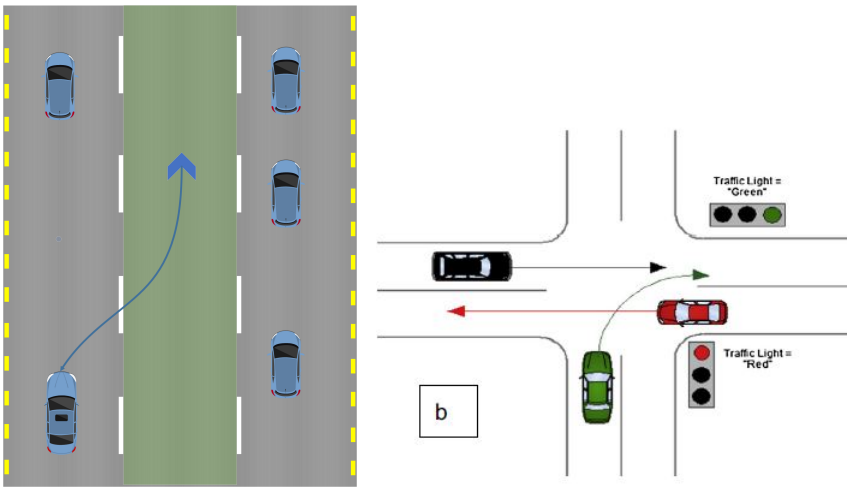
- The lower layer is the motion planning layer which then takes the input as the behaviour to be followed from the higher layer and accordingly plans the motion(by laying down a reference trajectory or local waypoints) for the vehicle to accomplish this task.
- The lowest layer is the local feedback control which takes as inputs the reference path and adjusts the vehicle's current position with respect to the corresponding reference point by providing actuation in the form of steering and acceleration. Thus the vehicular control loop is closed by taking sensor data from the surroundings, checking where the vehicle currently is with respect to where it should be instead, and accordingly sending actuator commands to it.

Thus, we can infer that the role of planning and decision-making is to lay down the path, both locally and globally, for the vehicle to follow and ensure that the vehicle does indeed follow them by using feedback from the environment. Further details of these methods can be found in [60], [67] and [40].

2-1-1 Route Planning

From Figure 2-2, an idea of what route planning means can be inferred. It involves using various heuristics to compute the optimal route from the start to the end. In Figure (a), a route has been mapped from Point A to Point B. We can visualise this route being the result of an internal optimal route computation where different points are mapped as vertices, linked by edges with some cost. The optimal route problem thus reduces to computing the optimal route through a graph, as in Figure (b).

This is a transportation problem, and many different solutions are proposed to solve such problems. There is the GSP(Generic Shortest Path Algorithm) [30], Label-Setting Algorithms like Dijkstra's [25] and Bi-Directional Dijkstra's Algorithm. There are also the Label-Correcting Algorithms like Bellman-Ford[7][32] and Heuristic Estimators like A^* [57]. It is important to check the time and space complexity of such algorithms for



(a) For Lane Changes (b) At an Intersection with Traffic Signals

Figure 2-3: Instances of Behavioral Decision Making(referred to as "Manoeuvre Planning" in [40])

automated driving applications, as there will be a significant number of nodes needed for many routing applications, and the route planner should be practically compatible in such scenarios.

2-1-2 Behavioral Decision Making

Figure 2-3 (a) and (b) serve as a pointer towards the role of the behavioural decision-making module. This layer acts as the "Brain" of the planning layers; where the way-points provided by the routing layer are assessed, interactions with other traffic participants(moving as well as stationary) are defined, and a decision is made in the best interests of all the concerned entities [40].

The instinctive approach to implement behavioural decision-making would be using finite state machines, where the behaviour is time triggered or event triggered[13]. However, in urban settings, there is much more randomness and uncertainty, creating the need for alternative decision-making methods. The decision-making system should either have a good obstacle prediction and risk management capability or a suitable decision theory-based framework [40] that can be termed interactive behaviour-aware planning. There are different categories according to which these algorithms can be classified based on the methods used to make decisions.

Game Theoretical

Game theoretical approaches aim to optimise an individual agent's cost function, resulting in optimal controls for the agent while also considering actions undertaken by other agents. Automated driving scenarios can be modelled as different types of games depending on the situation under consideration. Marden & Shamma [53] provides a comprehensive survey regarding the basics of game theory, the different types of games and equilibria.

Liniger & Lygeros [49] modelled the automated racing of two cars as a non-zero-sum non-cooperative game where the players are only rewarded for collision avoidance. Wang et al. [79] extend the work by introducing an ego factor term which determines how much importance a vehicle will place on its competitiveness as a trade-off with collision avoidance. Wang et al. [77] use a risk-sensitive game theoretic framework to model the stochastic behaviour of interacting agents. Fabiani & Grammatico [27] proposed a mixed-logical dynamical framework along with a set of driving rules to ensure efficient and safe driving of a selfish automated vehicle on a highway. Chiu et al. [19] propose a robust defensive driving model based on a general-sum dynamic game framework. This robustness is implemented by adding an adversarial phase (a time period where neighbouring agents have erratic behaviour) to the cost function to render the ego agent's trajectory robust to the non-ego agent's uncertainties. Fisac et al. [31] propose an algorithm where the dynamic game is broken down into a long-horizon strategic game and a short-horizon tactical game. The long horizon non-zero sum game has simplified dynamics and a complete closed-loop information feedback structure. In contrast, the shorter horizon game has accurate dynamics and a simplified information structure(informed by the long-term planner), with this approach also accounting for non-deterministic decision-making scenarios. Laine et al. [41] use a Bernoulli random variable to represent the probability of specific hypotheses. Each hypothesis represents an estimate of other agents' constraints and objectives in situations of uncertainty regarding the intentions of the agents.

Probabilistic

There are several approaches in the probabilistic domain originating from several possible sources of uncertainty in terms of the motions and intentions of agents. In such situations, scenario optimisation, intention estimation, dynamic programming and probabilistic sampling find a range of applications [72].

Fridovich-Keil et al. [33] model a Bayesian confidence regarding the agent motion model variance. This prediction is combined with a robust motion planner after selecting a high-probability collision-free path. Guan et al. [35] formulate the automated driving task as a Markov Decision Process(MDP) with an environment state space and agent action space. A state transition and reward model are built using a prediction of surrounding vehicles. Bai et al.[5] use DESPOT [68] to implement automated driving in crowded areas. DESPOT is a scenario optimisation framework which samples scenarios and searches for a near-optimal plan at each time step. Here, the sampling is done according to the confidence variable. Luo et al. [50] propose an Importance Sampling method to the DESPOT. The Importance Sampling samples scenarios according to their importance.

Learning-Based

Lenz et al. [46] compare deep neural networks, feature combinations and past inputs for motion prediction, even accounting for uncertainties. Bakker & Grammatico [6] propose the introduction of adjacent lanes in a structured observation grid(thus making collisions observable) and lane-changing decisions of neighbouring agents to be in the state vector. This enables future predictions regarding the collision, and action can

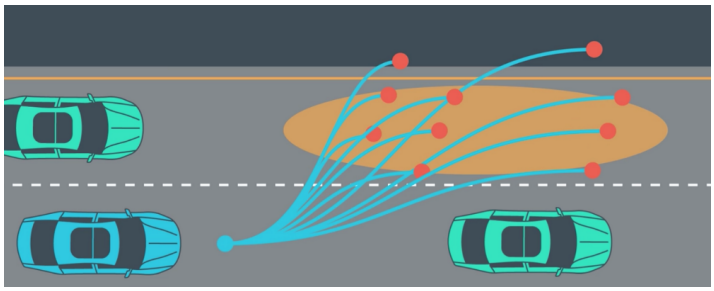


Figure 2-4: Motion Planning Layer

be taken accordingly to avoid them. Toghi et al. [74] represent the mixed-autonomy driving problem as a partially observable stochastic game and derive optimal policies using Deep Multi-Agent Reinforcement Learning to enable altruistic agents to drive safely and with inter-agent coordination.

Other Safety Approaches

With the safety of decision-making being the driving force behind this thesis, there are also several other approaches where the focus is on a safety certificate or a guarantee of safety. Safety constraints can be guaranteed using Control Barrier Functions(CBFs), where forward invariance of a safe set is obtained [69] [51] [76]. Polling systems-based coordination control methods where safety guarantees and time delay bounds are provided [55] [56]. Connected and Automated Vehicles(CAVs) based cooperative communication and control methods have a wide array of corresponding literature, where the focus is decentralized control with Vehicle-to-Vehicle(V2V) communication [63] [36] [54] [20] [83]. Planning space-based methods provide safety guarantees within a certain range of values. [78] [61].

2-1-3 Motion Planning

This layer is responsible for translating the behavioural decision-making level commands into a local trajectory or path the vehicle needs to follow to accomplish that decision(Figure 2-4). This layer needs to consider dynamic/static obstacles, smooth motion, and dynamics of the vehicle while laying out the way-points for the vehicle to follow.

Motion planning methods can be roughly categorised into three [67] classes based on methods used: Input Space Discretization (Motion primitives [29][62][80], Lane Graphs, Geometric Graphs [43][18][58][71][42], Sampling based graphs), Randomized Planning [44][45][39] and Receding Horizon Control [28][52][47].

2-1-4 Local Feedback Control

In this layer, the vehicle's current state is compared with the desired state(reference trajectory from the motion planning layer). From Figure 2-5, errors are computed and

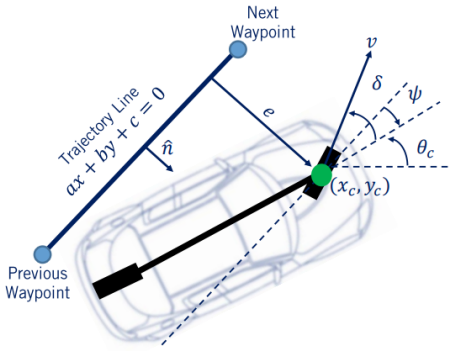


Figure 2-5: Lowest Layer- Local Feedback Control

nullified by designing controllers which send appropriate commands to the actuator. This is where the control loop is closed; thus, essential properties like robustness and stability need to be considered.

Several methods can be employed for this low-level control layer. Pure pursuit controllers are a method where the vehicle is made to follow a point on the reference path at a certain look-ahead distance [75] [21]. Front and rear wheel-based position feedback control use the wheel position to stabilize the nominal wheel path [73] [64]. Another method uses a Control Lyapunov function based on the vehicle state [81]. This can guarantee exponential stability locally. Other methods are output feedback linearization[22] and PID based approaches. MPC can also be used in the lower layer as a tracking controller.

A visual summary of the methods covered in this section can be seen in Figure 2-6.

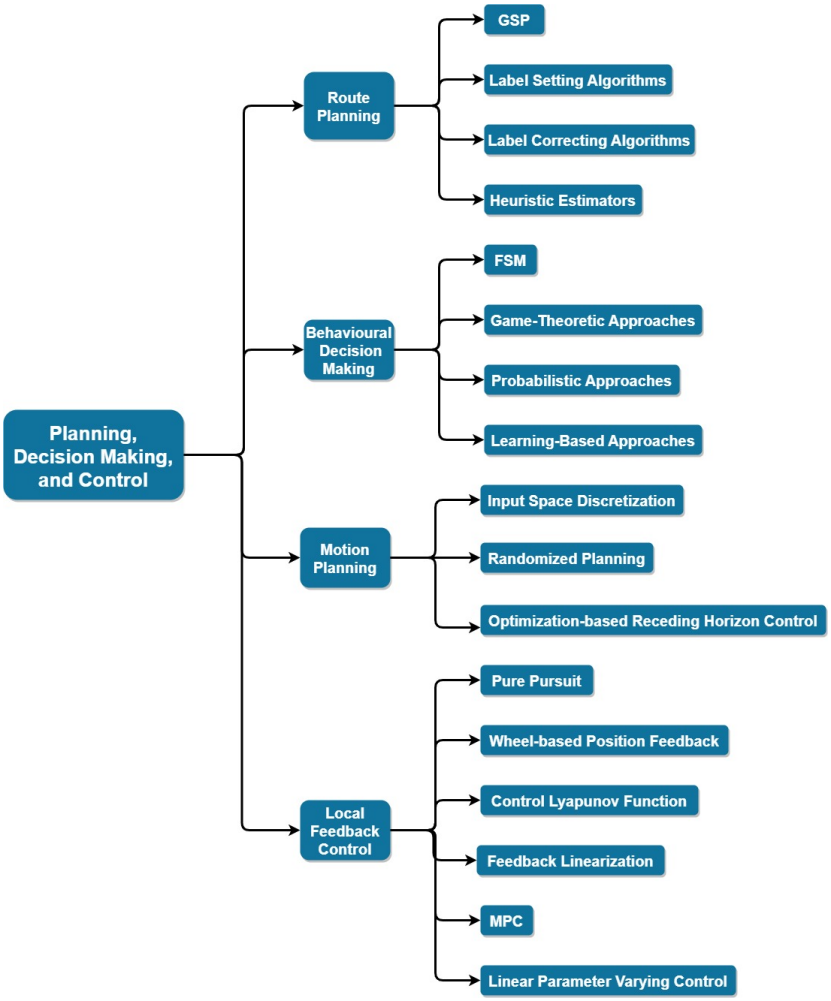


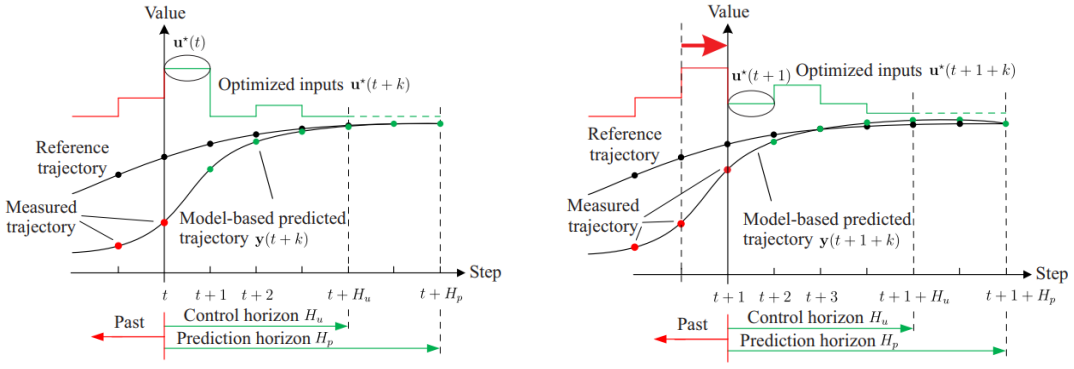
Figure 2-6: Summary of the Planning, Decision Making and Control Layers and Methods

Chapter 3

Background

3-1 Model Predictive Control

Model Predictive Control(MPC) is a control strategy used to control a system optimally, subject to certain constraints. It uses a system dynamics model, which predicts the system's behaviour and optimises the states and inputs over a certain time horizon. The optimisation problem generally comprises an objective function to be optimised and a set of constraints. The result of this optimisation is an optimal series of inputs over the time horizon. Only the first of these inputs is applied to the system, which is then repeated in a receding horizon manner. This problem is solved at every sample time to account for prediction, and observed mismatches [3]. Figure 3-1 illustrates the working of MPC by illustrating two consecutive iterations. t represents the current time step, H_p represents the prediction horizon, H_u is the control horizon, or the number of steps after which change in the control input is not allowed. Thus, $H_u \leq H_p$. At each time step, the optimisation problem is solved over horizon H_p(Figure 3-1(a)).



(a) MPC at Iteration 1

(b) MPC at Iteration 2

Figure 3-1: The working of MPC illustrated at 2 consecutive iterations [3])

The result is a vector of control inputs $\Delta\mathbf{u}^\star = \begin{pmatrix} \Delta\mathbf{u}^\star(t) & \dots & \Delta\mathbf{u}^\star(t+H_p-1) \end{pmatrix}^T$. The applied input is however only $\mathbf{u}(t) = u(t-1)+\Delta u^*(t)$ at time step $[t, t+1)$. At the next time step $t+1$, a new optimisation problem is solved, based on measured states, and the horizon is shifted, as seen in Figure 3-1(b). Based on the model used, the MPC problem can be linear or non-linear, and based on the convexity of the optimisation, it can be convex or non-convex. The general components of an MPC problem are covered below.

3-1-1 Dynamic Model and Predictions

Generally, an MPC problem uses the dynamic model of the system as a constraint-based on which the predicted states evolve over the future. Complex models provide better representations of the system behaviour but also take up computation time. Simplified models, or linearised models, provide good approximations in specific regions and are computationally less expensive. A general discrete-time model is represented by Equation 3-1:

$$\begin{aligned} \mathbf{x}(t+1) &= f(\mathbf{x}(t), \mathbf{u}(t)) \\ \mathbf{y}(t) &= g(\mathbf{x}(t), \mathbf{u}(t)) \end{aligned} \tag{3-1}$$

where states $\mathbf{x} \in \mathbb{R}^n$, control inputs $\mathbf{u} \in \mathbb{R}^m$, controlled outputs $\mathbf{y} \in \mathbb{R}^p$, states evolution function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and output function $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^p$. For the purpose of this thesis, a continuous-time second-order bicycle model is used with linearisation every 200ms. Further details of the implementation will be provided shortly.

3-1-2 Objective Function

The objective function represents the function to be optimised and is generally split into the terminal cost and the running cost. The running cost is usually a weighted error function between the states and inputs and the reference over the prediction horizon. The terminal cost is the weighted error between the states and inputs and the reference at the final prediction step H_p. A generalized objective function looks like the one in Equation 3-2.

$$\mathbf{J}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \underbrace{\sum_{t=1}^{H_p-1} \tilde{\mathbf{x}}(t)^T \mathbf{Q}_i \tilde{\mathbf{x}}(t) + \sum_{t=1}^{H_u-1} \tilde{\mathbf{u}}(t)^T \mathbf{R}_i \tilde{\mathbf{u}}(t)}_{\text{Running cost}} + \underbrace{\tilde{\mathbf{x}}\left(H_p\right)^T \mathbf{P} \tilde{\mathbf{x}}\left(H_p\right)}_{\text{Terminal cost}}, \tag{3-2}$$

where $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}_{ref}(t)$ and $\tilde{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_{ref}(t)$. $\mathbf{Q_i}$ and $\mathbf{R_i}$ represent the (possibly) time-varying weights on the state error and input error in the running cost while $\mathbf{P}$ is the weight on the terminal cost state errors.

3-1-3 Constraints

In an MPC formulation, constraints to the system can be directly incorporated as constraints to the optimisation problem. Thus, a saturation of control inputs, system

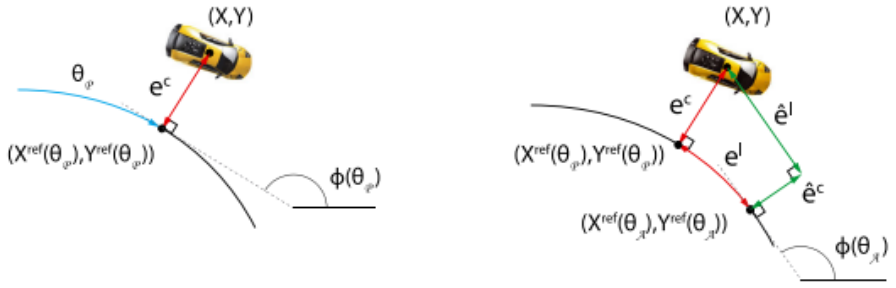


Figure 3-2: On the left is the contouring error e^c, on the right is the lag error e^l, with their linear approximations $\hat{e}^c$ and $\hat{e}^l$ respectively. [48]

dynamics, obstacles on the road, and road/lane boundaries, can all be directly used as constraints to the problem. The solution of the optimisation thus accounts for all this inherently. Constraints can provide safety, guarantee stability, provide a "hot-start" or an initial condition, and model complex environments. Constraints can be represented as sets as shown in Equation 3-3.

$$\begin{aligned}
&\mathbf{x}(t+k) \in \mathcal{X} \subseteq \mathbb{R}^n, k = 1, \ldots, H_p \\
&\mathbf{y}(t+k) \in \mathcal{Y} \subseteq \mathbb{R}^p, k = 1, \ldots, H_p \\
&\mathbf{u}(t+k) \in \mathcal{U} \subseteq \mathbb{R}^m, k = 0, \ldots, H_u - 1 \\
&\Delta\mathbf{u}(t+k) \in \Delta\mathcal{U} \subseteq \mathbb{R}^m, k = 0, \ldots, H_u - 1
\end{aligned} \tag{3-3}$$

Thus, a general MPC optimisation problem looks like Equation 3-4.

$$\begin{array}{lllll}
\min_{\mathbf{x},\mathbf{u}} & \sum_{t=0}^{N} J_t\left(\mathbf{x_t}, \mathbf{u_t}\right) & & => & \text{ObejctiveFunction} \\
\text{s.t. :} & \mathbf{x_{t+1}} = f\left(\mathbf{x_t}, \mathbf{u_t}\right),\ t = 0, 1, \ldots, N-1 & & => & \text{Model} \\
& g\left(\mathbf{x_t}, \mathbf{u_t}\right) \leq d_t,\ t = 0, 1, \ldots, N-1 & & => & \text{Constraints} \\
& \mathbf{x_0} = x_{\text{init}} & & => & \text{InitialCondition}
\end{array} \tag{3-4}$$

where $\mathbf{x}$, $\mathbf{u}$ are the states and control inputs respectively, N is the prediction horizon, t is an arbitrary time-step within the prediction horizon, $J_t(.)$ is the stage cost function to be optimised in terms of the states, inputs and references, subject to the system model, the initial condition and some other equality or inequality constraints.

3-2 Model Predictive Contouring Control

Model Predictive Contouring Control [66] [48] is a formulation that combines path tracking and path generation by planning a progress-optimal path that considers the non-linear projection of the vehicle's position onto the centre line. In MPCC, the reference path is parameterized by the arc length, $\theta \ \epsilon \ [0, L]$, where L is the total arc length, by using spline polynomials. Using the spline polynomial arguments makes any spatial reference path easily accessible. The angle of the tangent to the path with

respect to the reference point on the X-Axis is defined as

$$\Phi(\theta) \triangleq \arctan\left\{\frac{\partial Y^{\text{ref}}(\theta)}{\partial X^{\text{ref}}(\theta)}\right\}$$

The contouring error e^c is taken as the orthogonal distance from the car to the reference path. However, using the projection operator to minimise the orthogonal distance in itself resembles an optimisation problem, and is not feasible to be performed online. Thus, an independent approximation of this spline polynomial is introduced and the lag error e^l is defined as the link between these two approximations. It is a measure of the quality of the approximation(Figure 3-2).

$$e^l\left(X, Y, \theta_{\mathcal{A}}\right) \triangleq |\theta_{\mathcal{A}} - \theta_{\mathcal{P}}|$$

where $\theta_{\mathcal{A}}$ is the independent variable approximation of the projection $\theta_{\mathcal{P}}$. Thus in order to be independent of the projection $\theta_{\mathcal{P}}$, the lag error and contouring error are approximated as a function of the position X,Y and the approximated projection $\theta_{\mathcal{A}}$. The linearised contouring error $\hat{e}^c_k$ is the orthogonal component of the error between X, Y and $X^{ref}(\theta_{\mathcal{A}})$ and $Y^{ref}(\theta_{\mathcal{A}})$ and at time step k is defined as

$$\hat{e}^c\left(x_k\right) = \sin\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(X_k - X^{\text{ref}}\left(\theta_k\right)\right) - \cos\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(Y_k - Y^{\text{ref}}\left(\theta_k\right)\right)$$

The approximate lag error $\hat{e}^l_k$ is tangential component of the error between X, Y and $X^{ref}(\theta_{\mathcal{A}})$ and $Y^{ref}(\theta_{\mathcal{A}})$ and at time step k is defined as

$$\hat{e}^l\left(x_k\right) = -\cos\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(X_k - X^{\text{ref}}\left(\theta_k\right)\right) - \sin\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(Y_k - Y^{\text{ref}}\left(\theta_k\right)\right)$$

The problem formulation is as shown in Equation 3-5

$$\begin{aligned}
\min \quad & \textstyle\sum_{k=1}^{N} \begin{bmatrix} \hat{e}^c_k \\ \hat{e}^l_k \end{bmatrix}^T \begin{bmatrix} q_c & 0 \\ 0 & q_l \end{bmatrix} \begin{bmatrix} \hat{e}^c_k \\ \hat{e}^l_k \end{bmatrix} - q_v v_{\theta,k} + \Delta u_k^T R_\Delta \Delta u_k \\
\text{s.t.} \quad & x_0 = x(0) \\
& x_{k+1} = f\left(x_k, u_k\right) \\
& \hat{e}^c\left(x_k\right) = \sin\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(X_k - X^{\text{ref}}\left(\theta_k\right)\right) - \cos\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(Y_k - Y^{\text{ref}}\left(\theta_k\right)\right) \\
& \hat{e}^l\left(x_k\right) = -\cos\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(X_k - X^{\text{ref}}\left(\theta_k\right)\right) - \sin\left(\Phi^{\text{ref}}\left(\theta_k\right)\right)\left(Y_k - Y^{\text{ref}}\left(\theta_k\right)\right) \\
& \Delta u_k = u_k - u_{k-1} \\
& x_k \in \mathcal{X}_{\text{Track}} \\
& \underline{x} \leq x_k \leq \bar{x} \\
& \underline{u} \leq u_k \leq \bar{u} \\
& \Delta u \leq \Delta u_k \leq \overline{\Delta u}
\end{aligned} \tag{3-5}$$

and q_c and q_l are weights on the contouring error and lag error respectively, q_v is a weight on the $v_{\theta,k}$, which is a measure of the approximated progress along the path. R_Δ represents the weights on the control inputs. It can be seen that X^{ref} and Y^{ref} are now parameterised by θ_k, which are spline interpolation parameters. x_0 represents the starting point or the initial condition. The system dynamics, bounds on the states and inputs, linearised contouring error and lag error, form the rest of the constraints in this problem formulation.

3-2-1 Local Model Predictive Contouring Control

LMPCC [12] or a Local MPCC is a reformulation of MPCC [66] enabling real-time collision avoidance with a lightweight implementation. It is a local motion planning framework extended to robots in unstructured environments. Here, the global reference path is broken into segments and locally followed while ensuring continuity is maintained over the various local reference paths. This algorithm also ensures static and dynamic collision avoidance and visualises a polytope of the free space and predicted free spaces.

This model creates a static map by online computation of a set of convex regions in free space. The robot is considered a combination of n circles, while an ellipse represents dynamic obstacles. The dynamic obstacles at future time steps are represented with a constant velocity model with Gaussian uncertainty. This collision with dynamic obstacles is modelled using a collision region found by the Minkowsky sum of an ellipse and a circle. The global reference path consists of a sequence of multiple path-segments containing M way-points ($p_m^r = [x_m^p, y_m^p]$). The closest path segment to the robot is selected at every step of the optimisation, along with the following few path segments, depending on the robot's speed. At each step in the time horizon, the controller minimises the distance to the path segments while avoiding static and dynamic obstacles. This method is lightweight and runs in real-time in an automated system. The algorithm of LMPCC can be summarised by Algorithm 1.

LMPCC is just an MPCC problem written for a path-following task. There are three major changes made to the MPCC formulation to accomplish this:

- In each planning stage, θ_0 is initialised. This is a parameter used to monitor the progress over the reference path. Each time the closest path segment m is found, θ_0 is computed using line search in the same neighbourhood as the previous iteration.
- The global reference path is broken into M segments. In order to reduce the computational load, $\eta \leq M$ segments are picked to generate the local reference path, which is tracked in the optimisation problem instead of the global reference. η in the local reference path is a function of the prediction horizon length, the individual path segment lengths, and the robot's speed at each time instance. A conservative η is chosen considering the maximum longitudinal velocity of the robot v_{max} and imposing that the covered distance is lower than the lower bound of the travelled distance along the reference path. This is shown in Equation 3-6.

$$\underbrace{\tau \sum_{j=0}^{N-1} v_j}_{\text{Traveled distance}} \leq \tau N v_{\max} \leq \underbrace{\sum_{i=m+1}^{m+\eta} \left\| \boldsymbol{p}_{i+1}^{\mathrm{r}} - \boldsymbol{p}_i^{\mathrm{r}} \right\|}_{\text{Waypoints distance}} \leq \underbrace{\sum_{i=m+1}^{m+\eta} s_i}_{\text{Reference path length}} , \tag{3-6}$$

  Here N is the prediction horizon, τ is the length of the discretisation steps along the horizon, v_j is the predicted speed of the robot at step j, m is the index of the closest path segment to the robot, $\mathbf{p}_i^r$ is the i^{th} waypoint of the path segment and s_i is the length of each path segment.

- The η reference paths are concatenated to form a differentiable local reference path $\mathbf{L}^r$, which is tracked by LMPCC as shown in Equation 3-7.

$$\overline{\boldsymbol{p}}^r(\theta_k) = \sum_{i=m}^{m+\eta} \sigma_{i,+}(\theta_k)\,\sigma_{i,-}(\theta_k)\,\varsigma_i(\theta_k) \tag{3-7}$$

where

$$\sigma_{i,-}(\theta_k) = 1/\left(1 + e^{\left(\theta - \sum_{j=m}^{i} s_i\right)/\epsilon}\right),\ \sigma_{i,+}(\theta_k) = 1/\left(1 + e^{\left(-\theta + \sum_{j=m}^{i-1} s_i\right)/\epsilon}\right)$$

are two sigmoidal activation functions for each path segment $\varsigma_i(\theta_k)$ and ϵ is a design constant with a small value. This representation is used to ensure that a continuous local reference path is followed, and that the solver does not throw an error during gradient computation.

Algorithm 1 LMPCC Algorithm

for each time step t **do**
- $z_0 = z_{init}$
- Estimate path parameter θ_0 w.r.t closest path segment
- Select η ▷ No. of path segments needed to generate local reference
- Concatenate η path segments into continuous local reference $\overline{\boldsymbol{p}}^r(\theta_k)$
- Compute static collision free region along computed trajectory
- Get dynamic obstacles predicted pose
- Solve the optimisation problem
- Apply u* as the control input

end for

The LMPCC algorithm is summarised by Algorithm 1. Inside the for loop, we can see that the second, third and fourth steps correspond to the modified steps just covered to split the global reference into local path segments and track them in the optimisation. The remaining steps of the algorithm are similar to the original MPCC formulation given by Equation 3-5.

3-3 Robust Optimisation vs Stochastic Optimisation

There are broadly two schools of approaches for collision avoidance problems: robust optimisation and stochastic optimisation.

Robust optimisation problems aim to guarantee safety and consider all possible constraints in the optimisation problem. Robust MPC considers a worst-case scenario and ensures a safe motion for all possible circumstances. Robust optimisation assumes bounded uncertainties within which it defines safety corridors where the ego vehicle can traverse without collisions. Approaches such as [65] and [38] use a Robust Model Predictive Control model to realise real-time, safe, collision avoidance.

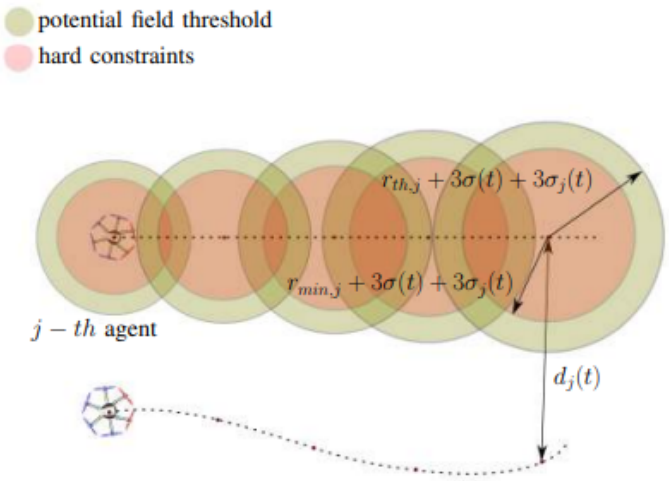


Figure 3-3: A representation of the robust obstacle avoidance constraints in [38]. The green area represents the soft constraints defined by $r_{min,j}$ while the orange area represents hard constraints enforced by $r_{th,j}$

[38] use an Extended Kalman Filter to propagate uncertainty with the collision constraint function defined as

$$\boldsymbol{G}_j(\boldsymbol{x}) = -\left\|\boldsymbol{p}(t) - \boldsymbol{p}_j(t)\right\|_2^2 + r_{\min,j}^2(t)$$

where $\mathbf{p}(t)$ and $\mathbf{p}_j(t)$ are the positions of the ego vehicle and obstacle j at time t respectively. $r_{min,j}$ represents a constant minimum distance that the ego vehicle should try and maintain from the obstacle j. If this is violated, a hard constraint $r_{th,j}$ threshold constant is enforced to avoid collisions. This constant is incorporated in the cost function. The constraints used here are non-convex, and the function $\mathbf{G}$ is smooth and continuous. An illustration of the constraints being implemented and uncertainty being propagated in the robust model is shown in Figure 3-3. d_j represents the distance between the drone and obstacle predictions. σ is the square root of the maximum eigen value of the ego's uncertainty and σ_j is the square root of the maximum eigen value of jth agent's uncertainty.

This method needs a bounded uncertainty distribution to maintain its constraint model. Even when provided with the bounded uncertainty model, the controller will produce very conservative behaviour due to the method in which the constraints are enforced. There may even be conditions where the ego vehicle might not take action.

On the other hand, stochastic optimisation designs safety constraints probabilistically (through chance constraints). The control action should guarantee that a constraint violation's chances are lower than a user-defined risk value. This formulation allows for a trade-off between safety and efficiency in an urban framework. [4] explore this trade-off between flexibility and generalisation using a CVaR(Conditional Value-at-Risk) term. Many disturbances are better represented as stochastic models rather than set-bounded ones. Less conservative actions can be taken with the addition of

a risk term, still bounded by an acceptable risk level. Directly dealing with these probabilistic constraints is intractable in real time, considering that the distribution could be shaped differently. Two methods of dealing with these are bounding them or approximating them. Bounding techniques guarantee that constraints will be satisfied, but approximation techniques often do not. However, if the bound is loose, the result can be a highly conservative solution, leading to high cost and even infeasibility [11].

Bounding them, such as in [8] with a long prediction horizon, helped the trade-off between safety and conservativeness in urban traffic scenarios. [11] bound the probability of collision that approximates the chance-constrained problem as a disjunctive convex problem which is solved using branch-and-bound techniques. This method introduces a degree of conservatism by introducing the bound and applies only to Gaussian distributions.

The scenario approach(which will be covered in detail later) generates samples of uncertain variables and ensures that the constraints are satisfied for all scenarios. This can apply to arbitrary distributions.

In the case of approximation methods, the method most commonly used is the particle control approach [9]. The disadvantages of these methods are that the complexity scales with the number of obstacles. Particle control ensures that the fraction of particles violating the constraints is at most Δ, and as the number of particles approaches ∞, the solution is exact. It can apply to arbitrary distributions, although the safety of this approach cannot be guaranteed in unknown environments. An illustration of this is shown in Figure 3-4.

The difference between particle control and scenario approaches is that particle control approximates the chance-constrained problem using samples, while the scenario methods establish a bound on the chance-constrained problem using samples.

To summarise briefly, robust optimisation attempts to provide a 100% safety guarantee for the trajectory by enforcing hard constraints on bounded uncertainties in every possible situation. While this guarantees safety, it often results in very conservative trajectories and infeasible trajectories. Stochastic optimisation attempts to model collision avoidance as a probabilistic constraint based on a risk factor. These are usually independent of the uncertainty model used, and they can be dealt with using bounds on the constraints or approximation methods to determine the safe path. By incorporating risk into the probabilistic framework, we can easily obtain a trade-off between safety and conservativeness of the resultant trajectory and integrate probabilistic noise models into the problem formulation.

3-3-1 Chance Constraints

We know real-world processes are subject to many uncertainties, such as state estimation errors, exogenous disturbances and modelling uncertainties. For example, a position and velocity estimator as a Kalman filter yields a Gaussian-distributed uncertainty in the predictions. In addition to that, the system model may not be perfect, and a vehicle could be subject to unpredictable disturbances, such as turbulence.

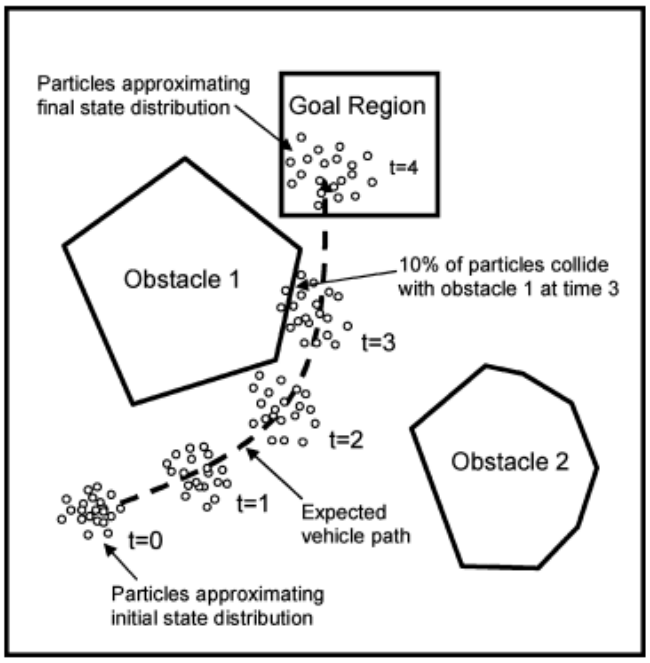


Figure 3-4: A representation of the particle control approach[9] designed such taht at most 10% of the particles fail to provide a safe approximation of the trajectory.

Due to such uncertainty, the result of an executed plan will inevitably deviate from the original plan, and hence there is a risk of certain conditions or constraints being violated. When the plan is deterministic, the optimisation plan usually pushes against constraint boundaries to find an optimum and is thus much more susceptible to risk. For example, in the case of obstacle avoidance, even a tiny perturbation to the planned path might result in a collision with obstacles. This risk can be reduced by introducing a safety margin between the path of the vehicle and obstacles, which might come at the cost of a less optimal path, but more robust to disturbances.

However, a guarantee of zero risk is usually impossible, and there is thus a non-zero probability of having a disturbance significant enough to make still the vehicle violate constraints. The risk must be accepted, but it is vital to establish a bound for this risk. The motion planner should be able to provide a guarantee that the system can consistently operate within these bounds. These constraints are called chance constraints [59].

In the case of collision avoidance, a typical chance constraint might take the form of Equation 3-8.

$$\mathbb{P}_k \left[\left\| \boldsymbol{x}_k^r - \boldsymbol{x}_k^o \right\|_2 > r \right] \geq 1 - \epsilon_k, \forall k \tag{3-8}$$

Here, $\mathbf{x}_k^r$ and $\mathbf{x}_k^o$ are the states of the robot and obstacle respectively at time step k, ϵ_k is the risk threshold at k and r is some distance threshold. This Equation reads out as: The probability of the distance between the robot and the obstacle at time step k being greater than r should be at least $1-\epsilon$ for all k time steps. Thus this equation and

chance constraints in motion planning aim to represent a bound on the risk or safety of constraints.

3-4 Scenario Approach

Optimisation-based methods use constraints in their formulation to avoid collisions. Classical methods of imposing these constraints consider the obstacle predictions deterministic(do not account for uncertainties). In the case of uncertainties, deterministic frameworks are very likely to be violated since they do not consider the distribution of uncertainties. Uncertainties can be added to deterministic predictions using robust optimisation. The acceptable risk can be set to zero if the probability density function is non-zero in a bounded domain of the robot's workspace and is zero elsewhere. However, the assumption of a bounded distribution may be limiting, and it could be conservative in the case of a large support domain.

In the case of unbounded uncertainties, stochastic optimisation in the form of chance constraints allows for constraints of the collision probability below a certain risk level. The chance of collisions for each step can also be constrained separately. More on chance constraints has previously been discussed in Section 3-3-1.

Directly evaluating these constraints is intractable, especially considering that the distribution could be arbitrarily shaped. Thus, they are often approximated or bounded. Due to the sample efficiency and safety guarantees, bound-based methods receive the most attention. Scenarios are one such method, where sampling points from a continuous distribution and bounding the risk with these samples results in a problem that ensures safety and real-time tractability. It would otherwise be infeasible to guarantee safety bounds for a continuous arbitrary uncertainty distribution and ensure that it has a real-time computational latency. The scenario approach ensures that we retain the features of the distribution even after discretisation, bound the risk and use it in real time.

The scenario approach is a stochastic optimization method, constraining the marginal risk of collision at each point using chance constraints. Positions of the dynamic obstacles are sampled from the chance constraints, and each of these samples represents a scenario for collision constraints. This approach uses individual samples for the collision constraints and is thus unaffected by the underlying uncertainty distribution. This method is thus flexible in dealing with both Gaussian and non-Gaussian uncertainty.

The scenario approach can be described as a model that chooses the best solution x^*(Figure 3-5), which works well for multiple scenarios $\left(\delta^{(1)} \ldots . .\delta^{(N)}\right)$, while optimizing the objective [14]. The risk of the solution depends on how many scenarios are subject to optimisation. Naturally, more scenarios might ensure less risk and thus greater safety, but the computation would be slow and infeasible for time-constrained applications. On the other hand, with fewer scenarios, we might get faster optimisation and more feasible solutions, but the risk threshold might be pushed up. Thus, there is a trade-off between the feasibility and safety of the solution. The scenario approach makes use of

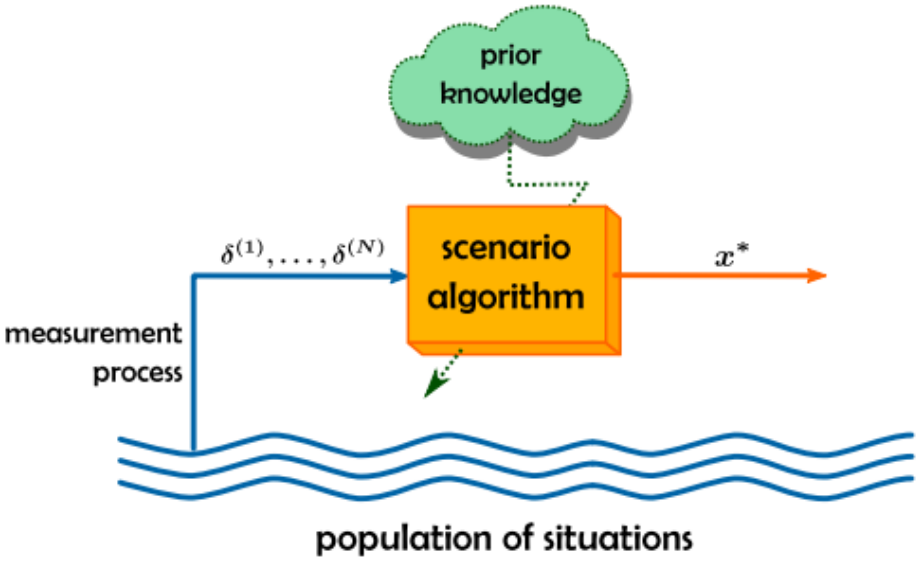


Figure 3-5: A scenario algorithm is a way to generate decisions able to cope with uncertainty. The picture shows in yellow the scenario box. It is constructed on the ground of prior knowledge about the problem at hand and is fed by a sample of situations (scenarios) elicited from a (typically infinite) population of situations. [14]

a wait-and-judge [16] quality evaluation of its solutions, where the decision maker can make the following decisions based on the quality of the result:

- Use or Discard the solution.
- Re-calibrate the formulation of the optimisation in order to be provided with a better solution.

We can now look at how the risk is bounded using the scenario approach, first for the convex optimisation case, then the non-convex optimisation case, and finally, in the case of the non-convex, non-linear optimisation applied to automated driving.

3-4-1 Convex Optimisation with Scenarios

In [15], the authors have attempted to demonstrate the feasibility of different uncertain convex optimisations. Uncertain convex optimisations are situations where constraints are known imprecisely. N constraints are randomly extracted from the uncertainty distribution, and the constraint satisfaction for these constraints is used as an alternative representation. The optimisation problem is thus:

$$\begin{aligned} &\min_{x \in \mathcal{X} \subseteq R^d} c^T x \\ &\text{s.t., } x \in \cap_{i \in (1,\ldots,N)} \mathcal{X}_{\delta^{(i)}} \end{aligned} \tag{3-9}$$

where R^d represents the dimensionality of x, c is a constant vector of weights, $\mathcal{X}$ and $\mathcal{X}_{\delta^{(i)}}$ represent the set of values of x and the set of scenarios from which x can be drawn respectively. In this problem formulation, a finite number of constraints N are checked, thus significantly reducing the computational optimisation cost if N is not too large. The result in the paper, help to define the violation probability of the method, based on N. Violation probability is defined as: $V(x) = \mathbb{P}\{\delta \in \Delta : x \notin \mathcal{X}_\delta\}$.

In Equation 3-9, due to the the scenarios being picked randomly, the solution of the equation x_N^* is a random variable and thus the violation probability $V(x_N^*) \leq \epsilon$ for some cases, while $V(x_N^*) > \epsilon$ for others, where ϵ is the risk value. The authors then attempt to establish a confidence with which this probability can have an upper risk bound of ϵ. Certain terminologies are defined:

- Support Constraint - These are constraints whose removal affects the value of the optimisation problem. Constraint $\delta^{(r)}, r \in \{1, \ldots, N\}$, is a support constraint for P_N if its removal changes the solution of P_N. P_N here is the scenario problem(Equation 3-9) with N extractions.
- Fully Supported Problem- These are problems where the number of support constraints is equal to d, the dimensionality of the problem.
- Non-Fully Supported Problem- These are the class of problems where the number of support constraints is less than or equal to d.

In the case of a fully supported problem, the probability of violation:

$$\mathbb{P}^N\{V(x_N^*) > \epsilon\} = \sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \tag{3-10}$$

In the case of a non-fully supported problem, this probability is an upper bound, and the violation probability is bounded:

$$\mathbb{P}^N\{V(x_N^*) > \epsilon\} \leq \sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \tag{3-11}$$

Thus, for general convex problems, the confidence with which the violation probability is less than ϵ is given by Equation 3-11.

3-4-2 Non-Convex Optimisation with Scenarios

Most real-world problems are non-convex. The scenario problem was recently extended to incorporate non-convex optimisation problems [17]. It is suggested that a generalisation guarantee of the problem is to be found only after a solution, or a-posteriori, depending on the length of the support subsample. The optimization problem considered here is:

$$\begin{array}{ll} \min_{\theta \in \Theta} & f(\theta) \\ \text{s.t.} & \theta \in \Theta_{\delta^{(i)}}, \text{ for all } i = 1, \ldots, N. \end{array} \tag{3-12}$$

where Θ and Θ_δ represent the set of values of θ and the set of scenarios from which θ can be drawn respectively.

When $f(\theta), \Theta$ and Θ_δ are convex, the problem takes the same form as the problem in the previous section. The problem here is, however, much more general since no assumptions are made on $f(\theta), \Theta$ and Θ_δ. The assumption made in [15] is that the number of support constraints in a convex optimisation problem with d optimisation variables never exceeds d. In a non-convex optimisation problem, this does not hold true. In non-convex optimisation problems, the removal of any support constraint might generate a new optimisation problem.

f(θ) is the non-convex objective function to be optimised. If it assumed that a unique solution(or a scenario decision) to Equation 3-12 exists (θ_N^*), then the Equation defines a mapping function $\mathcal{A}_N : \Delta_N \mapsto \Theta$ between θ_N^* and the picked scenarios $\left(\delta^{(1)}, \ldots, \delta^{(N)}\right)$, i.e. $\theta_N^* = \mathcal{A}_N\left(\delta^{(1)}, \ldots, \delta^{(N)}\right)$. N scenarios are picked during the optimisation. The objective of the paper is to find out how robust to unseen scenarios θ_N^* is. This can be defined by the violation probability, given by: $\mathrm{V}(\theta) = \mathrm{P}\left\{\delta \in \Delta : \theta \notin \Theta_\delta\right\}$. If $V(\theta) \leq \epsilon$, where $\epsilon \in (0, 1)$, the solution is ϵ - feasible ϵ - robust. The authors find the number of samples required to find a confidence bound (1- β), such that $V\left(\theta_N^*\right) \leq \epsilon$.

Some important terminologies used are:

- Support Subsample- With given samples $\left(\delta^{(1)}, \ldots, \delta^{(N)}\right) \in \Delta^N$, a support sample S is a k-tuple elements extracted from the sample(S= $\left(\delta^{(i_1)}, \ldots, \delta^{(i_k)}\right)$, where $i_1 < i_2 < \ldots < i_k$), which gives the same solution as the original sample. i.e. $\mathcal{A}_k\left(\delta^{(i_1)}, \ldots, \delta^{(i_k)}\right) = \mathcal{A}_N\left(\delta^{(1)}, \ldots, \delta^{(N)}\right)$.
- A support subsample S= $\left(\delta^{(i_1)}, \ldots, \delta^{(i_k)}\right)$ is irreducible if no element can be removed from it, giving the same solution.

The assumption made in the convex case that the support length is lesser than or equal to the problem's dimensionality does not hold in the non-convex case. The goal is, therefore, to find the minimal-length support subsample.

Let $\mathcal{B}_N : \left(\delta^{(1)}, \ldots, \delta^{(N)}\right) \mapsto i_1, \ldots, i_k$ where $i_1 < i_2 < \ldots < i_k$. $\mathcal{B}_N$ is the function that finds the support subsample from a given sample. Let $s_N^* := \left|\mathcal{B}_N\left(\delta^{(1)}, \ldots, \delta^{(N)}\right)\right|$ be the cardinality of $\mathcal{B}_N$(length of the support subsample $\left(\delta^{(i_1)}, \ldots, \delta^{(i_k)}\right)$). Since $\mathcal{B}_N\left(\delta^{(1)}, \ldots, \delta^{(N)}\right)$ is a random variable over the uncertainty distribution Δ^N, so is s_N^*. It is shown that for a confidence (β), risk $\epsilon \in [0, 1]$ is a function where $\epsilon(N) = 1$ and:

$$\sum_{k=0}^{N-1} \left(\begin{array}{c} N \\ k \end{array} \right) (1 - \varepsilon(k))^{N-k} = \beta \tag{3-13}$$

and the confidence bound of the violation probability for any $\mathcal{A}_N, \mathcal{B}_N$ and probability P is given by:

$$\mathrm{P}^N\left\{\ \mathrm{V}\left(\theta_N^*\right) > \varepsilon\left(s_N^*\right)\right\} \leq \beta.$$

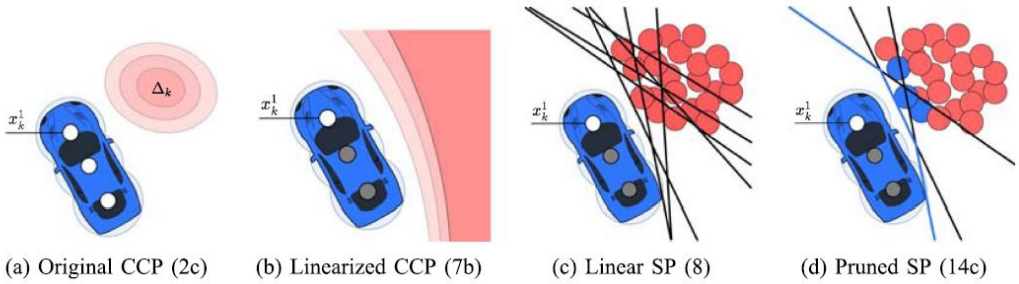


Figure 3-6: Transition of the chance-constrained problem from its original form(a) to a linearized form(b) to a sampled form of the linearization(c) to a pruned sampled form of the linearization(d). Robot is in blue, and the obstacle is in red. [24]

Then the length of the support subsample$(s_N^*) < \bar{s}$, and the value of risk $\epsilon(k)$ are:

$$\varepsilon(k) := \begin{cases} 1 & \text{if } k \geq \bar{s} \\ 1 - \sqrt[N-k]{\frac{\beta}{\bar{s}\binom{N}{k}}} & \text{otherwise.} \end{cases} \tag{3-14}$$

The authors thus establish a scenario approach for non-convex optimisation programs. [24] use these theorems in automated driving, where positions of the dynamic obstacles are used as scenarios for optimisation. This is discussed in the subsequent section.

3-5 Scenario-based Model Predictive Contouring Control

The work in this thesis is based on De Groot et al. [24], who propose a scenario-based trajectory optimisation method to deal with dynamic uncertainties of obstacles. The method does not consider any description of the obstacle's uncertainty, thus generalising the problem. Chance constraints are used to bound the marginal risk of collisions [10] [84]. The probabilistic chance constraints are first linearised and then sampled to generate scenarios. A scenario represents the probabilistic chance constraint for collision with a dynamic obstacle at the sampling point. However, most of these scenarios may be computed but never achieved, so the paper proposes a method to prune scenarios before optimisation while maintaining safety guarantees. A scenario optimisation problem can handle arbitrary uncertainty distributions. This implementation is applied to a Local Model Predictive Contouring Control(MPCC) [12] framework, which was discussed previously. This Scenario-MPCC or S-MPCC is built on a non-convex optimisation framework [17] where the predicted risk of motion planning is obtained pre-optimisation (unlike in [17]) by using the geometry of the problem formulation, leading to a reduced

and more efficient evaluation of samples. The optimisation problem here is defined as:

$$\begin{aligned} &\min_{\boldsymbol{u}\in\mathbb{U}} \sum_{k=1}^{N} J\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \\ &\text{s.t. } \boldsymbol{x}_{k+1} = f\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \boldsymbol{x} \in \mathbb{X} \\ &\qquad \boldsymbol{A}_k^T\left(\delta_k^i, \hat{\boldsymbol{x}}_k\right)\boldsymbol{x}_k \leq b_k\left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right), i \in \mathcal{H}_k. \end{aligned} \tag{3-15}$$

where the last line represent the linear inequality constraints which also include halfspaces. $\mathcal{H}_k$ is a subset of halfspace indices defined as

$$\mathcal{H}_k := \left\{ i \mid \exists \boldsymbol{x}_k \in \mathcal{P}_k, \boldsymbol{A}_k^T\left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right)\boldsymbol{x}_k = b_k\left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right)\right\}$$

where

$$A_k = \frac{\delta_k - \hat{x}_k}{\|\delta_k - \hat{x}_k\|}, \quad b_k = A_k^T\left(\delta_k - A_k r\right)$$
$$\mathbb{P}_k\left[\boldsymbol{A}_k^T\boldsymbol{x}_k \leq b_k\right] \geq 1 - \epsilon_k, \forall k, \delta_k \in \Delta_k$$

At stage k, S_k(sample size) number of scenarios are sampled from the uncertainty distribution, and δ_k is the sampled position of one of the S_k scenarios for one of the obstacles. $\hat{x}_k$ are the future predicted poses of the robot from the previous optimisation step, and r is the combined radius of the car and the obstacle discs. A_k and b_k are the halfspace parameters constructed with respect to this particular scenario. Only the indices of the halfspaces which span the boundary of the polytope $\mathcal{P}_k$ are taken into account as shown by $\mathcal{H}_k$, thus reducing the size of the optimisation problem. This set contains active indices during the optimisation, thus placing a bound on the support subsamples(by the cardinality $s_k^* \leq |\mathcal{H}_k|$). ϵ is the risk factor associated with each halfspace constructed. The line $\mathbb{P}_k\left[\boldsymbol{A}_k^T\boldsymbol{x}_k \leq b_k\right] \geq 1 - \epsilon_k, \forall k, \delta_k \in \Delta_k$ represents the linearised chance constraints based on the the normalised distance between the ego vehicle and the sampled scenarios of each obstacle. This means that the obstacle avoidance risk is represented probabilistically as a set of linear halfspaces for all scenarios sampled from an uncertainty distribution.

Algorithm 2 S-MPCC Algorithm [24]

```
Compute Number of Scenarios S from epsilon safety function, and lower bound
for t=1,2,.. do
    Get uncertainties from perception module
    for k=1 to N do
        Sample S times from the uncertainty region
        Compute halfspace parameters for each sampled obstacle position
        Construct halfspaces and determine safety region as polytopes
    end for
    Solve optimisation problem over horizon
    Apply first input u_1.
end for
```

This implementation is competitive in terms of computation times with state-of-the-art planning methods while also applicable to generic uncertainties, thus making it

a very suitable prospect for hardware implementation. This paper has implemented this on hardware with pedestrians modelled as the dynamic obstacles, with both Gaussian and non-Gaussian uncertainties. There is not much work in non-convex scenario optimization, but [17] and [34] are helpful resources. Generally, sampling-based chance-constrained approaches are considered intractable for real-time motion planning; this method can incorporate it, although with a higher processing cost. The sampling period used for this implementation is 10ms. In the case of Gaussian uncertainties, S-MPCC is very effective in preventing collisions and competing with other algorithms. In the case of non-Gaussian uncertainties, the risk threshold can be maintained at the cost of increasing the computation times. This is because, in the case of non-Gaussian uncertainties, an increased number of samples might be required to replicate the distribution features and ensure the same risk levels as in the case of Gaussian uncertainties. This increased number of scenarios leads to extra computational load and, thus, higher computation times.

3-5-1 Bounding the Risk in S-MPCC

The work proposes a few theorems to bound the risk and also prune samples in order to have a tractable solution in real-time. Theorem 1 of [24] proves that the probability of the optimisation solution violating the chance constraint at a stage k is:

$$\mathbb{P}_k^{S_k}\left[V_k\left(u_{SP}^*\right) > \epsilon_k\left(s_k^*\right)\right] \leqslant \beta_k\left(S_k\right) \tag{3-16}$$

where u_{SP}^* is the solution of the optimisation problem Equation 3-15, s_k^* is the smallest support subsample size and β is the confidence, defined by:

$$\beta_k\left(S_k\right) := \sum_{s=0}^{S_k-1} \left(\begin{array}{c} S_k \\ s \end{array} \right) \left[1-\epsilon_k(s)\right]^{S_k-s} \tag{3-17}$$

Regarding scenario pruning, the work explains that not all the scenarios need to be considered in order to define the free space. The convex collision-free region is found by the intersection of convex half-spaces, which are defined by their indices:

$$\mathcal{H}_k := \left\{ i \mid \exists x_k \in \mathcal{P}_k, A_k^T\left(\delta_k^i, \hat{x}_k\right) x_k = b_k\left(\delta_k^i, \hat{x}_k\right) \right\} \tag{3-18}$$

This means that only the half-spaces spanning the polytope contribute to the optimization problem. Discarding the remaining scenarios increases the time efficiency of the approach.

Theorem 2 of the work formalized the risk of the solution when R of the original S scenarios are discarded, leaving P = S - R, the remaining scenarios. Consider,

$$\beta(S,P) = \left(\begin{array}{c} S \\ P \end{array} \right) \sum_{s=0}^{P-1} \left(\begin{array}{c} P \\ s \end{array} \right) \left[1-\epsilon(s)\right]^{P-s}, \tag{3-19}$$

where ϵ(s) is a function such that ϵ(P) = 1. The probability of violation satisfies:

$$\mathbb{P}^S\left[V\left(u_{SP}^*\right) > \epsilon\left(s^*\right)\right] \leqslant \beta(S,P) \tag{3-20}$$

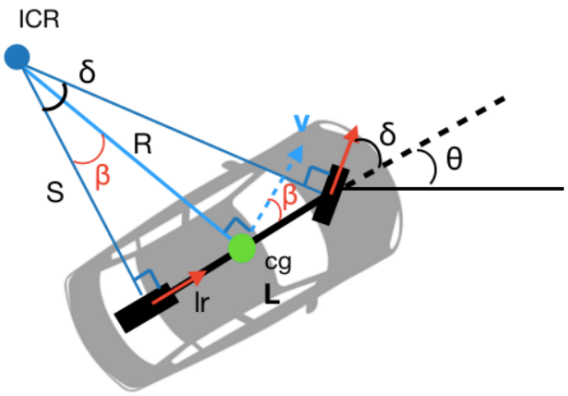


Figure 3-7: 2^{nd} order bicycle model, where it is assumed that l_r(distance between the rear wheel and the center of gravity cg) is known. Here $L = l_r + l_f$, S=$L/\tan(\delta)$, R=$S/\cos(\beta)$ and $\theta = \psi$ when comparing this model with Equation 3-21. ICR is the Instantaneous Center of Rotation. [23]

We can conclude from this section that using the scenario approach in the case of dynamic obstacle avoidance for local motion planning is a stochastic approach agnostic to the uncertainty type or distribution of the obstacle, unlike LMPCC [12].

Another positive point of this approach is its real-time feasibility due to its pruning of constraints and the formal definition of risk bounds with confidence parameters.

3-6 Technical Specifications

Some of the important technical specifications contributing to this implementation are covered below.

3-6-1 Vehicle Dynamics

For the dynamic model and predictions in the optimisation solver, a 2^{nd} order bicycle model is used [12]. This model has two inputs(acceleration a and angular velocity w) and six states(x-coordinate x, y-coordinate y, yaw ψ, velocity v, steering angle δ, and spline state s). The continuous-time state space model is given by Equation 3-21.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \\ \dot{\delta} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} v\cos(\psi+\beta) \\ v\sin(\psi+\beta) \\ \frac{v}{l_r}\sin(\beta) \\ a \\ w \\ v \end{bmatrix} \tag{3-21}$$

where

$$\beta = \arctan(\frac{l_r}{l_r + l_f}\tan(\delta))$$

where l_r and l_f are the distances from the Center of Mass(COM) of the vehicle to the rear and front respectively. The dynamic effect of steering can be seen through the representation of β(the slip angle) and its effect on the position and orientation indirectly. Figure 3-7 can be used as a reference to visualise the physical interpretation of these dynamics. The integrator step size used is 0.2s, which is thus the discretisation and correspondingly the model prediction update time step.

3-6-2 Objective Function

The objective function is the function to be optimised by the solver. The objective function used here as shown in Equation 3-22 is a function of both the inputs, the velocity and the contour and lag error approximations which are orthogonal and tangential error in the MPCC framework. [48]. Further details of these two error terms have been previously covered in Section 3-2.

$$\min \sum_{k=1}^{N} W_c e_c^2 + W_l e_l^2 + W_v \frac{(v - v_{ref})^2}{v_{ub} - v_{lb}} + W_a \frac{a^2}{a_{ub} - a_{lb}} + W_w \frac{w^2}{w_{ub} - w_{lb}} \tag{3-22}$$

where W_c, W_l, W_v, W_a, W_w are the weights on the contouring error, lag error, velocity error, acceleration and angular velocity respectively. The subscript 'ub' and 'lb' represent pre-defined upper bounds and lower bounds respectively. This is a non-linear objective function, although it might appear quadratic on first glance. This is due to the non-linear definition of the lag and contouring errors in terms of the spline-parameterised reference path, which is a decision variable in the MPCC formulation.

3-6-3 Constraints

The constraints to this problem(aside from the non-linear dynamic model) are only linear inequality constraints of the form $A_k^T x \leq b_k$ whose definition has been covered by Equation 3-23.

$$\boldsymbol{A}_k^T \left(\delta_k^i, \hat{\boldsymbol{x}}_k\right) \boldsymbol{x}_k \leq b_k \left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right), i \in \mathcal{H}_k. \tag{3-23}$$

where $\mathcal{H}_k$ are the set of boundary halfspaces defined as

$$\mathcal{H}_k := \left\{ i \mid \exists \boldsymbol{x}_k \in \mathcal{P}_k, \boldsymbol{A}_k^T \left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right) \boldsymbol{x}_k = b_k \left(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k\right) \right\},$$

$$A_k = \frac{\delta_k - \hat{x}_k}{\|\delta_k - \hat{x}_k\|}, \quad b_k = A_k^T \left(\delta_k - A_k r\right)$$

Here δ_k is the position of one of the S scenarios for one of the obstacles, $\hat{x}_k$ is the future predicted poses of the robot from the previous optimisation step, r is the combined radius of the car discs and the obstacle discs. A_k and b_k are the half space parameters constructed with respect to this particular scenario. Only the line segments(and not the interiors) of the halfspaces are taken into account as shown by $\mathcal{H}_k$.

Chapter 4

Proposed Approach and Methodology

Now that a literature review and mathematical background has been established, a formal proposal for this thesis is made.

4-1 From Pedestrians to Vehicles

The work in [24] proposed using a scenario approach to deal with dynamic uncertainties in the form of pedestrians crossing a road in an urban environment. The method showed promising results regarding collision probabilities, the time to completion, and the ability to handle different probabilistic uncertainty distributions.

Table 4-1: Differences between dynamic pedestrians and vehicles in an urban setting

Pedestrians	**Vehicles**
Modelled with small radii	Larger radii required for model
Lesser deviation in position uncertainty standard deviations	Greater deviation in position uncertainty standard deviations
Comparatively slower motion	Faster motion
Motions in most cases linear motion like walking on sidewalks or crossing the road	Complex set of motions like left and right turns in addition to straight motion
Size of disc to be modelled is fixed	Uncertain sizes (trucks, sedans, hatchbacks) should be considered
Relatively less complex motions could be generalised as unimodal or bi-modal in most cases	Different sets of complex possible decisions render uncertainties to be multimodal

This thesis aims to extend the work in [24] from dealing with pedestrians to dealing with vehicles. In any urban setting, with different road configurations and widths, the issue of dynamic obstacles in the form of vehicles is as crucial a challenge to deal with as pedestrians, if not more.

We now briefly assess the differences and difficulties posed when modelling vehicles compared to pedestrians(Table 4-1). From the table, we can conclude that a different representation is required for a vehicle compared to a pedestrian. This thesis will use the extended pedestrian model for vehicles as a baseline for comparison with the proposed approach.

There are three different approaches through which we can model the uncertainties of other vehicles:

- Stick to the single disc representation. Keeping the disc small under-approximates the vehicle and can lead to unsafe solutions. A large disc leads to over-approximation of the vehicle shape and, thus, an overly conservative and often infeasible solution.

- Modelling the vehicle as a series of n pedestrian obstacle discs(or larger) linked together by the vehicle geometry, and each disc has its uncertainty distribution and collision region

- Modelling the vehicle as a single disc with n uncertainty regions linked by vehicle geometry but having a single collision region.

At this point, the question of why the choice to use discs as a model and not another shape(e.g. ellipse, rectangle, which might be a better approximation of the vehicle shape). There are multiple reasons for this:

- In terms of speedy optimisation, a rectangle is not an ideal choice due to its non-differentiable shape and difficulty in deciding how to construct a halfspace appropriately due to the shape and orientation.

- An ellipse provides a better approximation than a rectangle in terms of its continuity and halfspace construction. However, its orientation dependence makes it computationally relatively inefficient. Every point on an ellipse has a different radius relative to its neighbourhood points, making distance-based pruning out of scenarios or halfspaces a more tricky and computationally demanding task. In addition, an elliptical vehicle model would lead to different constraints based on its orientation, resulting in computational inefficiency relative to a disc. In addition, a series of discs is a better approximation(it covers more vehicle area) than an ellipse.

- Using discs removes the dependence on orientation when constructing the constraints and polygons since it is symmetric in all directions. In addition, discs cover more area and are the ideal candidate to model obstacle vehicles in this scenario-based implementation.

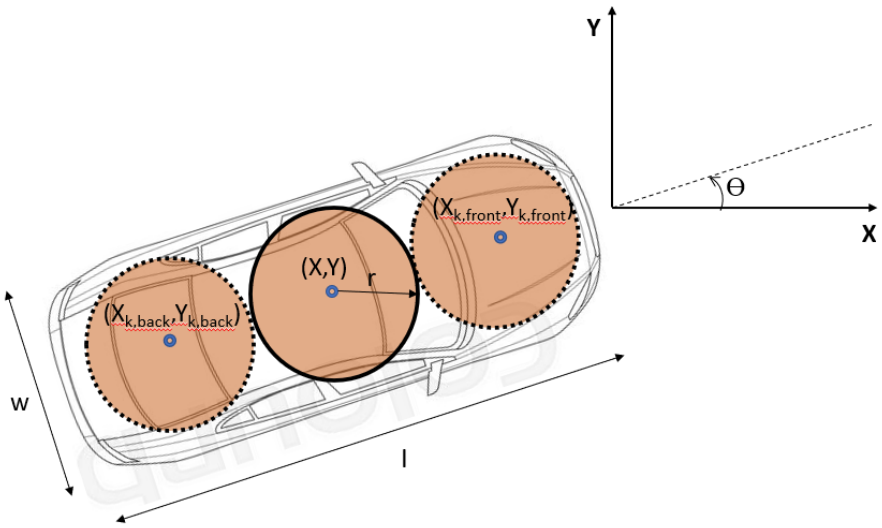


Figure 4-1: A car with 3 discs(n=3). $(X_{k,front}, Y_{k,front})$ and $(X_{k,back}, Y_{k,back})$ represent the k^{th} non-overlapping discs at the front and back of the center disc(X,Y) respectively.

4-2 Mathematical Extension

4-2-1 n-Disc Representation

Referring to a generalised representation of vehicle geometry like in Figure 4-1, we can conjure up a generalised equation to find the centre of each of the n discs that are used to represent the vehicle or the scenarios(the mathematical transformation remains the same).

Assume that the car has length l, width w and is oriented at an angle θ with respect to the global coordinate frame. If each disc has radius r, and if we want our car to have a total of n discs((n-1)/2 discs behind the centre disc and (n-1)/2 discs in front of the centre), and let X, Y be the x-y coordinate of the centre disc, the x-y coordinate of the k^{th} disc behind the car is

$$X_{k,back} = X - 2kr\cos\theta \tag{4-1}$$

$$Y_{k,back} = Y - 2kr\sin\theta$$

and the k^{th} disc at the front of the car is

$$X_{k,front} = X + 2kr\cos\theta \tag{4-2}$$

$$Y_{k,front} = Y + 2kr\sin\theta$$

where it is assumed that $l > w > r$ and that the radius r is selected. With r selected, the maximum number of discs that can be taken in front(or the back, due to the symmetry of the problem) of the car,

$$\frac{n-1}{2} = \frac{l-2r}{4r}$$

and thus the total number of discs that can be taken(including the center disc) are:

$$n = \frac{l}{2r} \tag{4-3}$$

As an example, for a car with length=3m, width=1m and r=0.5m, we can have 3 discs, one at the back, one at the center, one at front. Both the approaches taken in this thesis involve linear transformations based on car geometry at different stages of the SMPCC algorithm.

4-2-2 Constraints Extension

The generalised chance constraints are given by

$$\mathbb{P}_k \left[\left\| x_k^d - \delta_k^v \right\|_2 > r, \forall d, v \right] \geq 1 - \epsilon_k, \forall k \tag{4-4}$$

$\delta_k^v \in \Delta_k^v$ is the uncertain position of obstacle v at stage k and the r is the combination of the vehicle and obstacle radii. This chance constraint represents the risk level ϵ_k bound on the probability of collisions in between collision circles d of the vehicle and the collision circle of each dynamic obstacle v at the future prediction step k. The probability measure $\mathbb{P}_k$ is the modeled uncertainty.

[24] use the nonconvex scenario optimisation framework of [17] to solve this problem. In this extension, Equation 4-4 would be re-written as:

$$\mathbb{P}_k \left[\left\| x_k^d - a\delta_k^v - b \right\|_2 > r, \forall d, v \right] \geq 1 - \epsilon_k, \forall k \tag{4-5}$$

where a and b are constants representing the linear displacement of the position of the vehicle v at stage k.

[17] established a link between a Chance Constrained Problem and a Scenario Program enabling the conversion of the non-convex constraints from the form

$$\mathbb{P}[g(u, \delta) \leq 0] \geq 1 - \epsilon, \delta \in \Delta$$

to the form

$$g\left(u, \delta^i\right) \leq 0, \delta^i \in \Delta, \forall i \in \mathcal{S}$$

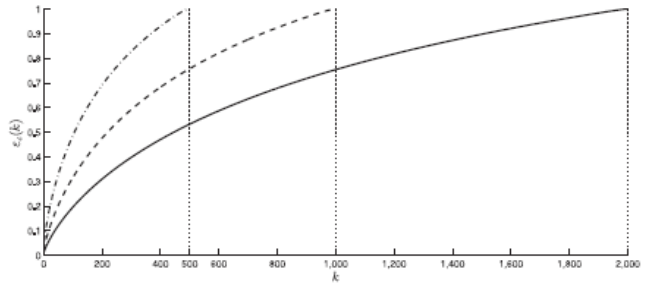
[24] use this property to convert linearized chance constraints to a linear halfspace. In this extension, this would look something like

$$\begin{aligned} &\tilde{A}_k = \frac{a\delta_k + b - \hat{x}_k}{\|a\delta_k + b - \hat{x}_k\|}, \quad \tilde{b_k} = \tilde{A_k}^T \left(a\delta_k + b - \tilde{A}_k r\right) \\ &\mathbb{P}_k \left[\tilde{A_k}^T x_k \leq \tilde{b_k} \right] \geq 1 - \epsilon_k, \forall k, \delta_k \in \Delta_k \end{aligned} \tag{4-6}$$

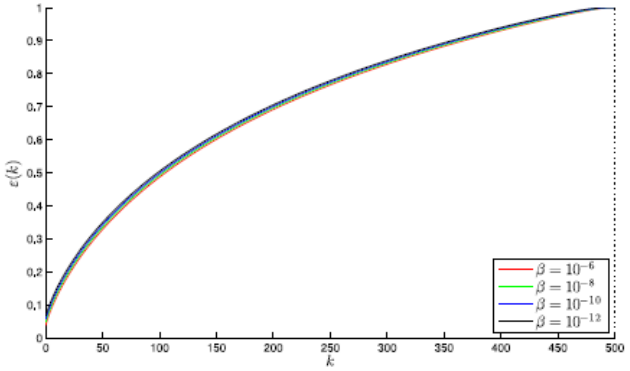
We can convert this linear chance constraint problem to a linear halfspace

$$\tilde{\boldsymbol{A}_k}^T \left(a\delta_k^i + b, \hat{\boldsymbol{x}}_k\right) x_k \leq \tilde{b_k} \left(a\delta_k^i + b, \hat{x}_k\right), \forall i \in \mathcal{S}_k, \forall k \tag{4-7}$$

This problem is of the same form as the constraints in [24], and do not affect the linearity or convexity of the optimisation problem.



(a) A plot of $\epsilon(k)$ for N=500, N=1000 and N=2000, when $\beta = 10^{-6}$



(b) A plot of $\epsilon(k)$ for N=500 and $\beta = 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12}$

Figure 4-2: Dependence of Risk $\epsilon(k)$ on different parameters. [17]

4-2-3 Safety Guarantee

The free space defined by the subset of scenarios is defined by

$$\mathcal{H}_k := \left\{ i \mid \exists x_k \in \mathcal{P}_k, A_k^T \left(\delta_k^i, \hat{x}_k \right) x_k = b_k \left(\delta_k^i, \hat{x}_k \right) \right\} \tag{4-8}$$

which in this case will be

$$\tilde{\mathcal{H}_k} := \left\{ i \mid \exists x_k \in \mathcal{P}_k, \tilde{A_k}^T \left(a\delta_k^i + b, \hat{x}_k \right) x_k = \tilde{b_k} \left(a\delta_k^i + b, \hat{x}_k \right) \right\} \tag{4-9}$$

This represents only halfspaces bordering the free-space to be be involved in the polytope construction, and also represents an upper bound $\bar{s}$ on the cardinality of $\mathcal{H}_k$, hence bounding the support samples.

Taking a look the previously defined safety guarantee,

$$\beta_k \left(S_k \right) := \sum_{s=0}^{S_k - 1} \left(\begin{array}{c} S_k \\ s \end{array} \right) \left[1 - \epsilon_k(s) \right]^{S_k - s} \tag{4-10}$$

For this extension, instead of S_k scenarios, we now have nS_k scenarios(S_k scenarios corresponding to each disc). If we interpret the vehicle as a single body, we now

represent it with nS_k scenarios, and the support subsample would also be ns^*. Thus, our new confidence function is now:

$$\beta_k\left(nS_k\right) := \sum_{s=0}^{nS_k-1} \left(\begin{array}{c} nS_k \\ s \end{array} \right) \left[1-\epsilon_k(s)\right]^{nS_k-s} \tag{4-11}$$

and thus our risk violation probability is now bounded by $\beta(nS_k)$:

$$\mathbb{P}_k^{nS_k}\left[V_k\left(u_{SP}^*\right) > \epsilon_k\left(ns_k^*\right)\right] \leqslant \beta_k\left(nS_k\right) \tag{4-12}$$

Referring to Figures 4-2, we analyse the trends in the risk with respect to changes in S_k and the confidence parameter β. It can be seen that there is a very weak dependence on $\epsilon(k)$ by β, and thus we can assume that the value of $\beta(nS_k)$ remains similar with the change in the number of scenarios. On the other hand, there is a rich coupling between the risk function and the number of scenarios(Equation 4-13). As expected, the risk function has a smoother increase over the stages k, for larger values of S_k. Physically this means that the better we approximate the uncertainty distribution, the less is the risk of the constraints being violated. As we now have nS_k scenarios, the risk function takes the appearance of

$$\varepsilon(k) := \begin{cases} 1 & \text{if } k \geq \bar{s} \\ 1 - \sqrt[nS_k-k]{\dfrac{\beta}{\bar{s}\left(\begin{array}{c} nS_k \\ k \end{array} \right)}} & \text{otherwise.} \end{cases} \tag{4-13}$$

Thus we have a smoother risk function and similar confidence in constraint violation probability with the n-Disc model for an obstacle vehicle.

The second theorem which formalized the risk of the solution when some of the original scenarios are discarded continues to stay. Considering,

$$\beta(nS,P) = \left(\begin{array}{c} nS \\ P \end{array} \right) \sum_{s=0}^{P-1} \left(\begin{array}{c} P \\ s \end{array} \right) \left[1-\epsilon(s)\right]^{P-s}, \tag{4-14}$$

where ϵ(s) is a function with ϵ(P) = 1. The probability of violation continues to satisfy:

$$\mathbb{P}^S\left[V\left(u_{SP}^*\right) > \epsilon\left(ns^*\right)\right] \leqslant \beta(nS,P) \tag{4-15}$$

From this entire section, we can summarise that:

- A n-Disc representation is proposed to model the obstacle vehicle.
- The discs are placed(translated) in a non-overlapping fashion, taking into account the geometry of the vehicle.
- The translation of these discs does not affect the nature of the constraints correspondingly produced, neither does it add any non-linear factor into any of the computations with respect to the linearisations, chance constraints or polygon construction.
- The number of samples now used are n times what was previously used per obstacle. This renders the risk function to be smoother over time, and maintains a very similar confidence bound as was the case for a single disc.

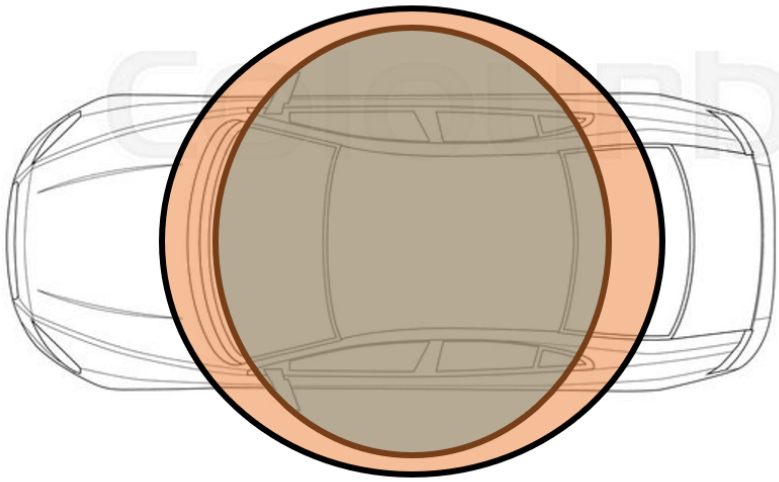


Figure 4-3: Vehicle as a single expanded disc. The Blue disc represents the vehicle model, while the Orange ellipsoid is representative of the uncertainty distribution from which scenarios are sampled. This configuration has one collision region, one corresponding to the single disc.

4-3 Analytical Interpretation

Having proposed three different methods to model the scenario-based implementation, we can go in-depth into each of these approaches and understand what they mean for the algorithm.

4-3-1 Vehicle as a Single Expanded Disc

In terms of implementation, this is identical to the work in [24], with the minor difference being that the radius of the disc is made much bigger to represent a vehicle better. An illustration of what this would look like is shown in Figure 4-3.

The explanation for this shape is: to be neither too conservative nor liberal with the radius of the disc.

- If the disc radius is too conservative, and the disc is confined to the limits of the car at the centre of the car, the front and the back region of the car are not considered as a part of the car at all, leaving them more prone to collision, as they are not considered in the optimisation problem.
- If the disc modelling in the car is made so large as to cover the entire car, it will over-extend well beyond the car's sides. One would think this will result in a very safe solution in terms of collisions. While that is true, the resulting circle would project so much beyond the ego vehicle that it would result in several infeasible optimisations since there would be no feasible trajectory for the obstacles without violating safety constraints.

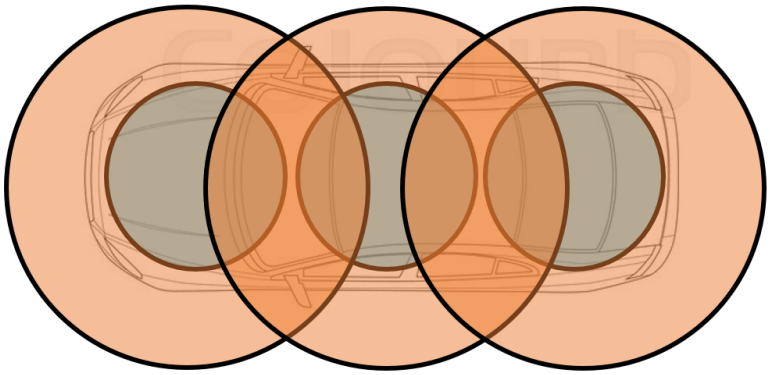


Figure 4-4: Vehicle as a series of linked discs. The Blue discs collectively represent the vehicle, while the orange ellipsoids are representative of the uncertainty distribution from which scenarios are sampled. This particular formation has n collision regions, one corresponding to each disc.

Hence, a radius must be chosen such that the ego vehicle is neither over-approximated nor under-approximated.

The radius of this disc is taken as 1.2m, while for the proposed two methods, it is taken as 0.8m. The dimension of the obstacle vehicles taken is 4.72m x 1.78m. Thus three non-overlapping discs of 0.8m perfectly envelop the car, while the single disc of radius 1.2m is already 0.62m beyond the car boundaries. Further increases lead to slower and infeasible solutions while reducing this number leads to the front and back being even less approximated.

In terms of the algorithm, the steps followed are identical to the one in Algorithm 2.

4-3-2 Vehicle as a series of Linked Discs

The idea here is that for each obstacle, $n-1$ new obstacles are cloned (where n is the required number of discs to model each vehicle), and their mean positions are translated based on the yaw of the vehicle, the radius of the discs, and the dimensions of the vehicle. The properties associated with a new obstacle are given to these newly cloned obstacles with the extra condition of being linked to each other through the car geometry. The rest of the algorithm works as it previously did, with the difference now being that each obstacle is now considered as n obstacles. This also implies that since each obstacle had uncertainty bounds and a collision region, each vehicle now has n collision regions instead of one. An illustration of what this means is shown in Figure 4-4.

4-3-3 Vehicle as a series of Linked Scenarios

The previous two methods may not have been the smartest or the most optimal ways to model a vehicle. The idea of this method aims to reduce some of the computation and

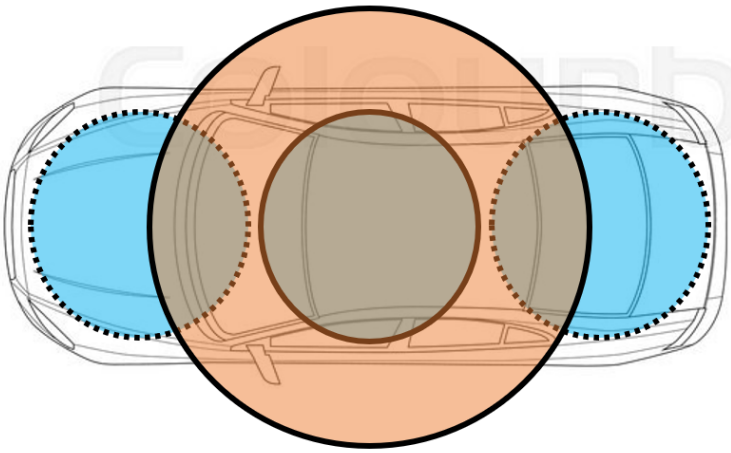


Figure 4-5: Vehicle as a series of linked scenarios. The Blue disc in the center represents the defined vehicle model, while the dotted blue circles represent the effective vehicle representation(due to the linked scenarios). The orange ellipsoid represents the uncertainty distribution from which scenarios are sampled. This particular formation has one collision region, one corresponding to the centre disc. This method combines the best properties of both the previously defined methods.

provide a better implementation. The proposal is that instead of modelling a vehicle as a series of obstacle discs, the vehicle continues to be modelled as a single disc by name, but that one uncertainty sample now spawns n-1 more scenarios linked together by the exact geometry of the car. This step is implemented at the stage where the samples' mean and standard deviation are translated to the location of each obstacle vehicle, thus bypassing a few computation steps made in the previous subsection but effectively representing the same region. An illustration of this is shown in Figure 4-5.

The difference between Method 2 and Method 3 is:

- In Method 2, the vehicle discs are linked to each other through the car geometry at the first step of the algorithm(feedback reception). The n scenarios for this implementation(one for each disc) are independently sampled and not linked to each other.
- In Method 3, the entire vehicle is considered as a single disc only and each scenario sampled from the uncertainty region of this disc is linked to n-1 other scenarios(one for each disc, as if a disc were present). The advantage of doing this is that we now functionally have a n disc representation without computationally having n discs but only one disc. This is expected to make the multi-vehicle extension computationally faster while maintaining the safety guarantee.

Chapter 5

Test Setup and Results

5-1 Test Setup

In this section, some of the other technical specifications of this implementation are laid down. Table 5-1 highlight some essential technical details which were run in the simulation. The scenario distribution was taken as a Gaussian distribution, and not a truncated Gaussian with a 3σ cut-off threshold like in [24], to avoid pruning out of any essential scenarios that could fall beyond the 3σ threshold. Although this threshold can be considered the safe limit, a Gaussian uncertainty model provides that additional safety of keeping scenarios(at the cost of variance), and is appropriate for this investigation of a multi-vehicle extension of scenario-based trajectory optimisation.

Figure 5-1 is a screenshot of the simulation taken to provide an idea of the visualisation. The small window with the title 'Carla UE4' is the CARLA simulator with the ego vehicle in the centre of the frame at a T-Junction. Another vehicle can be seen on the right side of this image.

The larger window in Figure 5-1 is the visualisation tool RViz which is a more technical visualisation of what is seen in the simulator. The ego vehicle is represented as a mini-car, and the blue discs are its predictions over the horizon. The rectangular shape around the car is the safe region polytope for each step of the prediction horizon. The series of circles are surrounding objects and their predicted paths along the prediction horizon(orange to green is the current step to N prediction steps, respectively). The red line in front and behind the ego vehicle represents the car's reference path. As the ego vehicle moves, its predictions, obstacle predictions and thus its safety region is updated in real-time in RViz.

CARLA provides several urban and rural maps for conducting simulations. In this thesis, there will be two major simulations carried out.

- Straight road:- Ego vehicle goes straight. The vehicle just ahead goes straight and turns left, another vehicle just ahead goes straight, and two other vehicles coming

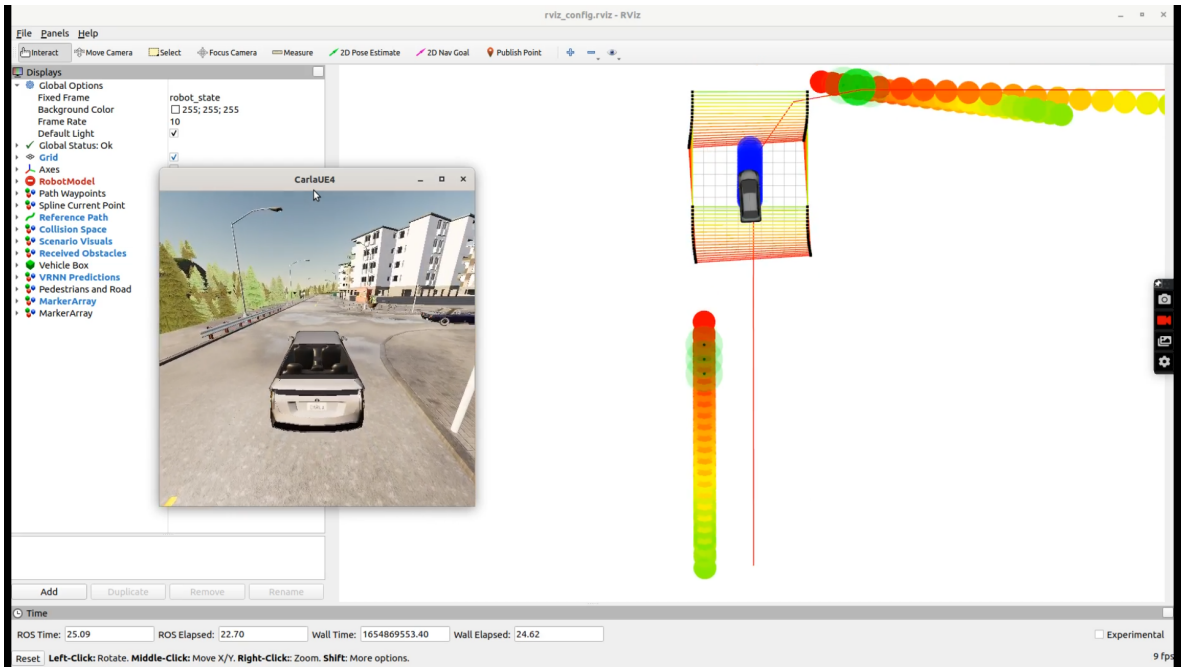


Figure 5-1: An instance of the multi-vehicle visualisation

Table 5-1: Technical Specifications of the Test Setup

Computing Specifications	HP Pavilion 15-cs3xxx, Intel Core i7 @ 1.3GHz, NVIDIA GeForce MX250 Graphics, Ubuntu 20.04 OS
Simulator	CARLA 0.9.10[26]
Physics Engine	Unreal Engine 4(UE4)
Visualisation Tool	RViz[37]
Framework	ROS Noetic Ninjemys[70](See Appendix A)
Optimiser	Forces Pro[82]
Optimisation Algorithm	Primal-Dual Interior Point Method
Global and Local Path Planner	CARLA and LMPCC respectively
Integrator Step Size	200ms
MPCC Objective Function Type	Non-Linear(due to non-linear contouring error and lag error computation)
Optimisation System Dynamics	Bicycle model with Dynamic Steering(Non-Linear)
Constraints	Linear Halfspaces
Number of ego vehicle discs	3
Number of obstacle discs	3
Prediction Horizon	20
Number of scenarios per sample	1000
Scenario Distribution	Gaussian
Risk ϵ	0.05
Confidence β	10^{-6}

in the opposite direction, one behind the other, go straight. This simulation is chosen in order to monitor the responsiveness of the ego vehicle to a crowd of obstacles on spawning.

- T-Junction:- Ego vehicle takes a right turn. Just ahead of the ego, one vehicle also takes a right turn. Another vehicle on the opposite side before the T-Junction goes straight. A third vehicle, just behind the second, takes a left into the same turn our ego vehicle turns into simultaneously. Coming into the T-Junction from the T stem is another car, which takes a left turn towards our ego vehicle's road. This simulation configuration is chosen to test the responsiveness of the ego vehicle to possible head-on collisions with other obstacles.

These situations will be run for all three vehicle disc models covered in the previous section. The evaluation metrics for these results are:

- Time taken to reach the goal. It is representative of how conservatively the vehicle deals with obstacles.
- Closest distance between ego vehicle and an obstacle. Another measure of the safety of the implementation.
- Computation Time. An important factor in determining the real-time feasibility and checking for improvements in the algorithm.
- Ego vehicle steering, velocity and acceleration. It helps in determining how aggressive the reaction to obstacles is.

For these simulations, in the case of a single disc, the radius of the disc modelling the car is taken as 1.2m, while in the cases where we have multiple obstacle discs linked together and multiple scenarios linked together, the radius is 0.8m, following the details covered in the previous chapter.

5-2 Results

In this section, the results of the simulations carried out in all three different configurations, and two different urban scenarios will be shown, and the results will be analysed.

For reference, from now on, 'Single Disc' refers to the implementation like in Figure 4-3, 'Multiple Discs' refers to modelling each obstacle as a series of linked obstacles, as shown in Figure 4-4, and 'Multiple Scenarios' refers to the vehicle being modelled as a series of linked scenarios, like in Figure 4-5. This nomenclature does not imply that Multiple Discs do not have Multiple Scenarios since, by the flow of the program, it will have multiple scenarios at the 'Scenario' stage. It is just a reference to differentiate the implementations by highlighting a distinct characteristic.

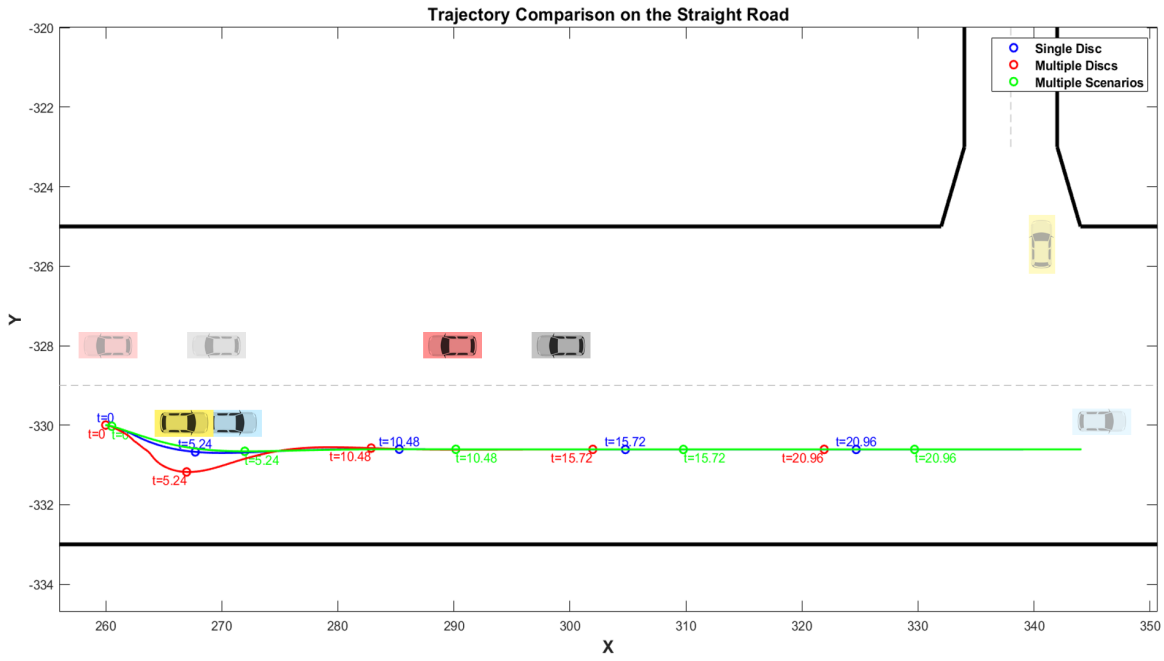


Figure 5-2: Trajectory followed by the ego vehicle in the three different implementations on the Straight Road. The thick black lines show the road limits. The gray dotted lines represent lane divisions. The circular points show the position of the ego vehicle at different time steps. The bright car icons represent the position of the obstacles at t=0. The faded car icons of the same colour represent the direction of the trajectories followed by the corresponding obstacles

5-2-1 Straight Road

The trajectory followed by the ego vehicle and the other obstacles are shown in Figure 5-2. Figure 5-3 is a CARLA and Rviz viewpoint of the same environment at t=0.

- The ego vehicle starts near the middle of the road while there are two other vehicles in the same lane spawned just in front of the ego.
- The first vehicle keeps going straight, while the second vehicle takes a left turn after going some distance.
- On the opposite lane, two vehicles are spawned back-to-back and they go straight in their respective lane(opposite direction).

Points to note are that the coordinates are in the global frame of the Town 1 in CARLA, and that the width of a lane spans roughly 4 coordinates. Figure 5-4 is the yaw feedback of the ego vehicle received from the simulator.

We now visualise the control commands given to the ego vehicle in the form of the velocity, acceleration and steering angle commands(Figures 5-5, 5-6, 5-7 respectively). Before we make an analysis of the three implementations, we also first visualise the minimum distance to the nearest obstacle throughout the trajectory(Figure 5-8). The

Figure 5-3: Simulation visualisation of the Straight Road case on spawning

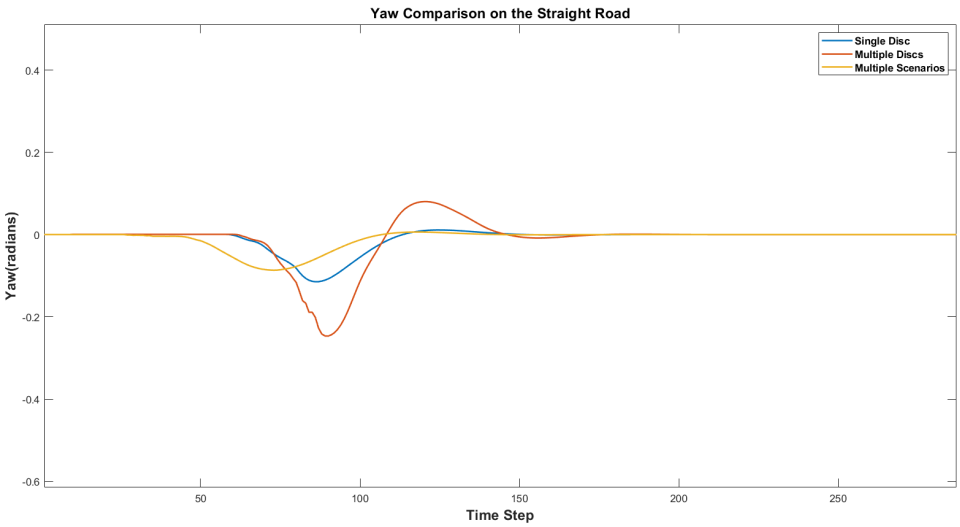


Figure 5-4: Yaws of the ego vehicle on the Straight Road

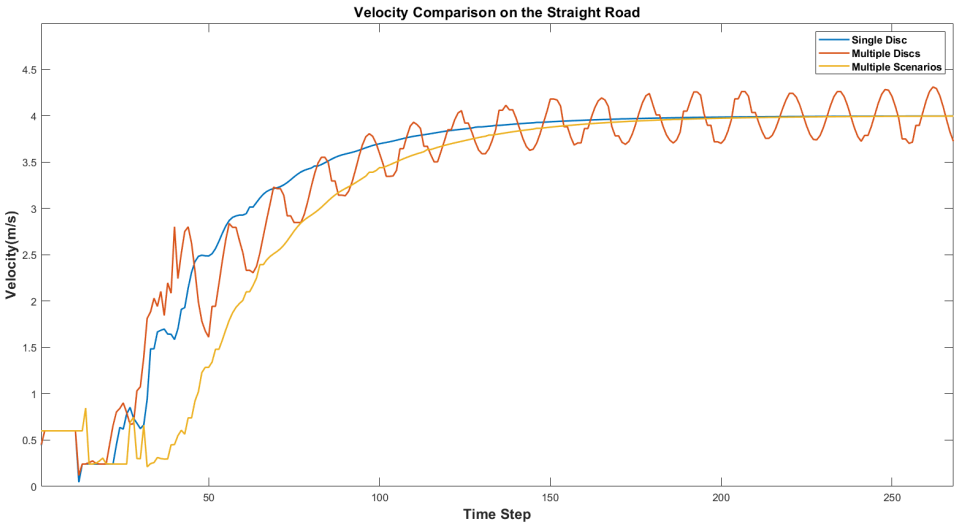


Figure 5-5: Velocity profile of the ego vehicle on the Straight Road

last thing to understand before analysing the results are the computation times and time to goal for each of the implementations (Table 5-2). The lowest computation times are highlighted.

We can now analyse the results we have seen.

- In terms of the trajectory in Figure 5-2, it is observed that the Multi Disc implementation swerves extra to its right and then overshoots very slightly to the other side in an attempt to follow the reference path. This can be confirmed by observing the yaw plots in Figure 5-4 where a slight deflection of around -0.24 radians is observed, and the correction can be seen. Correspondingly, the steering angle command instructing the car to turn that extra 0.24 radians and then correct them can be seen in the plot of Figure 5-7.

Table 5-2: Time To Goal and Computation Times for different parts of the algorithm for the Straight Road

	Time To Goal(s)	Computation Times(Minimum, Maximum, Average)(ms)		
		Scenario Updates	Full Control Loop	Optimisation
Single Disc	29.30	**0.001341**	**9.86111**	**2.35558**
		15.7656	**37.9061**	**23.6211**
		9.12744	**18.8476**	**9.05677**
Multiple Discs	31.13s	18.2915	26.5805	2.3702
		70.7839	89.9907	46.9688
		29.915	41.3661	10.7018
Multiple Scenarios	**26.16**	16.8869	25.4581	4.33009
		39.3932	65.1113	37.1549
		26.7276	39.8249	12.4562

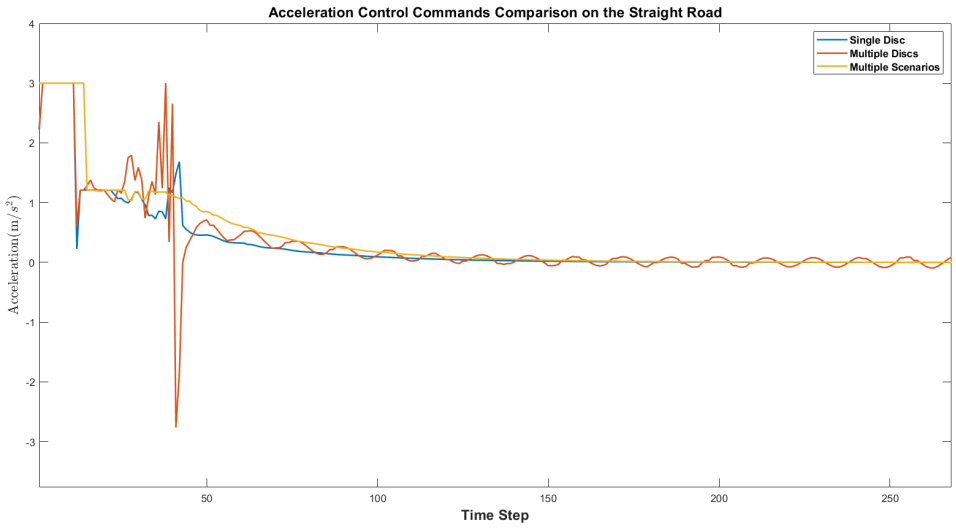


Figure 5-6: Acceleration commands to the ego vehicle on the Straight Road

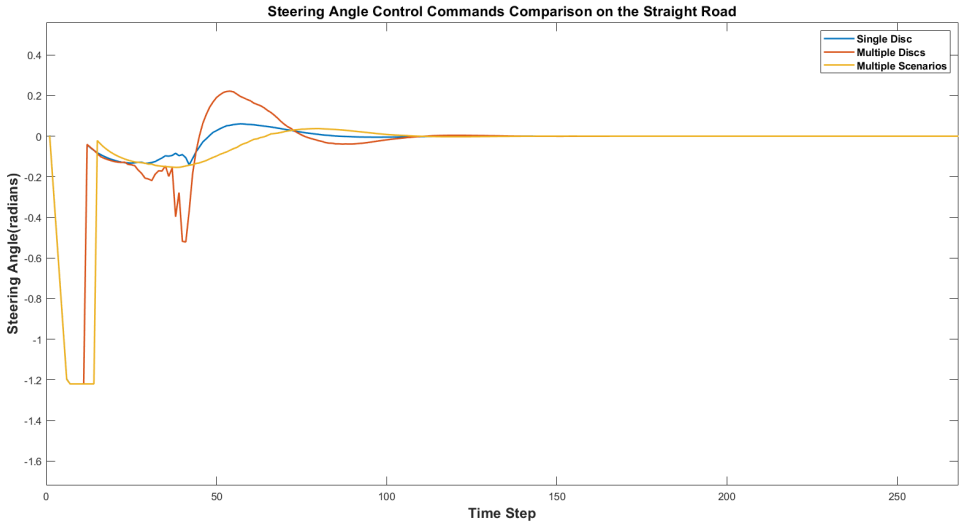


Figure 5-7: Steering commands to the ego vehicle on the straight road

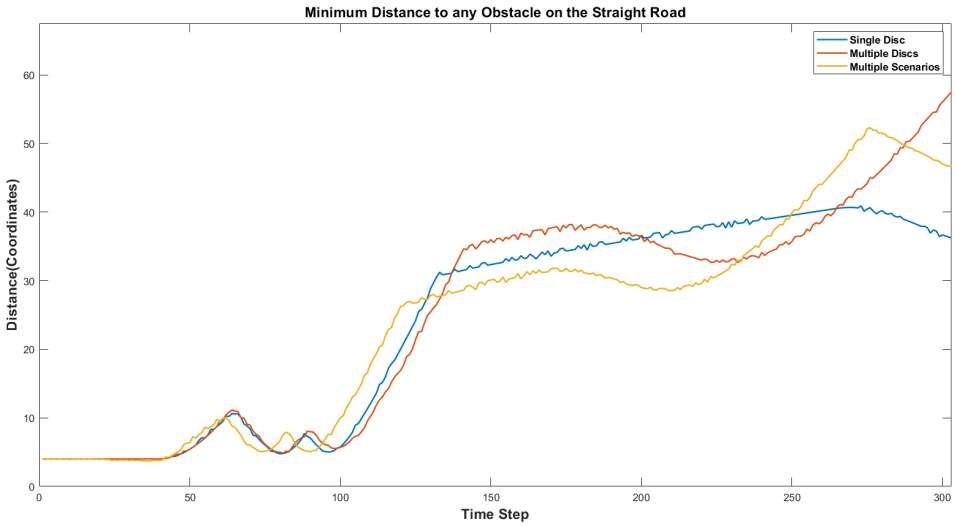


Figure 5-8: Minimum distances to nearest obstacles on the Straight Road

- Looking at the obstacle distance plot in Figure 5-8, at the time instants just before the steering commands to turn are sent(for the Multi Disc curve), we do not see any abnormalities. The deflection is minor, and can be attributed to the stochastic nature of the algorithm. The rest of the trends in the yaw and steering plots seem to agree with the trajectory reasonably well for all cases. It is noteworthy to observe the smoothness of the other two implementations and that there is a minimal deviation for the multi scenarios case.

- Turning the focus to the acceleration and velocity plots of Figure 5-6 and Figure 5-5 respectively, it is first helpful to note that the reference velocity is 4m/s, explaining why the speeds seem to settle at that value. The standout point from the velocity plot is the oscillatory response of the Multi Disc implementation. The reason for this has nothing to do with the implementation, but the computational load of the implementation on the simulator. In CARLA, the traffic manager is in asynchronous mode, and thus heavy traffic(especially heavy for multiple discs, since each vehicle is modelled as 3 obstacles) causes a drop in the frames per second, causing many frames to be skipped. This causes the observed oscillatory motion of the car, due to the skipping of frames. Relatively, the increase in speed corresponds to the increase in minimum distances from the nearest obstacle. Once the speed reaches the desired reference speed, the acceleration commands tend to be 0, as expected. In the case of the Single Disc, we can see that it reaches the reference speed much quicker and has much more aggressive acceleration commands than the Multiple Scenarios implementation. This could be explained by the Multiple Scenarios being a more realistic representation of how close to the ego vehicle the obstacle is, thus leading to a more conservative speeding up and a gentler acceleration profile by causality.

- Looking at Figure 5-8, we observe very similar initial trends for all three for long

periods until both multiple discs, and multiple scenarios slightly decrease and then increase rapidly. This is attributed to the period when the vehicle in front turns left and continues along the road. The Single Disc implementation showed a difference in trends at the end due to always being closer to the vehicle just ahead.

- Finally, we look at Table 5-2, and the results are not surprising. A single disc would take the least number of scenarios, obstacles, and computation and thus have the shortest computation times. It is to be noted that the optimisation times of all three implementations are pretty similar, but this is not surprising considering that the only difference would be the number of linear constraints. Regarding the control loop and the scenario update times, the Multiple Scenarios implementation is faster than the Multiple Discs since few computation steps are skipped to implement the Multiple Scenarios compared to Multiple Discs. This leads to decreased computation times in scenario update steps and the control loops. The time to reach the goal destination was the fastest in the Multiple Scenarios case.

A similar in-depth analysis will now be conducted for the T-Junction situation.

5-2-2 T-Junction

The trajectory followed by the ego vehicle and the other obstacles are shown in Figure 5-9. Figure 5-10 is a CARLA and Rviz viewpoint of the ego vehicle and obstacles at the T-Junction just before and during turning at the junction.

- The ego vehicle is spawned in the middle of the road. Just ahead of the ego vehicle is an obstacle(gray vehicle) which takes a right turn at the T-Junction
- A blue obstacle is spawned on the opposite lane of the ego vehicle and it continues straight along its path.
- A yellow vehicle is spawned just behind the blue vehicle and it takes a left turn into the T-Junction. The timing of this turn is perfectly in synchronisation with the ego vehicle turning into the T-Junction, thus providing a very appropriate test of the ego vehicle's response to our obstacle models.
- A red car is spawned on the opposite side of the lane inside the T-Junction. This vehicle turns left at the junction. This is also perfectly times to turn left just after allowing the previous yellow vehicle to turn inside.

It should be noted that the Single Disc trajectory took an aggressive trajectory by crossing a small part of the sidewalk and accelerating suddenly to try and get past the yellow obstacle vehicle, but still slowing down to let it pass. This led to a speedy time to goal; thus, all the time step plots we see might include similar trends but "time-shifted" data. The focus here is on the trends, though; they will now be visualised just like in

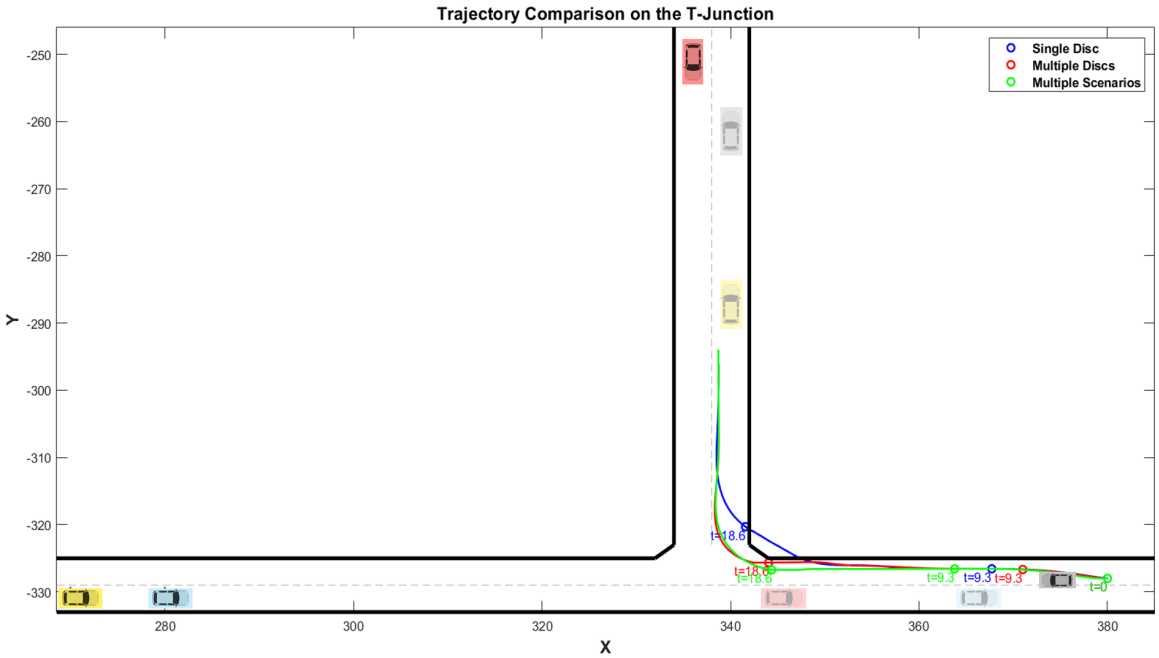
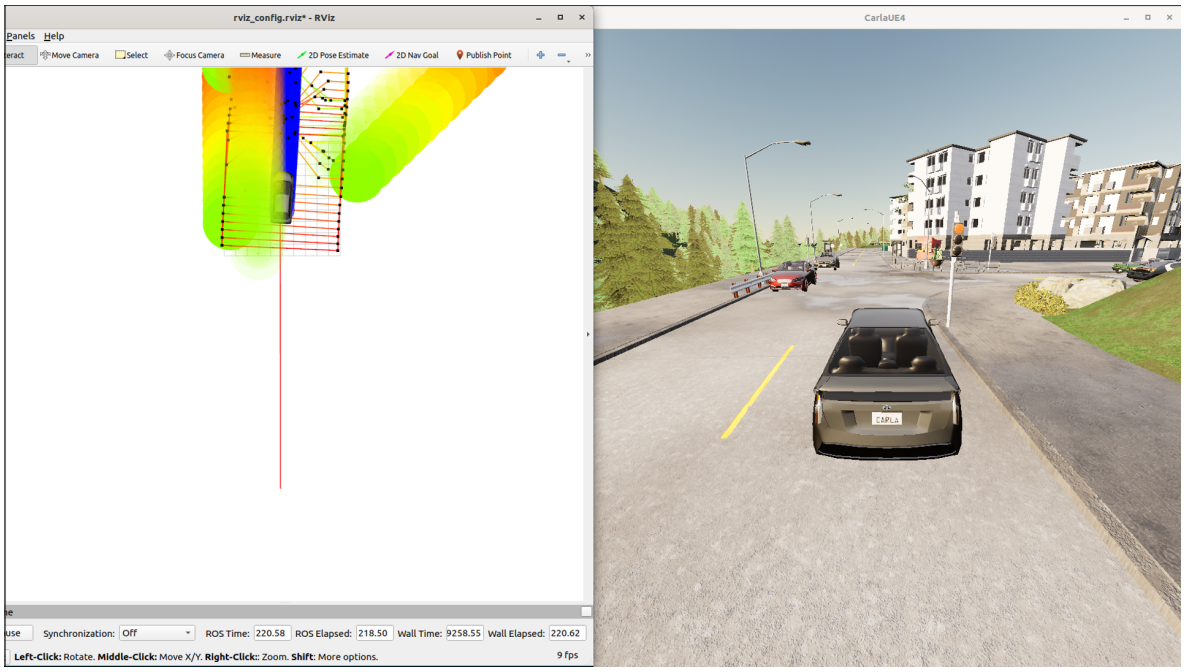
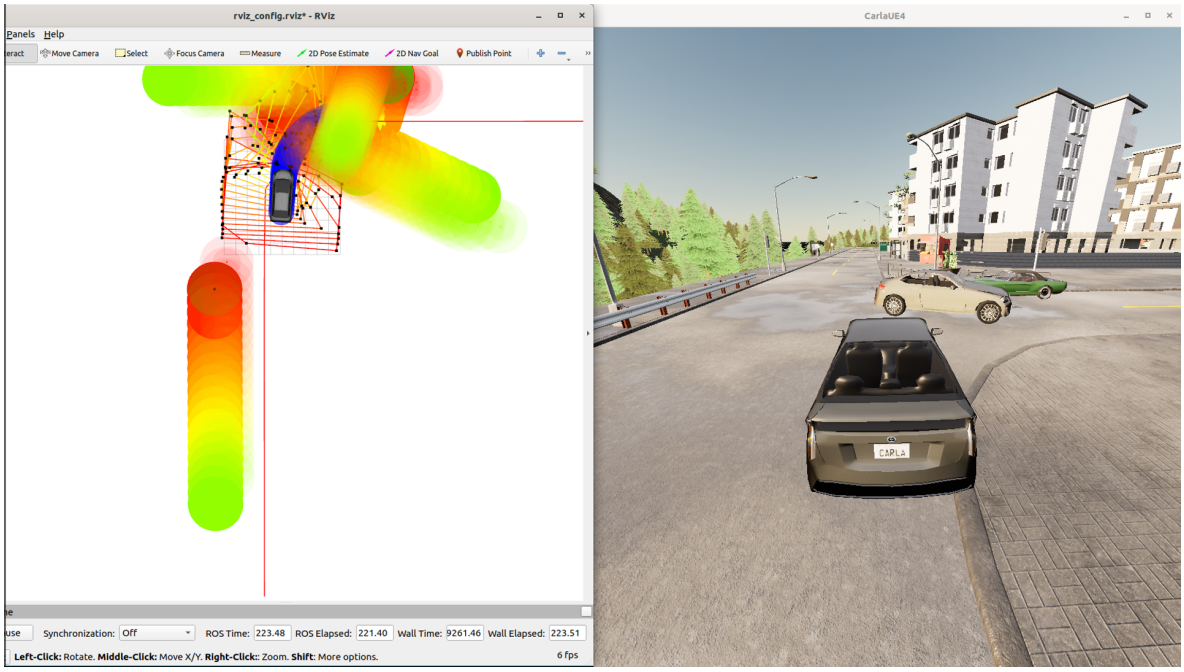


Figure 5-9: Trajectory followed by the ego vehicle in the three different implementations at the T-Junction. The thick black lines show the road limits. The gray dotted lines represent lane divisions. The circular points show the position of the ego vehicle at different time steps. The bright car icons represent the position of the obstacles at t=0. The faded car icons of the same colour represent the direction of the trajectories followed by the corresponding obstacles



(a) T-Junction while the ego vehicle turns. All the four obstacles are visible in this frame(two on the left lane, two in the T-junction, one leaving it and one entering it)



(b) T-Junction just before the ego vehicle is about to turn into it. The ego vehicle stops to let the obstacle pass cross first.

Figure 5-10: Simulation visualisation of the T-Junction case

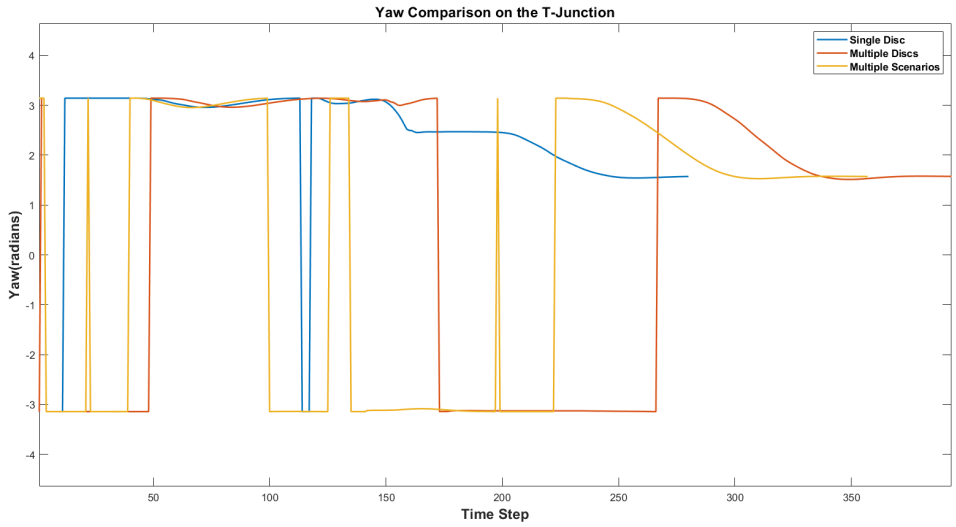


Figure 5-11: Yaws of the ego vehicle at the T-Junction

the previous sub-section, after which an analysis will be conducted. The yaw plots of the ego vehicle are shown in Figure 5-11. It should be noted that these yaw values are in the global CARLA frame, thus explaining the continual switching between π and $-\pi$ since the yaw range is $[-\pi, \pi]$.

We now visualise the control commands given to the ego vehicle in the form of the velocity, acceleration and steering angle commands(Figures 5-12, 5-13, 5-14 respectively). Before analysing the three implementations, we also visualise the minimum distance to the nearest obstacle throughout the trajectory(Figure 5-15). The last things to understand before analysing the results are the computation times and time to goal for each implementation (Table 5-3). The lowest computation times are highlighted.

- Through Figure 5-11, it can be seen that the yaw switches between $-\pi$ and π,

Table 5-3: Time To Goal and Computation Times for different parts of the algorithm for the T-Junction

	Time To Goal(s)	Computation Times(Minimum, Maximum, Average)(ms)		
		Scenario Updates	Full Control Loop	Optimisation
Single Disc	**25.96**	0.00195	8.47177	2.225764
		27.0908	**128.152**	114.808
		9.39033	**26.3929**	16.3145
Multiple Discs	38.21	14.8213	21.5146	**1.78828**
		37.8875	130.612	**102.898**
		29.8977	43.3787	**12.6314**
Multiple Scenarios	33.86	**0.001215**	**7.5641**	2.2842
		65.2766	196.675	158.471
		19.873	37.33	16.801

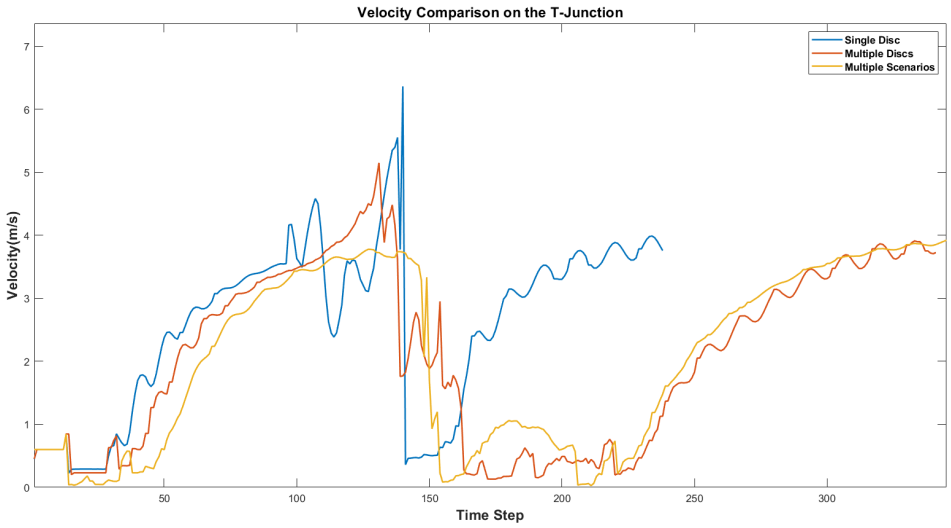


Figure 5-12: Velocity profile of the ego vehicle at the T-Junction

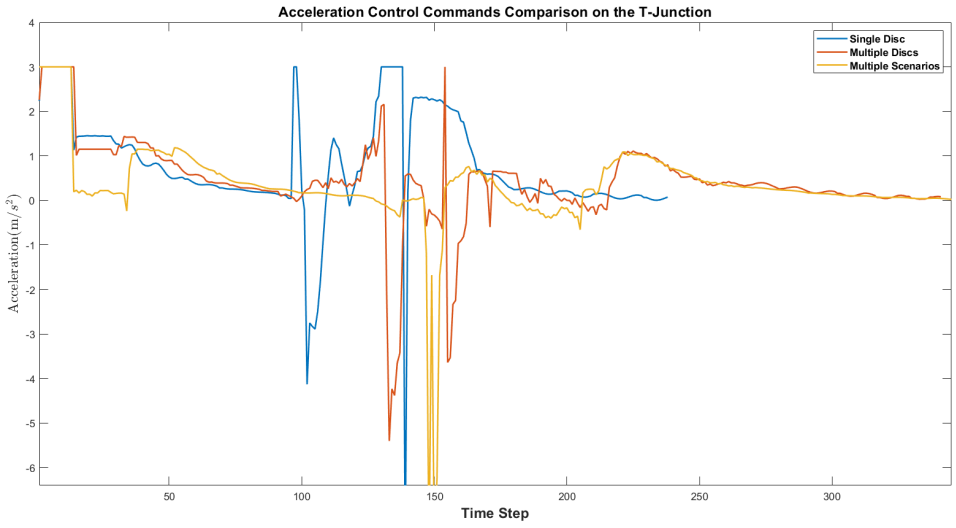


Figure 5-13: Acceleration commands to the ego vehicle at the T-Junction

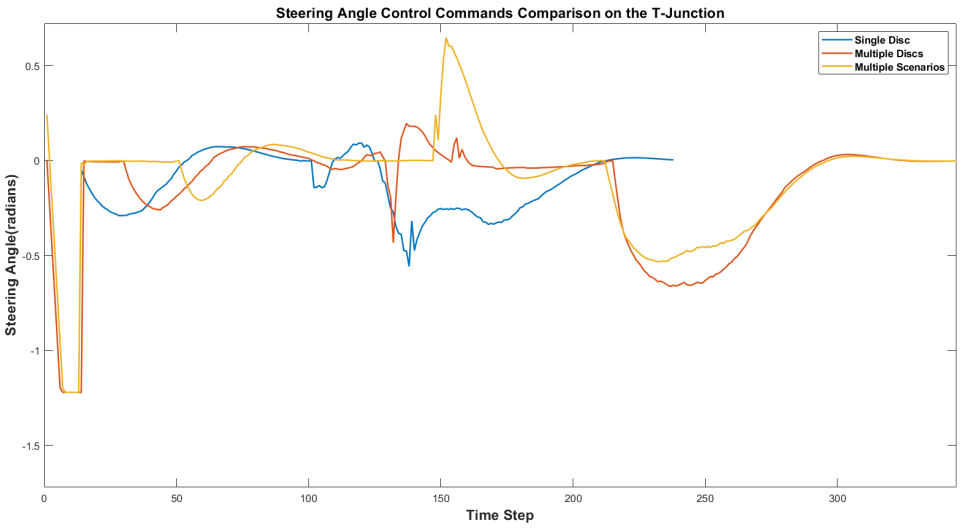


Figure 5-14: Steering commands to the ego vehicle at the T-Junction

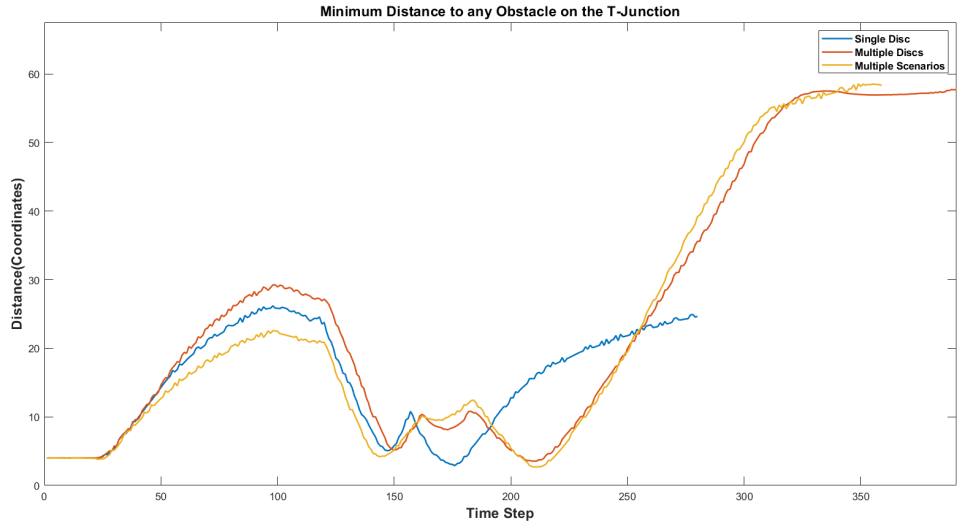


Figure 5-15: Minimum distances to nearest obstacles at the T-Junction

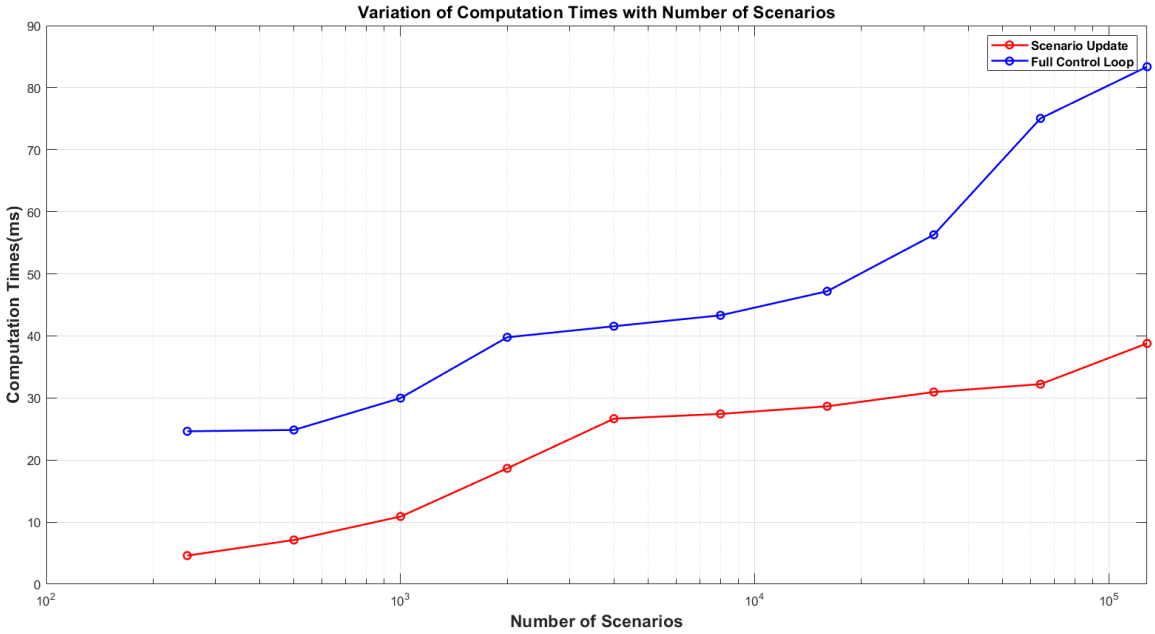


Figure 5-16: An illustration of the effect of the number of scenarios sampled on the computation times. Note that the x axis is on a logarithmic scale, while the y-axis is on a linear scale. The blue line represents the entire Control Loop(Scenario Updates+Optimisation+Extra Computations), while the red line represents only the Scenario Update times.

and that is because they correspond to nearly the same orientation at the point of discontinuity. Once the car reaches the T-Junction, it turns to the right and the yaw changes to $\pi/2$. In the case of the single disc, the yaw aggressively decreases for a short time before stabilising and reaching $\pi/2$. This happened because the car aggressively turned at the T-Junction, then stopped to allow a car to pass, after which it completed the turn. In the other cases, the ego vehicle approached the T-Junction, waited momentarily for the car to cross it, and followed suit. It can be seen that the Multiple Scenarios Approach run had the car turning shortly before the Multiple Discs run.

- We now look at the corresponding steering angle commands in Figure 5-14, where we observe that after an initial right turn when spawning to stay in the correct lane corresponding to the first dip, the steering returns to zero. After this, the single disc implementation takes a sudden right turn, as validated by the trajectory, and waits for a little. At the same time, the vehicle it stops for passes by and then resumes to complete the turn, after which the steering angle returns to zero. In the case of multiple discs, after the initial minor right turn, there is a sudden left and right steering command, which still keeps the car in the lane, but maybe shifts it a bit to the right side, as we can see in the trajectory. Due to this position, it must take a slightly sharper turn at the T-Junction, and this is reflected in the magnitude of the steering action taken. In the multiple scenarios steering angle graph, everything is ordinary apart from the big positive left steering spike. This corresponded to the trajectory part when the multiple disc implementation

shifted to the left lane and is attributed to having a cleaner, less steering turn at the T-Junction, and the trajectory graph, along with the steering command magnitude of the left turn, agrees with that.

- Looking at the acceleration commands and velocity profile in Figures 5-13 and 5-12 respectively, we assess each case individually. In the case of a Single Disc, we see a gradual increase in velocity on the straight part of the road approaching the T-Junction, and then we see a sudden spike, dip, spike, dip and then spike again in the acceleration plot. This corresponds to the car swerving onto parts of the crosswalk, slowing down, continuing on the path, allowing the yellow obstacle vehicle to pass and then speeding up again to reach the goal. The trend is pretty much identical and expected in the case of multiple discs and multiple scenarios. The cars speed up on the straight, slow down/stop at the T-Junction to allow the vehicle to pass, then turn and speed up again to complete the manoeuvre.

- The last plot we look at is the minimum distance plot in Figure 5-15, where we once again observe almost identical trends when comparing Multiple Discs and Multiple Scenarios. There is, however, a slight difference in minimum distances between them initially, which could be attributed to them sticking to different sides of the same lane. The first rise corresponds to the grey obstacle vehicle just ahead of the ego vehicle. The corresponding dip then corresponds to the blue obstacle vehicle, which continues to go straight along its path, the next dip corresponds to the red obstacle vehicle, and the final dip corresponds to the yellow obstacle car. In terms of Single Disc, we observe that it "misses" one of the dips we can see for the other two cases. This missing dip would correspond to the red obstacle since it takes a short turn across the crosswalk and is at all times closer to the yellow car than the red one, which is on the opposite side of the road.

- We now look at the Computation Times in Table 5-3, and we can see that, as expected, the time to goal for the Single Disc is much lesser than the other two implementations due to its relatively aggressive approach. The Time to goal for Multiple Scenarios was significantly lesser than the time to goal for Multiple Discs. In terms of individual computation times, the scenario updates are still the quickest in the Single Disc implementation, owing to the fewer scenarios due to one disc. The scenario updates for multiple scenarios are slightly lesser than those of multiple discs. In the control loop, a Single Disc is least understandably due to lesser computations, while multiple scenarios beat multiple discs due to reduced computations. The optimisation speeds are pretty comparable, with Multiple Discs being the quickest.

- From Figure 5-16, we can observe the reliance on the computation times of the scenario updates and, correspondingly, the control loop for different numbers of scenarios. Seeing the exponential effect that the number of scenarios has on computation times makes us reflect on the importance of having a trade-off between the number of scenarios sampled against the risk factor ϵ. Larger the number of scenarios, the lesser the risk factor, but the greater the computation time required.

However, even with 128000 scenarios the computation times remain real-time implementable, and that is noteworthy.

Now that all the results have been visualised and discussed, we make some conclusions based on all we have seen in the next chapter and discuss future progress directions.

Chapter 6

Conclusions and Future Directions

6-1 Conclusions

Going through a small recap of everything this thesis has covered, we started with the introduction, where a brief background of automated driving was laid down, and a few research questions regarding the transition of moving from pedestrians to vehicles, along with their impact on safety, were asked. The next chapter summarised relevant literature on motion planning and decision-making. This was followed by a background chapter that covered the basics of MPC, MPCC, LMPCC, Robust and Stochastic optimisation, Chance Constraints, Scenarios, and SMPCC, along with mathematical representations of risk bounding, safety and confidence in the case of convex and non-convex scenario optimisation problems. Chapter 4 highlighted the proposed approach and decided to represent vehicles as a series of discs, which was then improved upon by representing them as a series of linked scenarios. It was then mathematically shown that these implementations' chance constraints and safety guarantees remain as if a simple linear transformation had been applied, and that they relatively provide better safety guarantees due to having an increased number of scenarios. The next chapter discussed the test setup and environment where simulations were carried out, after which the results were visualised and discussed. We now attempt to make inferences based on these results.

Based especially on the results we saw in the previous chapter, we can conclude the following:

- The Single Disc implementation was generally always the fastest. This makes perfect sense as it is a single disc approximating the vehicle, and there are only 1000 scenario computations per vehicle per iteration. In terms of simulated performance on the straight road and T-Junction, this implementation seemed the more aggressive out of the three implementations. Once again, this makes good sense that the region covered by the scenarios is not a good fit for the vehicle.

Thus all computations would consider a relatively less safe approximation of the vehicle. The T-Junction was a perfect highlight of things that go wrong with using such an approximation of the vehicle.

- The multiple disc implementations consider the vehicle to be a series of linked obstacles. It is computationally heavier since n times the number of scenarios is taken through all the computations in each iteration(n depends on how many discs the vehicle could be modelled with). Each obstacle also has a collision region defined around it, and a single vehicle having n overlapping collision regions is not ideal or optimal. However, this implementation provides a safer or more conservative solution, which makes sense since the car is well-approximated shape-wise, with n times the number of samples compared to a single disc.
- The final implementation was multiple scenarios, where the single disc with a probability distribution of uncertainty around spawns n scenarios around the vehicle. It is effectively an n-disc representation with one collision region and a good approximation while avoiding the time complexity issue of Multiple Discs. The results confirm this in terms of similarity or trajectory and behaviour, but a marked difference in latency.

This can lead us to conclude that the Multiple Scenarios approach has served as a good approximation of the obstacle vehicle in an urban setting while being computationally efficient. It thus satisfies the requirements of being practical and real-time implementable.

6-2 Future Directions

The scenario approach to obstacle avoidance(especially for non-convex, non-linear programs) is a relatively unexplored but promising avenue for future research in planning and decision-making for automated driving and robotics. That being said, several hurdles must be overcome for this to be deployed at a commercial level. Some of the drawbacks of this implementation are:

- The real-time scalability of this method with respect to the number of dynamic obstacles is a question. Running the same algorithm with more vehicles and walking and crossing pedestrians resulted in frequent crashes on the testing system.
- The prediction model used for vehicles and even pedestrians here was a constant velocity model. In real life, that is an inaccurate representation of dynamic motion. The uncertainty distribution was assumed to be real-time here, while real uncertainties are non-Gaussian.

With these major drawbacks in mind, the proposal for future work is:

- The first recommendation would be to improve the scalability of dynamic vehicles through the scenario approach or an even better but still computationally efficient

obstacle representation. A shape like a rectangle used for this implementation while still preserving the latency, and dealing with its discontinuous shape, would be ideal.

- When it comes to trajectory prediction models, a wide variety of non-Gaussian trajectories spring up, rendering it a challenge for the algorithm to handle. There is much scope to explore this path.
- This scenario extension for vehicle representation should be tested on hardware platforms to assess its real-time feasibility.
- Having a vehicle represented as a series of overlapping discs can also be explored. This would provide an even better approximation of the vehicle. However, simultaneously, there would be many overlapping scenario points and collision regions, which would then need to be pruned out. While this could be a promising direction for future work, it will play with the safety risk bounds and might be much less conservative depending on the number of pruned-out scenarios.

Appendix A

ROS

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across various robotic platforms.

Why? Because creating truly robust, general-purpose robot software is complicated. From the robot's perspective, problems that seem trivial to humans often vary wildly between instances of tasks and environments. Dealing with these variations is so hard that no single individual, laboratory, or institution can hope to do it on their own.

As a result, ROS was built from the ground up to encourage collaborative robotics software development. For example, one laboratory might have experts in indoor mapping environments and could contribute a world-class system for producing maps. Another group might have experts at using maps to navigate, yet another group might have discovered a computer vision approach that works well for recognizing small objects in clutter. ROS was designed for groups like these to collaborate and build upon each other's work.

For this purpose, we cover the basic ROS concepts of topics, nodes, messages, publishing and subscribing. ROS is a robotics middleware. It cannot be called an Operating System, but it still provides hardware abstraction, low-level device control, message transfers between services, multi-threading, package management, etc. ROS is open-source. Users can choose the configuration of tools and libraries which interact with ROS core to account for custom applications and robots according to user demands. ROS processes are represented as nodes, which communicate with each other via topics. This is possible thanks to a process called ROS Master. The Master sets up peer-to-peer communication between node processes.

A-1 Design

A node represents a single process running the ROS graph. Every node has a name, which registers with the ROS master before it can take any other actions. Nodes are at the centre of ROS programming, as most ROS client code is in the form of a ROS node which takes actions based on information received from other nodes, sends information to other nodes, or sends and receives requests for actions to and from other nodes.

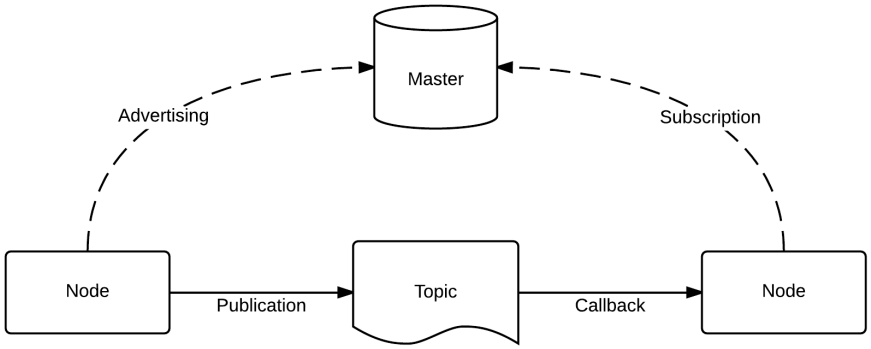


Figure A-1: ROS Topics and Nodes

Topics are named buses over which nodes send and receive messages. In order to send messages to a topic, a node must publish to said topic, while to receive messages, it must subscribe, as shown in Fig A-1. The publish/subscribe model is anonymous: no node knows which nodes are sending or receiving on a topic, only that it is sending/receiving on that topic. The messages passed on a topic vary widely and can be user-defined. The content of these messages can be sensor data, motor control commands, state information, actuator commands, or anything else.

A-2 Tools

ROS's core functionality is augmented by a variety of tools which allow developers to visualize and record data, easily navigate the ROS package structures, and create scripts automating complex configuration and setup processes. These tools significantly increase the capabilities of systems using ROS by simplifying and providing solutions to several common robotics developments. These tools are provided in packages like any other algorithm, but rather than providing implementations of hardware drivers or algorithms for various robotic tasks, and these packages provide task and robot-agnostic tools that come with the core of most modern ROS installations.

Catkin is the ROS build system. Catkin is based on CMake and is similarly cross-platform, open-source, and language-independent. The rosbash package provides a suite of tools that augment the bash shell's functionality. These tools include rosls, roscd, and roscp, which replicate the functionalities of ls, cd, and cp. The ROS versions of these tools allow users to use ROS package names in place of the file path where the package is located. The package also adds tab-completion to most ROS utilities. It includes rosed, which edits a given file with the chosen default text editor, and rosrun, which runs and executes in ROS packages.

roslaunch is a tool used to launch multiple ROS nodes locally and remotely, as well as set parameters on the ROS parameter server. roslaunch configuration files, written using XML can easily automate a complex startup and configuration process into a single command. roslaunch scripts can include other roslaunch scripts, launch nodes on specific machines, and even restart processes which die during execution.

A-3 Programming

ROSCPP is a C++ implementation of ROS. It provides a client library that enables C++ programmers to interface with ROS Topics, Services, and Parameters quickly. roscpp is the most widely used ROS client library and is designed to be the high-performance library for ROS.

Bibliography

[1] Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. SAE International On-Road Automated Driving Committee 2018.

[2] The history of the automata of leonardo da vinci, January 2021.

[3] Bassam Alrifaee. *Networked model predictive control for vehicle collision avoidance*. PhD thesis, Dissertation, RWTH Aachen University, 2017, 2017.

[4] Giorgio Arici, Marco C Campi, Algo Carè, Marco Dalai, and Federico A Ramponi. A theory of the risk for empirical cvar with application to portfolio selection. *Journal of Systems Science and Complexity*, 34(5):1879–1894, 2021.

[5] Haoyu Bai, Shaojun Cai, Nan Ye, David Hsu, and Wee Sun Lee. Intention-aware online pomdp planning for autonomous driving in a crowd. In *2015 ieee international conference on robotics and automation (icra)*, pages 454–460. IEEE, 2015.

[6] Louis Bakker and Sergio Grammatico. A multi-agent deep reinforcement learning framework for automated driving on highways. In *2020 28th Mediterranean Conference on Control and Automation (MED)*, pages 770–775. IEEE, 2020.

[7] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

[8] Tommaso Benciolini, Tim Brüdigam, and Marion Leibold. Multistage stochastic model predictive control for urban automated driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 417–423, 2021.

[9] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 26(3):502–517, 2010.

[10] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[11] Lars Blackmore, Masahiro Ono, and Brian C. Williams. Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094, 2011.

[12] Bruno Brito, Boaz Floor, Laura Ferranti, and Javier Alonso-Mora. Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robotics and Automation Letters*, 4(4):4459–4466, 2019.

[13] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[14] Marco C Campi, Algo Carè, and Simone Garatti. The scenario approach: A tool at the service of data-driven decision making. *Annual Reviews in Control*, 52:1–17, 2021.

[15] Marco C Campi and Simone Garatti. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization*, 19(3):1211–1230, 2008.

[16] Marco C Campi and Simone Garatti. Wait-and-judge scenario optimization. *Mathematical Programming*, 167(1):155–189, 2018.

[17] Marco Claudio Campi, Simone Garatti, and Federico Alessandro Ramponi. A general scenario theory for nonconvex optimization and decision making. *IEEE Transactions on Automatic Control*, 63(12):4067–4078, 2018.

[18] Bernard Chazelle. Approximation and decomposition of shapes. *Algorithmic and Geometric Aspects of Robotics*, 1:145–185, 1985.

[19] Chih-Yuan Chiu, David Fridovich-Keil, and Claire J Tomlin. Encoding defensive driving as a dynamic nash game. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10749–10756. IEEE, 2021.

[20] Alessandro Colombo and Domitilla Del Vecchio. Efficient algorithms for collision avoidance at intersections. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 145–154, 2012.

[21] R Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

[22] Brigitte d'Andréa Novel, Guy Campion, and Georges Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International journal of robotics research*, 14(6):543–559, 1995.

[23] Yin Dang. Simple understanding of kinematic bicycle model, Sept. 2021.

[24] Oscar de Groot, Bruno Brito, Laura Ferranti, Dariu Gavrila, and Javier Alonso-Mora. Scenario-based trajectory optimization in uncertain dynamic environments. *IEEE Robotics and Automation Letters*, 6(3):5389–5396, 2021.

[25] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[26] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[27] Filippo Fabiani and Sergio Grammatico. A mixed-logical-dynamical model for automated driving on highways. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 1011–1015. IEEE, 2018.

[28] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on control systems technology*, 15(3):566–580, 2007.

[29] Dave Ferguson, Thomas M Howard, and Maxim Likhachev. Motion planning in urban environments. *Journal of Field Robotics*, 25(11-12):939–960, 2008.

[30] Paola Festa. Shortest path algorithms. In *Handbook of optimization in telecommunications*, pages 185–210. Springer, 2006.

[31] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[32] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2015.

[33] David Fridovich-Keil, Andrea Bajcsy, Jaime F Fisac, Sylvia L Herbert, Steven Wang, Anca D Dragan, and Claire J Tomlin. Confidence-aware motion prediction for real-time collision avoidance1. *The International Journal of Robotics Research*, 39(2-3):250–265, 2020.

[34] Simone Garatti and MC Campi. Risk and complexity in scenario optimization. *Mathematical Programming*, pages 1–37, 2019.

[35] Yang Guan, Shengbo Eben Li, Jingliang Duan, Wenjun Wang, and Bo Cheng. Markov probabilistic decision making of self-driving cars in highway with random traffic flow: a simulation study. *Journal of intelligent and connected vehicles*, 2018.

[36] Michael R Hafner, Drew Cunningham, Lorenzo Caminiti, and Domitilla Del Vecchio. Cooperative collision avoidance at intersections: Algorithms and experiments. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1162–1175, 2013.

[37] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, and Chang-Hun Kim. Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, 60(2):337–345, 2015.

[38] Mina Kamel, Javier Alonso-Mora, Roland Siegwart, and Juan Nieto. Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243. IEEE, 2017.

[39] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[40] Christos Katrakazas, Mohammed Quddus, Wen-Hua Chen, and Lipika Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416–442, 2015.

[41] Forrest Laine, David Fridovich-Keil, Chih-Yuan Chiu, and Claire Tomlin. Multi-hypothesis interactions in game-theoretic motion planning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8016–8023. IEEE, 2021.

[42] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[43] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[44] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[45] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[46] David Lenz, Frederik Diehl, Michael Truong Le, and Alois Knoll. Deep neural networks for markovian interactive scene prediction in highway scenarios. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 685–692. IEEE, 2017.

[47] Xiaohui Li, Zhenping Sun, Qi Zhu, and Daxue Liu. A unified approach to local trajectory planning and control for autonomous driving along a reference path. In *2014 IEEE international conference on mechatronics and automation*, pages 1716–1721. IEEE, 2014.

[48] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

[49] Alexander Liniger and John Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2019.

[50] Yuanfu Luo, Haoyu Bai, David Hsu, and Wee Sun Lee. Importance sampling for online planning under uncertainty. *The International Journal of Robotics Research*, 38(2-3):162–181, 2019.

[51] Yiwei Lyu, Wenhao Luo, and John M Dolan. Probabilistic safe adaptive merging control for autonomous vehicles under motion uncertainty.

[52] David Madås, Mohsen Nosratinia, Mansour Keshavarz, Peter Sundström, Rolland Philippsen, Andreas Eidehall, and Karl-Magnus Dahlén. On path planning methods for automotive collision avoidance. In *2013 IEEE intelligent vehicles symposium (IV)*, pages 931–937. IEEE, 2013.

[53] Jason R Marden and Jeff S Shamma. Game theory and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:105–134, 2018.

[54] Alejandro Ivan Morales Medina, Nathan Van De Wouw, and Henk Nijmeijer. Cooperative intersection control based on virtual platooning. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1727–1740, 2017.

[55] David Miculescu and Sertac Karaman. Polling-systems-based control of high-performance provably-safe autonomous intersections. In *53rd IEEE conference on decision and control*, pages 1417–1423. IEEE, 2014.

[56] David Miculescu and Sertac Karaman. Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals. *IEEE Transactions on Automatic Control*, 65(2):680–694, 2019.

[57] Nils J Nilsson. A mobile automaton: An application of artificial intelligence techniques. Technical report, Sri International Menlo Park Ca Artificial Intelligence Center, 1969.

[58] Colm Ó'Dúnlaing and Chee K Yap. A "retraction" method for planning the motion of a disc. *Journal of Algorithms*, 6(1):104–111, 1985.

[59] Masahiro Ono, Brian C Williams, and Lars Blackmore. Probabilistic planning for continuous dynamic systems under bounded risk. *Journal of Artificial Intelligence Research*, 46:511–577, 2013.

[60] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[61] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Navigating congested environments with risk level sets. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5712–5719. IEEE, 2018.

[62] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[63] Jackeline Rios-Torres and Andreas A Malikopoulos. A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1066–1077, 2016.

[64] Claude Samson. Control of chained systems application to path following and time-varying point-stabilization of mobile robots. *IEEE transactions on Automatic Control*, 40(1):64–77, 1995.

[65] Georg Schildbach, Matthias Soppert, and Francesco Borrelli. A collision avoidance system at intersections using robust model predictive control. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 233–238. IEEE, 2016.

[66] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):2994–3008, 2017.

[67] Wilko Schwarting, Javier Alonso-Mora, and Daniela Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:187–210, 2018.

[68] Adhiraj Somani, Nan Ye, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *Advances in neural information processing systems*, 26, 2013.

[69] Tong Duy Son and Quan Nguyen. Safety-critical control for non-affine nonlinear systems with application on autonomous vehicle. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 7623–7628. IEEE, 2019.

[70] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.

[71] Osamu Takahashi and Robert J Schilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on robotics and automation*, 5(2):143–150, 1989.

[72] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[73] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[74] Behrad Toghi, Rodolfo Valiente, Dorsa Sadigh, Ramtin Pedarsani, and Yaser P Fallah. Social coordination and altruism in autonomous driving. *arXiv preprint arXiv:2107.00200*, 2021.

[75] Richard S Wallace, Anthony Stentz, Charles E Thorpe, Hans P Moravec, William Whittaker, and Takeo Kanade. First results in robot road-following. In *IJCAI*, pages 1089–1095, 1985.

[76] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

[77] Mingyu Wang, Negar Mehr, Adrien Gaidon, and Mac Schwager. Game-theoretic planning for risk-aware interactive agents. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6998–7005. IEEE, 2020.

[78] Mingyu Wang, Zijian Wang, Shreyasha Paudel, and Mac Schwager. Safe distributed lane change maneuvers for multiple autonomous vehicles using buffered input cells. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4678–4684. IEEE, 2018.

[79] Mingyu Wang, Zijian Wang, John Talbot, J Christian Gerdes, and Mac Schwager. Game-theoretic planning for self-driving cars in multivehicle competitive scenarios. *IEEE Transactions on Robotics*, 37(4):1313–1325, 2021.

[80] Moritz Werling, Sören Kammel, Julius Ziegler, and Lutz Gröll. Optimal trajectories for time-critical street scenarios using discretized terminal manifolds. *The International Journal of Robotics Research*, 31(3):346–359, 2012.

[81] Yan Wu, Lifang Wang, Junzhi Zhang, and Fang Li. Path following control of autonomous ground vehicle based on nonsingular terminal sliding mode and active disturbance rejection control. *IEEE Transactions on Vehicular Technology*, 68(7):6379–6390, 2019.

[82] Andrea Zanelli, Alexander Domahidi, Juan Jerez, and Manfred Morari. Forces nlp: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 93(1):13–29, 2020.

[83] Yue J Zhang, Andreas A Malikopoulos, and Christos G Cassandras. Optimal control and coordination of connected and automated vehicles at urban traffic intersections. In *2016 American Control Conference (ACC)*, pages 6227–6232. IEEE, 2016.

[84] Hai Zhu and Javier Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.