Probabilistic Motion Planning in Dynamic Environments

Parallelizable Scenario-Based Trajectory Optimization with Global Guidance

Probabilistic Motion Planning in Dynamic Environments

Parallelizable Scenario-Based Trajectory Optimization with Global Guidance

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus prof. dr. ir. T.H.J.J. van der Hagen,
Chair of the Board for Doctorates
to be defended publicly on
Friday 6 December 2024 at 12:30 o'clock

by

Oscar Maxim de Groot

Master of Science in Systems and Control
Delft University of Technology, Delft, The Netherlands,
born in Woerden, The Netherlands.

This dissertation received support from the Netherlands Organization for Scientific Research (NWO) within the SafeVRU project.

Keywords: Motion and Path Planning, Optimization and Optimal Control, Collision Avoidance, Constrained Motion Planning

Printed by: Gildeprint

Front & Back: AI generated image using Playground AI and DALL-E 2 inspired by the "Mistborn" bookseries by Brandon Sanderson.

ACKNOWLEDGEMENTS

This dissertation not only concludes my PhD, but also my time at the TU Delft. I am very grateful for the learning experience at the TU Delft and for the people that I have met in my time here. I thank no small part of this thesis to you. At least an equal part, I thank to my friends and family. In the end, you all have made my research worthwhile and enjoyable. I would therefore like to write a few words of appreciation for those that have had a part in this work.

First of all, I would like to thank my direct supervisors **Dr. Laura Ferranti** and **Dr. Javier Alonso-Mora** for always thinking with me, for providing feedback, support, and for making me aim high. I consider myself very fortunate to have worked under your supervision. I would additionally like to thank my promotor **Prof. Dr. Dariu M. Gavrila** for providing detailed and constructive feedback. I thank the results of this thesis to the support of you all.

To the members of my graduation committee, I would like to express my gratitude for being part of my committee, for your reading efforts and for providing feedback, **Dr. ir. Paolo Falcone**, **Prof. Dr. Tamas Keviczky**, **Dr. ir. Emilia Silvas** and **Dr. ir. Julian F. P. Kooij**. Special thanks to **Dr. ir. Paolo Falcone** for traveling to the Netherlands to join the committee in person. I furthermore would like to thank my paranymphs **Daniel** and **Khaled** for supporting me during the defense.

I want to thank all the PhDs and research staff that I have interacted with over the last few years. Having started during COVID, I realize that these interactions are what makes doing research engaging and enjoyable. For those early interactions, and for introducing me to robotics and the PhD, I want to thank **Bruno** and **Hai**. Thanks **Alvaro** for helping me find the office the first day after COVID... (Let's have a coffee/beer later), thanks **Maxi** and **Max (Spahn)** for the engaging discussions early on. Thanks also to **Ewoud**, **Andras**, **Jork**, **Frank** and **Ronald** for your company during our Prius live demo in 2020.

The most enjoyable projects, for me, are those that happen in collaboration. For that I would like to thank **Dennis** and **Thijs**, and also **Tasos**, for collaborating on our MPC code and for your ideas and discussions. Thanks **Diego** and **Luzia** for our lab sessions, trying not to get hit by the robot. Thanks to **Ronald** and **Mario** for endlessly picking up our dummy Hans in the pouring rain and for your patience and support in trying to get our planner up and running on the Prius. Thanks also to **Khaled**, **Rodrigo**, **Bruno**, **Leon** and **Chris** for your past and ongoing collaboration and discussions.

As robotics happens in the real-world, real people are needed to test them. I thank the experimental results in this thesis to the tireless colleagues that were willing to play pedestrian, and whom I may or may not have thanked with stroopwafels already. Thanks **Dennis**, **Anna**, **Khaled**, **Thijs**, **Luzia**, **Lorenzo**, **Bruno**, **Anish** and **Sanne**.

To the colleagues at AMR I have not yet named I owe many insightful and fun discussions. For that, thanks **Elia**, **Lasse**, **Max (Lodel)**, **Max (Spahn)**, **Anna**, **Andreu**, **Daniel**, **Saray**, **Tomáš** and **Niels**. Thanks to colleagues at the R2C lab, **Loreeenzoo**, **Manueeel**,

Tasos, **Bas** and **Jelle**. Thanks finally to the colleagues at IV, **Hidde**, **Alberto** and **Wilbert**, for the lunches at IO, and thanks **Ted**, **Tugrul**, **Vishrut**, **Varun**, **Chrys**, **Shiming**, **Yanggu**, **Mubariz** and **Emma**. I want to finally thank my MSc students, **Anish**, **Victor** and **Weiming** that I was fortunate to supervise. Thank you all for your dedication, the discussions and your collaboration.

I wish all of you the best of luck either in the rest of your PhD or, for those that already graduated, in the rest of your careers.

My friends and family made the last four years especially worthwhile. It is thanks to your company outside of work that I enjoyed my PhD. For that I would like to thank you all. For all the laughs, spelletjes and ijsjes, before and during the PhD, in Delft and in Amsterdam, I want to thank my friends and h(m)ousemates: **Corne** and **Johan**, my (almost) housemates: **Jari**, **Lennert** and **Maarten**, and **Jeroen**, **Alex**, **Marloes**, **Sander** and **Rogier**. Thanks **Daniel** for the great discussions over coffee.
I am very grateful to my family for their support throughout my life. Thanks **mom** and **dad**, I would not be here without the things I learned from both of you. Thanks **Jo** and **Cas** for being the best sister and brother I could have, and thanks **Resy** and **Teun** for being part of the family. Thanks to the rest of my family, **Pauline**, **Marcel**, **Meike**, **Wessel**, **Helen**, **Oma Blok**, **Joop**, **Dorothy**, **Peter**, **Lisanne** and **Oma Juul**.
Meeting my love, Sanne, was the most important thing that happened to me in the last four years. I want to thank **Miranda**, **Dick** and **Loes**, that have since welcomed me wholeheartedly in the family and thanks also **Annie**, **Ilse**, **Ruud**, **Romy**, **Ruben**, **Kim** and **Nick**. Finally, for being my everything, thanks **Sanne**. The last two years have been the best of my life. You are the kindest, most loving person I know. I hope that this is one among the many achievements that we will celebrate together. This thesis and the rest of my life is dedicated to *you*.

Contents

SUMMARY

Logistics and transportation can greatly benefit from the use of autonomous robots such as self-driving vehicles. Robots can help to move goods or people without human supervision. One of the main components that enable autonomous navigation among humans is motion planning. Motion planning is responsible for computing a collision-free trajectory that moves the robot to its destination, based on the perceived information. The motion planner should be efficient, robust and safe. This thesis contributes towards this goal by investigating the design of motion planning algorithms for autonomous navigation of mobile robots near humans.
Traditional motion planners for dynamic environments have two key limitations that this thesis aims to address. First, they assume that their model of dynamic obstacles (e.g., humans) is exactly correct, capturing it with a single deterministic prediction. In practice, the robot cannot observe human intentions and must account for its uncertainty about the human's future behavior. Second, motion planners usually compute a single trajectory around an obstacle as result of previously taken decisions without exploring alternative options. They react slowly or even fail to find a solution when unpredicted changes make this path undesirable. This results in poor planning performance in dynamic environments.
The goal of this thesis is to develop motion planners that account for the uncertainty of human motion predictions and that are consistent and robust in their decision-making in order to deal with unpredicted changes in dynamic environments. To accomplish this goal, this thesis proposes two motion planning frameworks: *scenario-based* and *topology-driven* trajectory optimization.

The first contribution of this thesis is Scenario-based Model Predictive Contouring Control (S-MPCC), a real-time capable probabilistic planning framework that incorporates any uncertainty associated with the motion predictions of dynamic obstacles. Contrary to existing methods that only account for small variations around a single predicted trajectory (unimodal uncertainty), the proposed planner accounts for multiple possible trajectories (multi-modal uncertainty). The planner therefore safely accounts for several outcomes, for instance, to express that a pedestrian may or may not cross in front of the robot. S-MPCC bounds the probability of collision in each time step with all obstacles through Chance-Constrained Optimization (CCO). The CCO problem is reformulated as an optimization without uncertainty by sampling trajectories from the predicted distribution, known as scenarios. Each scenario represents a possible position of all obstacles in one time step and the planner avoids collisions with all scenarios. This Scenario Program (SP), through a tailored linearization, can be solved efficiently online. S-MPCC therefore plans probabilistic safe trajectories independent of the underlying distribution of the uncertainty.

S-MPCC considers the probability of collision separately for each time instance in the planned trajectory. The second contribution of this thesis, Safe Horizon Model Predictive Control (SH-MPC), builds on S-MPCC to constrain the joint probability of collision with all obstacles over the duration of the planned trajectory. Existing methods that separately constrain the probability of collision in each time step (temporal marginal) and with each obstacle (obstacle marginal) lead to overly cautious motion planning when safety constraints are enforced. SH-MPC formulates a single chance constraint to bound the overall probability of collision. This CCO is reformulated as an SP where each scenario represents a possible trajectory for all obstacles. To certify the joint probability of collision with the SP, the number of scenarios that affect the motion plan need to be identified. SH-MPC estimates this quantity at a negligible computational cost during optimization. Consequently, SH-MPC plans trajectories in real-time under generic uncertainties that are less cautious than existing methods without compromising on safety.

The probabilistic safety of S-MPCC and SH-MPC is linked to the underlying accuracy of the prediction model of the obstacles that provides the scenarios. As third contribution, a joint prediction and planning framework, Partitioned Scenario Replay (PSR), is proposed that replays past observations of human motion as scenarios for scenario-based planning. PSR does not fit a distribution on observed data, but directly uses the data as empirical evidence of the underlying uncertainty and thereby provides a real-world safety guarantee.

A key limitation of the developed scenario-based planners and other optimization-based planners is that they locally refine an initial trajectory. This initial trajectory largely determines the quality of the final trajectory, while it does not consider other options. The fourth contribution of this thesis is Topology-driven Model Predictive Control (T-MPC) that concurrently optimizes trajectories, each attempting a different way to pass the obstacles. T-MPC is composed of a guidance planner and several parallel local planners. The guidance planner identifies guidance trajectories for several distinct maneuvers, relying on results from topology to distinguish trajectories. Each local planner is composed of an existing optimization-based planner (e.g., a scenario-based planner) and an additional set of constraints that are derived from one of the guidance trajectories. The guidance trajectories are optimized by the local planners in parallel and the results are compared to determine which trajectory gets executed. T-MPC is faster, more consistent and safer than several state-of-the-art planners. Contrary to similar existing work, it does not rely on an explicit lane structure and therefore enables both urban driving and mobile robotic applications.

The motion planners developed in this thesis are extensively validated in simulation and in experiments with a small-scale mobile robot and a full-scale self-driving vehicle navigating among pedestrians. The robot agnostic implementation of the proposed planners that were developed for this thesis are available open source.

SAMENVATTING

De opkomst van autonome robots en zelfrijdende auto's kan veel betekenen voor transport en de logistieke sector. Robots kunnen helpen bij het transporteren van goederen of mensen zonder dat menselijk toezicht nodig is. Één van de basis componenten die autonome navigatie rond mensen mogelijk maakt is *Motion Planning*. Deze component berekent, op basis van informatie van de omgeving, hoe de robot zich zonder te botsen naar zijn bestemming kan bewegen. De motion planner zou efficiënt, robuust en veilig moeten zijn. Deze dissertatie draagt hier aan bij door nieuwe motion planners to ontwikkelen voor autonome navigatie van robots in de nabijheid van mensen.
Traditionele motion planners in dynamische omgevingen hebben twee belangrijke limitaties die deze dissertatie ten doel stelt om te verhelpen. Ten eerste, nemen ze aan dat het model van dynamische obstakels (bijv. mensen) precies klopt. Dat model gebruikt dan een enkele deterministische voorspelling. In werkelijkheid kan de robot de intenties van mensen niet observeren en moet hij rekening houden met de onzekerheid die gepaard gaat met het voorspellen van menselijke gedrag. Ten tweede berekenen motion planners normaal gesproken een enkel traject om een obstakel te ontwijken waarin de beslissing in de vorige tijdstap bepalend is voor hoe het obstakel wordt ontweken. Alternatieve ontwijkende trajecten worden daarbij niet meegenomen. Dit resulteert doorgaans in langzame voortgang van de robot of kan leiden tot het falen van de motion planner als er onverwachte veranderingen plaatsvinden in de omgeving waardoor het vorige traject niet mogelijk is. Dit alles kan er toe leiden dat de motion planner niet goed werkt in dynamische omgevingen. Het doel van deze dissertatie is om motion planners te ontwikkelen die de onzekerheid in het voorspellen van menselijke beweging meenemen en die consistent en robuust beslissingen nemen zodat ze om kunnen gaan met onverwachte veranderingen in dynamische omgevingen. Om dit doel te bewerkstelligen worden in deze dissertatie twee motion planning methodes voorgesteld: *scenario-based* en *topology-driven* traject optimalisatie.

De eerste bijdrage van deze dissertatie is Scenario-based Model Predictive Contouring Control (S-MPCC), een real-time probabilistische planning methode die de onzekerheid in het voorspellen van de beweging van dynamische obstakels meeneemt. In tegenstelling tot bestaande methodes, die alleen kleine variaties rond een enkel voorspeld traject meenemen (unimodale onzekerheid) neemt deze planner meerdere mogelijke trajecten mee (multimodale onzekerheid). De planner beschouwd daardoor meerdere uitkomsten, bijvoorbeeld, zodat hij mee kan nemen of een voetganger wel of niet voor de robot gaat oversteken. S-MPCC limiteerd de kans op botsingen in iedere tijd stap met alle obstakels door middel van Chance-Constrained Optimization (CCO). De CCO wordt geherformuleerd als een optimalisatie zonder onzekerheid door de onderliggende distributie van voorspelde obstakel trajecten te samplen. Zulke samples heten ook wel *scenarios*. Ieder scenario representeerd een mogelijke positie van alle obstakels in één

tijd stap en de planner ontwijkt botsingen met alle scenarios. De bijbehorende Scenario Program (SP) kan dankzij een toegespitsde linearizatie efficiënt opgelost worden terwijl de robot beweegt. S-MPCC kan daarvoor probabilistich veilige trajecten plannen los van de distributie van de onzekerheid die daarin meegenomen wordt.

S-MPCC beschouwd de botskans in iedere tijd stap van het geplande traject los van elkaar. De tweede bijdrage van deze dissertatie, Safe Horizon Model Predictive Control (SH-MPC) bouwt voort op S-MPCC door de gezamelijke botskans met alle obstakels in iedere tijd stap in één keer te limiteren. Bestaande methodes die de botskans los limiteren per tijd stap (temporaal marginaal) en met ieder obstakel (obstakel marginaal) leiden tot overdreven voorzichtige beweging van de robot wanneer veiligheid vereist is. SH-MPC formuleerd een enkele constraint om de gezamelijke botskans te limiteren. De bijbehorende CCO wordt geherformuleerd als een SP waar ieder scenario een mogelijk traject van alle obstakels representeerd. Om de gezamelijke botskans te limiteren met de SP moet het aantal scenarios wat invloed uitoefend op het motion plan bepaald worden. SH-MPC schat dit aantal af met een verwaarloosbare hoeveelheid rekenkracht tijdens het optimaliseren. Daardoor berekent SH-MPC, in real-time en onder generieke onzekerheden, een robot traject wat minder voorzichtig is dan bestaande methodes, zonder de veiligheid te verslechten.

De probabilistische veiligheid van S-MPCC en SH-MPC is gelinkt met de onderliggende nauwkeurigheid van het voorspellings model van de obstakels dat de scenarios levert. Als derde bijdrage stelt deze dissertatie Partitioned Scenario Replay (PSR) voor, een methode die zowel voorspeld als plant. Deze methode speelt eerdere waarnemingen van menselijke beweging opnieuw af als scenario's voor scenario-based planning. In plaats van een model te fitten op de data, gebruikt PSR de data direct als empirisch bewijs voor de onderliggende onzekerheid. Daardoor kan het een veiligheids garantie geven die in de werkelijkheid blijft gelden.

Een belangrijke limitatie van de ontwikkelde scenario-based planners en andere op optimalisatie gebaseerde planners is dat ze een initieel traject alleen lokaal verbeteren. Dit initiele traject heeft daarom grote invloed op het uiteindelijk berekende traject en niet alle mogelijke opties worden meegenomen. De vierde bijdrage van deze dissertatie is Topology-driven Model Predictive Control (T-MPC) dat tegelijk meerdere trajecten optimaliseerd, waar ieder traject de obstakels op een andere manier ontwijkt. T-MPC bestaat uit een begeleidende planner en een aantal parallelle lokale planners. De begeleidende planner identificeert begeleidende trajecten voor een aantal verschillende maneuvers. Daarbij maakt het gebruik van resultaten uit topologie om trajecten van elkaar te onderscheiden. Iedere lokale planner bestaat uit een bestaande optimalisatie planner (e.g., een scenario-based planner) en een extra set van constraints die afgeleid worden van een van de begeleidende trajecten. De begeleidende trajecten worden daardoor in parallel door de lokale planners geoptimaliseerd en de resultaten worden vergeleken om te bepalen welk traject de robot het best kan gebruiken. T-MPC is sneller, consistenter en veiliger dan een aantal state of the art planners. In tegenstelling tot vergelijkbaar werk hoeft de omgeving geen banen te hebben om de methode toe te passen zodat deze

toepasbaar is voor zelfrijdende autos in stedelijke omgevingen en voor mobiele robots.

De ontwikkelde motion planners zijn uitgebreid getest in simulatie en in experimenten met voetgangers waar een kleine mobiele robot en een zelfrijdende auto bestuurd worden. De robot agnostische software implementatie van de voorgestelde planners is publiek beschikbaar.

ACRONYMS

ADE Average Displacement Error

BMT Box-Muller Transformation

CADRL Collision Avoidance with Deep Reinforcement Learning

CC-MPC Chance Constrained Model Predictive Control

CCO Chance Constraint Optimization

CDF Cummulative Density Function

COLREGS Convention on the International Regulations for Preventing Collisions at Sea

CPU Central Processing Unit

CP Collision Probability

ECU Electric Control Unit

EKF Extended Kalman Filter

FDE Final Displacement Error

GMM Gaussian Mixture Model

GO-MPC Goal Oriented Model Predictive Control

GPU Graphical Processing Unit

HGO Homotopy Globally Optimal

IID Independent and Identically Distributed

LMPCC Local Model Predictive Contouring Control

MPCC Model Predictive Contouring Control

MPC Model Predictive Control

MPPI Model Predictive Path Integral

NSO Nonconvex Scenario Optimization

PRM Probabilistic Roadmaps

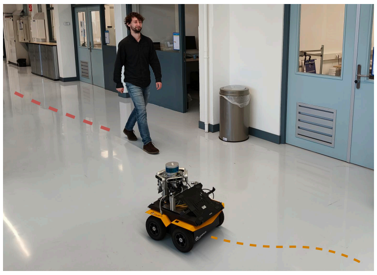
PSR Partitioned Scenario Replay

ROS	Robotic Operating System
S-MPCC	Scenario-based Model Predictive Contouring Control
SH-MPC	Safe Horizon Model Predictive Control
SHINE	Social Homology Identification for Navigation in Crowded Environments
SP	Scenario Program
SQP	Sequential Quadratic Programming
T-MPC	Topology-driven Model Predictive Control
TEB	Time Elastic Band
UVD	Universal Visibility Deformation
VRU	Vulnerable Road User

1 Introduction

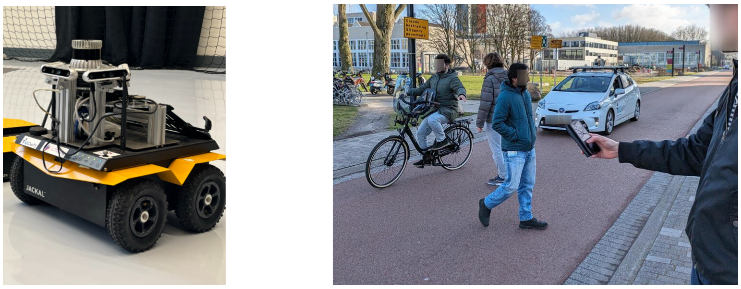
Before a human driver receives their driver's license, their decision-making is tested in challenging situations (see Fig. 1.1a). An expert driver intuitively knows what action to take in these situations. The driver understands the environment and how it will change over time, can estimate the behavior of the vehicle and knows how to react (brake, release the throttle or accelerate). Would a robot be able to make these decisions safely in the same way? This dissertation proposes new motion planning techniques that aim to address this question, constructively. This dissertation not only focuses on automated vehicles but also considers similar decision making problems for smaller mobile robots navigating among crowds (see Fig. 1.1b). Both types of robots face significant challenges as they move from controlled environments to more complex and dynamic environments where humans are present.

(a) A self-driving vehicle application. A cyclist seems to be crossing in front of the vehicle. Source: https://tests.quest.nl/vervoer/ken-jij-de-verkeersregels.



(b) A mobile ground robot application. Mobile robots could autonomously transport goods through crowded environments such as hospitals.

Figure 1.1: Two application domains of mobile robots.



(a) Mobile robot. (b) Automated vehicle.

Figure 1.2: Motion planning applications vary from (a) small scale mobile robotic platforms (driving or flying) to (b) full-scale automated vehicles.

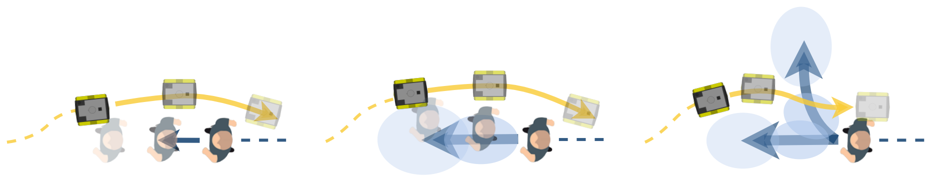
1.1. MOTIVATION

Mobile robots (see Fig. 1.2) have promising applications in automation and logistics, including warehouse automation [1], urban transportation [2] and maritime transportation [3]. In most applications where robots are currently deployed, the operating domain (e.g., warehouses) does not allow the robots to operate near humans. This restricts the usage of mobile robots to controlled environments created specially for them. Mobile robots can have a major impact on our society if they become capable of navigating among humans. There are still major challenges that need to be addressed before this can become a reality.

Autonomous navigation among humans is typically enabled through two main components: perception and motion planning. *Perception* is responsible for perceiving the robot's environment while *motion planning* is responsible for computing a collision-free trajectory that moves the robot to its destination, based on the perceived information. This dissertation focuses on motion planning and in particular, on designing strategies to avoid collisions with humans (i.e., dynamic obstacles).

Traditional planners regard humans as *deterministic*: they assume that future human motion is exactly known to the robot (see Fig. 1.3a). In reality, it is often not possible to exactly predict human behavior. First, because behavior varies per person. Consider, for instance, human drivers that have different driving styles or pedestrians that walk at varying speeds. Second, humans may have an intended destination in mind that the robot cannot observe. Without considering these variations, a planner may produce an unsafe action and collide.

Motion planners can produce safer actions by considering the uncertainty associated with human motion predictions. The uncertainty expresses possible outcomes together with the probability that they will happen. The probability distribution of the uncertainty can be uni-modal or multi-modal. *Uni-modal* distributions capture continuous variations with respect to expected behavior (see Fig. 1.3b) while *multi-modal* distributions additionally capture multiple discrete behaviors (e.g., whether a pedestrian crosses or not, see Fig. 1.3c). A common way to account for uncertainty in the planner is to limit the *probability* that a collision will happen. However, for generic (e.g., multi-modal) uncertainties, this is a complex task that has not been solved.



(a) Deterministic planning. (b) Planning under unimodal uncertainty. (c) Planning under multimodal uncertainty.

Figure 1.3: Motion planning methods illustrated on a simple problem with a mobile robot (yellow) and a pedestrian (blue). (a) Deterministic methods assume that they know exactly what the pedestrian will do. (b) By incorporating unimodal uncertainty (blue shaded regions), the planner accounts for continuous variations near an expected behavior. Consider, for example, that in the worst-case (depicted pedestrian positions) the planner keeps a safe distance. (c) The planner can account for multiple discrete behaviors (e.g., turning, not turning) by accounting for multi-modal uncertainty.

Next to safety, the planner should consider its performance criteria, such as its progress towards the goal. Standing still is generally safe for instance, but is not desirable. Balancing performance criteria and safety is a complex task. To reduce the complexity of this decision-making problem, the planner is usually composed of at least two components: a global and a local planner. The *global planner* finds a path to the destination, considering static obstacles (e.g., walls and the environment layout). The *local planner* follows this path while avoiding dynamic obstacles. Both planners still have severe limitations. The global planner usually does not consider dynamic obstacles and does not explicitly optimize the performance criteria, which leads to inefficient planning decisions. The local planner tends to plan a single trajectory, which may become unsafe or inefficient when unpredicted changes happen, leading to planning failure. This dissertation considers how motion planning algorithms can be made more adaptive and robust, by mitigating these limitations.

The goal of this dissertation is to *develop robust online motion planners for autonomous navigation in dynamic environments by planning multiple distinct trajectories that each account for generic uncertainties associated with the predictions of dynamic obstacles.*

1.2. BACKGROUND

The following provides a broad overview of the topics that concern this dissertation.

Optimization Optimization problems can be used in planning and decision-making to select decision variables (e.g., the robot's control action) based on a cost function and constraints. Its *cost function* specifies performance criteria (e.g., time-efficiency or tracking a desired speed) as a function of the optimization variables. The *constraints* specify what solutions are acceptable (e.g., collision-free). The decision variables that minimize the cost function can efficiently be found when the optimization problem is convex. However, robot motion planning problems are usually nonconvex because of the nonlinear robot dynamics and collision avoidance constraints. These nonconvex problems can be solved in real-time thanks to advancements in solving techniques but can only provide a locally optimal solution that is not necessarily the best decision for the specified problem. More details on numerical optimization algorithms can be found in [4].

Model Predictive Control The central planning technique used in this dissertation, Model Predictive Control (MPC) [5], applies optimization for control. In particular, MPC predicts the future over a finite time horizon, using a model of the robot dynamics and the environment, to make a decision for the current time instance (see Fig. 1.3). By repeatedly predicting forward in time, MPC can adapt to changes, for instance, when the environment does not exactly match its predicted state. The optimization problem is typically nonconvex for motion planning problems which means that the planned trajectory is one out of many local optimal trajectories. It may therefore not be the best trajectory for the specified problem. Constraints in the optimization problem enable MPC to plan trajectories that avoid collisions with dynamic obstacles.

Chance-Constrained Motion Planning An MPC that uses deterministic collision avoidance constraints does not consider the uncertainty in the motion of dynamic obstacles, which can lead to collisions. *Chance Constrained Optimization (CCO)* [6] allows constraints to be violated with a non-zero probability, known as the *acceptable risk*. This enables MPC to consider uncertainty when planning collision-free trajectories. CCO is difficult to solve in general. This dissertation considers whether human trajectories sampled from the underlying distribution can be used to reformulate the planning problem so that it can be solved in real-time.

Topology-Based Planning Each distinct way in which the robot can pass the obstacles usually corresponds to a local optimum for MPC. Topology is a mathematical field that studies the properties of space that are preserved under continuous deformations. For robot motion planning, its results can be applied to identify whether two trajectories pass obstacles differently [7]. Topology-based planning methods use this information to explore multiple local optima of the motion planning problem. This dissertation proposes a topology-based planning framework that explores multiple ways to pass *dynamic* obstacles to improve the robustness and performance of the motion planner.

1.3. CONTRIBUTIONS

The contributions of this dissertation are:

(1) **A real-time probabilistic motion planning framework that bounds the probability of collision with all obstacles over the duration of the planned trajectory independent of the probability distribution that describes their future motion.** The framework enables probabilistic planning under non Gaussian, possibly multi-modal uncertainty associated with the motion predictions of dynamic obstacles. Three planners are proposed within this framework:

(a) **A Scenario-based Model Predictive Contouring Control method (S-MPCC)** that bounds, in each time step, the marginal probability of collision with all obstacles. The proposed framework reformulates the collision avoidance constraints as linear constraints and samples these linearized constraints to obtain an optimization problem that can efficiently be solved online. Each sample (i.e., *scenario*) is associated with the possible positions of all obstacles in a particular time step. Simulation results on a mobile robot navigating among pedestrians showed that this method is safer and more time efficient than two baseline methods, while it can handle non Gaussian distributions.

(b) **A Safe Horizon Model Predictive Control method (SH-MPC)** that explicitly constrains the joint probability of collision with all obstacles over the duration of the planned trajectory. Different from S-MPCC, SH-MPC considers all time steps at once as each scenario represents trajectories of all obstacles for all time steps. To constrain the joint probability of collision, the scenarios that affect the solution to the optimization, known as the *support*, must be identified. SH-MPC includes a computationally efficient estimate of the support that enables real-time planning. Simulated and real-world results indicated that SH-MPC provides tighter probabilistic safety guarantees in crowded environments than marginal methods and that it applies seamlessly to non Gaussian probability distributions.

(c) **A joint data-based prediction and planning framework, Partitioned Scenario Replay (PSR)** that predicts human motion by replaying previously observed human trajectories and uses these replayed scenarios in a scenario-based planner. Because no modeling assumptions are made when real-world data is sampled directly, PSR provides a data-based safety guarantee on the probability of collision.

(2) A framework that concurrently optimizes multiple distinct trajectories with global guidance in dynamic environments, referred to as Topology-driven Model Predictive Control (T-MPC). Compared to state-of-the-art trajectory optimization methods, T-MPC reduces how often the planner becomes infeasible while computing lower-cost (i.e., higher quality) trajectories. By keeping track of the previously intended passing behavior and preferring this option in its decision making, it additionally plans more decisive motion in practice. T-MPC can directly build on top of existing local planners, such as the proposed scenario-based planners, to optimize multiple distinct trajectories in parallel. Simulations and real-world experiments showed that T-MPC outperformed several state-of-the-art methods in terms of safety and time efficiency.

Next to these main contributions, this dissertation developed the ROS/C++ software package `mpc_planner` that implements robot-agnostic MPC. The simulations and experiments in this thesis are all implemented by this software package. It is available at: https://github.com/tud-amr/mpc_planner. A self-contained environment with the planner is provided at: https://github.com/tud-amr/mpc_planner_ws.

This thesis studies scenario-based and topology-driven planning seperately. Which method is most suitable in practice depends on the requirements imposed by the environment. Topology-driven planning is generally applicable, while scenario-based planning is particularly useful when obstacle motion predictions are multi-modal. The two contributions could also be combined into a single planning framework where topology-driven planning allows the planner to compute multiple distinct trajectories and scenario-based planning allows each individual trajectory to account for multi-modal uncertainty.

1.4. RESEARCH BEYOND THIS THESIS

Next to the work in this thesis, other research topics were pursued jointly. The following provides a brief overview of these works.

Rule-Aware Trajectory Optimization for Autonomous Vessels The Convention on the International Regulations for Preventing Collisions at Sea (COLREGS) imposes rules for interactions between vessels. In [8], constraints are proposed for trajectory optimization that comply with the COLREGS when avoiding other vessels. This work shows how explicit rules on interactions can be encoded in MPC.

Probabilistic Risk Assesment for Chance Constraint Collision Avoidance The theory for the scenario-based methods developed in Chapters 3 to 5 can be conservative in practice, leading to overly cautious trajectories. In [9], this conservatism is reduced by running several planners in parallel where some of the deployed planners are not theoretically safe. The safety of the planned trajectories are then assessed a posteriori. A trajectory that was theoretically unsafe, may be safe in practice and typically has better performance. The results in [9] therefore allow scenario-based planners to find more efficient trajectories without compromising on safety.

Learning Navigation Decisions from Humans When navigating crowds, humans continuously make navigation decisions, for example, whether to pass on the right or left of other humans. Chapter 6 computes such options for autonomous navigation. However, it decides what option is best based on a cost function that is tuned by hand. In [10], a dataset with human trajectories is used to learn which option a human would prefer. In [11], a similar approach is taken to learn how fast a self-driving vehicle should drive. In this work, the MPC controls a self-driving vehicle in simulation. The algorithm learns a reference velocity from the drivers vision and feedback of a human supervisor. This allowed the vehicle to overtake other vehicle and to slow down in potentially dangerous situations. Both of these methods learn high-level navigation decisions from humans to behave more socially using MPC.

1.5. ORGANIZATION

Literature related to this dissertation is first reviewed in Chapter 2. The main contributions then follow, organized into three sections (see Fig. 1.4).

Planning Under Uncertainty Chapters 3-5 address how generic uncertainties associated with dynamic obstacles can be incorporated into the motion planner. Chapter 3 presents a sampling-based approach to bound the probability of collision with all obstacles in each time step of the planner. Chapter 4 further extends this approach by bounding the probability of collision with all obstacles over the duration of the motion plan. Chapter 5 introduces a data-driven generic motion prediction framework suitable for supplying predictions to the previously introduced sampling-based planners.

Global Guidance Chapter 6 addresses how the planner can make high-level navigation choices in dynamic environments. It proposes two planning components: a high-level guidance planner and a set of local motion planners. The local planners can incorporate uni-modal uncertainties and link to the methods proposed in Chapters 3-5, while the high-level component improves overall navigation behavior.

Application to a Self-Driving Vehicle Chapter 7 implements the techniques developed in Chapters 3-6 on a full scale self-driving vehicle. These techniques are demonstrated for autonomous summoning of a vehicle in the presence of Vulnerable Road Users (VRUs) such as pedestrians and cyclists.

Chapter 8 draws conclusions and highlights recent developments and future works.

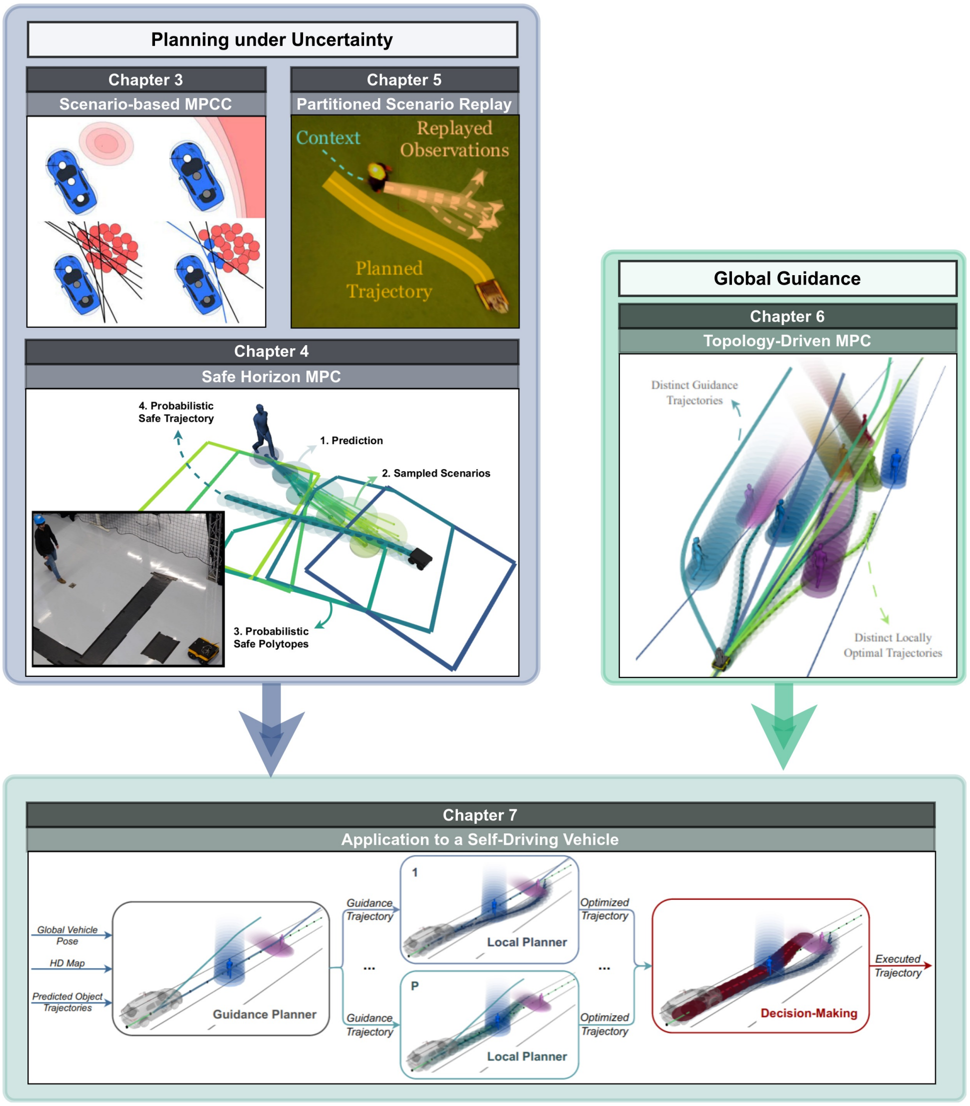


Figure 1.4: An overview of the chapters in this dissertation.

2

Related Work

This chapter reviews motion planning algorithms for dynamic environments related to the work of this dissertation. First, an overview of the general motion planning methods is given in Sec. 2.1. Then, trajectory optimization methods are considered in detail in Sec. 2.2. Methods to predict human motion are reviewed in Sec. 2.3. We finally review several experimental applications of self-driving vehicles in Sec. 2.4 and draw conclusions in Sec. 2.5.

2.1. Overview of Motion Planning Methods

A motion planner computes collision-free and time-efficient trajectories to move a robot from its current position to its destination. Motion planners can be categorized as reactive or predictive planners depending on how far ahead in time they compute their plan. The remainder of this section provides an overview of these planners in more detail, highlighting their strengths and weaknesses in the context of this work.

2.1.1. Reactive Planners

Reactive planning methods take decisions based on their current perceived knowledge of the environments using a short planning window. Examples of reactive planning methods are Dynamic Window Approach (DWA) [12], Optimal Reciprocal Collision Avoidance (ORCA) [13], [14] and Potential Fields [15]. DWA [12] singles out a collision-free circular trajectory with a constant rotational and translational velocity that best moves the robot to the goal while keeping its distance from obstacles. ORCA [13] and Non-Holonomic ORCA (NH-ORCA) [14] try to find the velocity closest to a robot's desired velocity that is collision-free for a short time horizon. Potential Fields [15] follow the gradient of a potential function that attracts towards the goal and repulses from obstacles. Reactive methods can respond quickly to changes it the environment as they do not plan far ahead in time. This makes them well suited for low-speed low-complexity scenarios, but leads to non-smooth and time-inefficient motion in scenarios with higher complexity.

2.1.2. PREDICTIVE PLANNERS

Predictive planners were proposed to tackle scenarios with higher complexity and to produce smoother, faster and safer robot motion. These planners plan their motion further ahead in time. Several state-of-the-art predictive methods are reviewed in the following.

Input Space Discretization Several methods discretize the input space. For instance, *Motion primitive* planners [16]–[18] generate a large number of trajectories that are dynamically feasible by construction. A single trajectory can be singled out by verifying the safety and performance requirements on all sampled trajectories and choosing that of the lowest cost. Motion primitives planners are simple and computationally efficient but discretize the possible maneuvers. This can lead to infeasibility and time-inefficient robot motion.

Sampling-based Planning Planning methods such as Rapidly exploring Random Trees (RRT) [19] and Probabilistic Roadmaps (PRM) [20] plan by randomly sampling and connecting states in the configuration space until a goal configuration is reached. Closed-Loop RRT (CL-RRT) [21] proposed an RRT that samples inputs to a stable closed-loop system consisting of the robot and a controller. RRT^{x} [22] continuously rewires the graph to adapt to dynamic environments. Recent work [23] greatly improved the computational efficiency of sampling-based planners for high-dimensional problems by using topological abstraction over fiber bundles. Sampling-based methods have the advantage that they work well in high-dimensional state spaces, but generally plan non-smooth trajectories. When dynamic constraints are enforced (i.e., trajectories need to be smooth), their sample efficiency drops, which often leads to time-inefficient trajectories.

Learning-based Planning Deep Reinforcement Learning (DRL) learns a navigation policy by training a neural network to plan collision-free and time-efficient trajectories. The policy is trained by running the planner in simulation on the intended navigation task. A reactive example of this type of method is Collision Avoidance with Deep RL (CADRL) [24] which learns to identify a collision-free velocity for social navigation. Social Graph-based Double Dueling Deep Q-Network (SG-D3QN) [25] introduces a DRL approach with predictive qualities to navigate crowds. DRL approaches have significantly improved in recent years and are well suited for social navigation problems that may be hard to tackle with model-based approaches. However, they remain limited to the navigation task that they were trained in and suffer from a sim-to-real-gap.

Trajectory Optimization Finally, this dissertation focuses on optimization-based planning methods, discussed in detail in the following section.

2.2. TRAJECTORY OPTIMIZATION

Trajectory optimization methods, in particular, Model Predictive Control (MPC) [5], [26]–[29] formulate trajectory planning as a nonlinear optimization problem where performance (e.g., progress and smoothness) is optimized under constraints (e.g., dynamic constraints and collision avoidance). Consequently, trajectory optimization methods allow one to plan high-quality smooth trajectories and incorporate the vehicle and environment model. These methods consider all possible actions (contrary to input discretization methods), plan smooth trajectories (contrary to sampling-based planners) and do not rely on the availability of a suitable data set (contrary to learning-based methods). These advantages make them applicable in a wide variety of applications [8], [30]–[34]. However, they result in locally optimal trajectories. The following sections focus on the modeling of the obstacles and their future motion, and how they are incorporated when planning collision-free trajectories. The following sections in particular consider deterministic approaches (where obstacle future motion is assumed to be known), interaction-aware approaches (where the interactions between obstacles and the robot are modeled) and probabilistic approaches (where uncertainty is considered in the future motion of obstacles).

2.2.1. Deterministic Approaches

Obstacles and their predicted future motion are typically incorporated in the optimization problem through nonconvex constraints. For example, by modeling the vehicle with a set of discs and obstacles with ellipsoids [5], [27]. An alternative is to linearize the constraints from the vehicle perspective, which leads to convex constraints in the state, but a limited feasible region. When rules are enforced on interaction (e.g., passing on the left), tailored collision avoidance constraints may be applied to enforce a particular behavior [8].

Limitations

Although collision avoidance in the constraints allows robots to avoid obstacles in most cases, it comes with limitations that affect planning performance.

- **Local Optimality:** The collision avoidance constraints make the optimization problem nonconvex. The planned trajectory is therefore only locally optimal.
- **Infeasibility:** The optimization problem can get infeasible, in which case no solution is returned.

Local optimality occasionally results in poor (e.g., slow or unsafe) trajectories. Additionally, when the initial guess of the optimization problem is not consistent over multiple planner cycles, the planner can repeatedly switch how it passes the obstacles, leading to indecisive behavior. Infeasibility, that can be caused by local optimality, prevents the robot from taking action and can lead to collisions. In the following, we review methods that aim to resolve these key limitations.

Convex Collision Avoidance

Since local optimality is a consequence of the optimization problem being nonconvex, it could be mitigated by formulating a convex problem. This is difficult, however, due to the nonlinear vehicle dynamics and nonconvex collision avoidance constraints. In [35], [36], Graphs of Convex Sets (GCS) are used to find almost globally optimal solutions to collision avoidance problems using a mixed-integer problem. However, it is not yet real-time and imposes limitations on the trajectory end point, supported dynamics and constraints.

High-Level Planning

A more practical alternative is to optimize more than a single trajectory, with the intent to compute multiple locally optimal trajectories. In [37], multiple trajectories are computed, each for a different cost function. It provides more trajectories, but not multiple local optima, making it hard to tune. Most authors [38]–[46] instead introduce a high-level planning layer that proposes a set of trajectories to be processed via trajectory optimization. These generated trajectories can be incorporated in the trajectory optimization as initial guess, through the objective function, or as constraints. Examples of high-level planners are: mixed-integer planners [38], Partially Observable Markov Decision Processes (POMDP) [39], graph-search based methods [40]–[43] and global planners [44]–[46]. Both the mixed-integer planner and POMDP are computationally expensive. The POMDP is only tractable when heuristically pruning the generated trajectories for selected cases.

Topology-Based High-Level Planning

Graph-based methods and global planners provide more efficient solutions. Both types of methods are based on the idea that local optima related to collision avoidance link to the topology of trajectories through the collision-free space. Roughly speaking, two trajectories are in the same *homotopy* class if they can be smoothly transformed into each other in the collision-free space [7] (e.g., when they evade the obstacles on the same side). Graph-based methods [40]–[43] leverage the structure in the environment, for instance, a lane structure in highway driving, to compose the environment into a graph. A graph search then provides the shortest P paths that are in distinct homotopy classes by construction. Without structure in the environment (e.g., in urban driving or mobile robotic applications), it is difficult to compute a trajectory in each homotopy class. Graphs can be constructed from static obstacles. In [47], Delauney triangulation is used to identify passable gaps between dynamic obstacles in a global planner. Voronoi graphs are used in [48] to identify homotopy classes with respect to static obstacles and in [45] include each dynamic obstacle and their predicted motion as a static obstacle. Trajectories are generated directly from the description of the homotopy class in [49] by modeling interactions as a physical vortex system. These graph-based and generative approaches can exhaust the possible homotopy classes, but scale poorly to crowded environments.

Instead, several works, such as [7], [44]–[46], compute distinct trajectories by filtering out homotopy equivalent trajectories during planning. For 3-D navigation among static obstacles, [44] introduces Universal Visibility Deformation (UVD) to compare trajectories. Trajectories are UVD equivalent if they can be connected without collision at several intermediate times. The authors present a Visibility-PRM [50] algorithm to generate UVD-distinct trajectories. In 2D dynamic environments, homotopy classes are typically compared via winding numbers [51] or the H-signature [7]. *Winding numbers* track the relative angle between the robot and dynamic obstacles over their trajectories. They were used in [48] to distinguish homotopy classes of trajectories with respect to dynamic obstacles. In [52], an MPC with winding numbers in the cost function was proposed to motivate passing progress. The *H-signature* uses *homology* classes as an approximation for homotopy classes. In [45], this approximation is applied for 2D navigation among static obstacles. Their planner, Time Elastic Band (TEB) Local Planner, identifies several trajectories in distinct homology classes (using regular PRM) and uses each to initialize a soft-constrained optimization-based planner. TEB is however limited to static obstacles, the global planner needs to reach a single goal and the underlying trajectory optimization is soft-constrained.

Sampling-Based Optimization

Another solution is to use sampling-based solvers for the trajectory optimization, instead of relying on the gradient. Model Predictive Path Integral control (MPPI) [53] samples inputs and evaluates the cost of the realized trajectories. By biasing the input distribution [54] or projecting samples [55] the local planner can escape local optima and attain better performance.

Reducing Infeasibility

The previously presented methods that reduce local optimality also help improve feasibility as the planner is less likely to be infeasible when multiple trajectories are optimized in parallel. The optimization may however still get infeasible because of incorrect perception information, too strict collision avoidance requirements or modeling errors. To mitigate infeasibility with MPC, some authors propose to use two trajectories where one features as a contingency plan [56]–[59] improving planner safety. The planner may still perform poorly when the contingency plan is activated.
A promising solution is to relax the least important constraints when the optimization is infeasible, such that the trajectory optimization returns the least harmful action. Versions of this idea have been proposed in [60], [61] but remain largely theoretical.

2.2.2. Interaction-Aware Planning

Next to the aforementioned limitations, standard trajectory optimization methods assume that obstacle motion is independent of the robot's action. In practice, humans change their behavior depending on the robot's behavior. The planner should account for this interaction. This problem is challenging, however, as the robot and human trajectories influence *each other*.
Traditional methods compute motion predictions for traffic participants and pass these to the motion planner [62]. If the predictions account for the robot's intended trajectory (e.g., [63]), then one can try to run several iterations of the decision loop as previously mentioned. Although it has been shown that the loop converges, under assumptions on the underlying prediction algorithm [64], this approach remains computationally intractable [33].

Accounting for Interaction

Sampling-based optimization offers a promising approach to include interaction. With MPPI, planning of other traffic participants can be performed jointly with that of the robot [65] or a prediction can be obtained for each sample and can be used to evaluate its cost.
Another solution is to first use a high-level interactive planner, similar to previously discussed high-level planners, to resolve how each agent should pass each other. Each plausible option can then be refined locally at a low computational cost. This method was

recently explored in [33] by using a topology-based guidance planner and local optimization. It is however not real-time and relies on structure in the environment. Similar approaches for multi-agent navigation, using topological braids, were proposed in [52], [66].
Game Theoretic approaches, such as [67], plan interaction-aware joint plans for all agents in the scene, assuming that other agents are rational. They circumvent the sequential prediction and planning problem by solving directly for equilibria of the joint problem. These methods however remain computationally intractable in real-world settings.

SOCIALLY-AWARE PLANNING

An implicit approach to interaction-aware navigation is to learn social behavior from humans. Goal Oriented MPC (GO-MPC) [68], [69] learns socially acceptable subgoals for a predictive controller. H-TEB [70] learns the topology class of socially acceptable passing maneuvers from human demonstrations using Inverse Reinforcement Learning (IRL). Similarly, Social Homology Identification for Navigation in crowded Environment (SHINE) [10] uses a deep neural network to learn social passing behavior from a dataset with human motion. All of these methods encode social behavior in the planner by learning directly from humans.
More research is necessary to evaluate social planners in practice. Several test benches for social navigation have recently been proposed [71]–[73] and a set of guidelines were put forward in [74].

2.2.3. PROBABILISTIC SAFE TRAJECTORY OPTIMIZATION

Due to large and multi-modal uncertainties in human motion prediction, collision avoidance with dynamic obstacles may, in the mean or nominal case (e.g., constant velocity), lead to collisions in practice. Many works therefore consider how to address this uncertainty. In optimization, uncertainty can be incorporated in the objective function to optimize performance considering the uncertainty (i.e., risk-aware) or in the constraints to ensure safety under uncertainty (i.e., chance-constrained).

RISK-AWARE PLANNING

Traditionally, *risk-aware* methods computed the expectation of the cost (e.g., [75]). The performance of these methods may still be poor when the distribution has long tails. The conditional Variance at Risk (cVaR) instead optimizes for the average worst-case, which is more suitable for robotics [76]. Several works have applied the cVaR for motion planning [77]–[79]. Although cVaR can improve performance under uncertainty, it does not guarantee safety. The remainder of this section therefore focuses on chance-constrained methods.

COLLISION-AVOIDANCE CHANCE CONSTRAINTS

The constrained trajectory optimization problem can be seen as a special case of optimization under uncertainty, for which two common approaches exist. *Robust optimization* [80] requires the constraints to be satisfied for all possible realizations of the uncertainty. An example of this is formal verification [81]. *Stochastic optimization* (see [82] for an overview) allows for a violation of the constraints, as long this happens with a probability smaller than an upper bound ϵ. Because the set of all possible realizations is often not available or too conservative, the remainder of this section focuses on stochastic optimization. More details on both methods in the context of Model Predictive Control (MPC) can be found in [83].

The probability of constraint violation in stochastic optimization is specified through *chance constraints*, which constrain the *probability* that a nominal constraint is satisfied. Exact evaluation of these chance constraints is, however, intractable in almost all applications. Existing works therefore focus on an approximation of the constraints through additional assumptions on the probability distribution associated with the uncertainty (e.g., Gaussian [84], [85]) or on the controlled robot (e.g., linear dynamics [86]). Recent works [87], [88] have resolved many of the assumptions, making the framework applicable to nonlinear robot dynamics and arbitrary probability distributions. However, the chance constraints in these and many other works are not imposed on the robot's planned trajectory, i.e., the timed sequence of planned positions, but rather on each of its individual positions over the planning horizon. This fails to accurately constrain the probability of colliding at *any* time[1]. In this regard, three types of chance constraint formulations have been considered in previous work: *Marginal*, *Conditional* and *Joint*.

Marginal Chance Constraints

Constraints on each position along the trajectory are referred to as *marginal* chance constraints. Let event A_k denote the case that no collisions occur at time k and $\mathbb{P}(A_k)$ therefore be the probability that the robot is safe at timestep k. Then the exact probability that a trajectory is safe over N steps is given by $\mathbb{P}(A) = \prod_{k=1}^{N} \mathbb{P}(A_k \mid A_{0:k-1})$. That is, the CP for each position is conditional on the probability of avoiding collisions up until the position is reached at time k. The problem is simplified if it is assumed instead that this event is independent for all states ($\tilde{\mathbb{P}}(A) \approx \prod_k \mathbb{P}(A_k)$). In [89], these marginal methods are further divided into *additive* and *multiplicative* approaches.

Additive approaches impose constraints on each marginal probability (i.e., $\mathbb{P}(A_k) \leq \epsilon_k$). Using Boole's inequality ($\mathbb{P}(\cup_k A_k) \leq \sum_k \mathbb{P}(A_k)$) the CP of the trajectory is bounded by the sum of the individual CPs. Under Gaussian uncertainty, the work [86] reformulated the constraints as an analytical constraint on the 1D Cumulative Density Function (CDF). The same idea is used in [84] in an MPC framework to prevent collision between robot and obstacle volumes. In [90], the bound on each marginal probability is updated, known as *risk allocation*, while maintaining the same total risk bound (i.e., $\sum_k \epsilon_k = \epsilon$). In [91], [92], marginal chance constraints are applied to the Rapidly expanding Random Trees (RRT) algorithm such that each node in the tree is statistically safe. When the uncertainty is non Gaussian, the CDF of the probability distribution is typically not available. In [87], [93], an MPC for motion planning is formulated where inequalities are posed on stochastic moments of the marginal probability distribution. A similar approach for linear dynamics is applied in [77] where the conditional Variance-at-Risk (cVaR) is used to minimize constraint violation.

The *multiplicative* formulation explicitly constrains the product of the marginal probabilities. It was applied in [94] to plan motion under sensing and actuation uncertainty. An alternative marginal formulation is proposed in [85], which bounds the largest

[1]Similarly, chance constraints imposed per obstacle fail to estimate the probability of colliding with *any* obstacle.

marginal constraint violation.
The limitation of marginal approaches is that the bound on the CP of the trajectory is inaccurate, as noted by [95] and [89]. It is shown in [89] that the trajectory CP approaches ∞ and 1 for the additive and multiplicative formulation, respectively, when the number of evaluations in the trajectory increases, regardless of the real CP. Marginal constraints only asses the risk correctly for the first time step and a single obstacle. The risk of the remainder is under- or overestimated. Overestimation of the risk and the associated unsafe space along the time horizon can cause the planning problem to become infeasible and may cause the robot to freeze. Due to these limitations, [95] conditioned marginal chance constraints on being collision-free at prior times and evaluated them by truncating the part of the distribution in collision in each time instance. This formulation is more accurate but is limited to Gaussian distributions.

Joint Chance Constraint

Some authors formulate a *joint* chance constraint on the CP of the planned trajectory. Joint chance constraints estimate the open-loop risk over the time horizon more accurately, making it less likely that the problem becomes infeasible and improving performance. The joint CP can be evaluated by using sampling-based methods [89]. In particular, prior works [89], [96], [97] consider Importance Sampling Monte-Carlo (ISMC) sampling to *approximate* the CP. An empirical estimate of the constraint violation is obtained by sampling the joint distribution and evaluating the constraint for each sampled *trajectory*. ISMC methods are well suited for estimating risk but are computationally expensive when planning a trajectory. An alternative is to formulate a mixed-integer problem (e.g., [96]) to decide which samples may be violated, but these problems are hard to solve in real-time.

Scenario Optimization

In this dissertation, *Nonconvex Scenario Optimization (NSO)* [98] is considered for evaluating the joint chance constraint through sampling. Scenario optimization is well established for convex optimization under uncertainty (see e.g., [99]–[102]) and has been applied to planning [103]. The framework was extended to the general nonconvex case in [98]. NSO is a sampling-based framework for optimization under uncertainty, similar to ISMC, but instead of *averaging* the samples, it *constrains* the solution to the samples, which is computationally more efficient when planning trajectories.

Distributionally Robust Optimization

A remaining problem with probabilistic safe methods is that they assume that the predicted probability distribution for obstacle motion is accurate. Several works [78], [104] have proposed to use the Wasserstein metric, which qualifies the difference between distributions, to make planners robust to this assumption. Although a promising idea, these approaches are not yet real-time capable.

2.3. Human Motion Prediction

Before planning a safe motion, the robot needs to predict the motion of nearby humans and the associated uncertainty. In the following, human trajectory prediction methods are reviewed, with a focus on pedestrian prediction. In general, these methods can be categorized by how they incorporate models and data. The planner presented in this thesis is agnostic to the prediction method employed.

Model-Based Human Motion Prediction

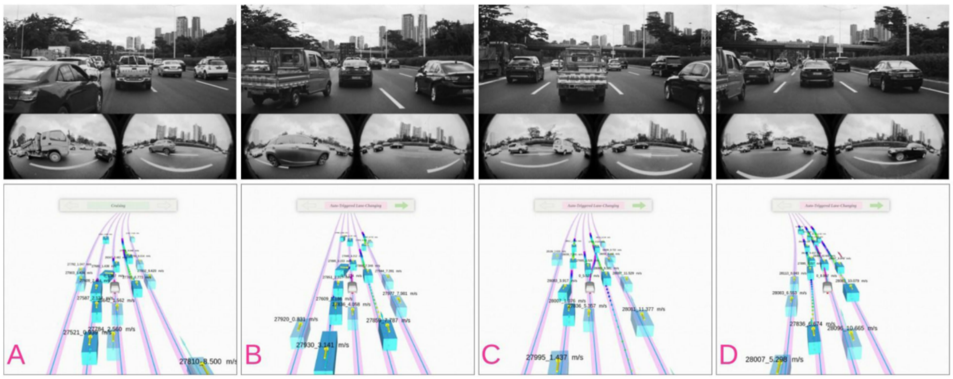
Model-based methods such as constant velocity or physics-based rules predict human trajectories based on model-based approximations and are popular for their simplicity [105], [106]. They fail, however, to capture contextual information.

Planning-Based Human Motion Prediction

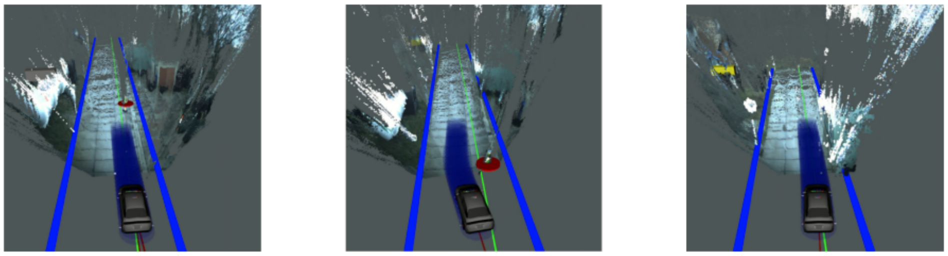
Planning-based methods apply motion planning methods from the perspective of the human to predict the human's future motion. In [107], pedestrian predictions are modeled by a relaxed maximum value Markov Decision Process (MDP) that infers pedestrian goal locations from previously observed trajectories. These predictions are used to avoid humans using a graph-based planner. Static obstacles are considered in [108], where predictions are informed by their distance to goals. Nonlinear optimization towards goals is used in [109]. Planning-based methods capture human intent more accurately, but due to the planning step suffer from long inference times. Additionally, wrongly inferred goals result in incorrect predictions and can lead to collisions.

Learning-Based Human Motion Prediction

Learning-based methods learn a distribution of probable human trajectories from a dataset (e.g., [110]). The temporal dependencies of human motion can be modeled by Recurrent Neural Networks [111], such as Long Short-Term Memory (LSTM). We distinguish between *uni-modal* and *multi-modal* prediction models.
Uni-modal methods, such as Social LSTM [112] and Social-STGCNN [113], predict a single trajectory (or mode) for each human. Hence, when multiple distinct trajectories (modes of the probability distribution) are possible, they tend to average the modes without representing any of the modes accurately [114], [115].
Multi-modal methods do account for multiple distinct trajectories. *Variational* methods such as the Conditional Variational AutoEncoder (CVAE) [116] (based on the Variational AutoEncoder (VAE) [117]), model latent variables to represent the data as a lower dimensional distribution. Trajectron++ [63] is a CVAE that incorporates dynamics and scene context to improve predictions. Variational Recurrent Neural Networks (VRNN) [118] are extended VAEs that model high dimensional sequences. This network was applied for Social-VRNN [119] where scene-aware multi-modal trajectory predictions are represented by a Gaussian Mixture Model (GMM). Y-Net [120] predicts several trajectories per endpoint using waypoints. NSP-SFM [121] incorporates a physics model and CVAE to learn realistic physical behavior.
Generative Adversarial Networks (GANs) [122] train a generator together with a discriminator network. The discriminator enforces the generator to produce realistic predictions, which can be queried after training. Examples are Social-GAN [123] that encodes pedestrian interactions and MG-GAN [124] that produces modes through multiple generators.
State-of-the-art learning-based prediction algorithms still have severe limitations. The accuracy of the learned distribution is limited because finite data is available and may



(a) EPSILON [39], driving on the highway. Top row: camera images. Bottom row: planner visualizations.



(b) SafeVRU [28], avoiding a pedestrian (visualized in red). The planned trajectory is visualized in blue.

Figure 2.1: Two examples of self-driving vehicle demonstrators. Images taken from the respective publications.

be insufficient for guaranteeing safety [125]. In addition, the model distribution has a predefined structure, assuming for example a static number of modes [119], [126], which may not accurately capture the real distribution and auxiliary uncertainties such as tracking and sensing errors. While these inaccuracies are always present, the planner is typically not aware of their magnitude, which may lead to collisions in practice. Finally, learning-based models are computationally expensive and resource intensive to train and also to deploy. Still, learning-based prediction algorithms currently achieve the best prediction accuracy on public benchmarks (see for example [127]).

2.4. SELF-DRIVING DEMONSTRATOR VEHICLES

A promising area where motion planners are being developed is for self-driving vehicles. Several groups have published experiments with experimental platforms. In [128], an autonomous vehicle (named *Bertha*) drove 103 km. Their planner uses trajectory optimization with inequality constraints to avoid obstacles. Obstacle predictions are deterministic but multi-modal (i.e., they consider all possible paths that an obstacle can take). IARA [129] drove 74 km in the night. Their planner [31] uses MPC largely based around [16] with deterministic obstacle predictions. IARA encountered limited interactions with obstacles given the time of the drive. Recently, EPSILON [39] drove about 12.5 km in highway and suburban environments, using a POMDP that considered interaction to guide an optimization-based planner. On a smaller scale, [130] demonstrated an optimization-based motion planner that improved passenger comfort.

Few methods have demonstrated safe navigation in urban environments. These environments are more challenging as they are less structured and require safe navigation among Vulnerable Road Users (VRUs). Steering versus braking evasion was considered in [131]. In [132], a 5 km route was completed in an urban environment that included traffic lights, crosswalks and roundabouts. The planner used motion primitives, checking collisions with deterministic predictions of dynamic obstacles. Their test environment featured only few VRUs. SafeVRU [28] was the first demonstrator to incorporate obstacle motion uncertainty into the motion planner. The planner, based on a trajectory optimization method [27], accounted for level sets of the predictions that followed a Gaussian distribution. The demonstrator avoided a real pedestrian, in a small-scale experiment.

In summary, these demonstrations only featured few VRUs and were performed largely outside of urban environments. Self-driving cars are being deployed in urban environments by several companies (for example by Waymo and Cruise in San Fransisco). However, they still have a limited operational domain, there are major safety concerns [133], [134] and they are not as fast as human drivers yet (having to take long detours [135]). Remaining challenges of self-driving cars and, in particular, the planner include scaleability to large numbers of VRUs and the ability to account for multi-modal uncertainty of VRUs' predicted motion. This would allow the planner to reduce the traveling time without compromising on safety.

2.5. CONCLUSIONS

There are still significant challenges to enable motion planning in dynamic environments. Optimization-based approaches currently have significant advantages over other methods as they incorporate the dynamic model and dynamic collision avoidance constraints while computing commands in real-time. Their main limitations (local optimality and infeasibility) can possibly be resolved through the combination of multiple controllers, combining the strengths of each or by introducing novel solver techniques (e.g., sampling-based optimization). While deterministic collision avoidance may be enough in simple scenarios, uncertainty in human predictions needs to be incorporated in more complex environments. At the same time, these methods impose strong assumptions (uni-modal distribution) and remain too conservative (marginalizing over time and per obstacle) to be efficiently used in practice. Finally, motion planners cannot yet explicitly account for the interaction with humans in real-time, although the state-of-the-art is quickly progressing in this direction.

This dissertation builds on optimization-based planning in order to propose novel safe and time-efficient motion planning methods. The first part composed of Chapters 3 to 5 aims to reduce conservatism and relax the assumptions of probabilistic safe motion planning. The second part, in Chapter 6, considers how the limitations of optimization-based methods can be mitigated through high-level planning. Chapter 7 discusses the implementation of these methods on a full-scale self-driving vehicle.

3

Marginal Scenario-Based Trajectory Optimization

Deep in the human unconscious is a pervasive need for a logical universe that makes sense. But the real universe is always one step beyond logic.

Frank Herbert - Dune

3.1. Overview

The first planner proposed in this dissertation is concerned with the uncertain predictions of dynamic objects (e.g., pedestrians). Techniques to provide such predictions (e.g., [63], [136]) were discussed in Chapter 2. Traditional planners [5], [27] do not account for the presence of uncertainties while planners that do consider uncertainties are usually limited to Gaussian probability distributions [28], [84], [85], [92]. The real probability distribution of human motion predictions is non-Gaussian as their dynamics are nonlinear and their intentions cannot be observed by the robot.

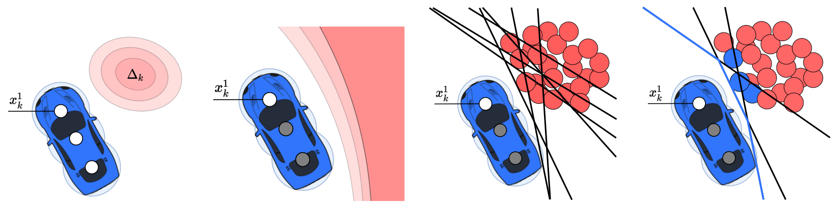
Non-Gaussian probability distributions are difficult to incorporate analytically in planning, contrary to their Gaussian counterparts. One analytical approximation is considered in a simulated setting in [87]. The probability of collision, independent of the shape of the probability distribution, can be evaluated by sampling. However, standard Monte Carlo based approaches [89], [96], [97] are computationally inefficient when used with optimization-based planners.

3.1.1. Approach

This dissertation proposes a different sampling-based reformulation of the problem through Nonconvex Scenario Optimization (see Fig. 3.1). Instead of directly solving

Parts of this chapter have been published in:

O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, “Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments”, *IEEE Robotics and Automation Letters (RA-L)*, pp. 5389–5396, 2021



(a) Original CCP (3.2c) (b) Linearized CCP (3.7b) (c) Linear SP (3.8) (d) Pruned SP (3.14c)

Figure 3.1: The proposed approach exemplified for one robot's disc and one dynamic obstacle for a single stage. The robot and the obstacle are drawn in blue and red, respectively. Fig. 3.1a shows the 1 σ to 3 σ interval of the uncertainty in red shades. Fig. 3.1b shows the probabilistic collision region when linearized from the robot disc at the front. Fig. 3.1c shows the sampled locations in red and boundaries of the constraints in black. Fig. 3.1d shows the resulting minimal polytope in blue.

the chance-constrained motion planning problem, it solves an associated deterministic problem obtained as follows. First, it applies a linearization of the chance constraints (see Fig. 3.1b), then it samples from the linearized chance constraints a large set of deterministic constraints, known as *scenarios* (see Fig. 3.1c). The number of scenarios drawn is linked with the associated probability of collisions. As a result, the original planning problem is reformulated as a deterministic problem, known as a scenario program (see Fig. 3.1d).

3.1.2. Contribution

The contributions of this chapter are:

1. A novel trajectory optimization framework for motion planning in uncertain dynamic environments, Scenario-based Model Predictive Contouring Control (S-MPCC) that builds on nonconvex scenario-optimization framework [98] and the model predictive contouring control (MPCC) design of [27]. While sampling-based chance-constrained approaches are generally considered intractable for real-time motion planning, S-MPCC is competitive in terms of computation times with state-of-the-art planning methods, while applicable to generic uncertainties.

2. In contrast with the general a posteriori results in [98], this chapter additionally shows that the risk of the motion plan can be obtained *before optimization*. The support subsample, which is the key indicator for the risk in [98], is obtained through the geometry of the problem, leading to efficient evaluation of the samples.

3.2. Problem Formulation

This chapter considers the motion planning problem of a mobile robot, whose dynamics can be represented by the following nonlinear discrete-time system:

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{3.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denote the states and inputs, respectively. The robot can move within a workspace (e.g., the 2D plane when considering ground robots). In the

workspace, the robot must avoid collisions with dynamic obstacles. This chapter models the collision region of the robot $\mathcal{V}_k$ at time k as the union of n_c circles, and the collision region of the dynamic obstacles $\mathcal{D}_k^v$ at time k as a single circle.
The position of dynamic obstacles along the planning horizon of the robot is uncertain. This chapter denotes the uncertainty of the obstacles at stage k with a tuple $(\Delta_k, \mathcal{D}_k, \mathbb{P}_{k,\text{real}})$, where Δ_k is a probability space equipped with a σ-algebra $\mathcal{D}_k$ and a probability measure $\mathbb{P}_{k,\text{real}}$. The probability space is allowed to be unbounded and non Gaussian. It is assumed that at each step a perception module provides the motion planner with an independent model of the uncertainty, formalized as follows.

Assumption 1. *The planner is provided with a model $\mathbb{P}_k$ of the real probability measure $\mathbb{P}_{k,real}$ for each k.*

Assumption 2. *Random variables $\boldsymbol{\delta}_j \sim \mathbb{P}_j$ and $\boldsymbol{\delta}_l \sim \mathbb{P}_l$ are independent for all stages $j, l \in \{1, \ldots, N\}$, where $j \neq l$.*

Assumption 2 implies that the dependency induced, for example, by the dynamics of an obstacle, is handled by the perception module such that the uncertainties are independent as viewed from the perspective of the motion planner. The assumption is common in state-of-the-art perception modules, for example [137], [138], [136]. The assumption is necessary to consider the probability of collision in each time step separately. While it holds for several perception models, it may not always hold for the true distribution, in particular, when that distribution consists of several modes. For example, when the obstacle motion is realized in the direction of one of the modes (e.g., when an obstacle moves distinctly to the left instead of right), this information may change the distribution in later time steps to follow that mode more strictly (e.g., obstacle will likely keep moving left). This chapter does not capture this temporal dependency. In the next Chapter (Chapter 4), Assumption 2 is not necessary.
Similar to existing scenario optimization schemes, it is additionally assumed that the probability distribution of the obstacles' future motions is not affected by the robot's trajectory. Interaction between obstacles and the robot is therefore not considered.
Under the, possibly unbounded, uncertainty of the dynamic obstacles, this chapter constrains the marginal probability of collision at each time step of the trajectory using *chance constraints*, similarly to [84], [102]. Each chance constraint is subject to an acceptable risk level ϵ_k, which can be tuned accordingly. This implies that the proposed method cannot give a non-conservative bound on the collision risk of the full motion plan. However, by frequently recomputing the motion plan, for example in an MPC framework, the actions in the near future are probabilistically safe and risk in later stages is reconsidered when the robot moves closer. This chapter formulates the motion planning problem as follows:

$$\min_{\boldsymbol{u} \in \mathbb{U}} \quad \sum_{k=1}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.2a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \boldsymbol{x} \in \mathbb{X} \tag{3.2b}$$

$$\mathbb{P}_k \left[||\boldsymbol{x}_k^d - \boldsymbol{\delta}_k^v||_2 > r, \forall d, v \right] \geq 1 - \epsilon_k,\ \forall k, \tag{3.2c}$$

where $\boldsymbol{u} = \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\} \in \mathbb{U}$ are the optimized system inputs subject to input constraints, $\boldsymbol{\delta}_k^v \in \Delta_k^v$ is the uncertain position of obstacle v at stage k and $J(\boldsymbol{x}_k, \boldsymbol{u}_k) \geq 0$ is the cost function specifying performance metrics. The radius r is the sum of vehicle and obstacle radii. To simplify the notation, this chapter assumes this radius to be a constant. The chance constraint, (3.2c), constrains the probability of collisions between each collision circle d of the vehicle and the collision circle of each dynamic obstacle v at prediction step k to be below the risk level ϵ_k, as visualized in Fig. 3.1a. The probability measure $\mathbb{P}_k$ refers to the modeled uncertainty. Problem (3.2) is a chance constrained optimization problem. As discussed in Section 3.4, to solve this problem, this chapter relies on the nonconvex scenario optimization (NSO) framework of [98], for which an overview is provided in the following section. This framework can in general provide a bound on the risk with respect to the unknown probability distribution $\mathbb{P}_{\text{real}}$, by sampling from the real system. In real-time planning, however, collecting samples online is intractable. Instead, this chapter proposes to sample from the model distribution $\mathbb{P}$, as defined in Assumption 1. For consistency of notation, the results of [98] are presented here using the model $\mathbb{P}$.

3.3. Nonconvex Scenario Optimization

The NSO framework allows us to replace chance constraints with deterministic constraints by sampling. Consider the Chance Constrained Problem (**CCP**)

$$\min_{\boldsymbol{u} \in \mathbb{U}} \quad J(\boldsymbol{u}) \tag{3.3a}$$

$$\text{s.t.} \quad \mathbb{P}\left[g(\boldsymbol{u}, \boldsymbol{\delta}) \leq 0\right] \geq 1 - \epsilon, \ \boldsymbol{\delta} \in \Delta, \tag{3.3b}$$

where $\boldsymbol{u}$ are decision variables, $\boldsymbol{\delta} \in \Delta$ is the realization of the uncertainty and the function $g : \mathbb{X} \times \Delta \rightarrow \mathbb{R}$ is a nonlinear function associated with the nonconvex constraint $g(\boldsymbol{x}, \boldsymbol{\delta}) \leq 0$. The authors of [98] established a link between CCP (3.3) and the deterministic Scenario Program (**SP**):

$$\min_{\boldsymbol{u} \in \mathbb{U}} \quad J(\boldsymbol{u}) \tag{3.4a}$$

$$\text{s.t.} \quad g(\boldsymbol{u}, \boldsymbol{\delta}^i) \leq 0, \ \boldsymbol{\delta}^i \in \Delta, \ \forall i \in \mathcal{S}. \tag{3.4b}$$

Its solution is denoted by $\boldsymbol{u}_{SP}^*$. Each of the S constraints in (3.4b) is constructed by drawing a sample $\boldsymbol{\delta}^i$ from Δ, and formulating the constraint $g(\boldsymbol{u}, \boldsymbol{\delta}^i) \leq 0$ in the scenario where the sample $\boldsymbol{\delta}^i$ is a realization of the uncertainty. Since each of the samples specifies a scenario, the samples themselves are called *scenarios* and the constraints (3.4b) are known as *scenario constraints*. The *violation probability*, $V : \mathbb{U} \rightarrow [0, 1]$, given by

$$V(\boldsymbol{u}) := \mathbb{P}\left[\boldsymbol{\delta} \in \Delta : g(\boldsymbol{u}, \boldsymbol{\delta}) > \mathbf{0}\right], \tag{3.5}$$

defines the probability that input $\boldsymbol{u}$ violates a newly observed scenario. The solution of the SP in (3.4) depends on randomly sampled scenarios and hence its violation probability is a random variable over the product probability measure, given by $\mathbb{P}^S = \mathbb{P} \times \ldots \times \mathbb{P}$ (S times). To link the SP of (3.4) with the CCP of (3.3), the goal is to constrain the probability that $V(\boldsymbol{u}_{SP}^*)$ satisfies risk bound ϵ, a probability which is referred to as the confidence. A key definition in this direction is the *support subsample*.

Definition [98]: A *support subsample* of an SP is a subset of scenarios $\mathcal{S}_{\text{support}} \subseteq \mathcal{S}$ that results in the same optimizer as the original SP. The cardinality of the support subsample, that is, the support subsample size, is denoted by s. The smallest support subsample size is denoted by s^*.

Theorem 1 in [98] provides the following confidence bound

$$\mathbb{P}^S[V(\boldsymbol{u}_{SP}^*) > \epsilon(s^*)] \leq \sum_{s=0}^{S-1} \binom{S}{s} [1-\epsilon(s)]^{S-s} = \beta. \tag{3.6}$$

Here $\epsilon(s) : \{0, \ldots, S\} \to [0,1]$ can be designed subject to (3.6) and $\epsilon(S) = 1$, an example can be found in [98, Sec. II].
Equation (3.6) theoretically links the sampling size S, confidence parameter β (complement of the confidence) and risk ϵ, based on the observed support sample size. Notice that in this work, as a consequence of using model distribution $\mathbb{P}$, the bound (3.6) applies to the modeled uncertainty rather than the real robot, in contrast with [98, Th. 1].

3.4. PROPOSED APPROACH

The proposed method relies on the Model Predictive Contouring Control (MPCC) framework [27] to define the objective to optimize to plan a suitable path for the robots. It differs from [27] in the way that dynamic obstacles are dealt with, as detailed in the rest of the section. As such this chapter will refer to the approach as Scenario-MPCC (S-MPCC). To present the method, the following derivations consider a single dynamic obstacle and one of the discs used to represent the vehicle[1].

3.4.1. CHANCE CONSTRAINTS LINEARIZED IN THE ROBOT POSITION

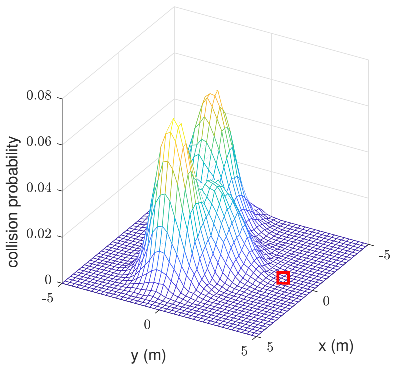
Chance constraints (3.2c) are nonconvex in the robot position when sampled (see discs in Fig. 3.1c) and the associated SP may have many local optima and a sizable support subsample. This chapter therefore considers a linearization of the collision regions (depicted by the lines in Fig. 3.1c) before sampling. The linearization greatly reduces the number of scenario constraints that affect the motion plan in the optimization (the support subsample). This enables us to reduce the sampling size initially without compromising on the conservatism of the motion plan. The constraints are modified as

$$\boldsymbol{A}_k = \frac{\boldsymbol{\delta}_k - \hat{\boldsymbol{x}}_k}{||\boldsymbol{\delta}_k - \hat{\boldsymbol{x}}_k||}, \qquad b_k = \boldsymbol{A}_k{}^T(\boldsymbol{\delta}_k - \boldsymbol{A}_k r), \tag{3.7a}$$

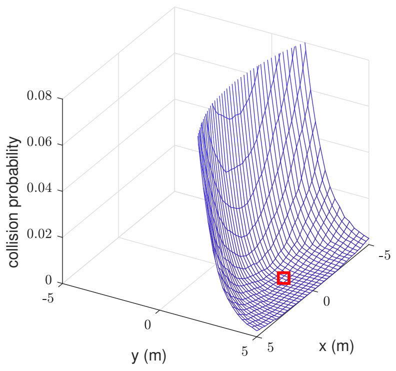
$$\mathbb{P}_k\left[\boldsymbol{A}_k^T \boldsymbol{x}_k \leq b_k\right] \geq 1 - \epsilon_k, \forall k, \boldsymbol{\delta}_k \in \Delta_k, \tag{3.7b}$$

where the collision region is linearized with respect to $\hat{\boldsymbol{x}}_k$, the k-step ahead prediction of the robot position. The trajectory of the previous planning cycle, forward propagated, is used as predictor. That is[2], $\hat{\boldsymbol{x}}_{t|k} = \boldsymbol{x}_{t-1|k+1}$ and $\hat{\boldsymbol{x}}_{t|N} = \boldsymbol{x}_{t-1|N}$. Hence, this chapter

[1]Section 3.4.4 shows how this case extends linearly to multiple dynamic obstacles and multiple discs.
[2]$\boldsymbol{x}_{t|k}$ denotes the k-step ahead prediction of the robot trajectory for the MPC planning cycle at time t



(a) Quadratic CCP (3.2c)



(b) Linearized CCP (3.7b)

Figure 3.2: Grid wise evaluation of the collision probability, with $r = 0.5$ m of chance constraints (3.2c) and (3.7b) in an example where $\boldsymbol{\delta}$ follows a Mixture-of-Gaussians (MoG) distribution. The red square denotes the linearization point.

searches for collision-free solutions around the planned trajectory of the previous planning cycle. By using the previous trajectory to overapproximate the collision regions, the linearization enforces the passing behavior of the previous trajectory onto the newly optimized trajectory, possibly reducing the search space. Consequently, the planning problem may become infeasible, in particular, if the obstacle distribution shifts between time steps. Different initial trajectories could be used to linearize the collision regions, as is further explored in Chapter 6. Sec. 3.4.3 shows that after linearization, the free-space of the resulting SP is convex in the robot position. A comparison between chance constraints (3.2c) and (3.7b) for an example is provided in Fig. 3.2. The linearized chance constraints capture less of the shape of the distribution, but are accurate near $\hat{\boldsymbol{x}}_k$ and thus sufficient for motion planning. Note that the linearizations are performed for each stage of the trajectory, as illustrated in Fig. 3.1b.

3.4.2. Scenario Program

For each of the chance constraints in (3.7b) a set of deterministic constraints is constructed by sampling from the uncertainty. The red circles in Fig. 3.1c represent these samples and the black lines are the *scenarios* (Sec. 3.3). The resulting SP is given by

$$\min_{\boldsymbol{u}\in\mathbb{U}} \quad \sum_{k=1}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.8a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \boldsymbol{x}\in\mathbb{X} \tag{3.8b}$$

$$\boldsymbol{A}_k^T(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k)\boldsymbol{x}_k \leq b_k(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k), \forall i \in \mathcal{S}_k,\ \forall k. \tag{3.8c}$$

The theoretic properties of SPs, discussed in Sec. 3.3, are limited to CCPs with one chance constraint. However, (3.7b) describes multiple chance constraints, one for every stage of the planned trajectory. In the following, this chapter shows that multiple chance constraints can be handled separately, resulting in a probabilistic feasibility property per stage.

Theorem 1. *Under Assumption 2, the probability that the solution of SP (3.8) violates its associated chance constraint at stage k, satisfies*

$$\mathbb{P}_k^S[V_k(\boldsymbol{u}_{SP}^*) > \epsilon_k(s_k^*)] \leq \beta_k(S_k), \tag{3.9}$$

where

$$\beta_k(S_k) := \sum_{s=0}^{S_k-1} \binom{S_k}{s} [1-\epsilon_k(s)]^{S_k-s}. \tag{3.10}$$

Proof of Th. 1. The proof follows along the lines of the convex proof [102, Th. 4.1]. In the following, this chapter derives the result for $k = 1$. The proof is analogous for all other k. This chapter uses the notation $\boldsymbol{\omega}_k = \{\boldsymbol{\delta}_k^1, \ldots, \boldsymbol{\delta}_k^{S_k}\}$ to denote the collection of all samples per stage. Consider the complement of the confidence of the first stage, when the samples of all other stages have been drawn,

$$\mathbb{P}_1^S[V_1(\boldsymbol{u}_{SP}^*(\boldsymbol{\omega}_1)) > \epsilon_1(s_1^*) \mid \boldsymbol{\omega}_2, \ldots \boldsymbol{\omega}_N],\ \boldsymbol{\omega}_1 \in \Delta^{S_1}. \tag{3.11}$$

Under Assumption 2, the samples $\boldsymbol{\omega}_1$ are drawn independently from the samples $\boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_N$. Moreover, since $\boldsymbol{\omega}_2, \ldots, \boldsymbol{\omega}_N$ have been observed, their respective constraints can be merged into the feasible set

$$\tilde{\mathbb{X}}^{2:N} = \prod_{k=2}^{N} \{\boldsymbol{x}_k \mid g(\boldsymbol{x}_k, \boldsymbol{\omega}_k) \leq 0\}. \tag{3.12}$$

This results in the following modified optimisation problem

$$\min_{\boldsymbol{u} \in \mathbb{U}} \quad \sum_{k=1}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.13a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \boldsymbol{x} \in \mathbb{X} \tag{3.13b}$$

$$\mathbb{P}_1[g(\boldsymbol{x}_1, \boldsymbol{\delta}_1) \leq 0] \geq 1 - \epsilon_1 \tag{3.13c}$$

$$\boldsymbol{x}_{2:N} \in \tilde{\mathbb{X}}^{2:N}. \tag{3.13d}$$

This problem is a nonconvex CCP of the form (3.3) with one chance constraint, hence (3.6) can be applied, which shows that the confidence of the first stage satisfies the proposed theorem for $k = 1$ and analogous derivations apply for $k = 2, \ldots, N$. Even though constraints (3.13d) are deterministic, the solution to the optimization problem has not changed compared to SP (3.8). It is concluded that the result holds. □

3.4.3. PROBABILISTIC SAFETY GUARANTEES

The key insight that makes the proposed approach tractable is that due to the geometric structure of the problem, the free space may be described by only a small subset of the scenarios. To see this, first note that each scenario constraint in (3.8c) defines a half-space. The collision-free space, if it exists, is formed by the intersection of half-spaces and is convex, as i) each half-space is convex and ii) the intersection of convex constraints is convex. This results in a free space polytope $\mathcal{P}_k$ (see Fig. 3.1d), spanned by those half-spaces that form the boundary of the polytope. This subset of half-spaces can be defined by their indices as

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

The usefulness of the set $\mathcal{H}_k$ is twofold. First, (3.8c) may be replaced with only those half-spaces that span polytope $\mathcal{P}_k$, greatly reducing the size of the online optimization problem. Second, the set $\mathcal{H}_k$ contains indices of the constraints that may be active during optimization and hence the support subsample is bounded by its cardinality, that is, $s_k^* \leq |\mathcal{H}_k|$. This chapter use the latter fact to establish the link between the CCP subject to (3.7b) and SP (3.8). There always exists an upper bound, $\bar{s}$, for the cardinality of $\mathcal{H}_k$ and for the considered problem this chapter experimentally identified that this upper bound $\bar{s}$ is much smaller than the sample size. That is, for uncertainty distributions where the samples are not cluttered at the boundary, only few scenarios are active.
The sampling size S_k can now be computed offline, using *(i)* Theorem 1, *(ii)* upper bound $\bar{s}$, *(iii)* confidence parameter β_k, and *(iv)* risk ϵ_k. The SP solved online is given by:

$$\min_{\boldsymbol{u}\in\mathbb{U}} \quad \sum_{k=1}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{3.14a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \boldsymbol{x} \in \mathbb{X} \tag{3.14b}$$

$$\boldsymbol{A}_k^T(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k)\boldsymbol{x}_k \leq b_k(\boldsymbol{\delta}_k^i, \hat{\boldsymbol{x}}_k),\ i \in \mathcal{H}_k. \tag{3.14c}$$

Algorithm 1 summarizes the proposed method. Online, the distribution is sampled and the minimal polytope and support subsample size are identified (Lines 4-8). Then optimization problem (3.14) is solved (Line 9) and the first input is used as control input (Line 10). In the following, this chapter provides a result for improving performance by discarding outlier scenarios.

Algorithm 1: S-MPCC

```
 1: Compute $S_k$ from $\epsilon_k$, $\bar{s}$, for all $k$
 2: for all $t = 1, 2, \ldots$ do
 3:   $\Delta_k^t \leftarrow$ Retrieve uncertainty from perception module
 4:   for all $k = 1, \ldots, N$ do
 5:     Sample $\boldsymbol{\delta}_k^i \in \Delta_k^t$, $i = \{1, \ldots, S_k\}$
 6:     Compute $\boldsymbol{A}_k^i, b_k^i$ from (3.7a) for all samples
 7:     Find $\mathcal{H}_k$ and verify $|\mathcal{H}_k| \leq \bar{s}$
 8:   end for
 9:   $\boldsymbol{u}_t \leftarrow$ Solve (3.14)
10:   Output: $\boldsymbol{u}_{t|1}$
11: end for
```

Theorem 2. *Consider solving the CCP* (3.3) *using the SP* (3.4)*, where after sampling, part of the scenarios are discarded. Suppose that we have a discarding algorithm $\mathcal{R}$ that removes R of the S scenarios, leaving $P = S - R$ scenarios to be considered for the optimiza-*

tion. Let $\epsilon(s)$ be a function such that $\epsilon(P) = 1$ and

$$\beta(S,P) = \binom{S}{P} \sum_{s=0}^{P-1} \binom{P}{s} [1-\epsilon(s)]^{P-s}.$$

Then the probability that the solution of the SP (3.4) is infeasible for the original CCP (3.3) satisfies the upper bound

$$\mathbb{P}^S[V(\boldsymbol{u}_{SP}^*) > \epsilon(s^*)] \leq \beta(S,P). \tag{3.15}$$

Proof. Consider the partitioning of the probability space:

$$\Delta_{\mathcal{I}_p}^S = \{\boldsymbol{\delta}^S \in \Delta^S \mid \mathcal{R}(\boldsymbol{\delta}^S) = \mathcal{I}_p\}, \tag{3.16}$$

The sets $\Delta_{\mathcal{I}_p}^S$ are events where the picking algorithm selected the indices $\mathcal{I}_p$. Define the set where the risk bound is violated

$$\mathcal{B}_{\mathcal{I}_p} = \{\boldsymbol{\delta}^S | \mathcal{R}(\boldsymbol{\delta}^S) = \mathcal{I}_p, V(\boldsymbol{u})_{\mathcal{I}_p}^*) > \epsilon(s_{\mathcal{I}_p}^*)\}. \tag{3.17}$$

Notice that the last condition is upper bounded by (3.6) with $S = P$. But the distribution of the samples is biased due to the samples that were removed from the iid sample set. This chapter obtains the following bound on the biased sample set

$$\mathbb{P}^S[\mathcal{B}_{\mathcal{I}_p}] \leq \sum_{s=0}^{P-1} \binom{P}{s} [1-\epsilon(s)]^{P-s}. \tag{3.18}$$

This result holds for all index sets which also contains all the possible biases introduced by $\mathcal{R}$. Hence, the upper bound

$$\mathbb{P}^S[\mathcal{B}] = \mathbb{P}^S\Big[\bigcup_{\mathcal{I}_p} \mathcal{B}_{\mathcal{I}_p}\Big] \leq \beta(S,P), \tag{3.19}$$

$$\tag{3.20}$$

is attained by independence of the samples. □

Remark 1. Bound (3.19) is conservative. For example, when $\mathcal{R}$ is a random discarding algorithm, then the samples are still iid and (3.6) can be used directly with $S = P$, giving

$$\sum_{s=0}^{P-1} \binom{P}{s} [1-\epsilon(s)]^{P-s} = \beta,$$

which is generally much tighter than (3.19). However, even if the bound is conservative, it can be used to remove extreme scenarios, leading to generally better performance.

3.4.4. Multiple Dynamic Obstacles and Discs

To apply the strategy above to more than one obstacle, this chapter uses the fact that scenario optimization is distribution agnostic. The predictions of the obstacles are combined into a probability space $\Delta_k = \begin{bmatrix}\Delta_k^0 & \dots & \Delta_k^V\end{bmatrix}^T$, where samples are denoted $\boldsymbol{\delta}_k = \begin{bmatrix}\boldsymbol{\delta}_k^0 & \dots & \boldsymbol{\delta}_k^V\end{bmatrix}^T$. Although the stacked distribution $\boldsymbol{\delta}_k$ could be used to model the correlation between the movement of obstacles, S-MPCC samples each component separately from individual probability distributions. The chance constraints (3.7b) need to include all obstacles and are modified as follows

$$\mathbb{P}_k\left[\boldsymbol{A}_k^T(\boldsymbol{\delta}_k^v, \hat{\boldsymbol{x}}_k)\boldsymbol{x}_k \leq b_k(\boldsymbol{\delta}_k^v, \hat{\boldsymbol{x}}_k), \forall v\right] \geq 1-\epsilon_k,\ \boldsymbol{\delta}_k \in \Delta_k,\ \forall k.$$

The rest of the method follows analogously to the single obstacle approach but where the scenarios are drawn for each obstacle, resulting in more scenarios to process before obtaining the free space polytope.

In the case of multiple vehicle discs, S-MPCC formulate multiple chance constraints of the form (3.2c), one for each collision disc. It then applies the method described in this Section per disc as samples for each of the discs are independent.

3.5. S-MPCC With Gaussian Uncertainties

A common class of uncertainties are the (truncated) Gaussian uncertainties. This section presents a detailed formulation of Algorithm 1, namely Algorithm 2, one can use in the case of (truncated) Gaussian uncertainty.

The first step of Algorithm 2 is to determine the sample size. This chapter uses $\epsilon_k = 1 - 0.9889$, equivalent to the probability mass under the 3 σ interval of a bivariate Gaussian (generally considered as safe). Since the risk has logarithmic dependency on β_k [98], β_k is generally small. This chapter picks $\beta_k = 1 \cdot 10^{-6}$, i.e., one in a million SPs may not be feasible for the original CCP[3]. The removal size $R = 50$ is empirically determined, verifying that outliers are removed. Upper bound $\bar{s}$ is guessed and increased until it is never exceeded in practice, leading to $\bar{s} = 20$. Evaluating (3.19) results in $S_k \approx 53050$ (line 1). The main dependency of the sample size is the acceptable risk ϵ_k. Sampling more scenarios results in a higher probability of safety, but at the cost of more conservative trajectories and increased computation times.

By assuming a (truncated) Gaussian distribution for the obstacle motion, it is possible to perform part of the sampling operation offline, to improve real-time performance. Under this assumption, samples from a standard bivariate Gaussian distribution can be drawn offline to be transformed to the mean and variance of the predicted distribution, online. To be precise, a number of batches with S_k bivariate Gaussian samples are generated offline, centered at the origin and with $\boldsymbol{\Sigma} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix (line 2-4). These samples are obtained using the Box-Muller Transformation (BMT) [139], which also allows us to draw radially truncated Gaussian samples by simply changing the support domain of u_1 to $[e^{-\frac{r^2}{2}}, 1]$ [140]. Most of the samples will be in the center of the distribution and will not be relevant online. Hence, this chapter runs the online algorithm for scenario selection (explained later), offline and aggregate the set of selected

[3]Note that the designer can choose to keep safety margin in the obstacle radius such that a failure does not have to result in a collision.

Algorithm 2: Detailed S-MPCC for (truncated) Gaussian

```
 1: Determine $S_k$ from $\epsilon_k$, $\beta_k$, $\bar{s}$, $R$
 2: $\boldsymbol{u}^i \leftarrow \mathbb{U} \times \mathbb{U}$, $\forall i = \{1, \dots, S_k\}$ (uniform random)
 3: $z_0^i = \sqrt{-2 \ln u_1^i} \cos(2\pi u_2^i)$, $\forall i = \{1, \dots, S_k\}$ (BMT)
 4: $z_1^i = \sqrt{-2 \ln u_1^i} \sin(2\pi u_2^i)$, $\forall i = \{1, \dots, S_k\}$ (BMT)
 5: Verify relevance of samples $\boldsymbol{z}$, prune irrelevant
 6: for all $t = 1, 2, \dots$ do
 7:     for all $k = 1, \dots, N$ do
 8:         $\boldsymbol{\delta}_k^i \leftarrow$ (3.21)
 9:         $\hat{\boldsymbol{\delta}}_k^l \leftarrow$ apply $\mathcal{R}$ to closest $R + l$ scenarios in $\boldsymbol{\delta}_k^i$
10:         $\mathcal{P}_k \leftarrow$ intersection algorithm on $\mathcal{H}(\hat{\boldsymbol{\delta}}_k^l) \cup \mathcal{H}_k^{\text{range}}$
11:     end for
12: end for
```

scenarios. Scenarios that are not in this set are pruned offline (line 5). In the 3 σ example, approximately 95% of the scenarios are removed offline.

Online, the offline samples only need to be transformed from the standard bivariate normal distribution to the estimated mean and variance of the uncertainty (line 8), which is computed using

$$\boldsymbol{\delta}_k^i = \boldsymbol{A}_k^T \boldsymbol{z}_k^i + \boldsymbol{\mu}_k, \quad \boldsymbol{A}_k^T \boldsymbol{A}_k = \boldsymbol{\Sigma}_k. \tag{3.21}$$

This chapter selects for each obstacle only one batch of samples. The obstacle predictions are sampled with that batch for all stages and all time steps. This provides the motion planner with consistent constraints. To further reduce the computational load, S-MPCC searches online only for the $l + R$ scenarios closest to considered vehicle position, where $l = 150$ is used in the following experiments. It is assumed that this set contains the support subsample. S-MPCC then applies the discarding algorithm $\mathcal{R}$, which removes the R scenarios furthest from the mean of the distribution (line 9). It constructs half-spaces from the remaining l scenario and add four half-spaces to constrain the vehicle in a square workspace. To find the minimal polygon in 2D from this set of half-spaces, S-MPCC uses an intersection based algorithm (line 10). The algorithm explores the intersections in the inner polygon in a counter-clockwise fashion. The lines traveled form the minimal polygon. In the following simulations and experiments, this chapter incorporates the proposed dynamic obstacle avoidance method in the MPCC framework [27]. A cost term that activates when the robot gets close to the boundaries of the free space polygon is added to penalize movement close to pedestrians.

3.6. RESULTS

This section presents simulation and real-world results for a mobile robot navigating among pedestrians. Moreover, it present a qualitative analysis and performance results of S-MPCC against two baselines: MPCC [27] and Collision Avoidance with Deep RL (CADRL) [24].

3.6.1. EXPERIMENTAL SETTINGS

The experimental platform is the Clearpath Jackal robot equipped with an Intel i5 CPU@2.6GHz. For the robot and pedestrian's localization an OptiTrack system [141] is used. The simulations use the open-source ROS implementation of the Jackal Gazebo for the robot simulation and Social Forces model [142] for pedestrian simulation.

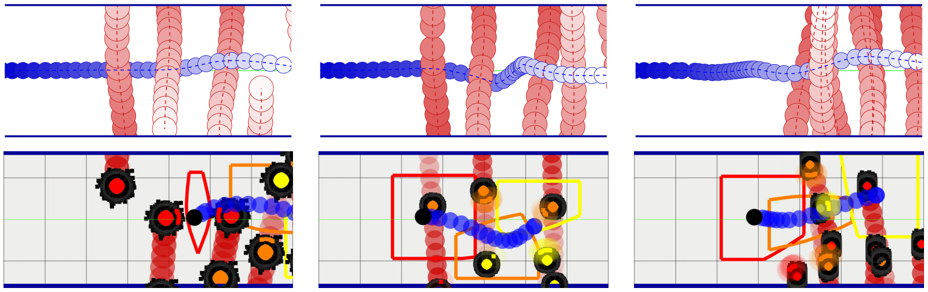
ForcesPro [143] is used to solve SP (3.14). The robot dynamics are described by a continuous-time second-order unicycle model [144]. The model is discretized with steps of 200 ms. The time horizon is set to 3 seconds divided into 15 stages. The sampling period for control is 50 ms.

3.6.2. SIMULATION RESULTS

This chapter compares the proposed method against two methods for Gaussian uncertainties. The first is a baseline MPCC approach [27] in which the ellipses used to represent the obstacles are obtained from the level sets of a known Gaussian distribution of the uncertainties. For comparison, the same tuning is used for both approaches (the interested reader can refer to [27] for details on the definition of the cost function and general constraints). The main difference between the two approaches is the handling of dynamic obstacles (i.e., ellipsoidal level sets vs. scenario constraints). The second method for comparison is CADRL [24]. Their open source ROS implementation is used in the following simulations. Similar to MPCC, this chapter employs ellipsoidal level sets as the collision region of the obstacles.

The simulation environment consists of a straight road where pedestrians are crossing freely, as depicted in Fig. 3.3. The robot objective is to follow the centerline of the road. This chapter evaluates S-MPCC for 2, 4 and 6 pedestrians. The uncertainty of the pedestrian predictions is Gaussian with a variance of $\boldsymbol{\Sigma} = 0.1^2\boldsymbol{I}$. The pedestrian radius is set to zero. Fig. 3.3a depicts one simulation of S-MPCC with 6 pedestrians. Aggregated results over 100 simulations are presented in Table 3.1. In all tested cases, collisions are prevented by S-MPCC, while additionally the risk, evaluated over the perceived uncertainty, remains below the specified 3 σ threshold. The MPCC method frequently switches between locally optimal trajectories resulting in collisions when it becomes infeasible. CADRL is reactive, which in the simulated environment leads it to positions where collisions may not be avoided. This behavior becomes worse with more obstacles. Interestingly, this chapter finds that S-MPCC results in smoother trajectories than both methods which results in earlier arrival at the goal. The downside is that the computation time of S-MPCC is higher. The computation time is largely dependent on the sampling size that increases for smaller acceptable risk values. The computation times could be reduced by discarding less scenarios or by considering only the pedestrians close to the estimate $\hat{\boldsymbol{x}}_k$. The simulation were repeated with 6 pedestrians in this case. The computation time was reduced to 6.86 ms mean and 40.94 ms maximum.

Evaluation of S-MPCC for non Gaussian uncertainties is depicted in Fig. 3.3. Here, the previous Gaussian predictions are radially truncated at 3.5 σ (Fig. 3.3b) and truncated in their width at 2.5 σ (Fig. 3.3c). In this scenario, width truncated uncertainties incorporate the domain knowledge that pedestrians are expected to cross at a crosswalk. Level set based approaches are not applicable in this case, as the geometry of the level sets depends on the specified risk threshold. Pedestrian locations are changed to simulate a



(a) Gaussian (b) Radially truncated Gaussian (c) Width truncated Gaussian

Figure 3.3: Simulations using S-MPCC with 6 crossing pedestrians for 3 types of uncertainties. The top row visualizes the robot (blue) and pedestrian (red) trajectories, where newer positions are depicted with lighter shades. The bottom row visualizes the free space and active samples at stages 1, 8 and 15 in red, orange and yellow. All samples considered online are shown in black. The robot's current and predicted occupied area are denoted in black and blue, respectively.

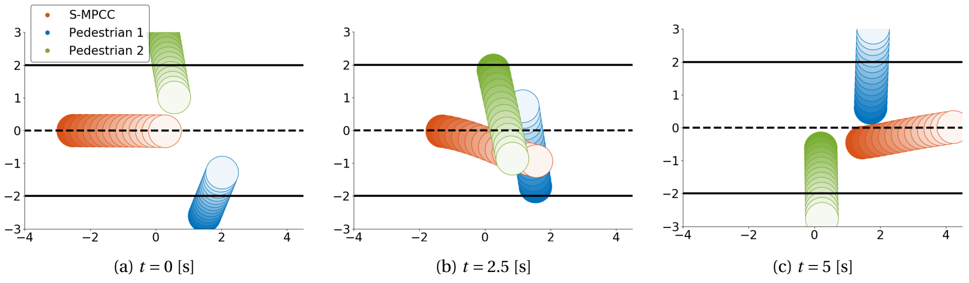
Table 3.1: Statistic results of the probability of collision with respect to the estimated uncertainty for the first stage (evaluated using Monte Carlo sampling) and violations of the specified risk, the task completion time and the computation times. The results are collected from 100 simulations of a crossing scenario for $n \in \{2, 4, 6\}$ pedestrians.

Ped.	Max Collision Prob. Stage 1 (# Violations)			Time to Completion Mean (Std.) [m]			Computation Time Mean (Max) [ms]		
	CADRL	MPCC	S-MPCC	CADRL	MPCC	S-MPCC	CADRL	MPCC	S-MPCC
2	0.71 (12)	0.13 (11)	**0.00007 (0)**	7.67 (0.97)	7.61 (0.10)	**7.14 (0.33)**	3.72 (**15.49**)	**1.47** (16.48)	6.48 (22.88)
4	0.83 (17)	0.14 (4)	**0.00006 (0)**	7.94 (1.04)	8.13 (0.49)	**7.54 (0.32)**	4.07 (22.29)	**1.80 (19.44)**	10.32 (43.91)
6	0.86 (43)	0.12 (13)	**0.00034 (0)**	8.68 (1.99)	8.27 (0.76)	**7.40 (0.45)**	4.83 (30.31)	**2.12** (**20.17**)	18.37 (65.56)

crosswalk. In contrast to the previous simulations, an obstacle radius of 0.3 m is specified and a variance of $\boldsymbol{\Sigma} = 0.08^2 \boldsymbol{I}$. This chapter evaluates the probability of collision in the first stage, with respect to the estimated uncertainty over 100 tests using Monte Carlo sampling. This chapter finds a maximum risk of 0.00305 for radial truncation and 0.02038 for width truncation. The violation of S-MPCC in the case of width truncation corresponds to a single case where the horizon is not long enough to correctly assess the risk of the full task a priori. This leads the robot to a state where the planner cannot find a trajectory that satisfies the risk bound along the horizon and the optimization becomes infeasible. By increasing the horizon, the risk can be anticipated earlier, improving feasibility at the cost of larger computation times. The maximum risk over the other simulations was at most 0.0070.

3.6.3. REAL-WORLD RESULTS

S-MPCC is evaluated further on real navigation situations with pedestrians. In the experiment, the robot navigates on a road following the lane central line when two pedestrians cross the robot's path. The noise on the pedestrian predictions is modeled with Gaussian distributions truncated at 3.5 σ. Fig. 3.4 provides snapshots of one experiment. While pedestrians are not following constant velocity behavior (that matches the mean predictions), the robot evades the two pedestrians succesfully by accounting for the encoded uncertainty.



(a) $t = 0$ [s] (b) $t = 2.5$ [s] (c) $t = 5$ [s]

Figure 3.4: Experimental results with the robot avoiding two crossing pedestrians. The orange circles depict the robot's plan, while the blue and green circles the pedestrians' (constant velocity) predictions. The solid black lines depict the road boundaries.

3.7. Conclusions and Future Work

This chapter presented a Scenario-based Model Predictive Contouring Control (S-MPCC) method for mobile robot motion planning in the presence of dynamic obstacles with arbitrary position distributions. The main idea was to pursue a scenario-based method (translating probabilistic constraints into deterministic ones), generating scenarios from a model of the uncertainty. By using geometry considerations, this chapter was able to prune the possible outcomes (scenarios), while providing a bound on the marginal risk with respect to the modeled probability distribution. This chapter demonstrated in simulations that the proposed method outperformed two recent baselines, in the sense that it generated trajectories that were significantly safer and more efficient. This came at a higher processing cost, but the method is still real-time capable. Furthermore, S-MPCC was demonstrated in a real-world experiment with a moving robot platform navigating among pedestrians. To further reduce the uncertainties and improve the navigation of the robot, incorporating the interactions between robot and pedestrians would be useful. The risk bounds that S-MPCC provides on the modeled uncertainty can still be improved by alleviating the standing assumption that requires the probability distributions per stage to be independent. Additionally, the risk bound on the planned trajectory is relatively conservative. A tighter bound can be useful for planning safer long term motion, especially when the robot dynamics are slow.

4

Joint Scenario-Based Trajectory Optimization

4.1. Overview

The previous chapter constrained the probability of collision in the planner through scenario optimization for *generic* uncertainties. In line with almost all existing work, such as [84], [86], [87], [91], [93], [94], this planner constrained the probability of collision in each time step of the trajectory (marginalizing over time) rather than the during trajectory as a whole. Many existing works additionally limit their constraints to the probability of collision *per obstacle* (marginalizing per obstacle), instead of jointly considering all obstacles. These two marginalizations of the probability of collision ignore the correlation between time steps and obstacles, and therefore lead to conservative motion plans. By considering the correlation in time and all obstacles (i.e., the joint probability of collision), more efficient trajectories can be planned without compromising safety.

4.1.1. Approach

This chapter achieves this goal by devising a novel scenario-based planner, Safe Horizon MPC (see Fig. 4.1), that considers the joint probability of collision. The proposed method predicts the uncertainty associated with obstacle motion forward in time (Step 1). From these predictions, scenarios are sampled that each describe the trajectories of all dynamic obstacles during the planning horizon (Step 2). Collision avoidance constraints are constructed around each of the scenarios (Step 3). The robot trajectory is optimized with respect to the constraints (Step 4), providing probabilistic collision avoidance. The proposed method is the first real-time capable method that bounds the CP

Parts of this chapter have been submitted as:

Paper: O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Scenario-Based Motion Planning with Bounded Probability of Collision", Jul. 2023, *under review*.

Video: O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Scenario-Based Motion Planning with Bounded Probability of Collision*, Jun. 2024, Available: `https://youtu.be/vyuNMO1giF0`.

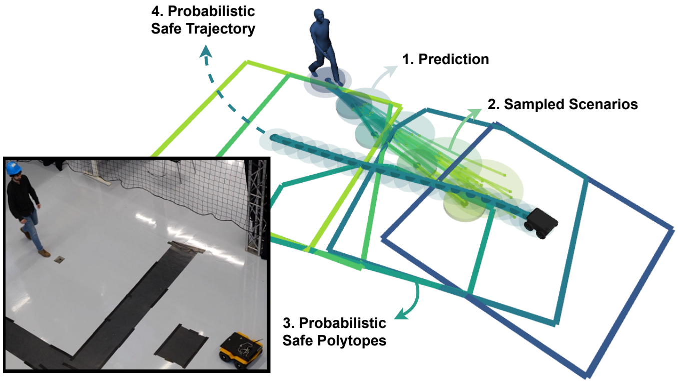

4. Probabilistic Safe Trajectory
1. Prediction
2. Sampled Scenarios
3. Probabilistic Safe Polytopes

Figure 4.1: Overview of the proposed scenario-based motion planner. Predicted obstacle trajectories are sampled to obtain scenarios, where each scenario represents a trajectory for all obstacles over the planning horizon. By ensuring safety for all scenarios, probabilistic safety of the motion plan is guaranteed.

of the planned trajectory jointly, in contrast with the existing state-of-the-art where the same quantity is conservatively approximated through its marginals.
These results are enabled by addressing two key limitations of Nonconvex Scenario Optimization (NSO) for robotic applications. First, the safety verification of NSO needs to solve the optimization S time, with S the number of samples, which is computationally intractable. This chapter verifies safety at a negligible cost during optimization. Second, the planned trajectory must always be feasible for the sampled scenarios, which is difficult to guarantee for generic uncertainties (e.g., unbounded distributions). This chapter proposes a relaxed formulation of the scenario program that is always feasible.

4.1.2. CONTRIBUTION

The contributions of this chapter are:

1. A novel trajectory optimization method, *Safe Horizon MPC*, that explicitly constrains the collision probability over the full duration of the planned trajectory. This distinguishes it from previous work, where the collision probability is constrained per planning time instance and per obstacle. The idea is that each sample from the distribution of the uncertain obstacle trajectories, representing a possible trajectory for all obstacles, corresponds to a single collision avoidance constraint over the horizon. The more samples are drawn, the higher the probability that the constraint is satisfied. By relying on sampling, the proposed planner is distribution agnostic.

2. An approach that, under a convexity assumption on the iterations of the underlying optimization algorithm (that holds, for example, for Sequential Quadratic Programming), identifies the scenarios that hold the solution in place (known as the *support*) during optimization, in contrast with the general framework of [98] where the support is computed after optimization. SH-MPC leverages this information to certify the motion plan online.

4.2. PROBLEM FORMULATION

This chapter considers a controlled robot with nonlinear discrete-time dynamics

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denote the states and inputs, respectively. The robot state is assumed to contain its x-y position $p = [x, y] \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$. Humans in the environment of the robot pose constraints on the navigation envelope. This chapter considers that the motion prediction of the dynamic obstacles is uncertain by modeling the positions of at most M obstacles as random variables. In particular, the uncertain position of Obstacle j at time step k is denoted as $\boldsymbol{\delta}_{k,j} \subseteq \boldsymbol{\delta}_k$, where $\boldsymbol{\delta}_k$ contains all obstacle positions at time k. The *joint uncertainty* $\boldsymbol{\delta} = \left[\boldsymbol{\delta}_1^T, \ldots, \boldsymbol{\delta}_N^T\right]^T \in \Delta$ stacks the uncertainty over all time steps. Here, Δ denotes the probability space[1] of the joint uncertainty, which is endowed with a σ-algebra $\mathcal{D}$ and a probability measure $\mathbb{P}$.

4

4.2.1. CHANCE CONSTRAINED PLANNING PROBLEM

The planning problem is visualized in Fig. 4.2. This chapter models the robot area and obstacle area with the union of n_d discs and a single disc, respectively. The dynamic uncertainty of other road users affects the navigation envelope of the robot. The probability space of the joint uncertainty $\Delta = \mathbb{R}^{2MN}$ captures the future N positions of M obstacles. To constrain the probability of a collision with *any* obstacle along the horizon, this chapter formulates a single chance constraint for collision avoidance. This leads to the following Chance Constrained Problem (CCP)

Problem 1 (CCP)**.**

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.1a}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{x}_{init} \tag{4.1b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, \ldots, N-1 \tag{4.1c}$$

$$\mathbb{P}\left[\bigwedge_{k=1}^{N} \bigwedge_{c=0}^{n_d} \bigwedge_{j=0}^{M} \left(||\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}||_2 \geq r\right)\right] \geq 1-\epsilon \tag{4.1d}$$

where $\boldsymbol{x}_{\text{init}}$ denotes the initial state of the robot, objective J is designed to achieve control objectives, $\mathbb{X} \subseteq \mathbb{R}^{Nn_x}$ and $\mathbb{U} \subseteq \mathbb{R}^{Nn_u}$ denote state and input constraints over the horizon, respectively, and (4.1d) is the collision avoidance chance constraint. The goal is to compute a control input $\boldsymbol{u}$ and trajectory $\boldsymbol{x}$ under the uncertainty $\boldsymbol{\delta}$ that is collision-free with a probability of at least $1-\epsilon$, where ϵ denotes the maximum collision probability over the planned trajectory.

When $\mathbb{P}$ is estimated by a prediction model the collision avoidance constraint is formulated with respect to an estimate $\hat{\mathbb{P}}$ of $\mathbb{P}$ and the chance constraint relates to the estimate of the probability distribution.

[1]See e.g., [147] for more details.

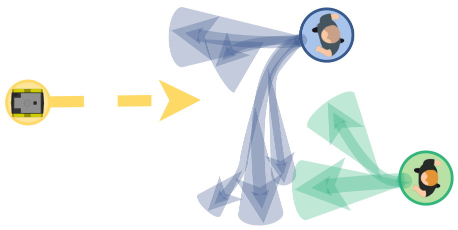


Figure 4.2: The chance constrained motion planning problem considered in this chapter with the robot (yellow) navigating under probabilistic motion predictions of obstacles (blue/green). The distribution of motion prediction can take any form but is visualized here with several modes (arrows/shaded regions).

4.2.2. SCENARIO-BASED PLANNING PROBLEM

Directly evaluating chance constraint (4.1d) is not computationally feasible in closed loop. The goal is to formulate a sampled deterministic version of the CCP, known as a Scenario Program (SP) [98]. The challenges for safe robot navigation within this framework are to determine the number of samples that must be drawn and, consequently, to identify the samples that affect the optimization.

4.2.3. PAPER ORGANIZATION

In the following, this chapter first considers a more general CCP formulation that can be solved by the proposed framework. This chapter provides a brief summary of the nonconvex scenario optimization framework of [98] to show how this general class of CCPs can be solved via its associated SP. This chapter then presents the main results in Sec. 4.4, which shows how this SP can be solved in closed loop. Finally, this chapter applies the main results in Sec. 4.5 in simulation by generating safe motion plans for a robot navigating among pedestrians.

4.3. NONCONVEX SCENARIO OPTIMIZATION: OVERVIEW AND LIMITATIONS

In the following, this chapter summarizes the main results of the NSO framework of [98] that are used to build the motion planning framework. To this end, consider the following generalization of Problem 1,

Problem 2 (General CCP)**.**

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{4.2a}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{x}_{init} \tag{4.2b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0,\dots,N-1 \tag{4.2c}$$

$$\mathbb{P}\left[g(\boldsymbol{x},\boldsymbol{\delta}) \le 0\right] \ge 1-\epsilon,\ \boldsymbol{\delta}\in\Delta. \tag{4.2d}$$

The constraints $g(\boldsymbol{x},\boldsymbol{\delta}) \le 0$ must be satisfied with a probability of at least $1-\epsilon$. The main idea of scenario optimization is to solve Problem 2 by imposing deterministic constraints for a set of *scenarios* $\Omega = \{\boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(S)}\} \in \Delta^S$, where[2] each scenario is independently extracted from $\mathbb{P}$. The number of sampled scenarios is known as the *sample size* S. Using scenarios, the SP for Problem 2 is formulated as

Problem 3 (General SP)**.**

$$
\begin{aligned}
\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \quad & \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) && \text{(4.3a)}\\
s.t. \quad & \boldsymbol{x}_0 = \boldsymbol{x}_{init} && \text{(4.3b)}\\
& \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ k = 0, \dots, N-1 && \text{(4.3c)}\\
& g(\boldsymbol{x}, \boldsymbol{\delta}^{(i)}) \le 0,\ \boldsymbol{\delta}^{(i)} \in \Omega,\ i = 1, \dots, S, && \text{(4.3d)}
\end{aligned}
$$

4

where chance constraint (4.2d) has been replaced by the deterministic constraints for each of the scenarios in (4.3d).
To simplify notation, this chapter defines a *decision* as $\boldsymbol{\theta} := \left[\boldsymbol{x}^T, \boldsymbol{u}^T\right]^T \in \Theta$, where $\Theta = \mathbb{X} \times \mathbb{U}$. Each scenario $\boldsymbol{\delta}^{(i)}$ imposes a constraint $\boldsymbol{\theta} \in \Theta_{\boldsymbol{\delta}^{(i)}}$ on the decision, with $\Theta_{\boldsymbol{\delta}^{(i)}} \subset \Theta$. Formally, to make a decision based on the scenarios, a *decision algorithm* $\mathcal{A}$ mapping the scenarios to a decision (i.e., solving Problem 3) is defined. This (sub)optimal decision $\boldsymbol{\theta}^* = \mathcal{A}(\Omega)$ is called the *scenario decision.* The probabilistic guarantees derived from the SP depend on the following assumption.

Assumption 3. *For any finite $S = 1, 2, \dots$ and for any multi-sample $\Omega \in \Delta^S$, it holds that*

$$
\mathcal{A}(\Omega) \in \Theta_{\boldsymbol{\delta}^{(i)}},\ i = 1, \dots, S. \tag{4.4}
$$

This chapter focuses its attention on the planning problem by considering only finite sample sizes S. Assumption 3 is not trivially satisfied for motion planning. It requires that a feasible trajectory exists for all possible extractions in the support, which is particularly problematic for unbounded distributions (e.g., Gaussians). This chapter will consider this assumption in more detail in Sec. 4.4.1.
Intuitively, the more scenarios that were used to compute the scenario decision, the lower is the probability that the resulting decision will violate the constraints. Formally, the *violation probability*, $V : \Theta \to [0, 1]$, given by

$$
V(\boldsymbol{\theta}) := \mathbb{P}\left[\boldsymbol{\delta} \in \Delta : \boldsymbol{\theta} \notin \Theta_{\boldsymbol{\delta}}\right], \tag{4.5}
$$

defines the probability that a decision $\boldsymbol{\theta}$ violates a newly observed scenario. This chapter also refer to this probability as the *risk* of the decision. Since the decision $\boldsymbol{\theta}$ depends on the realization of the randomly sampled scenarios, the violation probability $V(\boldsymbol{\theta})$ is in itself a random variable over the product probability measure, given by $\mathbb{P}^S = \mathbb{P} \times \ldots \times \mathbb{P}$ (S times). This chapter is therefore interested in lower bounding the *confidence*, which is the *probability* that the scenario decision achieves a risk of at most ϵ. The key variable to obtain this bound is the support subsample, defined as follows.

[2] Δ^S represents the S-fold Cartesian product of Δ associated with drawing S random samples

Definition 1. *[98]: Given a multi-sample $\Omega = \{\boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(S)}\}$, a support subsample $\mathcal{C} = \{\boldsymbol{\delta}^{(i_1)}, \dots, \boldsymbol{\delta}^{(i_n)}\}$ is a tuple of n elements extracted from the multi-sample with $i_1 < i_2 < \dots < i_n$, which gives the same solution as the original sample, that is,*

$$\mathcal{A}(\boldsymbol{\delta}^{(i_1)}, \dots, \boldsymbol{\delta}^{(i_n)}) = \mathcal{A}(\boldsymbol{\delta}^{(1)}, \dots, \boldsymbol{\delta}^{(S)}). \tag{4.6}$$

The cardinality of the support subsample is referred to as the *support size*, that is, $n := |\mathcal{C}|$. In the context of this chapter, a scenario is said to be *of support* if it is an element of the considered support subsample. A scenario that can be excluded from a support subsample without changing the solution is said to be *not of support*. For example, a sampled human trajectory $\boldsymbol{\delta}^{(i)}$ can be excluded from the support if it does not change the robot's optimal behavior under the current set of human trajectory samples.
The support size captures the number of scenarios necessary to hold the solution of the SP in place and is strongly correlated with its risk. This correlation can be used to derive a probabilistic guarantee on the solution of the SP using only the support and sample size. Denoting the *confidence* as $1-\beta$, Theorem 1 in [98] provides the following bound

$$\mathbb{P}^S[V(\boldsymbol{\theta}^*) > \epsilon(n)] \leq \sum_{n=0}^{S-1} \binom{S}{n} [1-\epsilon(n)]^{S-n} = \beta. \tag{4.7}$$

That is, the probability that the scenario decision $\boldsymbol{\theta}^*$ exceeds the acceptable risk ϵ, is upper bounded by β. The function $\epsilon(n) : \{0, \dots, S\} \to [0,1]$ is designed subject to (4.7) and $\epsilon(S) = 1$, which divides the risk over the range of the support from 0 to S. The following mapping divides the risk evenly over all support values

$$\epsilon(n) = \begin{cases} 1, & n > \bar{n}, \\ 1 - \left(\frac{\beta}{S}\binom{S}{n}\right)^{\frac{1}{S-n}}, & n \leq \bar{n}. \end{cases} \tag{4.8}$$

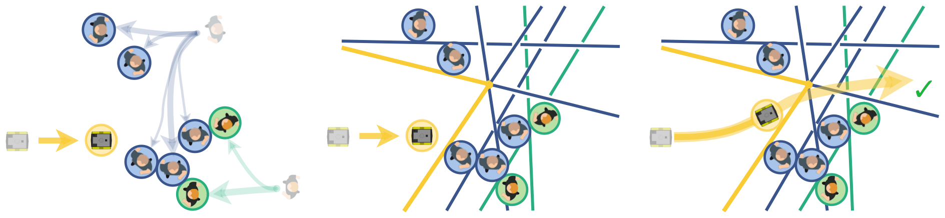
Notice that the violation probability increases with the support size. The more scenarios that are necessary to support a decision, the higher risk that decision is.
A general algorithm to determine the support is the *greedy* algorithm of [98]. After solving Problem 3, this algorithm removes one scenario at a time, solving Problem 3 again. If the solution changes for a scenario, then that scenario is part of the support. The samples remaining after checking all scenarios constitute a support subsample.

4.3.1. LIMITATIONS OF NSO FOR MOTION PLANNING

The NSO framework is not directly applicable to the motion planning problem because of two limitations.

Limitation 1. *Assumption 3 is not always satisfied. Drawing samples from the probability distribution of future obstacle motion can lead to an infeasible problem (e.g., if samples are close to the robot). For unbounded distributions (e.g., Gaussians), such samples can always be extracted.*

Limitation 2. *Certifying the trajectory requires solving S additional optimization problems of similar complexity to the original problem (the greedy support estimation), which is computationally intractable for real-time trajectory optimization.*



(a) Scenarios are sampled from the trajectory distributions. Each time instance of SP (Eq. 4.9) is associated with a set of sampled obstacle positions as visualized by the green and blue circled pedestrians.

(b) Linear constraints are constructed between sampled obstacles and the robot, and are reduced to a probabilistic safe polytope for each time instance and robot disc.

(c) Problem 5 is solved via Algorithm 3. The resulting trajectory is certified up to a probabilistic bound.

Figure 4.3: Schematic illustration of Safe Horizon MPC applied to a mobile robot.

The goal of this chapter is to address these limitations so that Problem 1 can be solved with an SP to bound the joint CP.

4.4. Safe Horizon Model Predictive Control

This section introduces the proposed safe motion planning framework. The SP is reformulated to obtain a real-time solvable problem that satisfies Assumption 3. Then, this chapter shows how the support can be estimated during optimization. Finally, the sample size of the SP is derived.

4.4.1. Motion Planning Scenario Program

Safe Horizon MPC bounds the joint CP by solving an SP. In the planning problem, the scenarios extracted from $\mathbb{P}$ represent realizations of the future motion of all obstacles in the near future (see Fig. 4.3a). Using these scenarios, this chapter can construct the following SP from the CCP in Problem 1

Problem 4 (Motion Planning SP)**.**

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}\\ d\in\mathbb{R}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) + w_d\|d\|_2^2 \tag{4.9a}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{x}_{init} \tag{4.9b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k), \; k = 0,\dots,N-1 \tag{4.9c}$$

$$\bigwedge_{i=0}^{S}\bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\bigwedge_{j=0}^{M}\left(\|\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}^{(i)}\|_2 \geq r + d\right), \tag{4.9d}$$

where (4.9d) represents the scenario constraints (bundled with the “and” operator). Compared to the general SP in Problem 3, Problem 4 adds a single joint slack variable $d\in\mathbb{R}$ over the collision constraints that denotes the minimum distance kept to all obstacles over the duration of the trajectory. Because of the slack introduced through d, there is always a solution to Problem 4 and Assumption 3 is satisfied (addressing Limitation 1).

The violation probability associated with Problem 4 is

$$\mathbb{P}\left[\boldsymbol{\delta} \in \Delta \mid \bigwedge_{k=1}^{N} \bigwedge_{c=0}^{n_d} \bigwedge_{j=0}^{M} \left(||\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}||_2 \geq r + d\right)\right] \geq 1 - \epsilon.$$

For $d = 0$, the violation probability matches that of Problem 1 and is bounded through the NSO framework. When $d \leq 0$, the optimization could not find a sufficiently safe solution. The robot can use this information to adjust its behavior and ensure safety (e.g., by slowing down or using a fallback plan).

4.4.2. Improved Computational Efficiency

Problem 4 cannot be solved efficiently. The collision-free space described by (4.9d) is nonconvex and can lead to high support (i.e., requiring many samples). This chapter considers a more efficient overapproximation of the collision avoidance constraints with lower support.

This chapter linearizes the collision regions with respect to the previously planned robot trajectory (denoted $\hat{\boldsymbol{p}}$). After linearization, each scenario is associated with a linear constraint (depicted in Fig. 4.3b). For a previous robot position $\hat{\boldsymbol{p}}_k$ and obstacle position $\boldsymbol{\delta}_k$ the constraints are given by

$$\mathcal{H}(\hat{\boldsymbol{p}}_k, \boldsymbol{\delta}_k, d) = \{\boldsymbol{p}_k \mid \boldsymbol{A}(\hat{\boldsymbol{p}}_k, \boldsymbol{\delta}_k)^T \boldsymbol{p}_k \leq b(\hat{\boldsymbol{p}}_k, \boldsymbol{\delta}_k, d)\}, \tag{4.10}$$

where

$$\boldsymbol{A}(\hat{\boldsymbol{p}}_k, \boldsymbol{\delta}_k) = \frac{\boldsymbol{\delta}_k - \hat{\boldsymbol{p}}_k}{||\boldsymbol{\delta}_k - \hat{\boldsymbol{p}}_k||}, \; b(\hat{\boldsymbol{p}}_k, \boldsymbol{\delta}_k, d) = \boldsymbol{A}^T \boldsymbol{\delta}_k - (r + d).$$

The linearized collision region contains the original collision region for zero slack, preserving the probabilistic guarantees when $d = 0$. In addition, the linearization is applied locally to each timestep k and robot disc c, resulting in a locally accurate approximation of the original collision regions. For the linearized constraints, this chapter obtains the following SP:

$$\min_{\substack{\boldsymbol{u} \in \mathbb{U}, \boldsymbol{x} \in \mathbb{X} \\ d \in \mathbb{R}}} \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) + w_d ||d||_2^2 \tag{4.11a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{4.11b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \; k = 0, \ldots, N-1 \tag{4.11c}$$

$$\bigwedge_{i=0}^{S} \bigwedge_{k=1}^{N} \bigwedge_{c=0}^{n_d} \bigwedge_{j=0}^{M} \left(\boldsymbol{p}_k^c \in \mathcal{H}\left(\hat{\boldsymbol{p}}_k^c, \boldsymbol{\delta}_{k,j}^{(i)}, d\right)\right). \tag{4.11d}$$

The linearization exploits the geometry of the motion planning problem to reduce the size of the support. Most of the linearized constraints become redundant (they are fully contained in other constraints) and can be excluded from the planning problem.

To drastically reduce the computational demand of this formulation, the constraints in (4.11d) can be reordered as

$$\bigwedge_{k=1}^{N} \bigwedge_{c=0}^{n_d} \left[\bigwedge_{i=0}^{S} \bigwedge_{j=0}^{M} \left(\boldsymbol{p}_k^c \in \mathcal{H}\left(\hat{\boldsymbol{p}}_k^c, \boldsymbol{\delta}_{k,j}^{(i)}, d\right)\right)\right], \tag{4.12}$$

to pair constraints that apply to a single robot disc position $\boldsymbol{p}_k^c$. Because of the overlap between the constraints, each of these constraint pairings can be described by a small subset of the constraints for $d = 0$ (see Fig. 4.3b). The constraints (4.11d) can, therefore, be reduced to free-space polytopes before optimization, which significantly reduces computation times. This chapter denotes these polytopes, for disc c and time step k as

$$\mathcal{P}_k^c = \left\{ \boldsymbol{p}_k^c \,\middle|\, \bigwedge_{(i,j)\in\mathcal{I}_k^c} \boldsymbol{p}_k^c \in \mathcal{H}\left(\hat{\boldsymbol{p}}_k^c, \boldsymbol{\delta}_{k,j}^{(i)}, 0\right) \right\}, \tag{4.13}$$

where the indices of scenarios that form the boundary of the polytope are collected in the set $\mathcal{I}_k^c$. Merging constraints into this polytope typically reduces the number of constraints by a factor of $10^2 - 10^3$. The final SP that is solved online is given by

Problem 5 (Safe Horizon MPC)**.**

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}\\ d\in\mathbb{R}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) + w_d ||d||_2^2 \tag{4.14a}$$

$$s.t. \quad \boldsymbol{x}_0 = \boldsymbol{x}_{init} \tag{4.14b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \; k = 0, \ldots, N-1 \tag{4.14c}$$

$$\boldsymbol{p}_k^c \in \mathcal{P}_k^c, \quad \forall k, \forall c. \tag{4.14d}$$

The SH-MPC problem can be solved in real-time while bounding the joint CP of its trajectory (see Fig. 4.3c).

4.4.3. ESTIMATING THE SUPPORT

The joint CP that is bounded by SH-MPC depends on the sample size S and the support n. In the following, this chapter proposes an estimate of the support $\hat{n} \geq n$ that is efficiently computed simultaneously with the trajectory, during optimization.

For a convex SP, the support can easily be computed as its support constraints are active [100], [148]. Computing the support in the nonconvex case is much harder as this property does not hold. This chapter shows here that the support can be estimated through the active constraints after each *iteration* of the nonconvex optimization. An iteration refers to the procedure that is repeated to solve the nonconvex optimization, such that the decision algorithm can be described by a repeated sequence of iterations $\mathcal{A}^l : \Theta \times \Delta^S \to \Theta$, $l \in 0, \ldots, L$ as

$$\mathcal{A} = \mathcal{A}^L(\ldots \mathcal{A}^1(\mathcal{A}^0(\boldsymbol{\theta}^0, \Omega), \Omega) \ldots, \Omega). \tag{4.15}$$

The support of an iteration l is a subset of scenarios Ω_l for which $\mathcal{A}(\theta^{l-1}, \Omega) = \mathcal{A}(\theta^{l-1}, \Omega_l)$. That is, replacing all scenarios with the support of iteration l, does not change the decision after iteration l. The support of the decision algorithm can be connected with that of its iterations by the following lemma.

Lemma 1. *Consider a decision algorithm $\mathcal{A}$, separated according to* (4.15)*. Its support $\mathcal{C}$ satisfies*

$$\mathcal{C} \subseteq \Omega \setminus \Omega_{ns}, \quad \Omega_{ns} = \bigcap_{l=0}^{L} \Omega_{ns}^l, \tag{4.16}$$

where

$$\Omega^l_{ns} = \left\{\boldsymbol{\delta}^{(i)} \in \Omega \mid \mathcal{A}^l(\boldsymbol{\theta}^l, \Omega) = \mathcal{A}^l(\boldsymbol{\theta}^l, \Omega \backslash \boldsymbol{\delta}^{(i)})\right\}, \tag{4.17}$$

is the set of scenarios not of support in iteration l.

Proof. Scenarios in Ω_{ns} can be excluded for all l without changing the solution, that is, by (4.16) and (4.17),

$$\mathcal{A}(\boldsymbol{\theta}^0, \Omega \backslash \boldsymbol{\delta}^{(i)}) = \mathcal{A}(\boldsymbol{\theta}^0, \Omega) = \boldsymbol{\theta}^*, \tag{4.18}$$

for all $\boldsymbol{\delta}^{(i)} \in \Omega_{\mathrm{ns}}$. Therefore, by Definition 1 all scenarios in Ω_{ns} are *not* of support for $\mathcal{A}$. The support of $\mathcal{A}$ is therefore in the complement of this set with respect to the set Ω and the result follows. □

4

The support set obtained through Lemma 1 is an overestimation. It is possible that a scenario changes the solution of an intermediate iteration without changing the final solution.
To apply this lemma to SH-MPC, this chapter notes that a local optimum of Problem 5 can be computed by iteratively linearizing the problem and solving a convex optimization. In this case, each iteration of the solver is a convex scenario optimization[3] for which the support constraints are active [100]. This chapter explicitly imposes this assumption on the solver.

Assumption 4. *Each iteration $\mathcal{A}^l$ of decision algorithm $\mathcal{A}$ solves a convex optimization problem.*

The active constraints can be identified by verifying which of the scenario constraints of the convex program are exactly satisfied. This chapter therefore requires the additional assumption that the constraints of the convex problem[4] are satisfied in each iteration.

Assumption 5. *The solution computed by each iteration $\mathcal{A}^l$ is feasible with respect to its inequality constraints.*

Using these assumptions, this chapter proposes the following support estimation.

Theorem 3. *If iterations $\mathcal{A}^l$ of the decision algorithm satisfy Assumptions 4 and 5, then the support of Problem 3 satisfies*

$$\mathcal{C} \subseteq \bigcup_{j=0}^{L} \Omega^j_{active} = \hat{\mathcal{C}}, \quad \hat{n} := |\hat{\mathcal{C}}|, \tag{4.19}$$

where, with g^l the inequality constraints of $\mathcal{A}^l$,

$$\Omega^l_{active} = \{\boldsymbol{\delta}^{(i)} \in \Omega \mid \exists k \; g^l(\boldsymbol{x}^l_k, \boldsymbol{\delta}^{(i)}_k, d^l) = 0\}, \tag{4.20}$$

denote the active constraints in iteration l.

[3]For example, in Sequential Quadratic Programming (SQP), each iteration $\mathcal{A}^l$ refers to the inner QPs.
[4]In general, the scenario constraints are nonlinear through the robot dynamics. When the problem is made convex, constraints are derived from the linearized dynamics. This chapter only requires that these are satisfied.

Proof. Under Assumption 4, the support constraints of iteration l are in the set Ω^l_{active} (all constraints are satisfied by Assumption 5). Under convexity, Ω^l_{active} is therefore the complement of the set Ω^l_{ns}. Invoking Lemma 1 and using De Morgans Law [149], (4.19) is obtained. □

The support of SH-MPC can therefore be estimated by the aggregated set of active scenarios over all iterations, addressing Limitation 2. In practice, this chapter uses Sequential Quadratic Programming (SQP) [4] Chapter 18 to solve Problem 5, which satisfies Assumption 4 (convexity of iterates) and Assumption 5 (feasibility of intermediate iterates).

4.4.4. DETERMINING THE SAMPLE SIZE

With the support of SH-MPC estimated through Theorem 3 and directly available after optimization, it remains to determine the sample size S for SH-MPC such that the joint CP of its trajectory is at most ϵ.
Problem 5 can only be solved once due to the strict real-time requirements on the planner. This chapter therefore proposes to find a sufficiently high S that certifies the joint CP almost always. To that end, this chapter defines a support limit $\bar{n}$ describing a support size that is expected not to be exceeded and uses it to certify the optimized trajectory in practice.

Theorem 4. *SH-MPC (Problem 5) with sample size S, computed from (4.8) with support limit $\bar{n}$, does not exceed a risk of ϵ (with confidence $1-\beta$) if its support is lower than the support limit, i.e., if $n \leq \bar{n}$.*

Proof. SH-MPC satisfies Assumption 3 and with $\epsilon(n)$ as in (4.8), Theorem 1 in [98] ensures that (4.7) holds. Since $\epsilon(n)$ in (4.8) is monotonically increasing in n (i.e., $\epsilon(n+1) > \epsilon(n)$, $\forall n < S$) and given that the computed S ensures that $\epsilon(\bar{n}) \leq \epsilon$, it holds that $\epsilon(n) \leq \epsilon, \forall n \leq \bar{n}$. □

Theorem 4 shows that SH-MPC solves the CCP in Problem 1 if its support is lower than the support limit (and if $d = 0$).
To set the support limit in practice, the intended problem is solved repeatedly while keeping track of the highest observed support. This chapter uses this empirical worst-case support as the support limit. This approach is conservative as the support limit is higher than necessary in many iterations, leading to conservative trajectories. Conservatism can be reduced by running several instances of SH-MPC in parallel (along the lines of [9]). This is considered future work.

4.4.5. ALGORITHM OUTLINE

Algorithm 3 summarizes the SH-MPC framework. Offline, the sample size is computed based on the joint CP ϵ, confidence β, and support limit $\bar{n}$ (Line 2). This chapter provides Jupyter notebook [150] that performs this computation using a bisection of (4.8). Online, the predicted probability distribution is received from the perception module and scenarios are sampled from it (Line $4-5$). This chapter solves l iterations of Problem 5, determining the active scenarios and aggregating the support set in each iteration (Line $7-9$). If the slack is zero and the support limit is not exceeded, then the final trajectory is certifiably safe and its first input is executed (Line $10-12$).

Algorithm 3: Safe Horizon MPC

```
1  Input: $\epsilon$, $\beta$ and $\bar{n}$
2  $S \leftarrow$ Bisection of Eq. 4.8 using $\epsilon, \beta, \bar{n}$
3  while True do
4      $\mathbb{P} \leftarrow$ Motion prediction received from perception
5      $\Omega \leftarrow$ Draw $S$ samples from $\mathbb{P}$
6      for $l = 1, \ldots, L$ do
7          $\boldsymbol{x}^l, \boldsymbol{u}^l, d^l \leftarrow$ Solve an iteration of Problem 5
8          $\Omega^l_{\text{active}} \leftarrow$ Determine active scenarios (Eq. 4.20)
9          $\hat{n}^l \leftarrow$ Aggregate the support (Eq. 4.19)
10     if $d^L = 0$ and $n \leq \bar{n}$ then
11         Trajectory attains a collision risk of at most $\epsilon$
12         Actuate $\boldsymbol{u}^L_0$
```

4.5. RESULTS

This section compares SH-MPC in simulation against two MPC baselines and validates the approach experimentally. A video of the simulations and experiments is available in [146].

4.5.1. SIMULATION SETUP

The simulations consider a mobile robot moving through an environment with pedestrians (see Fig. 4.4) in which the robot is modeled by a kinematic unicycle model [144]. This chapter assumes that the distribution of pedestrian motion is known in order to evaluate the performance of the planner in isolation (i.e., without prediction errors). A Gaussian case is validated first, where the baselines may leverage the shape of the distribution to approximate the probabilistic collision-free space accurately. In this case, the pedestrian dynamics are given by

$$\boldsymbol{\delta}_{k+1} = \boldsymbol{\delta}_k + (\boldsymbol{v} + \boldsymbol{\delta}_{w,k})dt, \quad \boldsymbol{\delta}_{w,k} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w), \tag{4.21}$$

where $\boldsymbol{\Sigma}_w \in \mathbb{R}^{2\times 2}$ is a diagonal covariance matrix (i.e., random variables in the x and y direction are independent). Its diagonal entries $\sigma_{wx} = \sigma_{wy} = 0.3$ are kept constant over the horizon. The nominal velocity is denoted by $\boldsymbol{v} \in \mathbb{R}^2$. The pedestrian and robot radius are 0.3m and 0.325m, respectively.

4.5.2. IMPLEMENTATION OF SH-MPC

All baselines and SH-MPC use the Model Predictive Contouring Control formulation in [27], which tracks a reference path and reference velocity while penalizing robot inputs. Problem 5 is solved using the Forces Pro SQP solver [143] with at most 12 iterations. This chapter empirically selected the support limit $\bar{n} = 10$. For a joint CP of $\epsilon = 0.05$ and a confidence of $\beta = 0.01$ the sample size is $S = 1351$. If the support limit is exceeded or the slack is nonzero, the robot slows down. The linearization of the constraints requires the previous plan to be feasible. This chapter uses a projection step to ensure that this holds in all time steps. This projection step consists of a projection orthogonal to the direction of robot movement, which almost always results in a feasible plan. In the remaining cases, a feasibility program using Douglas-Rachford Splitting [151] is applied. SH-MPC is implemented in C++/ROS and will be released open source.

4.5.3. BASELINES

This chapter compares SH-MPC against two baselines that constrain the marginal CP, that is, the independent CP per time instance and/or obstacle.

1. **CC-MPC** [84]: Marginal CP per time instance and obstacle.
2. **S-MPCC** [88]: Marginal CP per time instance.

CC-MPC is strictly applicable to Gaussian distributions. It approximates the collision probability via the Cumulative Density Function (CDF) of the Gaussian distribution. The linearized collision avoidance chance constraint

$$\mathbb{P}\left[\boldsymbol{a}_{k,j}^T(\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}) \geq r\right] \geq 1 - \epsilon_k,\ \boldsymbol{a}_{k,j} = \frac{\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}}{||\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}||},$$

is equivalent, under a Gaussian distribution of $\boldsymbol{\delta}_{k,j}$, to

$$\boldsymbol{a}_{k,j}^T(\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}) - r \geq \operatorname{erf}^{-1}(1 - 2\epsilon_k)\sqrt{2\boldsymbol{a}_{k,j}^T \boldsymbol{\Sigma} \boldsymbol{a}_{k,j}},$$

where erf^{-1} is the inverse standard error function and $\boldsymbol{\Sigma}$ is the covariance matrix of the uncertainty. This constraint is imposed separately for each time step and obstacle.
S-MPCC handles arbitrary distributions. It solves an SP where scenario constraints are posed on the marginal distributions. For each time k, it samples from the independent obstacle distribution at k to obtain a collision-free polygon. It is assumed that all constraints in the polygon are of support (set to 20 in these simulations), which for $\epsilon_k = 0.0025$ requires $S = 75946$. Samples in the center of the distribution are pruned in the Gaussian case to reduce the number of samples considered online.
Since SH-MPC is characterized by a single bound ϵ on the trajectory CP, while the baselines specify bounds ϵ_k on the CP for each k and for each obstacle, this chapter considers three versions of the baselines. The first set $\epsilon_k = \epsilon$, which is not provably safe but relies on updates of the controller to remain safe. The second version sets $\epsilon_k = \frac{\epsilon}{N}$, accounting for the marginal approximation over time since $\sum_k \epsilon_k = \epsilon$, but ignoring marginalization per obstacle. The third version accounts for both marginalizations, setting $\epsilon_k = \frac{\epsilon}{NM}$ such that $M\sum_k \epsilon_k = \epsilon$. Only this last version attains the same safety guarantee as SH-MPC. This version is considered in crowded environments where the marginal CP is violated otherwise.

4.5.4. WEIGHTS AND PARAMETERS

The only difference between the planners is their collision avoidance constraints. Planners have identical solvers, weights and cost functions. Weights of the MPC problem are given in Table 4.1. The horizon consists of $N = 20$ steps, with a discretization step of 0.2s, giving a time horizon of 4.0s. The control rate is 20Hz, corresponding to a sampling time of 50ms. The computer running the simulations is equipped with an Intel i9 CPU@2.4GHz.

Table 4.1: Weights of the MPC problem.

Contour	Lag	Velocity	Acceleration	Ang. Velocity
0.005	0.1	0.05	0.05	0.05

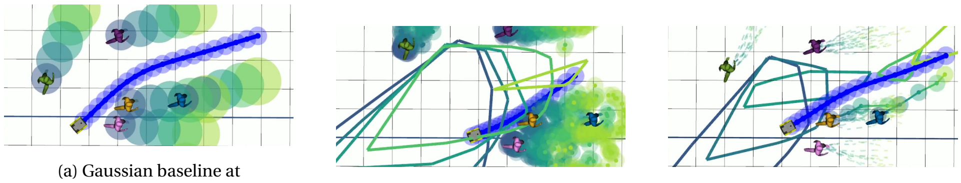

(a) Gaussian baseline at $\epsilon_k = 0.0003125$. (b) S-MPCC at $\epsilon_k = 0.0025$. (c) SH-MPC at $\epsilon = 0.05$.

Figure 4.4: Snapshot of the 8 pedestrian simulation under Gaussian pedestrian motion. The robot plan and associated collision areas are drawn in blue. For all methods, visualizations are shown in blue to green colors for stages 0, 5, 10, 15 and 19, respectively. (a) Circles show the level sets of the Gaussian distribution at the specified ϵ. (b) Sampled pedestrian positions (excluding pruned samples in the center) are drawn as points with their collision area, borders of the safe polytopes are drawn as colored lines. (c) Similar to (b) but depicting sampled *trajectories* as dashed lines and support constraints are highlighted.

4.5.5. BASELINE COMPARISON - GAUSSIAN UNCERTAINTIES

This chapter considers an environment with multiple dynamic pedestrians following the dynamics in Eq. (4.21) where $\boldsymbol{v}$ describes a constant velocity. The actual CP of the motion plan is validated offline after the experiments through Monte Carlo sampling for all methods, where the dynamics in Eq. (4.21) are used to generate the samples. This chapter computes the CP by dividing the number of samples where the robot and obstacle discs overlap at any stage by the total number of samples (set to 10^5). The marginal CP (CP_k) is computed without taking prior collisions into account. Scenarios with 4 and 8 pedestrians are considered, respectively. Fig. 4.4 depicts snapshots of the simulations with 4 pedestrians.

Results for 4 pedestrians are summarized in Table 4.2. CC-MPC bounds the marginal CP (CP_k) accurately. However, when $\epsilon_k = 0.05$, its maximum joint CP is 0.1876, which exceeds 0.05. The trajectories are safe under $\epsilon_k = 0.0025$ in practice with a maximum overall CP of 0.0096, although it is theoretically not accounting for the obstacle marginalization. SH-MPC attains a similar safe maximum CP of 0.0081. Both versions of CC-MPC

Table 4.2: Statistical results over 100 experiments of the marginal CP (“CP_k”) and trajectory CP (“CP”), the task duration, traveled distance, minimum distance to the pedestrians and computation times for the unimodal simulation with 4 pedestrians. For the CPs we report the maximum observed over all experiments and compare it to the specified bound (a dash indicates that no bound is specified on the particular CP). Other results are reported as “average (standard deviation)” unless stated otherwise. Methods are grouped by their safety guarantee. The joint CP of SH-MPC is in this case similar to that of the baseline at $\epsilon_k = 0.0025$.

Method	Max CP_k/Spec. (%)	Max CP/Spec. (%)	Dur. [s]	Min Dist. [m]	Runtime (Max) [ms]
S-MPCC ($\epsilon_k = 0.05$)	0.0034 / 0.0500 (7)	0.0038 / - (-)	10.7 (1.2)	0.30 (0.04)	30 (63)
CC-MPC ($\epsilon_k = 0.05$)	0.0469 / 0.0500 (94)	0.1876 / - (-)	10.0 (0.1)	0.07 (0.02)	**9** (35)
S-MPCC ($\epsilon_k = 0.0025$)	0.0001 / 0.0025 (6)	0.0002 / - (-)	13.0 (0.9)	0.42 (0.20)	44 (124)
CC-MPC ($\epsilon_k = 0.0025$)	0.0022 / 0.0025 (89)	0.0096 / - (-)	**9.9** (0.1)	0.19 (0.03)	**9** (31)
SH-MPC ($\epsilon = 0.05$)	0.0077 / - (-)	0.0081 / 0.0500 (16)	11.7 (1.4)	0.29 (0.06)	24 (59)

Table 4.3: Results for the unimodal simulation with 8 pedestrians. Displayed results follow the notation in Table 4.2. SH-MPC maintains a similar joint CP as with 4 pedestrians, contrary to the baselines.

Method	Max CP$_k$/Spec. (%)	Max CP/Spec. (%)	Dur. [s]	Min Dist. [m]	Runtime (Max) [ms]
CC-MPC ($\epsilon_k = 0.05$)	0.0929 / 0.0500 (186)	0.2264 / - (-)	**10.3** (2.0)	**0.12** (0.03)	**10** (45)
CC-MPC ($\epsilon_k = 0.0025$)	0.0047 / 0.0025 (187)	0.0159 / - (-)	16.9 (4.5)	0.24 (0.07)	**10** (39)
CC-MPC ($\epsilon_k = 0.0003125$)	0.0008 / 0.0025 (30)	0.0019 / - (-)	18.4 (2.4)	0.32 (0.07)	**11** (33)
SH-MPC ($\epsilon = 0.05$)	0.0073 / - (-)	0.0092 / 0.0500 (18)	**16.0** (1.9)	0.34 (0.03)	27 (66)

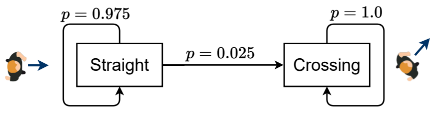


Figure 4.5: Markov-chain modeling a crossing pedestrian.

move more efficiently than SH-MPC in this scenario. CC-MPC uses the distribution and can tightly constrain the marginal CP, while in this scenario with 4 pedestrians, ignoring the correlation between obstacles does not make the trajectory unsafe.

Results for 8 pedestrians are summarized in Table 4.3. In this environment, the marginal CP of CC-MPC exceeds the specification. This is a result of the marginalization per obstacle as illustrated by Fig. 4.6 on a 1-dimensional example. This chapter therefore also compares against the Gaussian baseline at $\epsilon_k = \frac{0.0025}{8} = 0.0003125$, which attains the same safety guarantees as SH-MPC. The results indicate that SH-MPC moves through this environment significantly faster than the baseline with the same safety guarantees. SH-MPC shows no significant change in its joint CP compared to the previous simulation as it considers the joint distribution while the risk of the baselines increased significantly. Additionally, the safe baseline with $\epsilon_k = 0.0003125$ is excessively conservative with a CP of 0.0019 versus 0.0092 of SH-MPC.

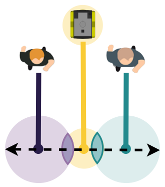
4.5.6. BASELINE COMPARISON - GAUSSIAN MIXTURE

The distribution is modified in the following to incorporate a probability that the pedestrians will cross. This scenario is encoded with a Markov Chain (see Fig. 4.5) that changes pedestrian movement from horizontal to diagonal in addition to the Gaussian process noise of the previous simulations. The pedestrian dynamics are defined as

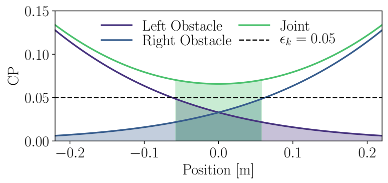
$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + (B\boldsymbol{v} + \boldsymbol{\delta}_{w,k})dt, \quad \boldsymbol{\delta}_{w,k} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{w,k}), \tag{4.22}$$

Table 4.4: Results for the multimodal simulation with 8 pedestrians. Displayed results follow the notation in Table 4.2. SH-MPC maintains a similar joint CP when the probability distribution describing pedestrian motion is non Gaussian.

Method	Max CP$_k$/Spec. (%)	Max CP/Spec. (%)	Dur. [s]	Min Dist. [m]	Runtime (Max) [ms]
CC-MPC ($\epsilon_k = 0.05$)	0.1027 / 0.0500 (205)	0.3175 / - (-)	16.6 (4.7)	**0.16** (0.18)	100 (340)
CC-MPC ($\epsilon_k = 0.0025$)	0.0053 / 0.0025 (213)	0.0196 / - (-)	18.2 (4.6)	0.35 (0.13)	106 (299)
CC-MPC ($\epsilon_k = 0.0003125$)	0.0007 / 0.0025 (28)	0.0028 / - (-)	18.8 (4.4)	0.45 (0.12)	106 (323)
SH-MPC ($\epsilon = 0.05$)	0.0081 / - (-)	0.0107 / 0.0500 (21)	**16.3** (4.3)	0.36 (0.14)	**27** (67)



(a) Example case



(b) Joint and marginalized risk in the direction of the black dashed arrow in (a).

Figure 4.6: A 1D illustration of the case where two obstacles constrain the robot. Even though the marginal probability of collision for each obstacle is less than ϵ_k (shaded tails) in the center region, the joint probability of collision in the feasible region (green shaded area) is larger than ϵ_k.

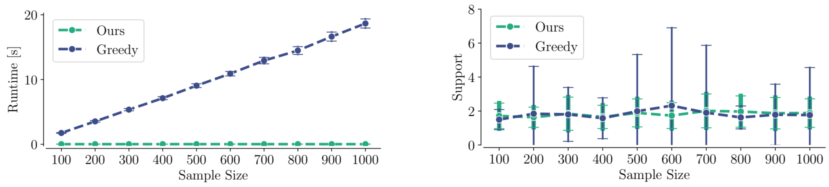


Figure 4.7: The mean and standard deviation of the runtime (top) and estimated support size (bottom).

where B is either $B_h = \begin{bmatrix}1 & 0\end{bmatrix}^T$ or $B_d = \begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix}^T$ depending on the state of the Markov Chain. The uncertainties associated with this motion can be modeled as a Gaussian Mixture Model (GMM), where each possible state transition in the Markov Chain leads to a separate mode with an associated probability (in total 21 modes). For example, the mode where the pedestrian crosses after two steps occurs with probability $p_2 = (1 - 0.025)0.025$. This chapter applies CC-MPC to the GMM distribution by formulating the constraints at a risk of ϵ_k for all modes, which can lead to conservatism when multiple modes influence the plan. Since previous simulations showed that S-MPCC performs worse than the other two methods, this chapter does not include it in this comparison. The environment contains 8 pedestrians.

Results are summarized in Table 4.4. SH-MPC outperforms the baselines on almost all metrics. The computation times of the Gaussian method are also excessive due to the many modes to be considered, while the computation times of SH-MPC are unaffected.

4.5.7. EMPIRICAL ANALYSIS AND SENSITIVITY

SUPPORT ESTIMATION

This section compares the proposed support estimation in Eq. (4.19) with the default Greedy algorithm of [98] in a scenario with one static pedestrian. The sample size is manually varied between 100 and 1000 and collects, for 100 iterations of each sample size, the runtime of the optimization with support estimation and the estimated support size. Fig. 4.7 shows that, while the runtime of the proposed support estimation is negligible compared to the optimization, the greedy algorithm takes roughly $S + 1$ times as long to solve.

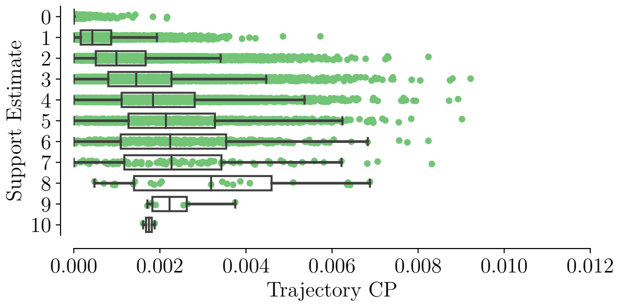


Figure 4.8: The empirical distribution of the joint CP for SH-MPC for each estimated support over 33564 planner iterations (in simulations of Sec. 4.5.5). Dots denote individual planner iterations. Boxplots denote the statistics per support value.

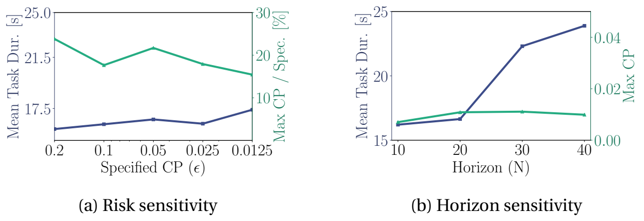


Figure 4.9: Sensitivities of the empirical CP and task duration with respect to (a) the specified CP with constant $N = 20$ and (b) the horizon length with constant specified CP $\epsilon = 0.05$, evaluated over 25 experiments in the setting of Sec. 4.5.5.

Empirical Risk Distribution

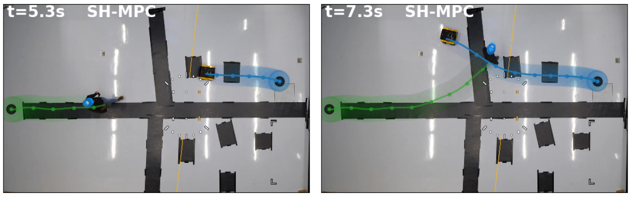
This chapter visualizes the empirical distribution of the joint CP for each value of the support estimate in Fig. 4.8 computed for the simulations of Sec. 4.5.5. In line with the theory, the trajectory CP is on average higher for higher support values. The support limit ($n = 10$) is reached in only 2 out of 33564 cases. In the other cases, the implemented support limit is conservative.

Sensitivity to ϵ and N

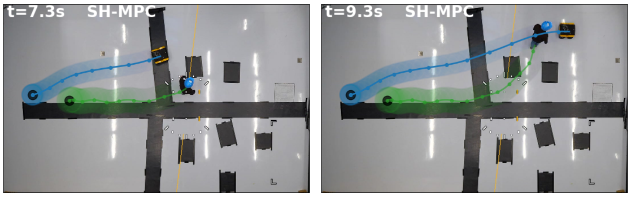
This section validates the sensitivity of SH-MPC to varying risk specifications (ϵ) and horizon lengths (N) in the scenario of Sec. 4.5.5. Fig. 4.9a shows that reducing the specified risk results in longer task durations and that the approach becomes slightly more conservative for lower risk specifications. In line with the theory, Fig 4.9b indicates that changing the horizon length does not significantly affect the CP. Instead, the trajectories become more cautious when the same risk must be guaranteed over a longer duration.

4.5.8. Real-World Experiments

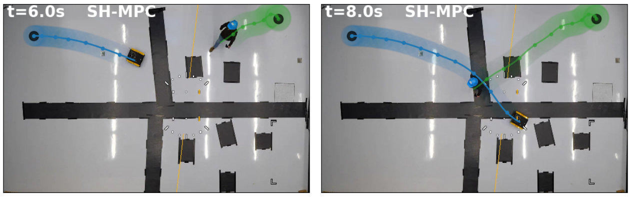
This chapter validates SH-MPC in the real world by applying the planner on a mobile robot navigating among pedestrians. The experimental setup consists of a Clearpath Jackal and up to 3 pedestrians in a 7m×9m space. The positions of all agents are detected with a motion capture system. The robot is given a reference path along the center of the



(a) The pedestrian turns in a straight encounter.



(b) The pedestrian turns with the robot passing from behind.



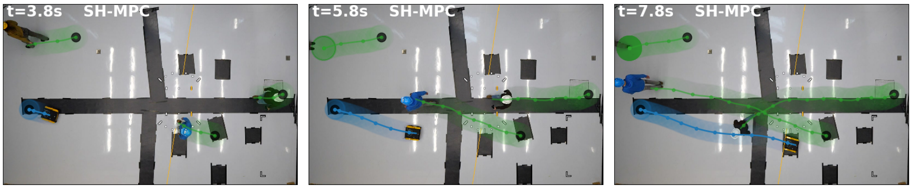
(c) The pedestrian speeds up.

Figure 4.10: Trajectories of the robot and one pedestrian overlayed with top-view camera images. By accounting for the possible future motion of the pedestrian, SH-MPC remains safe and smooth even when the pedestrian deviates from constant velocity behavior.

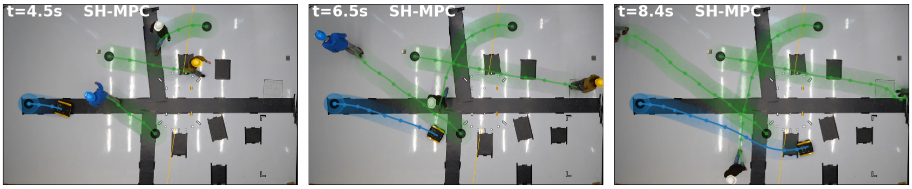
space and a reference velocity of 1.5m/s. When the robot reaches the end of the reference path, it rotates without MPC control and is given the reversed reference path. The robot is controlled at 20Hz by SH-MPC which models the dynamics as a second-order unicycle. This chapter specifies an acceptable risk of $\epsilon = 0.05$ over a 20 step and 4s horizon with confidence $\beta = 0.01$ and support limit $\bar{n} = 6$. The pedestrian motion predictions follow Gaussian constant velocity dynamics (Eq. 4.21), where an Extended Kalman Filter (EKF) is used to estimate the velocity.

Fig. 4.10 depicts 3 experiments with 1 pedestrian. SH-MPC keeps its distance from the pedestrian so that its motion remains smooth even when the pedestrian deviates from constant velocity behavior. In Figs. 4.10a and 4.10b, it keeps sufficient lateral distance when passing, while in Fig. 4.10c, it reacts to the pedestrian speeding up its pace. Fig. 4.11 shows 3 experiments with 3 pedestrians. The robot evades the pedestrians smoothly in this more crowded environment. In Figs. 4.11a and 4.11b, the robot smoothly evades multiple pedestrians. In Fig. 4.11c, the planner first evades 2 pedestrians, then reverses to ensure the safety of the third, fast-moving pedestrian.

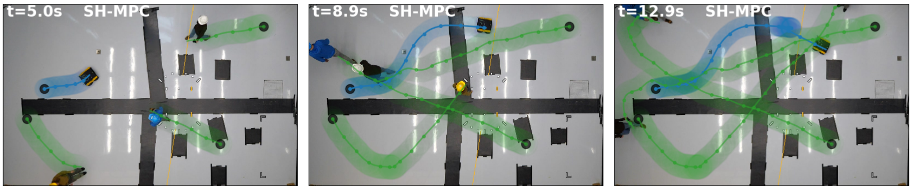
No collisions were observed in any of the experiments. Because the proposed planner is a local planner, it does sometimes get infeasible when its intended passing maneuver becomes impossible within the specified risk. This can be resolved by considering multiple distinct maneuvers in parallel (e.g., as considered in [152]) but this is outside of the



(a) The robot evades two pedestrians smoothly.



(b) During a passing maneuver, the robot keeps a sufficient distance.



(c) The robot passes two pedestrians, then reverses to ensure the safety of a third, fast-moving pedestrian.

Figure 4.11: Trajectories of the robot and three pedestrians overlayed with top-view camera images. The robot consistently avoids collisions with pedestrians.

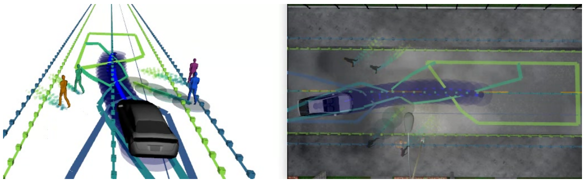
scope of this chapter.

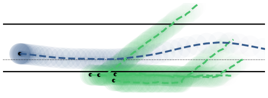
4.5.9. Autonomous Navigation in an Urban Environment

SH-MPC can be applied to different robot morphologies and scenarios. To demonstrate this, this chapter deploys the proposed approach on a simulated self-driving vehicle in Carla simulator [153]. The dynamics are modeled with a second-order bicycle model [30] and the collision region consists of 3 discs. A collision-free polytope is constructed for each of the discs. The pedestrians are programmed to follow the same dynamics as in Sec. 4.5.6, i.e., a GMM modeling crossing behavior. This chapter does not model the interaction between the vehicle and the pedestrians. The control frequency is 10 Hz. The computation time was measured to be 88 ms on average and 135 ms maximum. Fig. 4.12 visualizes snapshots of the simulations. In Case A (see Fig. 4.12b), the planner keeps enough distance to let the pedestrians cross while driving as close to the path as is safe. In Case B (see Fig. 4.12c), the vehicle passes behind the pedestrians while keeping its distance from the pedestrian that is not crossing.

4.6. Discussion

As expected from the theoretical analysis, the experiments showed that SH-MPC can consistently bound the CP of the overall trajectory. Where methods that impose marginal constraints need to decide between performance ($\epsilon_k = \epsilon$) and safety ($\epsilon_k = \frac{\epsilon}{NM}$),


(a) A snapshot of the proposed approach in the Carla simulator.


(b) Case A: Safe overtake

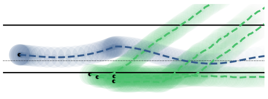

(c) Case B: Passing behind

Figure 4.12: (a) Snapshot from Carla simulations. Visualization follows Fig. 4.4c for the frontal vehicle disc. (b, c) Observed trajectories of the vehicle (dark blue) and pedestrians (green) with start positions as black dots in two cases.

SH-MPC makes this trade-off more explicit by ensuring that the CP remains consistent under different operating conditions, such as with regards to number of obstacles, probability distributions, and the horizon length.

The gap between the risk guarantee and the obtained risk can be reduced, for example, by assuming some knowledge of the distribution or by running multiple scenario programs in parallel. Alternatively, the risk could be analyzed in continuous time (see for example [154]) to reduce discretization errors.

The proposed method is widely applicable. It handles dynamic obstacles (e.g., cyclists, cars or non-cooperative robots) and controls systems with nonlinear dynamics. In addition, the joint distribution of the uncertainty can capture interactions of dynamic obstacles with other obstacles or the robot (e.g., to predict that pedestrians evade other pedestrians and the vehicle). It cannot yet account for interaction during planning, where the dynamics of the robot and pedestrians directly influence each other, as the probability measure $\mathbb{P}$ cannot depend on the optimization variables in scenario optimization.

The guarantees provided by SH-MPC rely on an accurate model of the uncertainty, which may be challenging to obtain, for example, in the case of human motion prediction. Nevertheless, the proposed method provides a planner that attains a desired level of risk with respect to the predicted probability distribution. The prediction model could also be replaced by recorded samples, in line with the typical scenario approach, to reduce modeling errors and provide formal guarantees with respect to the true motion of the obstacles.

In terms of computational efficiency, SH-MPC is online capable and scalable under typical operating conditions. For extremely low-risk specifications (e.g., $\epsilon \leq 5 \cdot 10^{-3}$), computational requirements may become excessive as a result of the increase in sample size. This can be addressed, for example, by either pruning the samples, given that only the extreme samples are of interest (see for example [88]), or by solving an approximate scenario optimization [155]. In addition, most computations of SH-MPC are parallel linear computations (for each sample), which potentially leave room for further optimization, e.g., by delegating computations to a Graphical Processing Unit (GPU).

4.7. CONCLUSIONS

This chapter presented a novel method for planning under uncertainty, Safe Horizon Model Predictive Control (SH-MPC), that bounds the collision probability of the planned trajectory over its duration and with respect to all obstacles. The method uses a scenario optimization formulation where samples of the involved uncertainty are used as constraints to limit the collision probability of the motion plan. The number of samples is the main indicator for the collision probability and, with SH-MPC, could be computed before deploying the controller.

Simulations, with a mobile robot and an autonomous vehicle, showed that SH-MPC better approximates the collision probability over the duration of the motion plan than existing methods that rely on the marginal probability of collision. The main baseline, which achieved tight evaluations of the risk for each time step, was shown to be conservative over the duration of its motion plan and there was significant variation in its overall risk when this chapter varied the number of obstacles or their distribution. The overall risk of SH-MPC remained less conservative and more consistent between different environments and distributions, which resulted in faster trajectories when the environment was crowded. In addition, this chapter showed excellent scaling of the computation time with respect to the number of obstacles and under varying distributions.

5
Partitioned Scenario Replay

The mystery of life isn't a problem to solve, but a reality to experience.

Frank Herbert - Dune

5.1. Overview

The scenario-based planners introduced in previous chapters ensure safety through samples that represent the uncertainty in the future motion of nearby humans. The uncertainty associated with human motion is typically predicted by learning a probability distribution of future motion conditioned on contextual information (e.g., the position of other humans and the robot) [63], [119], [120], [123], [124]. Scenarios are obtained by sampling from this distribution. While this learned distribution approximates the real distribution, it can deviate from the data points to improve the overall fit. This makes it hard to provide guarantees on sampled trajectories since not all data points are respected.

Samples would not deviate from the data, if they are made up directly from the data points themselves. That is, if human motion was directly predicted by previously observed motion of other humans. Unfortunately, not every observed trajectory can be replayed at any time since the context in which behavior is observed needs to be considered for accurate prediction.

5.1.1. Approach

This chapter proposes Partitioned Scenario Replay (PSR) for data-driven prediction and planning, illustrated in Fig. 5.1. Instead of learning a probability distribution from previously observed human trajectories and the associated contextual information, PSR first

Parts of this chapter have been published in:

O. de Groot, A. Sridharan, J. Alonso-Mora, and L. Ferranti, "Probabilistic motion planning and prediction via partitioned scenario replay", in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 7546–7552

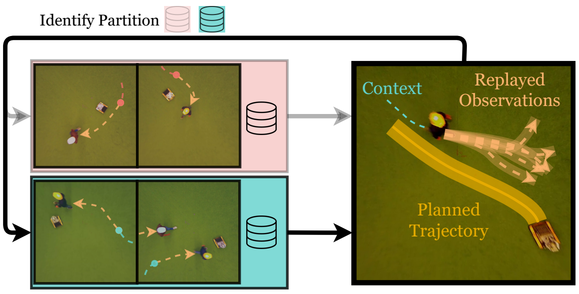


Figure 5.1: Illustration of PSR. Human trajectories are collected in a database and partitioned based on the associated context (highlighted in red and cyan). During planning, the current context identifies one of the partitions and trajectories in this partition (e.g. cyan) are replayed as motion predictions that the planned trajectory (in yellow) must avoid.

partitions human trajectories offline based on the context in which they were observed. In each online iteration of the planner, PSR uses the context to decide from which partition trajectories are reintroduced (or *replayed*) to predict the human's motion. PSR uses scenario-based trajectory optimization [157] to avoid all of the replayed trajectories, which inherently provides a probabilistic safety guarantee on the planned trajectory *in the real world*.
PSR can quickly generate an arbitrarily large number of samples, contrary to learning-based prediction algorithms (e.g., [63], [119]) and can improve its predictions *online*.

5.1.2. CONTRIBUTION

The contributions of this chapter are the following:

1. A joint data-driven method (PSR) for prediction and planning under arbitrary uncertainty that provides a real-world safety guarantee for collision avoidance (see Theorem 5), while optimizing planning task performance. The framework divides the training data into *partitions*. During planning, the current sensor data is mapped to one of the partitions and data in this partition is replayed to represent the uncertainty in the scenario-based planner [157]. The approach is fast to train and query and supports diverse inputs such as contextual information (e.g., the road layout, tracking information or social cues) or pre-processed contextual data (e.g., classifier or autoencoder outputs).

2. A scenario optimization for non-stationary (i.e., context-dependent) probability distributions.

The prediction component of PSR attains close to state-of-the-art performance on the ETH/UCY data-set [110], in terms of the Average Displacement Error (ADE) and the Final Displacement Error (FDE). In comparison, our method is much simpler, computationally more efficient and includes a planner with a real-world safety guarantee.
This chapter deploys PSR without prior data in the real world on a mobile robot, where it allows the robot to evade pedestrians, even when no initial model for pedestrian motion was provided.

5.2. PROBLEM FORMULATION

5.2.1. HUMAN MOTION PREDICTION

Accurate prediction of human motion relies on contextual information. A context state $\boldsymbol{x}_{obs}^{\text{full}} \in \mathbb{X}_{obs}^{\text{full}}$ is associated with each human, that is partially unobservable from the robot's perspective (e.g., human intentions). The observable subset of the context state is denoted by

$$\boldsymbol{x}_{obs} \in \mathbb{X}_{obs} \subseteq \mathbb{X}_{obs}^{\text{full}}, \tag{5.1}$$

which is available to the prediction and planning pipeline in each iteration. These observations can contain *deterministic* mappings of sensor data such as classification or autoencoder outputs. The uncertain position of a single human at future time k is denoted by $\boldsymbol{\delta}_k \in \mathbb{R}^2$ and its future N positions by $\boldsymbol{\delta} = \{\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_N\}$. This chapter assumes that there exists a probability distribution $\mathbb{P}$ that describes human motion and denotes with $\mathbb{P}_x = \mathbb{P}[\boldsymbol{\delta} \mid \boldsymbol{x}_{obs}]$ the probability distribution conditioned on the observed context. Finally, this chapter assumes that a dataset,

$$\mathcal{D} = \{(\boldsymbol{x}_{obs}^{(1)}, \boldsymbol{\delta}^{(1)}), \ldots, (\boldsymbol{x}_{obs}^{(D)}, \boldsymbol{\delta}^{(D)})\}, \tag{5.2}$$

containing observed contextual information and the associated realization of the uncertainty (i.e., trajectories) is available. This dataset can be recorded at runtime, accumulating over time, or can be a public dataset recorded in a similar setting (e.g., [110]).

5

5.2.2. SCENARIO-BASED ROBOT MOTION PLANNING

A robot with nonlinear discrete-time dynamics is considered

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{5.3}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ denote the states and inputs, respectively and n_x and n_u are the number of states and inputs, respectively. The vehicle state is assumed to contain its x-y position $\boldsymbol{p} = [x, y] \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$. For simplicity, consider collision avoidance with a single human. A collision avoidance constraint $g(\boldsymbol{x}_k, \boldsymbol{\delta}_k) \leq 0$ imposes that the vehicle does not collide with the human at time k. For example, $g(\boldsymbol{x}_k, \boldsymbol{\delta}_k) = r - ||\boldsymbol{p}_k - \boldsymbol{\delta}_k||_2$, with r the summed radius of robot and human. With the human's future motion uncertain, the following Chance Constrained Problem (CCP) is formulated:

$$\min_{\boldsymbol{u}\in\mathbb{U}, \boldsymbol{x}\in\mathbb{X}} \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{5.4a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{5.4b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \; k = 0, \ldots, N-1 \tag{5.4c}$$

$$\mathbb{P}\left[\bigwedge_{k=1}^{N} \left(g(\boldsymbol{x}_k, \boldsymbol{\delta}_k) \leq 0\right)\right] \geq 1 - \epsilon, \boldsymbol{\delta} \in \Delta, \tag{5.4d}$$

where formally the joint uncertainty $\boldsymbol{\delta}$ belongs to a probability space $\Delta = \mathbb{R}^{2N}$, associated with a σ-algebra $\mathcal{F}$ and probability measure[1] $\mathbb{P}$, and this chapter uses $\bigwedge$ to denote the "and" operation. Our goal is to solve the CCP in Eq. 5.4 to find robot control inputs $\boldsymbol{u}$ that avoid collisions with humans with a probability of at least ϵ in the real-world.

[1]For more details, see [158].

5.3. Preliminary - Scenario Program

The CCP in Eq. 5.4 can be solved via a sampling-based reformulation, known as a Scenario Program (SP) [157]. In the reformulation, one collects an independent set of samples $\{\boldsymbol{\delta}^{(1)},\ldots,\boldsymbol{\delta}^{(S)}\}$ from $\mathbb{P}$ referred to as *scenarios* and formulates a deterministic variant of the constraint (5.4d) for each sample. This gives the following SP

$$\min_{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}} \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{5.5a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{5.5b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k), \; k = 0,\ldots,N-1 \tag{5.5c}$$

$$\bigwedge_{k=1}^{N} \left(g(\boldsymbol{x}_k,\boldsymbol{\delta}_k^{(i)}) \le 0 \right), \; i = 1,\ldots,S. \tag{5.5d}$$

The SP is deterministic and can therefore be solved using a NonLinear Program (NLP) solver. Independent of the probability distribution $\mathbb{P}$, the required number of scenarios S can be computed from a desired risk ϵ (collision avoidance probability), confidence $1-\beta$ (probability that (5.4d) is satisfied by the SP) and support n (number of scenarios that affect the solution). A Jupyter notebook to perform this computation is provided with this chapter [150].

5.4. Partitioned Scenario Replay

The SP in Eq. 5.5 offers a data-driven way to compute a solution to the CCP in Eq. 5.4 in the real-world, using recorded observations of human motion. However, it does not consider the context in which these trajectories were recorded. This leads in practice to poor predictions since human motion is strongly context dependent. In the following, the SP is extended to a context dependent distribution that leads to a provably safe prediction and planning framework.

5.4.1. Partitioning the dataset

By leveraging on the properties of scenario-based motion planning [157], probabilistic collision avoidance can be ensured in the real-world in two steps: an offline *training* phase and an online *replay* phase. In the training phase, realizations of an uncertainty distribution $\mathbb{P}$ are collected in a dataset $\mathcal{D}$. In the online replay phase, S samples from the dataset are reintroduced in the planner as scenarios $\boldsymbol{\delta}^{(1)},\ldots,\boldsymbol{\delta}^{(S)}$ for the SP in Eq. 5.5. The distribution $\mathbb{P}$ and dataset $\mathcal{D}$ need to be carefully considered, however. Ignoring context, any sample in $\mathcal{D}$ is an independent sample of $\mathbb{P}$. Although this means that the SP in Eq. 5.5 can be solved by drawing scenarios from $\mathcal{D}$, its solution will be conservative as any sample may be replayed at any time.
To incorporate contextual information, the CCP in Eq. 5.4 should consider the conditioned distribution $\mathbb{P}_x$. The associated SP in Eq. 5.5 is then constructed by accumulating and replaying samples from $\mathbb{P}_x$. However, as the domain of the observed information $\mathbb{X}_{obs}$ is generally continuous, the probability of observing any particular case, $\boldsymbol{x}_{obs} \in \mathbb{X}_{obs}$ is zero. Samples from $\mathbb{P}_x$ therefore cannot be accumulated.

This chapter proposes instead to partition the space of observed information $\mathbb{X}_{obs}$ into a finite number of subsets or *partitions*. Each partition is constructed such that the probability of observing data belonging to the partition is non-zero. To formalize this idea, this chapter constructs $P > 0$ partitions $\mathbb{X}^p_{obs} \subseteq \mathbb{X}_{obs}$ as follows:

$$\bigcup_{p=0}^{P} \mathbb{X}^p_{obs} = \mathbb{X}_{obs} \qquad \text{(Partitioning)} \tag{5.6}$$

$$\bigcap_{p=0}^{P} \mathbb{X}^p_{obs} = \emptyset \qquad \text{(No Overlap)} \tag{5.7}$$

$$\mathbb{P}\left[\boldsymbol{x}_{obs} \in \mathbb{X}^p_{obs}\right] > 0. \qquad \text{(Density)} \tag{5.8}$$

Considering the partitioned contextual information, this chapter formulates the following chance constraint,

$$\mathbb{P}\left[\bigwedge_{k=1}^{N} \left(g(\boldsymbol{x}_k, \boldsymbol{\delta}_k) \le 0\right) \mid \boldsymbol{x}_{obs} \in \mathbb{X}^p_{obs}\right] \ge 1 - \epsilon, \tag{5.9}$$

for which the associated SP is given by (5.5), but where samples of $\boldsymbol{\delta}$ come from a subset of the dataset.

5.4.2. PREDICTION AND PLANNING ALGORITHM

This chapter proposes the following prediction and planning framework, outlined in Algorithm 4. Offline, each data point of the dataset $\mathcal{D}$ is assigned to the associated partition (see (1) in Fig. 5.2 and lines 4-7 in Algorithm 4), resulting in a partitioned dataset $\mathcal{D} = \bigcup_{p=0}^{P} \mathcal{D}^p$, where

$$\boldsymbol{x}^{(i)}_{obs} \in \mathbb{X}^p_{obs} \rightarrow (\boldsymbol{x}^{(i)}_{obs}, \boldsymbol{\delta}^{(i)}) \in \mathcal{D}^p. \tag{5.10}$$

Online, given a currently active partition $\mathbb{X}^p_{obs}$, PSR satisfies chance constraint (5.9) by solving the SP in Eq. 5.5 with samples from $\mathcal{D}^p$ (see (2) in Fig. 5.2 and lines 8 – 13 in Algorithm 4). The risk of the planning and prediction pipeline is then certified in the sense that (5.9) is satisfied in the real-world. The risk ϵ that can be guaranteed depends on the number of data points in the smallest partition $\text{argmin}_p |\mathcal{D}^p|$, since the sample size cannot be larger than any of the datasets. In practice, a desired risk ϵ is set and a sample size S is computed for a given confidence β and support n using Notebook [159]. Then PSR ensures that each partition is large enough, in the sense that

$$|\mathcal{D}^p| \ge S, \ \forall p. \qquad \text{(Data Requirement)} \tag{5.11}$$

This chapter obtains the following main result.

Theorem 5. *Consider the CCP in Eq. 5.4 and the associated SP in Eq. 5.5. Assume that dataset $\mathcal{D}$ containing Independent and Identically Distributed (IID) trajectories and observations is partitioned into P partitions according to partitioning rules* (5.6)-(5.8) *and that for the desired risk ϵ, data requirement* (5.11) *is satisfied. Then, the trajectory computed by the SP in Eq. 5.5 where scenarios are sampled from the current partition p (i.e., $\{\boldsymbol{\delta}^{(1)}, \ldots, \boldsymbol{\delta}^{(S)}\} \subseteq \mathcal{D}^p$) is collision free in the real-world, in the sense that* (5.9) *is satisfied.*

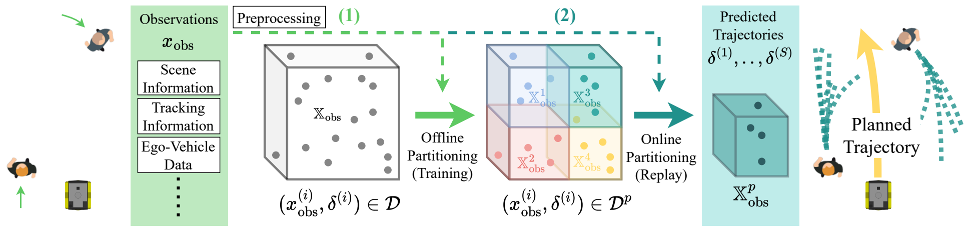


Figure 5.2: Schematic overview of PSR. Raw or processed observations are recorded into a dataset $\mathcal{D}$ containing contextual information and associated trajectories. (1) The dataset is partitioned into P partitions. (2) The current contextual information identifies a single partition. Trajectories in this partition are replayed to predict motion of nearby humans.

Algorithm 4: Partitioned Scenario Replay

```
   Input: Dataset $\mathcal{D}$, Observed features $\boldsymbol{x}_{obs}$, Trajectories $\boldsymbol{\delta}$,
          Sample size $S$, Partition threshold $S_{th}$
1  // Process new data
2  for each pedestrian do
3      Append $(\boldsymbol{x}_{obs}, \boldsymbol{\delta})$ to dataset $\mathcal{D}$
4  // Partition the dataset (training phase)
5  if $|\mathcal{D}| > S_{th}$ then
6      $\mathbb{X}^p_{obs}\ \forall p \leftarrow$ Size Constrained K-Means($\mathcal{D}$, $S$)
7      $\mathcal{D}^p \leftarrow$ Partition($\mathcal{D}$, $\mathbb{X}^p_{obs}$) $\forall p$ (Eq. (5.10))
8  // Retrieve scenarios (replay phase)
9  for each pedestrian do
10     Find $p$ for which $\boldsymbol{x}_{obs} \in \mathbb{X}^p_{obs}$    (Assign partition)
11     $(\boldsymbol{\delta}^{(0)}, \ldots, \boldsymbol{\delta}^{(S)}) \leftarrow \mathcal{D}^p$    (Retrieve scenarios)
12 // Trajectory optimization
13 $\boldsymbol{x}, \boldsymbol{u} \leftarrow$ Solve SP in Eq. 5.5 for retrieved scenarios;
   Output: $\boldsymbol{u}_0$
```

Proof. First note that (5.7) ensures that each observation $\boldsymbol{x}_{obs} \in \mathbb{X}_{obs}$ identifies a single partition, while together with (5.6) it is guaranteed that there is always a single partition for each observation. Hence, one may consider the motion planning problem of the CCP in Eq. 5.4 as P different motion planning problems (each with its own dataset $\mathcal{D}^p$, collected for the same problem) where one problem is active at each time instance. For each problem, the scenario approach certifies the risk ϵ based on S (and β, n) [98, Theorem 1]. Finally, (5.11) ensures that S samples can be sampled from each partition, proving the result. □

In practice, Theorem 5 provides a safety guarantee that helps to understand how safe the predictions are based on the size of the collected dataset. Several observations are in order. First, note that safety and performance are traded-off through the size of the partitions. A partition is safe when (5.11) holds. With more data available, the partition volumes shrink, leading to more accurate predictions and faster motion plans.

More insight can come from the two extreme applications of Theorem 5. If the dataset has S samples, it fits in a single partition. The planner evades all previously seen human trajectories and will be overly conservative in practice, but safe by Theorem 5. In the other extreme, many low variance partitions with at least S samples exist. It may happen that a new data point lands far away from each partition in which case it is likely that the low variance predictions do not capture the true future motion. Theorem 5 captures this in the risk ϵ. That is, given that S samples were observed in this partition without observing the new sample, the probability of seeing the current case is in the ϵ tail of the distribution.

5.4.3. PARTITIONING ALGORITHM

Deciding how the observation space $\mathbb{X}_{obs}$ is divided into P partitions, according to partition rules (5.6)-(5.8), can be seen as an unsupervised learning problem. In this work, prioritizing simplicity and computational efficiency, observations are normalized and K-means clustering [160] is applied to the resulting data points. K-means partitions the dataset in K clusters during training, where each data point belongs to the cluster with the nearest mean. To satisfy the data requirement in Eq. 5.11, this chapter applies a size constrained K-means clustering [161] strategy with a minimum cluster size of S (see lines 4-7 in Algorithm 4). Offline, this strategy computes a clustering of the observation space based on the dataset (i.e., constructing $\mathbb{X}^p_{\text{obs}}, \forall p$). In each online time step, the observations that form the current context $\boldsymbol{x}_{\text{obs}}$ are normalized. Denoting the mean of cluster p as $\bar{\mathbb{X}}^p_{\text{obs}}$, the active partition is selected as the cluster with the smallest distance from its mean to the current observation, that is, $\text{argmin}_p \, ||\boldsymbol{x}_{\text{obs}} - \bar{\mathbb{X}}^p_{\text{obs}}||_2^2$ (line 10 in Algorithm 4). Data in this partition are used as scenarios (line 11 in Algorithm 4).

Note that Theorem 5 does not assume the partitioning to be static. When the active partition has more than S samples, this chapter replays the S samples with the most similar current velocity magnitude.

5.4.4. OBSERVATIONS

The observations need to be deterministic, but each observation can be either continuous or discrete. Examples of observations are sensor data (e.g., relative position, velocity, orientation, etc.), categorical data (e.g., obstacle is pedestrian, cyclist) or Boolean data (e.g., has/has-not seen robot).

Any deterministic algorithm that pre-processes the data is also admissible. For example, a learning-based classifier to infer the crossing intentions of pedestrians or an autoencoder to encode complex scene information into a lower dimensional latent space and partition in this space.[2]

5.4.5. CONTINUAL APPLICATION

Theorem 5 guarantees probabilistic safety in the real-world for a given risk. While safety is guaranteed, more collected data leads to smaller partitions which, when partitioned effectively, leads to less conservative predictions of human motion. Because of this abil-

[2]This is considered future work.

ity to improve safely, PSR can be deployed without prior data in a real-world environment. In this setting, new observations are continuously recorded and the partition algorithm is repeated at regular intervals (e.g., when dataset is larger than a threshold S_{th}, see line 5 in Algorithm 4). This makes the proposed approach well suited for practical applications where no model is available.

5.5. Results

5.5.1. Comparison with Learning-Based Prediction

This chapter compares PSR prediction against learning-based methods on the ETH dataset for pedestrian motion prediction [110]. The dataset contains 35k+ pedestrian tracks each with 3.2s motion history and 4.8s ground-truth trajectories. Each trajectory contains 20 steps with a 0.4s time step. The data is split in 5 smaller datasets where one dataset is used as test set and the others are available for training. The validation set is incorporated in the training set, since the validation set is not required for PSR.

Evaluation Metrics: This chapter evaluates motion predictions with two metrics, similarly to prior works [63], [120], [121], [123]:

1. *Average Displacement Error (ADE)*: the average ℓ_2 distance between the ground truth and predicted trajectories.

2. *Final Displacement Error (FDE)*: the ℓ_2 distance between the ground truth and predicted trajectories at the prediction horizon T.

To make the motion uncertainty invariant to absolute position and orientation, PSR represents velocities v_k in the frame positioned at the pedestrian center and oriented forward. The features used for comparison are the velocity x and y component over the past 3 steps (more steps do not improve performance). Additionally, an ablation study is added that uses only the last velocity. Example predictions of PSR are depicted in Fig. 5.3 and quantitative results are listed in Table 5.1. PSR achieves close to state-of-the-art performance. The gap to the state-of-the-art is partially due to the inherent safety guarantee that accounts for less likely outcomes, degrading average performance. PSR could be further improved by using encoded scene information (e.g., an autoencoder) on top of the agent's velocity information.

A key feature of PSR is its computational efficiency given that PSR simply loads samples in its partition for each pedestrian. PSR predicts 101 trajectories for 2 pedestrians in Sec. 5.5.2 at 20Hz. This makes it significantly faster than other prediction methods which are usually limited to the order of 10 samples in real-time. For these methods, the planner cannot use the predicted distribution in detail even if the distribution is learned accurately.

5.5.2. Real-world evaluation

This chapter demonstrates PSR (see Algorithm 4) on a mobile robot (Clearpath Jackal) navigating among pedestrians using the continual PSR approach described in Sec. 5.4.5. The experimental setup is depicted in Fig. 5.4a. An open environment is mimicked by asking the pedestrians to move from one side to another, standing still for a brief time afterwards. Data is not saved when a pedestrian is standing still. The robot's task is to

Table 5.1: Comparison in ADE and FDE of state-of-the-art prediction models and PSR on the ETH/UCY data set [110].

Method	Metrics	ETH	Hotel	UNIV	ZARA1	ZARA2	AVG
[123] Social GAN	ADE	0.81	0.72	0.60	0.34	0.42	0.58
	FDE	1.52	1.61	1.26	0.69	0.84	1.18
[113] Social-STGCNN	ADE	0.64	0.49	0.44	0.34	0.30	0.44
	FDE	1.11	0.85	0.79	0.53	0.48	0.75
[63] Trajectron++	ADE	0.39	0.12	**0.20**	**0.15**	**0.11**	0.19
	FDE	0.83	0.21	0.44	0.33	0.25	0.41
[120] Y-Net	ADE	0.28	0.10	0.24	0.17	0.13	0.18
	FDE	0.33	0.14	0.41	**0.27**	0.22	0.27
[121] NSP-SFM	ADE	**0.25**	**0.09**	0.21	0.16	0.12	**0.17**
	FDE	**0.24**	**0.12**	**0.38**	**0.27**	**0.20**	**0.24**
PSR	ADE	0.60	0.22	0.41	0.24	0.18	0.33
	FDE	0.94	0.40	0.79	0.41	0.33	0.57
PSR (No History)	ADE	0.68	0.25	0.37	0.24	0.18	0.34
	FDE	1.12	0.44	0.74	0.43	0.34	0.61

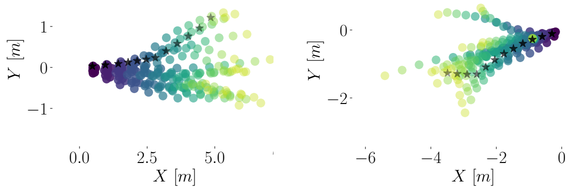
Table 5.2: Experimental settings with T_s the control time step and T_{int} the timestep of predictions.

ϵ	β	$\bar{n}$	S	T_s	T_{int}	N
0.25	0.01	5	101	0.05s	0.1s	30

drive from corner to corner while avoiding collisions with the pedestrians. The robot and pedestrian positions are detected with a marker-based tracking system. The experiment ran for 45 minutes.

Pedestrian trajectories are continually collected and the dataset is partitioned whenever its size increased by 10%. Pedestrian trajectories are saved every 10 steps to ensure that samples are independent. As observations, the x and y components of the previous three velocities and the current velocity magnitude are used for simplicity. The SP in Eq. 5.5 is solved online for linearized constraints (see [157]) with Forces Pro [143]. Experimental settings are listed in Table 5.2.

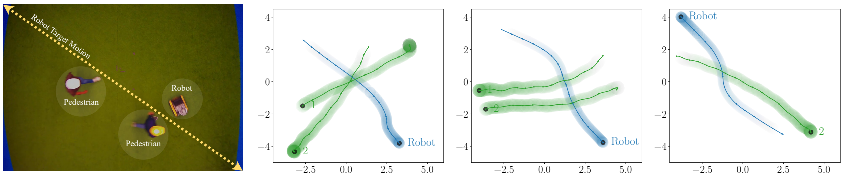
Fig. 5.4 depicts three experiments. Figs. 5.4b and 5.4c show that the robot avoids collisions, while PSR did not encode any model for the pedestrians. In Fig. 5.4d the pedestrian turns towards the robot, but the planner still evades the pedestrian smoothly. This indicates that the predicted distribution captures the deviation in behavior. Partitions



(a) An example of the variance captured in regular walking.

(b) An example of multi-modal trajectory prediction.

Figure 5.3: Examples of PSR predictions on the ETH dataset, showing 20 samples. Predicted trajectories are drawn from the first step ahead (purple) to the final time step (yellow). The ground-truth is depicted by black stars. Positions are drawn with increasing transparency along the time horizon.



(a) Experimental setup. (b) Evasion in between. (c) Evasion in front. (d) Pedestrians turns.

Figure 5.4: Experimental setup and three observed trajectories of the robot (blue) and pedestrians (green) towards the end of the experiment. Trajectories are depicted with increased transparency over time and start positions are indicated by a black dot.

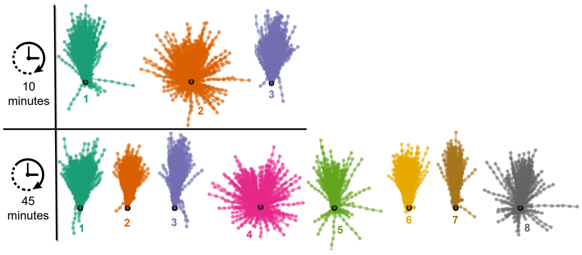


Figure 5.5: Trajectories in the partitions at two time instances.

at two time instances are depicted in Fig. 5.5. After 45 minutes the partitions capture *regular walking at different speeds* (nrs. 1, 2, 6), *fast walking at different speeds* (nrs. 3, 7) and *moving from stand still* (nrs. 4 and 8). This shows that partitions reduce in variance over time.

On the planner side, planning performance can be further improved by guiding the robot into the most suitable local optimum (e.g., using global dynamic guidance [152]), but this is outside the scope of this chapter.

5.6. CONCLUSION

This chapter presented a data-driven framework for human motion prediction and planning that collected and categorized observed trajectories into several partitioned datasets. During planning, trajectories from one partition were replayed for each human to predict their future motion. This allowed us to provide probabilistic safety guarantees on collision avoidance in the real-world. This chapter showed that PSR attained close to state-of-the-art prediction performance in ADE and FDE, while providing a safety guarantee. PSR was then deployed on a mobile robot and navigated successfully around pedestrians in the real-world even when starting the framework without prior data.

Our future work will combine learning-based context processing with PSR to generalize its applicability.

6

Topology-Driven Trajectory Optimization

"The vision of time is broad, but when you pass through it, time becomes a narrow door."

Frank Herbert - Dune

6.1. Overview

This chapter constitutes the second part of this dissertation. The focus in this chapter is not on the uncertainty associated with dynamic obstacles, but rather on the optimization-based planner itself. Traditionally, these methods compute a single locally optimal trajectory and their planning performance is tied to the quality of that local optimum. This chapter explores whether this weakness can be exploited to compute multiple locally optimal trajectories in parallel, to improve overall planner performance.

In dynamic environments, any planner must make both high-level and low-level decisions. The high-level decision determines how each obstacle should be avoided (e.g., left or right). The low-level decision determines the exact shape of a trajectory that is both collision-free and dynamically feasible. While these decisions operate on separate levels

Parts of this chapter have been published in:

- **Conference:** O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso–Mora, "Globally Guided Trajectory Planning in Dynamic Environments", in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 10 118–10 124
- **Journal:** O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Topology-Driven Parallel Trajectory Optimization in Dynamic Environments", Sep. 2024, *accepted for publication in IEEE Transactions on Robotics (T-RO)*
- **Video:** O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Video Topology-Driven Parallel Trajectory Optimization in Dynamic Environments*, Sep. 2024, Available: `https://youtu.be/E6UI8EAab50`

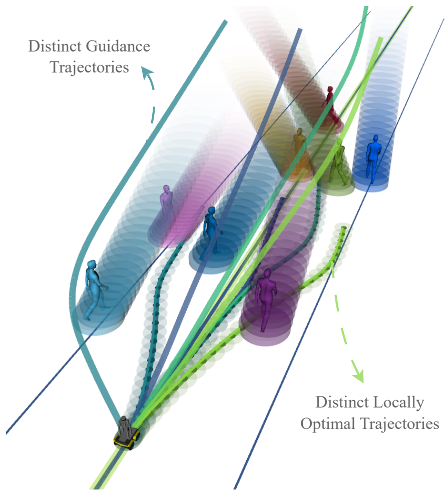

Distinct Guidance Trajectories
Distinct Locally Optimal Trajectories

Figure 6.1: T-MPC first computes distinct guidance trajectories in the state space (time is visualized in the upwards direction). Each guidance trajectory initializes a local planner, resulting in several distinct locally optimized trajectories. The locally optimized trajectories each pass the obstacles (predicted future motion visualized as cylinders) in a distinct way.

of the planning problem, they are often not differentiated, which can degrade planner performance in terms of time efficiency and safety.

Trajectory optimization makes the high-level decision implicitly through the initial guess and the tuning of the cost function [27], [84]. Other planners do not distinguish between the high-level and low-level decisions [17], [19], only consider static obstacles in the high-level decision [45] or require a structured environment to make the high-level decision [43], [128], [132].

6.1.1. Approach

This chapter presents a planning framework, Topology-driven Model Predictive Control (T-MPC) (see Fig. 6.1), that accounts for the two levels of the planning problem explicitly. The idea is to divide the planner into two components. The first component is a global planner that identifies several distinct, high-level navigation options by considering the topology of the dynamic collision-free space. The underlying topology allows T-MPC to determine whether trajectories pass obstacles differently. T-MPC then uses each high-level trajectory as initialization for an optimization-based planner. The low-level planning problems, that form the second component, are independent and are solved in parallel. The proposed framework does not modify the cost function of the optimization-based planner and selects the executed trajectory by comparing their optimal costs.

6.1.2. Contribution

T-MPC is different from existing works in four ways. First, it considers homotopy classes in the *dynamic* collision-free space, that includes time, to incorporate the motion of dy-

namic obstacles (contrary to [44], [45]). Second, it does not modify the cost function (i.e., the performance criteria) of the local planner (contrary to [35], [37]). Third, T-MPC does not rely on a structured environment (contrary to [39], [42], [43]). Finally, the proposed guidance planner can handle the case where its goal is blocked (contrary to [44], [45]) by considering multiple goal positions.
In addition, T-MPC *enforces* the final trajectories to be in distinct homotopy classes using constraints in the local planner and shows that it is not sufficient to initialize the solver in a homotopy class (contrary to [45]). By consistently planning distinct trajectories, the previously followed trajectory can be reidentified and given preference to make the planner more consistent and decisive.
The contributions of T-MPC are:

1. A planning framework for dynamic environments that optimizes trajectories in multiple distinct homotopy classes in parallel. The framework extends existing optimization-based local planners, improving their time efficiency, safety and consistency.

2. A fast guidance planner that computes homotopy distinct trajectories through the dynamic collision-free space towards multiple goal positions.

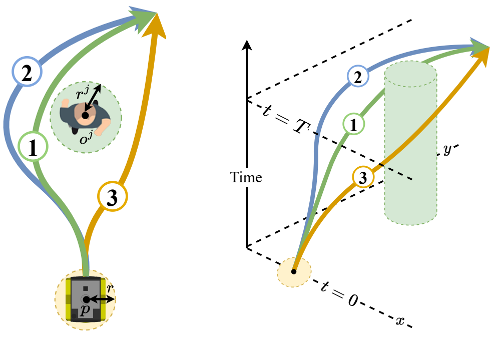
This chapter validates the proposed framework in simulation, on a mobile robot navigating among interactive pedestrians. The proposed framework is shown to accommodate different trajectory optimization approaches (e.g., [27] in the nominal case, and [84] to accommodate Gaussian uncertainties added to the motion of the dynamic obstacles). The proposed framework enhances the performance of [27], [84] out of the box and outperforms three relevant baselines ([17], [45], [152]). This chapter finally demonstrates T-MPC in the real world on a mobile robot navigating among five pedestrians.

6.2. PROBLEM FORMULATION

Discrete-time nonlinear robot dynamics are considered in this chapter

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \tag{6.1}$$

where $\boldsymbol{x}_k \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ are the state and input at discrete time instance k, n_x and n_u are the state and input dimensions respectively and the state contains the 2-D position of the robot $\boldsymbol{p}_k = (x_k, y_k) \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$.
The robot must avoid moving obstacles in the environment. The position of obstacle j at time $k=0$ is denoted $\boldsymbol{o}_0^j \in \mathbb{R}^2$ and it is assumed that for each obstacle, predictions of its positions over the next N time steps are provided to the planner (i.e., $\boldsymbol{o}_1^j, \dots, \boldsymbol{o}_N^j$) at each time instance. The collision region of the robot is modeled by a disc of radius r and that of each obstacle j by a disc with radius r^j (see Fig. 6.2a).
For high-level planning with dynamic collision avoidance, the simplified state space $\mathcal{X} := \mathbb{R}^2 \times [0,T]$ is considered, with $[0,T]$ being a continuous finite time domain (see Fig. 6.2b). The area of the workspace occupied by the union of obstacles at time t is denoted by $\mathcal{O}_t \subset \mathbb{R}^2$ and the obstacle set in the state space is thus $\mathcal{O} := \bigcup_{\forall t \in [0,T]} (\mathcal{O}_t, t) \subset \mathcal{X}$. The collision free state space (or *free space*) is denoted $\mathcal{C} := \mathcal{X} \setminus \mathcal{O}$. A trajectory is a continuous path through the state space, $\boldsymbol{\tau} : [0,1] \to \mathcal{X}$. The goal of the robot is to traverse along



(a) Planning scene. (b) State space view.

Figure 6.2: (a) Depiction of the planning problem and (b) equivalent in the state-space. Trajectory 1 and 2 are in the same homotopy class while trajectory 1 and 3 are in distinct homotopy classes.

a given reference path $\boldsymbol{\gamma} : [0,1] \to \mathbb{R}^2$ without colliding with the obstacles while tracking a constant reference velocity v_{ref}. It is allowed to deviate from the path.

6.2.1. Optimization Problem

The planning problem is formalized as the following trajectory optimization problem over a horizon of N steps

$$\min_{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{6.2a}$$

$$\text{s.t.} \quad \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \forall k \tag{6.2b}$$

$$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{6.2c}$$

$$g(\boldsymbol{x}_k, \boldsymbol{o}_k^j) \leq 0,\ \forall k, j, \tag{6.2d}$$

where the cost function J in (6.2a) expresses the planning objectives (e.g., following reference path $\boldsymbol{\gamma}$). Robot dynamics and initial conditions are imposed by (6.2b) and (6.2c), respectively and collision avoidance constraints are imposed by (6.2d).

Because dynamic obstacles puncture holes in the free space, the free space associated with the constraints (6.2d) is nonconvex. Nonlinear optimization algorithms, solving this problem, return just one of possibly many local optimal trajectories. The initial guess provided to them determines which local optimal trajectory is returned. It is generally unclear how close this trajectory is to the globally optimal trajectory (i.e., the best trajectory under the specified cost). In this work, the aim is to leverage this weakness to explore in parallel multiple locally optimal trajectories (provided as initial guesses on $\boldsymbol{x}$) that evade obstacles in a distinct way.

6.2.2. Homotopic Trajectories

To achieve the goal above, the concept of homotopic trajectories is relied upon, which can be formalized as follows:

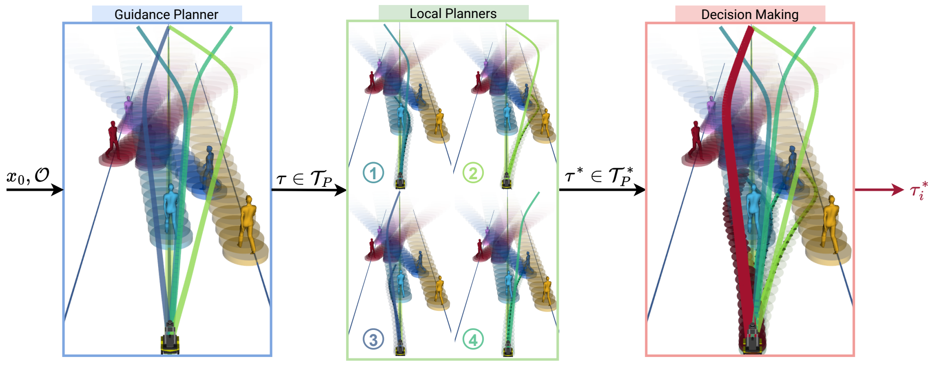


Figure 6.3: Schematic of T-MPC. An environment with several obstacles and a robot is visualized in x, y, t (time in the upwards axis). Obstacle motion predictions are denoted with cylinders. (1) A guidance planner (Sec. 6.3.1) finds $P = 4$ trajectories (visualized with colored lines) from the robot initial position to one of the goals. Each of these trajectories is in a distinct homotopy class in the state space. (2) Each trajectory guides a local planner (Sec. 6.3.3) as initial guess and through a set of homotopy constraints. Four guidance trajectories and optimized trajectories (as occupied regions for each step) are visualized. (3) The optimized trajectories are compared through their objective value (Sec. 6.3.5) and a single trajectory (in red) is excuted by the robot.

Definition 2. *[7] (Homotopic Trajectories) Two paths connecting the same start and end points $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$ respectively, are homotopic if they can be continuously deformed into each other without intersecting any obstacle. Formally, if $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in T$ represent two trajectories, with $\boldsymbol{\tau}_1(0) = \boldsymbol{\tau}_2(0) = \boldsymbol{x}_s$ and $\boldsymbol{\tau}_1(1) = \boldsymbol{\tau}_2(1) = \boldsymbol{x}_g$, then $\boldsymbol{\tau}_1$ is homotopic to $\boldsymbol{\tau}_2$ iff there exists a continuous map $\eta : [0,1] \times [0,1] \to \mathcal{C}$ such that $\eta(\alpha, 0) = \tau_1(\alpha) \forall \alpha \in [0,1]$, $\eta(\beta, 1) = \tau_2(\beta), \forall \beta \in [0,1]$ and $\eta(0,\gamma) = \boldsymbol{x}_s$, $\eta(1,\gamma) = \boldsymbol{x}_g \forall \gamma \in [0,1]$.*

If two trajectories are homotopic, they are said to be in the same homotopy class. An example is depicted in Fig. 6.2. To distinguish between trajectories in different homotopy classes, the homotopy comparison function

$$\mathcal{H}(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j, \mathcal{O}) = \begin{cases} 1, & \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \text{ in the same homotopy class} \\ 0, & \text{otherwise} \end{cases} \tag{6.3}$$

is used. Comparing homotopy classes can be computationally inefficient. The H-signature [7], winding numbers [51] and UVD [44] are supported to approximately compare homotopy classes in real-time. Details of the three methods are provided in Appendix A.

6.3. TOPOLOGY-DRIVEN MODEL PREDICTIVE CONTROL

In this section, T-MPC is proposed, a topology-guided planner that optimizes trajectories in multiple distinct homotopy classes in parallel.

The proposed planner consists of two components: a high-level guidance planner and multiple identical low-level local planners (see Fig. 6.3). The *guidance planner* G generates homotopy distinct trajectories through the free space

$$G(\boldsymbol{x}_0, \mathcal{P}_g, \mathcal{C}) = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_P\} =: \mathcal{T}_P, \tag{6.4}$$

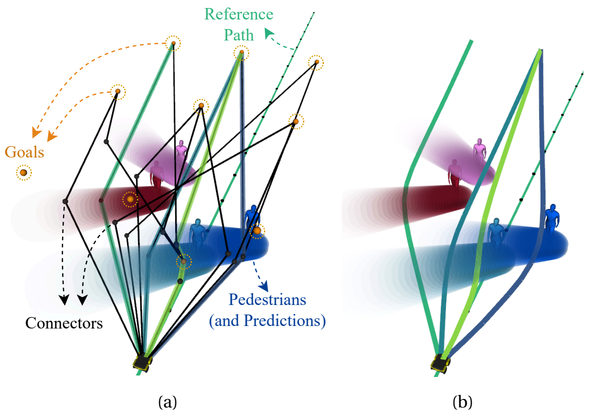


Figure 6.4: Illustration of the guidance planner in the state-space (time in the upwards axis). Visualization follows Fig. 6.3. (a) The visibility-PRM graph (black lines and dots) explores the free space toward the goals placed at $t = T$ around the reference path (orange dots). The homotopy distinct guidance paths (colored lines) are obtained by searching the graph. (b) The final trajectories are smoothened.

where $\boldsymbol{x}_0$ denotes the robot initial state and $\mathcal{P}_g$ denotes a set of goal positions. Each *local planner* is initialized with one of the guidance trajectories and optimizes the trajectory in the same homotopy class. With N the horizon of the global and local planners, each local planner defines a mapping $L : \mathcal{X}^N \to \mathcal{X}^N$,

$$L(\boldsymbol{\tau}_i) = \boldsymbol{\tau}_i^*. \tag{6.5}$$

To ensure that the local planner optimizes in the provided homotopy class, this chapter appends a set of constraints derived from the guidance trajectory. These constraints are appended to existing collision avoidance constraints to adapt the planner to the globalized framework. The proposed planner computes locally optimal trajectories $\mathcal{T}_P^* := \{\boldsymbol{\tau}_1^*, \ldots, \boldsymbol{\tau}_P^*\}$ in several distinct homotopy classes.

6.3.1. GUIDANCE PLANNER - OVERVIEW

The goal of the guidance planner is to quickly compute several homotopy distinct trajectories through the free space. Similarly to [44], [45], this chapter performs this search via Visibility-Probabilistic RoadMaps (Visibility-PRM [50]), a sampling-based global planner. The modifications that this chapter makes ensure that the graph remains consistent over successive iterations.

The guidance planner is outlined in Algorithm 5 and visualized in Fig. 6.4. Details of the algorithm are given in Sec. 6.3.2. A high-level overview is given here. First, **Visibility-PRM** constructs a sparse graph through the state space from the robot position to a set of goals, where each connection is homotopy distinct (line 1). The goals represent end points for the guidance planner and are placed along the reference path. For each goal, **DepthFirstSearch** (line 2) searches in this graph for the shortest P trajectories that reach it. Any homotopy equivalent trajectories are filtered out by **FilterAndSelect** (line 3), ensuring that the remaining trajectories are in distinct homotopy classes. The P trajectories

Algorithm 5: Guidance Planner

```
    Input: $\mathcal{C}$, $\boldsymbol{x}_0$, $\boldsymbol{x}_N$, previous graph $\mathcal{G}^-$, previous trajectories $\mathcal{T}_P^-$
  1 $\mathcal{G} \leftarrow$ Visibility-PRM$(\mathcal{C}, \boldsymbol{x}_0, \boldsymbol{x}_N, \mathcal{G}^-)$
  2 $\{\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_{N_{GS}}\} \leftarrow$ DepthFirstSearch$(\mathcal{G})$
  3 $\mathcal{T}_P = \{\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_P\} \leftarrow$ FilterAndSelect$(\{\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_{N_{GS}}\})$
  4 $\mathcal{G}^- \leftarrow$ IdentifyAndPropagate$(\{\boldsymbol{\tau}_0, \ldots, \boldsymbol{\tau}_P\}, \mathcal{T}_P^-)$
    Output: $\mathcal{T}_P$
```

that seem most promising are selected by a heuristic that prefers its goal to be as close as possible to the reference path at the reference velocity (see Fig. 6.4a). **IdentifyAndPropagate** (line 4) verifies if any of the selected trajectories are equivalent to trajectories of the previous planning iteration. This reidentification makes it possible to follow the same passing behavior over multiple planning iterations. Finally, nodes in the Visibility-PRM graph are propagated by lowering their time state by the planning time step.

Through this process, the proposed method obtains in each iteration P piecewise linear trajectories $\mathcal{T}_P = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_P\}$ that each connects the robot position to one of the goals (see Fig. 6.3). These trajectories are finally smoothed and cubic splines are fitted to make them differentiable (see Fig. 6.4b). More details can be found in [152]. The smoothening procedure produces only a small displacement in trajectories to maintain their homotopy class.

These trajectories serve as initializations for the local planners, described in Sec. 6.3.3.

6

6.3.2. Guidance Planner - Detailed Description

Each step of Algorithm 5 is detailed in the following.

Visibility-PRM computes sparse paths through the free space by randomly sampling positions and creating either a *Guard* or *Connector* node at the sampled position. The type of node depends on the number of Guards that it can directly connect to without colliding (i.e., which Guards are *visible*). A *Guard* is added if no other Guards are visible. A *Connector* (see black dots in Fig. 6.4) is added when exactly two Guards are visible and its connection to the Guards is feasible (e.g., satisfying velocity and acceleration limits). Similar to [44], it is also checked if any Connectors link to the same Guards (referred to as *neighbors*). If there are neighbors, the new connection is kept if it is distinct from existing connections, which is verified with homotopy comparison function (6.3). If it is equivalent and more efficient than the existing connection (e.g., if its connection is shorter), then the existing connector is replaced with the new connector. In regular visibility-PRM [50], the graph is initialized with a Guard at the start and goal positions and new nodes are drawn up to a time or node limit. More details of the algorithm can be found in [152, Algorithm 1].

Multiple Goals in Visibility-PRM This chapter addresses the limitation that a single goal must be reached by Visibility-PRM, which causes the planner to fail when that goal cannot be reached. This chapter proposes to add a Goal node type to Visibility-PRM. Goals inherit the properties of Guards but are inserted initially and are likely visible to each other. When a Connector can connect to multiple Goals, the Goal with the lowest distance to the point on the reference path reached with the reference velocity (i.e., the ideal goal) is singled out. By supporting multiple goals, the robustness of the guidance planner is improved. In practice, a grid of goals centered around the reference path is deployed (see Fig. 6.4).

Homotopy Comparison This chapter uses the homotopy comparison function (6.3) to verify if two trajectories are in the same homotopy class. This function was implemented (6.3) with the H-signature [7], winding numbers [51] and UVD [44]. Appendix A provides details on these methods. For the experiments, this chapter uses the H-signature that joins the two trajectories to be compared into a loop and verifies if that loop encircles any moving obstacles. If it does, then the two trajectories pass obstacles differently and belong to different homotopy classes.
DepthFirstSearch searches for P paths to each goal, with each search implemented similar to [45, Algorithm 1].
FilterAndSelect uses homotopy comparison function (6.3) to remove equivalent trajectories to different Goals found by DepthFirstSearch. The set of filtered trajectories $\mathcal{T}_F$ therefore satisfy

$$\mathcal{H}(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j, \mathcal{O}) = 0,\ \forall i, j, i \neq j,\ \boldsymbol{\tau}_i, \boldsymbol{\tau}_j \in \mathcal{T}_F. \tag{6.6}$$

The P lowest cost trajectories in $\mathcal{T}_F$ constitute the output $\mathcal{T}_P$.
IdentifyAndPropagate uses homotopy comparison function (6.3) to link new trajectories to trajectories found in the previous iteration. It checks for each previous trajectory $\boldsymbol{\tau}_i^- \in \mathcal{T}_P^-$ if

$$\exists \boldsymbol{\tau}_j \in \mathcal{T}_P, \mathcal{H}(\boldsymbol{\tau}_i^-, \boldsymbol{\tau}_j) = 1. \tag{6.7}$$

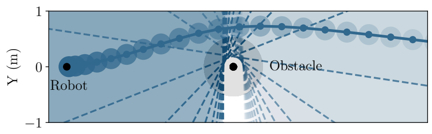
A unique identifier, tied to the homotopy class, is passed from $\boldsymbol{\tau}_i^-$ to $\boldsymbol{\tau}_j$ if the latter exists. This identifier can be used to decide which trajectory to follow (see Sec. 6.3.5).

6.3.3. LOCAL PLANNER

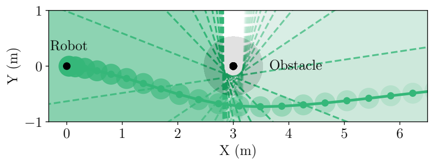
To refine the trajectories of the guidance planner, this chapter applies P local planners in parallel. Each local planner refines one of the guidance trajectories $\boldsymbol{\tau}_i$ and needs to ensure that the final trajectory is dynamically feasible and that it satisfies any other imposed constraints. The following definition is posed:

Definition 3. *(Local Planner) The local planner is an algorithm $L : \mathcal{X}^N \to \mathcal{X}^N$ that respects constraints.*

This definition captures many existing optimization-based planners. This chapter defines the local planner through the trajectory optimization in Eq. (6.2), where two modifications are made to ensure that the optimized trajectories are in the homotopy class of the associated guidance trajectory. First, the trajectory optimization of each local planner uses its guidance trajectory as the initial guess for $\boldsymbol{x}$. The initial guess speeds up convergence but does not guarantee that the optimized trajectory remains in the same homotopy class when there are obstacles. In the next section (Sec. 6.3.4), an example where initialization in distinct homotopy classes still leads to identical optimized trajectories is provided.

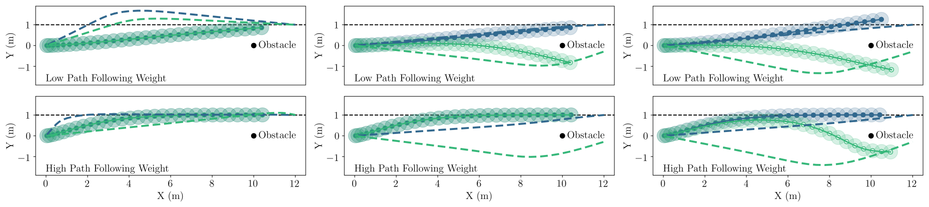


(a) Local planner 1 plans to evade the obstacle left.



(b) Local planner 2 plans to evade the obstacle right.

Figure 6.5: Two distinct locally planned trajectories for a robot (black dot) evading a *static* obstacle (black region and dot). For both planners, the topology constraints for each time step are depicted in their respective colors showing the constraint boundaries (broken lines) and their feasible region (colored regions with increasing transparency over time).



(a) Without homotopy comparison (6.3).

(b) Without constraints (6.9e).

(c) The proposed method.

Figure 6.6: Planned trajectories (lines with shaded discs) tracking a reference path (dashed black line) while avoiding a static obstacle with two guidance trajectories (dashed lines) for a low (0.01) and high (0.3) path following weight. (a) Without homotopy comparison (6.3), guidance trajectories are not distinct and optimized trajectories are identical. (b) Without homotopy constraints, increasing the path following weight results in identical trajectories. (c) With homotopy comparison (6.3) and homotopy constraints (6.9e), optimized trajectories are distinct.

To ensure that the homotopy class of the guidance trajectory is respected, this chapter adds to each local planner a set of constraints $g_H(\boldsymbol{x}_k, \boldsymbol{o}_k^j, \boldsymbol{\tau}_{i,k})$. For this purpose, the proposed method constructs for each time instance k and obstacle j a linear constraint between the guidance trajectory and obstacle position (see Fig. 6.5). With guidance trajectory $\boldsymbol{\tau}_i$ and obstacle trajectory $\boldsymbol{o}$, these constraints are given by $\boldsymbol{A}_k\boldsymbol{x}_k \leq b_k$, where

$$\boldsymbol{A}_k = \frac{\boldsymbol{o}_k - \boldsymbol{\tau}_{i,k}}{||\boldsymbol{o}_k - \boldsymbol{\tau}_{i,k}||}, \quad b_k = \boldsymbol{A}_k^T(\boldsymbol{o}_k - A_k(\beta(r + r_{\text{obs}}))). \tag{6.8}$$

The relaxation factor $0 \leq \beta \leq 1$ scales the distance that the constraints enforce from each obstacle. A key observation is that, with other collision avoidance constraints in place, the topology constraints can be relaxed ($\beta \approx 0$) such that they are inactive at the obstacle boundary. Since the constraints do ensure that the trajectory remains on the same side of each obstacle, the optimized trajectory is in the same homotopy class as the initialization provided by the guidance planner.

The resulting homotopy preserving local planner is given by

$$\begin{aligned}
J_i^* = \min_{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}} \quad & \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) && (6.9\text{a})\\
\text{s.t.} \quad & \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k),\ \forall k, && (6.9\text{b})\\
& \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} && (6.9\text{c})\\
& g(\boldsymbol{x}_k, \boldsymbol{o}_k^j) \leq 0\ \forall k, j && (6.9\text{d})\\
& g_H(\boldsymbol{x}_k, \boldsymbol{o}_k^j, \boldsymbol{\tau}_{i,k}) \leq 0\ \forall k, j. && (6.9\text{e})
\end{aligned}$$

The topology constraints and initialization of the optimization realize the local planning mapping of Eq. (6.5) which, as a function of the guidance trajectory, returns a distinct local optimal trajectory (see Fig. 6.3).

6.3.4. ENFORCING CONSISTENCY OVER TIME

The proposed method computes distinct trajectories through two algorithmic features: The guidance trajectories are distinct with respect to the homotopy comparison function (6.3) and the homotopy constraints (6.9e) ensure that each trajectory before and after the local optimization is in the same homotopy class (i.e., the local planner does not change the homotopy class of the trajectory that it is optimizing). The necessity of these two components is illustrated with an example.

Consider a planning scenario with a robot and static obstacle (both at $y = 0$) and the reference path at $y = 1$. The robot's trajectory is planned for a low and high weight on following the path[1]. Fig. 6.6 shows the planned trajectories after optimization. Without homotopy comparison (6.3) (see Fig. 6.6a), guidance trajectories are not distinct and lead to identical optimized trajectories. Without homotopy constraints (6.9e) (see Fig. 6.6b), trajectories are distinct for a low path following weight but become identical by increasing the path following weight. This is possible as the final state of the optimized trajectory is free to move to the other side of the obstacle. The proposed method (see Fig. 6.6c) maintains the two trajectories in both cases, irrespective of the tuning of the objective function.

6.3.5. DECISION MAKING

The robot can only execute one trajectory. Since the cost function of the local planners (6.9a) matches that of the original trajectory optimization (6.2a), the quality of the guided plans are directly comparable[2] through their optimal costs J_i^*.

The local planners output P optimized trajectories

$$\mathcal{T}_P^* = \{\boldsymbol{\tau}_1^*, \ldots, \boldsymbol{\tau}_P^*\}. \tag{6.10}$$

[1]The path following weight is the contouring weight from [27].

[2]As trajectory end points can be distinct, their quality needs to be represented in the cost function. We include a terminal cost that accounts for the deviation of the end point from the reference path.

Since each local planner minimizes the same cost function, the lowest cost trajectory[3]

$$\boldsymbol{\tau}_i^*, \quad i = \operatorname{argmin}_i J_i^*, \tag{6.11}$$

is the best trajectory under the specified objective. This chapter refers to executing $\boldsymbol{\tau}_i^*$ as obtained from (6.11) as the *minimal cost* decision.

In practice, frequently switching the homotopy class of the executed trajectory can degrade motion planning performance and lead to collisions even if, in each time instance, the selected trajectory attains the lowest cost. This chapter therefore considers a generalization of the decision-making process where the previously selected trajectory is given precedence. This is possible as a consistent set of trajectories in distinct homotopy classes is maintained where the previously executed trajectory is marked. This *consistent* decision is given by

$$\boldsymbol{\tau}_i^*, \quad i = \operatorname{argmin}_i w_i J_i^*, \tag{6.12}$$

where $w_i = c_i$ if this trajectory was previously selected, with c_i a constant $0 \leq c_i \leq 1$, and $w_i = 1$ otherwise. If $c_i = 0$, then the planner will pick the trajectory with the same homotopy class of the previous iteration, while for $c_i = 1$, the minimal cost decision is recovered. In practice, this decision-making scheme improves navigation behavior over consecutive iterations.

6.3.6. THEORETICAL ANALYSIS

The following formalizes to what extent the proposed planner resolves the nonconvexity of the free space. First, note that due to the cost function and/or nonlinear robot dynamics, the trajectory optimization in Eq. (6.2) remains nonconvex, even when it is constrained to stay in a single homotopy class. There may therefore be multiple local optima in each homotopy class. This means that the proposed planner does not provably return a globally optimal solution to the optimization in Eq. (6.2). This chapter propose instead a weaker notion of globalization[4].

Definition 4. *(Homotopy Globally Optimal) Denote the highest-cost local-optimum of optimization* (6.2) *in homotopy class i as $\boldsymbol{\tau}_i^-$. A trajectory $\boldsymbol{\tau}$ is said to be a Homotopy Globally Optimal (HGO) solution if its cost is lower or equal to that solution in each homotopy class, that is, if $J(\boldsymbol{\tau}) \leq J(\boldsymbol{\tau}_i^-)$, $\forall i$, for all homotopy classes that admit a feasible trajectory.*

To prove when the proposed scheme computes an HGO solution, this chapter pose three conditions. These conditions link the solution of Eq. (6.9) to that of Eq. (6.2).

Condition 1. The homotopy constraints are not active ($g_H(\boldsymbol{x}_k, \boldsymbol{o}_k^j, \boldsymbol{\tau}_{i,k}) < 0 \;\forall i, k, j$) in the final solution of (6.9).

Condition 2. The guidance planner finds a trajectory in each homotopy class where a dynamically feasible trajectory exists.

Condition 3. The executed trajectory is selected via (6.11).

[3] $J_i^* = \infty$ when the optimization is infeasible.

[4] In the following, with some abuse of notation, $J(\boldsymbol{\tau})$ refers to the optimal cost of the optimization initialized with trajectory $\boldsymbol{\tau}$.

Theorem 6. *If Conditions 1-3 hold, T-MPC is HGO for optimization problem* (6.2).

Proof. Under Condition 1, the solution for each optimization (6.9), $\boldsymbol{\tau}_i^*$, is locally optimal for (6.2) since homotopy constraints (6.9e) are the only distinction between the two problems. Therefore, $J(\boldsymbol{\tau}_i^*) \leq J(\boldsymbol{\tau}_i^-)$. If Condition 2 is satisfied, then $\mathcal{T}_P$ contains a guidance trajectory in every feasible homotopy class. Therefore, under Condition 3, the final trajectory $\boldsymbol{\tau}^*$ executed by T-MPC satisfies $J(\boldsymbol{\tau}^*) \leq J(\boldsymbol{\tau}_i^*) \leq J(\boldsymbol{\tau}_i^-)$, $\forall i$ and the HGO property is obtained □

This shows under what conditions the proposed planner finds a provably HGO trajectory. Although these conditions are useful for analysis, they are not necessarily satisfied in practice. Condition 1 can fail if there is no local optimum in a homotopy class or when the linearization around the guidance trajectory restricts the optimization.
Condition 2 is hard to guarantee in crowded environments. In 2-D navigation with M obstacles, there can be 2^M homotopy classes that do not wind around obstacles. Although robot dynamic constraints and bundled obstacles may reduce this amount in practice, the number of classes can still be too large. Limiting the planner to P classes allows us to plan in real-time, but the executed trajectory may not be HGO.
Condition 3 ensures that the lowest cost trajectory is executed but can lead to non-smooth driving behavior over consecutive iterations and it may be preferable to use the consistent decision in (6.12) instead.
While these conditions may not always be satisfied and the HGO property is not provably obtained in each iteration, this chapter will show that the proposed planner always improves on the local planner in isolation.

6.3.7. NON-GUIDED LOCAL PLANNER IN PARALLEL

The constraints and initialization provided by the guidance planner allow the local planner to escape poor local optima. Once the planner is in the correct homotopy class, the restrictions imposed by the guidance planner (i.e., homotopy constraints) may degrade performance. For this reason, an extension of the proposed planner is considered where the regular local planner without guidance (i.e., the optimization in Eq. (6.2)) is added to the set of parallel guided local planners. Since this planner is less restricted and does not rely on the global planner, it can occasionally find a better solution.
Next to practical benefits, this allows us to trivially establish that the proposed scheme does not achieve a higher cost solution than the local planner in isolation.

Theorem 7. *Consider the planner in Fig. 6.3 that includes a non-guided planner with solution $\bar{\tau}^*$. If a trajectory is selected according to* (6.11)*, then $J(\boldsymbol{\tau}^*) \leq J(\bar{\boldsymbol{\tau}}^*)$.*

Proof. Decision (6.11) picks the lowest cost solution from $J(\boldsymbol{\tau}_0^*), \ldots, J(\boldsymbol{\tau}_P^*), J(\bar{\boldsymbol{\tau}}^*)$, which cannot exceed $J(\bar{\boldsymbol{\tau}}^*)$. □

If the guided plans are always higher or equal cost compared to the non-guided planner, then this planner architecture reduces to the local planner (the non-guided planner is always selected). If they ever have a lower cost, then guidance *must* improve the planner in the sense that it reduces the cost of the executed trajectory. The following section will show that the latter holds true. This chapter refers to the method where the non-guided local planner is added in parallel as T-MPC++.

6.3.8. COMPUTATION TIME ANALYSIS

T-MPC plans guidance trajectories before optimization. The computational complexity of the guidance planner (see Algorithm 5) is analyzed in the following, considering the number of PRM samples n, obstacles M and distinct trajectories P.

Visibility-PRM The time complexity of regular Visibility-PRM is dominated by the visibility check. When adding a node, it checks its visibility in the worst case to all nodes (if all nodes are Guards), where each visibility check considers all obstacles. Its time complexity therefore is $O(n^2M)$. The algorithm additionally verifies that new connections are distinct. The time complexity of a single homotopy comparison is $O(M)$: the H-signature or winding numbers are evaluated for each obstacle. The homotopy class is compared roughly n times if a new distinct connection neighbors all connectors. Its time complexity therefore is $O(n^2M)$ and does not change the time complexity of Visibility-PRM.

DepthFirstSearch Each node links to at most one goal. Hence searching the graph for at most P paths to each goal at worst considers each node once. Its time complexity is $O(n)$.

FilterAndSelect Sorting P trajectories has time complexity $O(P \log P)$. Filtering homotopy distinct trajectories from the sorted list must compare a trajectory to P others in the worst case and has time complexity $O(P^2M)$.

IdentifyAndPropagate Similarly, comparing the homotopy class of new and existing trajectories has time complexity $O(P^2M)$. Propagating the graph has time complexity $O(n)$.

Total The total time complexity of the guidance planner is $O((n^2+P^2)M)$. In practice, this time complexity can be approximated by $O(n^2M)$ (Visibility-PRM dominates the time complexity), given that the number of relevant homotopy classes is typically small (i.e., $n \gg P$). Thanks to the propagation of nodes from the previous iteration, n can be relatively small as well (e.g., $n < 100$). For the use case of this chapter, a relatively small n and P are usually sufficient to construct a sparse graph from which the relevant homotopy classes can be extracted.

6.4. SIMULATION RESULTS

In the following, the planner is compared against several baselines on a mobile robot navigating among pedestrians.

6.4.1. IMPLEMENTATION

The implementation for T-MPC is written in C++/ROS and will be released open source[5]. The guidance planner will also be released as a standalone package.

[5]See `https://github.com/tud-amr/mpc_planner`

Table 6.1: Experimental settings.

Parameter Name	Parameter Value	Parameter Description
N	30	Global and local planner horizon
ΔT	0.2 s	Integration time step
h	0.05 s	Planning time step
n	30	Visibility-PRM sample limit
$T_{\max}$	10 ms	Visibility-PRM time limit
Eq. 6.3	H-signature	Homotopy comparison function
P	4	# of distinct guidance trajectories
G	5×5	Grid of goals (longitudinal × lateral)
r	0.725 m	Combined obstacle and robot radius
w_c	0.05	Optimization contouring weight
w_l	0.75	Optimization lag weight
w_v	0.55	Optimization velocity tracking weight
w_ω	0.85	Optimization rotational velocity weight
w_a	0.34	Optimization acceleration weight
Decision	Eq. (6.12)	Type of decision-making
c_i	0.75	Discount factor for trajectory in previously followed homotopy class

For the deterministic simulations, the optimization-based planner LMPCC [27] is implemented as the local planner. The robot dynamics follow second-order unicycle dynamics [144]. The objective of the robot, with weights w, is given by[6]

$$J = w_c J_c + w_l J_l + w_v J_v + w_\omega J_\omega + w_a J_a \tag{6.13}$$

for each time instance k in the horizon N. Herein, J_c, J_l are the contour and lag error used to follow the reference path, $J_v = ||v - v_{\text{ref}}||_2^2$ tracks a desired velocity and $J_\omega = ||\omega||_2^2$, $J_a = ||a||_2^2$ weigh the control inputs consisting of the rotational velocity ω and acceleration a. Collision avoidance constraints are imposed with $g(\boldsymbol{x}_k, \boldsymbol{o}_k^j) \leq 0$,

$$g(\boldsymbol{x}_k, \boldsymbol{o}_k^j) = 1 - (\Delta \boldsymbol{p}_k^j)^T \boldsymbol{R}(\phi)^T \begin{bmatrix} \frac{1}{r^2} & 0 \\ 0 & \frac{1}{r^2} \end{bmatrix} \boldsymbol{R}(\phi)(\Delta \boldsymbol{p}_k^j), \tag{6.14}$$

here $\Delta \boldsymbol{p}_k^j = \boldsymbol{p}_k - \boldsymbol{o}_k^j$, $\boldsymbol{R}(\phi)$ is a rotation matrix with orientation ϕ of the robot and $r = r_{\text{robot}} + r_{\text{obs}}$. These nonconvex constraints directly formulate that the robot region should not overlap with that of the obstacles. Each parallel local optimization is solved with Forces Pro [143]. Parameters of the full planner are listed in Table 6.1. Weights of the guidance and local planners are manually tuned. The planning scheme, including guidance and local planners, is updated in each iteration in a receding horizon manner.

6.4.2. SIMULATION ENVIRONMENT

The first simulation environment (see Fig. 6.1) consists of a mobile robot (Clearpath Jackal) moving through a 6 m wide corridor with up to 12 pedestrians. The robot follows the centerline with a reference velocity of 2 m/s and is controlled at 20 Hz. The

[6]The repulsive forces around obstacles from [27] are not used as they lead to more conservative plans and slow down the optimization problem.

pedestrians follow the social forces model [142] using implementation [164]. They interact with other pedestrians and the robot and are aware of the walls. A constant velocity model is used to predict the future pedestrian positions for the planner. The pedestrians have a radius of 0.3 m. A radius of 0.4 m is specified in the planners to account for discretization effects, allowing collisions to be clearly identified. Pedestrians spawn on two sides of the corridor with the objective to traverse the corridor. The random start and goal locations are the same for each planner.

6.4.3. Comparison to Baselines

The planner is compared to four baselines. Baselines are selected based on the availability of an open-source implementation and their application to navigation in 2D dynamic environments. The following baselines are considered:

- **Motion Primitives** *(global planner)* [17]: A non-optimization-based global planner that respects the robot dynamics.
- **TEB Local Planner** *(topology-guided planner)* [45]: One of the most used local planners in the ROS navigation stack [165] that considers multiple homotopy classes.
- **LMPCC** *(local planner)* [27]: An open-source non parallelized MPC (see Sec. 6.4.1). The previous solution shifted forward in time is supplied as the initial guess of the optimization.
- **Guidance-MPCC** *(topology-guided planner)* [152]: The previous conference work. The guidance planner used in this work is updated to make it more competitive with T-MPC++.

The same weights are used for the MPC planners (LMPCC, Guidance-MPCC, T-MPC, T-MPC++). Baseline actuation limits and tracking objectives are adapted to match the MPC objectives. TEB Local Planner tuning uses its default but with increased collision avoidance weight (from 10 to 20) to decrease collisions in crowded environments.
The simulations are performed with $4, 8$ and 12 pedestrians.
Evaluation Metrics. The planners are compared based on the following metrics.

1. *Task Duration:* The time it takes to reach the end of the corridor.
2. *Safety:* The percentage of experiments in which the robot does not collide with pedestrians or the corridor bounds.
3. *Runtime:* Computation time of the control loop.

It is noted that collisions in simulation may not correspond to collisions in practice but do provide insight into the safety of the planners (more details in Sec. 6.6).
The simulations are performed on a laptop with an Intel i9 CPU@2.4GHz 16 core CPU. The implementation of T-MPC and T-MPC++ uses P and $P+1$ CPU threads, respectively. Threads are terminated when they exceed the control period of 50 ms and in this case, the best trajectory out of the completed optimization problems is selected.

Table 6.2: Quantative results for interactive navigation simulations of Sec. 6.4.3 over 200 experiments with pedestrian motion *prediction* following a constant velocity model. Task duration (Dur.) and runtime are reported as "mean (std. dev.)". Without obstacles, the task duration is 12.9 s. Best planner performances per column are denoted in **bold**. Underlined results indicate that T-MPC++ significantly outperforms the respective method for $p = 0.001$ (U-test).

# Ped.	Method	Dur. [s]	Safe (%)	Runtime [ms]
0	-	12.9 (0.0)	-	-
4	Frenét-Planner [6]	14.0 (0.9)	77	11.6 (2.6)
	TEB Local Planner [8]	**13.0** (1.1)	**100**	**5.8** (5.2)
	LMPCC [3]	13.1 (0.4)	98	11.3 (2.6)
	Guidance-MPCC [32]	**13.0** (0.4)	92	13.9 (1.3)
	T-MPC (ours)	**13.0** (0.2)	**100**	18.3 (3.5)
	T-MPC++ (ours)	**13.0** (0.1)	**100**	19.4 (3.7)
8	Frenét-Planner [6]	15.1 (1.7)	64	11.6 (2.4)
	TEB Local Planner [8]	13.8 (1.7)	**98**	**7.5** (5.1)
	LMPCC [3]	13.8 (1.3)	96	13.7 (4.2)
	Guidance-MPCC [32]	**13.2** (0.7)	92	13.4 (1.4)
	T-MPC (ours)	13.3 (0.7)	96	20.2 (4.8)
	T-MPC++ (ours)	**13.2** (0.6)	96	21.4 (4.9)
12	Frenét-Planner [6]	16.5 (2.4)	42	14.1 (6.3)
	TEB Local Planner [8]	14.9 (2.4)	92	**8.7** (5.4)
	LMPCC [3]	14.0 (1.5)	90	12.9 (4.5)
	Guidance-MPCC [32]	**13.6** (1.1)	86	13.6 (1.6)
	T-MPC (ours)	14.1 (1.3)	90	18.3 (5.1)
	T-MPC++ (ours)	**13.6** (1.0)	**93**	20.1 (5.4)

The results over 200 experiments are summarized in Table 6.2 and the task duration is visualized in Fig. 6.7. The Motion Primitives planner is not safe and has a significantly longer task duration than the other planners even in the least crowded case. In the two more crowded environments, LMPCC has a significantly longer task duration than T-MPC++ and is less safe, in part because a single trajectory is planned. When the optimization becomes infeasible, it often does not recover fast enough to avoid oncoming obstacles. TEB Local Planner is marginally safer than LMPCC and maintains competitive average task durations to the other methods in the four and eight pedestrian scenarios under less computational cost. In the crowded scenario, T-MPC++ completes the task significantly faster. Additionally, in all cases, T-MPC++ has a much smaller standard deviation of the task duration indicating that its behavior is more consistent (visible also in Fig. 6.7). TEB Local Planner soft constrains collision avoidance which, in crowded environments, leads the robot into poor behaviors (e.g., reversing) due to the shape of the cost function. To further compare T-MPC++ and the TEB Local Planner, the trajectories are visualized in Fig. 6.8. The proposed planner results in smoother and more consistent trajectories and can follow the reference path more closely. The smoothness of the planners is quantitatively compared through the standard deviation on second-order input commands. The deviation on acceleration (σ_a) and rotational acceleration (σ_α) for TEB Local Planner are higher ($\sigma_a = 0.16$, $\sigma_\alpha = 0.12$) than for T-MPC++ ($\sigma_a = 0.04$, $\sigma_\alpha = 0.05$).

Out of the guidance planners, Guidance-MPCC attains the same mean task duration as T-MPC++ but is less consistent (high std. dev.) because the guidance planner, which does not account for the robot dynamics, determines the planner's behavior. This results in larger tracking errors under the same cost function, for example in the 12 pedestrian

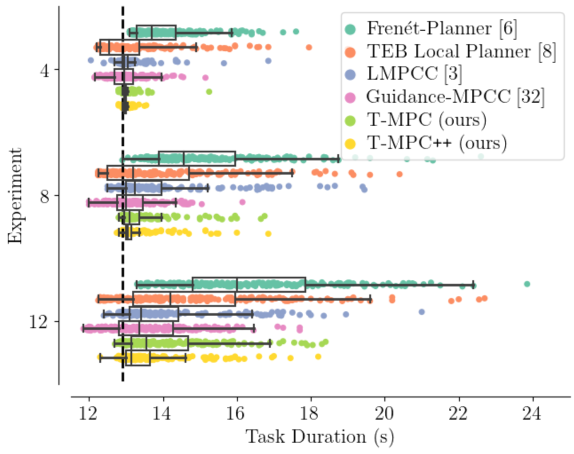
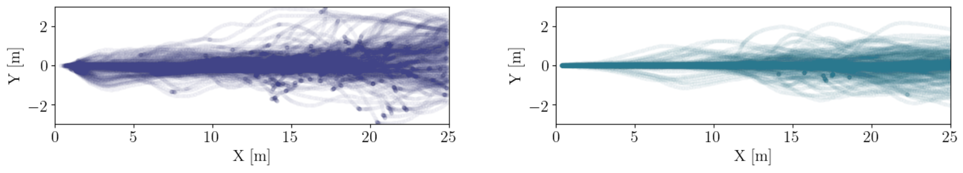


Figure 6.7: Visualization of the task duration (i.e., the time taken to reach the goal) in Table 6.2. The dashed vertical line denotes the task duration without obstacles. The proposed method achieves the smallest variation in task duration and the shortest task duration in crowded environments.



(a) TEB-Planner [45].

(b) T-MPC++ (ours)

Figure 6.8: Trajectories of 200 experiments with 12 pedestrians for the TEB-Planner and T-MPC++. The proposed method results in smoother and more consistent robot navigation.

case, the mean path and velocity errors of Guidance-MPCC are 0.45m and 0.47m/s, respectively, compared to values of 0.20m and 0.42m/s for T-MPC++. It also leads to more collisions than LMPCC in all scenarios. T-MPC is generally faster than LMPCC, except for the 12 pedestrian case. In this environment, the local planner may find solutions that the guidance planner did not, given that the space is cluttered. T-MPC++ demonstrates superior navigation performance over the other planners: it is significantly faster (with the exception of Guidance-MPCC), varies less in its task duration (lower std. dev.) and is safer in almost all cases (the TEB Local Planner is safer in the 8 pedestrian case). T-MPC++ has higher computational demands than the other planners. Compared to the other MPC planners, T-MPC++ first computes guidance plans. It is measured that this step takes approximately 5 ms on average (included in the runtime of Table 6.2) in all scenarios.

6.4.4. Crowded Baseline Comparison

T-MPC++ is further compared against TEB Local Planner in a square-shaped crowded environment with 50 pedestrians. The robot's task is to move diagonally through the

Table 6.3: Quantative results in crowded environment of Sec. 6.4.4 over 200 experiments. Notation follows that of Table 6.2. Without obstacles, the task duration is 20.6 s. The runtime is denoted as "mean (max)".

# Ped.	**Method**	**Dur. [s]**	**Safe (%)**	**Runtime* [ms]**
0	-	20.6 (0.0)	-	-
50	TEB Local Planner [8]	22.4 (2.4)	**92**	**9.4** (177.1)
	T-MPC++ (ours)	**21.0** (0.9)	**92**	21.2 (46.9)

environment at $v_{\text{ref}} = 1.5\,\text{m/s}$. The 12 pedestrians closest to the plan are considered, with preference for nearby pedestrians in both methods. Pedestrians are removed when they reach the goal to prevent unpredictable turns. Table 6.3 presents the results. T-MPC++ is significantly faster and the standard deviation of the task duration is less than half that of TEB Local Planner. TEB Local Planner is on average computationally faster as a new plan is not computed in each iteration. When it does compute a plan, its computation time can exceed the planning frequency of 20Hz as shown by the maximum in Table 6.3 (it exceeded 50 ms in 0.26% of its iterations). In contrast, T-MPC++ is explicitly limited to 50 ms such that its maximum computation time remains below 50 ms.

6.4.5. Sensitivity Studies

To provide more insight into the key parameters of the approach, the sensitivity of the parameters is studied.

Sensitivity to the Number of Trajectories P

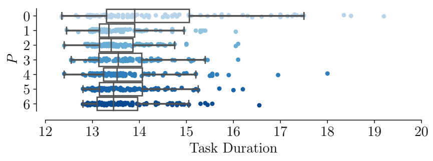
To study how the number of guidance trajectories impacts the task duration, 100 experiments are run in the 12 pedestrians environment and $P = 0, \dots, 6$ guidance trajectories are computed. The $P = 0$ case corresponds to the non-guided local planner, LMPCC [27]. Fig. 6.9 displays statistics on task duration. It is observed from Fig. 6.9a that guidance trajectories reduce the task duration compared to the local planner. Fig. 6.9c shows for both T-MPC and T-MPC++ how often the planner becomes infeasible. It indicates that the availability of at least two plans makes it more likely that a trajectory is found and shows that the non-guided planner added in T-MPC++ further improves feasibility.

Sensitivity to the Consistency Parameter c_i

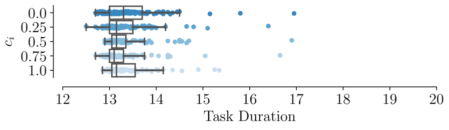
The sensitivity of the consistency parameter c_i (see Eq. (6.12)) is varied within its range $c_i \in [0, 1]$, including $c_i = 0$ that enforces the robot to follow the trajectory in the previous homotopy class, if it still exists, and $c_i = 1$ that expresses no preference.
Fig. 6.9 compares task duration and infeasibility. Both extreme values show poor performance. For $c_i = 0$, the task duration increases, while for $c_i = 1$, the planner becomes infeasible more often (it is indecisive). $c_i = 0.75$ is deployed, which best retains the feasibility of the optimization.

6.4.6. Empirical Cost Comparison

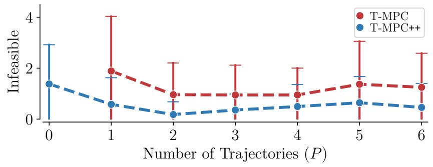
To verify that T-MPC++ is able to find lower-cost local optima than the local planner in isolation (as discussed in Sec. 6.3.6), the optimal cost of the executed trajectory attained by LMPCC (local planner), T-MPC and T-MPC++ that use identical cost functions is compared. This study is performed in the crowded environment of Sec. 6.4.4 over 50



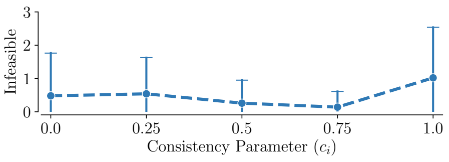
(a) Task duration for varying P.



(b) Task duration for varying c_i.



(c) Infeasible iterations (mean and std) per experiment for varying P.



(d) Infeasible iterations (mean and std) per experiment for varying c_i.

Figure 6.9: Sensitivity study of the number of (a, c) guidance trajectories P and (b, d) the consistency parameter c_i. (a, b) Task duration of T-MPC, individual experiments denoted by dots. (c, d) Mean and standard deviation of the number of control iterations in which the optimization is infeasible per experiment.

Table 6.4: Comparison of the attained cost in the crowded environment of Sec. 6.4.4 over 50 experiments. Notation, including the cost that is tested for significance, follows that of Table 6.2. The cost excludes infeasible planner iterations.

# Ped.	Method	Dur. [s]	Safe (%)	Cost	Runtime [ms]
50	LMPCC [3]	21.9 (1.7)	84	2.25 (9.36)	**7.1** (4.8)
	T-MPC (ours)	21.6 (1.5)	84	1.60 (6.21)	20.6 (7.7)
	T-MPC++ (ours)	**21.0** (0.9)	**96**	**1.05** (4.67)	21.7 (7.4)

experiments. Table 6.4 indicates that T-MPC finds lower-cost local optima than LMPCC and does so consistently (lower std. dev.). T-MPC++ further reduces this cost and its deviation. The average cost of T-MPC++ is less than half that of the non-guided planner.

6.4.7. T-MPC UNDER OBSTACLE UNCERTAINTY

To illustrate that T-MPC applies to different local planner formulations, T-MPC is deployed on top of CC-MPC [84]. CC-MPC is a local planner that considers the probability of collision with obstacles when their motion is represented by a Gaussian distribution at each time step. The motion of the obstacles follows the uncertain dynamics

$$\boldsymbol{o}^j_{k+1} = \boldsymbol{o}^j_k + (\boldsymbol{v}^j_k + \boldsymbol{\eta}^j_k)dt,\ \boldsymbol{\eta}^j_k \sim \mathbb{P}^j_k, \tag{6.15}$$

Here $\boldsymbol{v}_k$ is the velocity that follows the social forces model as in previous experiments. The distribution of $\boldsymbol{\eta}^j_k \in \mathbb{R}^2$ follows a bivariate Gaussian distribution, $\boldsymbol{\eta}^j_k \sim \mathcal{N}(\boldsymbol{\mu}^j_k, \boldsymbol{\Sigma}^j_k)$, where $\boldsymbol{\mu}^j_k = \boldsymbol{0}$ and $\boldsymbol{\Sigma}^j_k = \sigma \boldsymbol{I}$. In this simulation $\sigma = 0.3$. Instead of deterministic collision avoidance constraints (6.14), a chance constraint with risk $0 < \epsilon < 1$

$$\mathbb{P}\left[||\boldsymbol{p}_k - \boldsymbol{o}^j_k||^2_2 \geq r\right] \geq 1 - \epsilon,\ \forall k, j \tag{6.16}$$

that specifies collision avoidance to hold with a probability of $1-\epsilon$ for each agent and time instance is formulated. CC-MPC [84] reformulates this constraint using the Gaus-

Table 6.5: Quantative results for simulations with uncertain obstacle motion of Sec. 6.4.7 over 200 experiments. Notation follows that of Table 6.2.

#	Method	Task Duration [s]	Safe (%)	Runtime [ms]
High Risk	CC-MPC [4]	<u>15.8</u> (2.3)	91	**17.0** (8.7)
	TCC-MPC++ (ours)	**14.1** (0.8)	**96**	34.5 (7.9)
Medium Risk	CC-MPC [4]	<u>16.5</u> (2.7)	92	**17.4** (9.7)
	TCC-MPC++ (ours)	**15.1** (1.4)	**93**	35.8 (8.2)
Low Risk	CC-MPC [4]	<u>17.2</u> (2.5)	90	**18.5** (10.2)
	TCC-MPC++ (ours)	**16.1** (1.3)	**97**	38.1 (7.9)

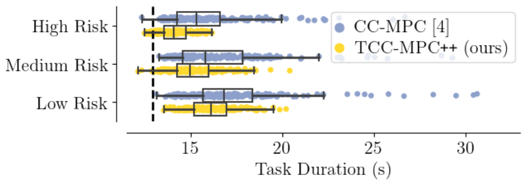


Figure 6.10: Visualization of the task duration in Table 6.5.

sian 1-D CDF in the direction of the obstacle. In this formulation, the collision avoidance constraint is linearized and reduces to

$$(\boldsymbol{A}_k^j)^T(\boldsymbol{p}_k - \boldsymbol{o}_k^j) - r - r^j \geq \operatorname{erf}^{-1}(1-2\epsilon)\sqrt{2(\boldsymbol{A}_k^j)^T\boldsymbol{\Sigma}_k^j(\boldsymbol{A}_k^j)},$$

with $\boldsymbol{A}_k^j$ as in (6.8) and where erf^{-1} is the inverse standard error function.
T-MPC is applied with the non-guided CC-MPC in parallel (referred to as TCC-MPC++). The uncertainty directly affects the local planner, while the guidance planner only avoids the mean obstacle trajectories. It may happen that some guidance trajectories are not feasible for the local planner.
Both planners are deployed in the scenario with 12 randomized pedestrians and compared under high ($\epsilon = 0.1$), medium ($\epsilon = 0.01$) and low ($\epsilon = 0.001$) risk settings. The results are shown in Table 6.5 and visualized in Fig. 6.10. TCC-MPC++ consistently outperforms CC-MPC in isolation, leading to significantly faster and more consistent task completion and fewer collisions. The local planner often collides when it becomes infeasible since it cannot recover. The initialization provided by the guidance planner resolves infeasibility and improves robustness.

6.5. Real-World Experiments

The proposed planner is demonstrated in a real-world setting on a mobile robot driving among pedestrians.

6.5.1. Experimental Setup

The experiment takes place in a 5m×8m square environment where participants walk among the robot. The robot and pedestrian positions are detected by a motion capture system at 20Hz. The pedestrian positions are passed through a Kalman filter and constant velocity predictions are passed to the planner. The robot is given a reference path between two opposite corners and turns around once a corner is reached.

6.5.2. One Pedestrian

In the first set of experiments, a single pedestrian interacts with the robot. TCC-MPC++ is run to evade the pedestrian. Fig. 6.11 shows the results of two experiments. In Fig. 6.11a, the pedestrian turns and speeds up to pass the robot in front. The robot changes its behavior from passing in front to passing behind to let the pedestrian pass. In Fig. 6.11b, the pedestrian changes its intended passing side from the right to the left side of the robot. The planner detects the change in direction and switches sides, passing the pedestrian smoothly.

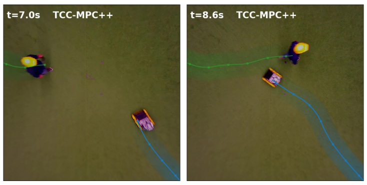
6.5.3. Five Pedestrians

In the second set of experiments, LMPCC, T-MPC++, CC-MPC and TCC-MPC++ are run for 5 minutes each with 5 pedestrians in the space. The participants were instructed to walk naturally toward a point on the other side of the lab. Before starting the recorded experiments, participants were asked to walk for 3 minutes without the robot, to get used to the environment. The participants were not aware of the planner running in each experiment and the order of planners was randomized. Several runs of CC-MPC and TCC-MPC++ are visualized in Fig. 6.12 and Fig. 6.13, respectively. The reader is encouraged to watch the associated video [163]. Fig. 6.12a highlights a case where CC-MPC conservatively avoids two pedestrians, not detecting the pathway in between the pedestrians. In Fig. 6.12b, CC-MPC gets infeasible and is not able to replan fast enough, requiring the robot to wait for a pedestrian to pass. In all experiments of Fig. 6.13, the robot passes pedestrians efficiently and smoothly. Fig. 6.13a and 6.13b highlight cases where the planner has to navigate through the crowd and does so successfully. In Fig. 6.13c, the planner falls in line with a pedestrian to pass another pedestrian.

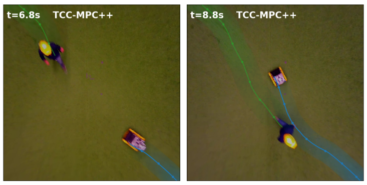
After the experiments, the participants unanimously preferred the planners running in experiments 1 and 4, which were both guided planners. In general, participants reported that the guided planners felt safer and more predictable than the non-guided planners.

6.6. Discussion

The results in this chapter have indicated that the proposed global and local planning framework can improve the safety, consistency and time efficiency of the planner. Further insights related to the planning framework are discussed in the following.

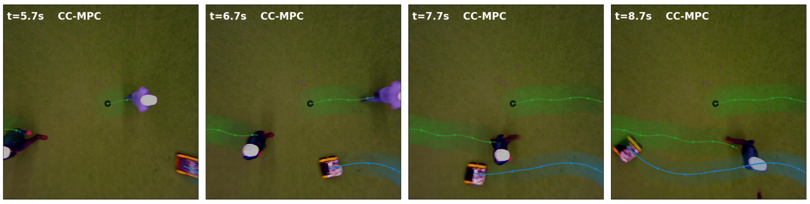


(a) Encounter with a turning pedestrian.

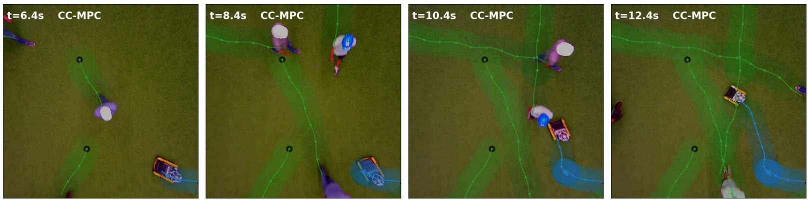


(b) Pedestrian switching intended passing side.

Figure 6.11: Overlayed top view camera images of real-world experiments of TCC-MPC++ with one pedestrian. Blue and green overlays denote the robot and pedestrian trajectories, respectively. Timestamps of each image denoted in the upper left corner.



(a) The baseline fails to find the more efficient trajectory passing between two uncoming pedestrians.



(b) The baseline becomes infeasible and does not recover quickly (note the time scale).

Figure 6.12: Trajectories of baseline CC-MPC [84] overlayed on camera images. Black dots denote pedestrian start positions.

6.6.1. Safety in Dynamic Environments

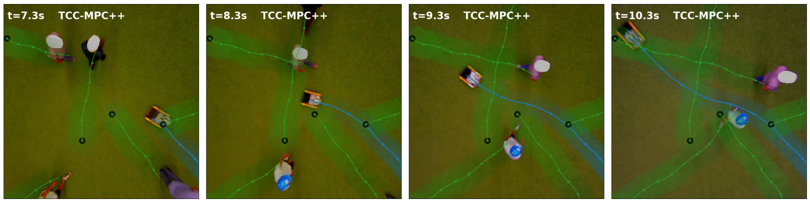
Planners deployed in the real world must be safe (i.e., collisions are unacceptable) and should not impede humans more than necessary. Table 6.2 indicated that all planners collided at least once in simulation. To identify the source of collisions, the experiments were repeated using the social forces model for both the pedestrian simulation model and the prediction model of the planner (i.e., removing prediction mismatch). The results are summarized in Table 6.6. Collisions are almost reduced to zero for T-MPC++ in this case, which shows that prediction mismatch causes most of the collisions.

In the real-world experiments, no collisions were observed. The video of the experiments contained seven instances where pedestrians had to take evasive action using LMPCC (3 out of 63 interactions) and CC-MPC (4 out of 61 interactions) and 0 out of 61 and 60 interactions for T-MPC++ and TCC-MPC++. This indicates that pedestrians take direct evasive action when the robot impedes their safety, deviating from the social forces model. Collisions in simulation therefore seem to correspond to cases where the human must take evasive action in practice.

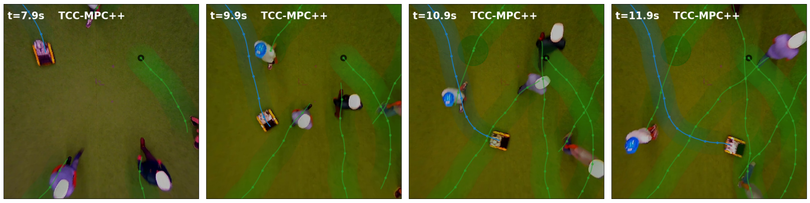
It was also observed in the real world that evasive action was necessary when the baseline planner became infeasible. Safety guarantees provided through constraints only hold when the optimization problem is feasible and can impose danger when no solution is found in time. The proposed approach reduced this danger by planning more

Table 6.6: Quantative results for the simulations of Sec. 6.4.3 repeated with the pedestrian motion predictions following the social forces model.

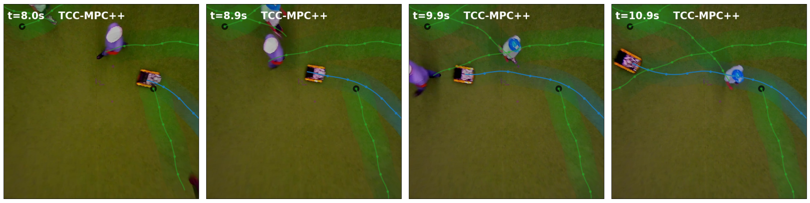
# Peds.	Method	Task Duration [s]	Safe (%)	Runtime [ms]
4	T-MPC++ (ours)	13.0 (0.1)	100	25.4 (4.9)
8	T-MPC++ (ours)	13.2 (0.4)	100	27.7 (6.2)
12	T-MPC++ (ours)	13.6 (0.7)	98	26.7 (7.0)



(a) The planner passes between two humans smoothly.



(b) Noticing the oncoming pedestrians, the planner passes behind them smoothly.



(c) The planner follows another passing pedestrian to pass.

Figure 6.13: Trajectories of TCC-MPC++ overlayed on camera images for three examples.

than a single trajectory and no dangerous cases of infeasibility were observed for T-MPC.

6.6.2. Advantages of Parallel Optimization

Deploying several local planners in parallel makes it more likely that the planner returns a trajectory (in time) as a feasible trajectory can be provided by not a single, but several optimization problems. This effect is likely more pronounced when the planning problems are more diverse. In addition, parallelization reduces the maximum computation times. The fastest solved optimization immediately provides a trajectory and other problems can be ignored if necessary. The parallel planner computation time is, at worst, equal to that of a single planner but is almost always faster. Several CPU cores are necessary to parallelize the planner but are usually available. These two advantages inherently improve all performance metrics (e.g., safety and time efficiency) as a solution is more often available. Because redundancy and reduced computation times are key for real-world applications, parallelization may be the key to safely and efficiently deploying optimization-based planners in practice.

6.6.3. Selection of the Homotopy Class

The decision-making in Sec. 6.3.5 used the optimal costs of the local planners to decide which trajectory to execute and preferred the homotopy class of the last followed trajectory. In practice it was observed that the robot stayed closer to the reference path and velocity, and passed pedestrians behind rather than in front when necessary. It was additionally observed that due to measurement and prediction noise, making a more consistent decision led to better navigation. With high consistency, another behavior must be much better before the robot switches its behavior. This provides a margin for error when estimating the cost of the trajectory. While the proposed decision-making led to fast navigation, it ignored social norms. The decision-making could be made more socially compliant by learning to pick the homotopy class that humans take from data (see e.g., [10], [48], [70]). Finally, as noted by [45] homotopy classes merge and split when obstacles are passed or appear in the planning horizon. Reacting to these events could make the planner more responsive in practice.

6.6.4. Limitations and Future Work

One of the remaining limitations of the framework is the lack of interaction between the humans and the robot. The social forces model that this chapter used in simulation is interactive but does not accurately model human-robot interactions. It may be possible to reduce the complexity of interaction with humans to an explicit decision on the topology class of interaction (along the lines of [66]) that simplifies the planning problem. Additionally, the guidance planner can possibly be extended to incorporate non-Gaussian (i.e., multi-modal) uncertainty in obstacle motion (e.g., [157]).

6.7. Conclusion

This chapter presented a two-fold planning approach to address the inherent local optimality of optimization-based planners. The proposed planner consisted of a high-level global planner and a low-level optimization-based planner. By accounting for the topology classes of trajectories in the dynamic free space, trajectories with distinct passing behaviors were generated and were used to guide several local optimization-based planners in parallel.

This chapter simulated a mobile robot navigating among pedestrians and showed that the proposed guided planner resulted in faster and more consistent robot motion than existing planners, including a state-of-the-art topology-guided planner. The same improvement was observed qualitatively in the real world, where the planner navigated successfully among five pedestrians.

Future work aims to deploy the proposed method, considering uncertainty in obstacle motion, on a self-driving vehicle navigating in urban environments.

A. Appendix - Homotopy Comparison

This appendix details and compares three implementations of the homotopy comparison function (6.3) for 2D motion planning in dynamic environments.

A.1. H-signature

The H-signature [7] approximates homotopy classes by homology classes, formally defined as follows.

Definition 5. *[7] (Homologous Trajectories) Two trajectories* $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2 \in T$ *connecting the*

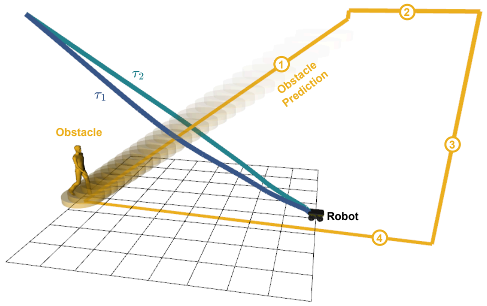


Figure 6.14: Illustrating example for the H-signature [7]. The two trajectories $\boldsymbol{\tau}_1$ (blue) and $\boldsymbol{\tau}_2$ (green) are in distinct homology classes as the loop that they form contains obstacle skeleton (orange). In practice, link (3) is placed far away.

same start and end points $\boldsymbol{x}_s$ and $\boldsymbol{x}_g$ respectively, are homologous iff $\boldsymbol{\tau}_1$ together with $\boldsymbol{\tau}_2$ (the latter in the opposite direction) forms the complete boundary of a 2-dimensional manifold embedded in $\mathcal{X}$ not containing or intersecting any of the obstacles.

If two trajectories are homotopic, they are homologous. The reverse does not hold. The H-signature in $3D$ computes the virtual magnetic field resulting from the current loop of each obstacle and its prediction, integrated over trajectory $\boldsymbol{\tau}$. If two trajectories $\boldsymbol{\tau}_1, \boldsymbol{\tau}_2$ enclose the loop of obstacle j, then $h^j(\boldsymbol{\tau}_1) \neq h^j(\boldsymbol{\tau}_2)$. Hence, two trajectories are equivalent if $h^j(\boldsymbol{\tau}_1) = h^j(\boldsymbol{\tau}_2)\ \forall j$ and are distinct otherwise. The H-signature for a finite-time state space is computed by constructing a 1D skeleton of the obstacle prediction that is looped outside of the workspace and time horizon. The skeleton is composed of the following lines. (1) The obstacle's prediction for $0 < t < T$. (2) A line upwards to $t = T + \epsilon$ where $\epsilon > 0$ is a small constant and a line going outside of the workspace. (3) A line down to $t = -\epsilon$. (4) A line to the obstacle position at $t = 0$. This skeleton ensures that trajectories can only enclose the predicted obstacle motion for $0 < t < T$.
It is assumed that obstacle trajectories are piecewise linear (e.g., discrete-time trajectories). The integration of the magnetic field $\boldsymbol{B}$ can then be computed analytically (see [7]) per segment i of Obstacle j's skeleton $\overline{\boldsymbol{o}_i^j \boldsymbol{o}_i^{j'}}$ as follows:

$$\boldsymbol{p} = \boldsymbol{o}_i^j - \boldsymbol{r},\ \boldsymbol{p}' = \boldsymbol{o}_i^{j'} - \boldsymbol{r},\ \boldsymbol{d} = \frac{(\boldsymbol{o}_i^{j'} - \boldsymbol{o}_i^j) \times (\boldsymbol{p} \times \boldsymbol{p}')}{||\boldsymbol{o}_i^{j'} - \boldsymbol{o}_i^j||^2},$$

$$\boldsymbol{\Phi}(\boldsymbol{o}_i^j, \boldsymbol{o}_i^{j'}, \boldsymbol{r}) = \frac{1}{||\boldsymbol{d}||^2}\left(\frac{\boldsymbol{d} \times \boldsymbol{p}'}{||\boldsymbol{p}'||} - \frac{\boldsymbol{d} \times \boldsymbol{p}}{||\boldsymbol{p}||}\right), \quad \boldsymbol{B}(\boldsymbol{r}) = \frac{1}{4\pi}\sum_{i=0}^{I} \boldsymbol{\Phi}(\boldsymbol{o}_i^j, \boldsymbol{o}_i^{j'}, \boldsymbol{r}),$$

with I being the number of segments in Obstacle j's skeleton. The integral $\int_{\boldsymbol{l}} \boldsymbol{B}(\boldsymbol{r}) d\boldsymbol{r}$ over looped robot trajectories $\boldsymbol{l}$ yields 1 if the obstacle is enclosed and 0 otherwise. To compare trajectories that reach different goals in the guidance planner, their end points are connected directly at $t = T$ with an additional line. The GSL library [166] is used to perform the integration, and computed H-signatures for each trajectory are cached to prevent re-computation.

A.2. WINDING NUMBER

The winding number [51] is a topological invariant that indicates how the robot and obstacle j rotated around each other. It is computed as follows. The relative position of obstacle j to the robot for time step k is $\boldsymbol{d}_k^j = \boldsymbol{p}_k - \boldsymbol{o}_k^j$. The relative angle $\angle\theta_k^j$ is the angle of $\boldsymbol{d}_k^j$ in a fixed global frame. Between time steps k and $k+1$, the relative angle changes by $\Delta\theta_k^j = \theta_{k+1}^j - \theta_k^j$. The winding number accumulates these changes over all time steps, $\lambda(\boldsymbol{\tau}, \boldsymbol{o}^j) = \frac{1}{2\pi}\sum_{k=1}^{N}\Delta\theta_k^j$. The sign of the winding number λ indicates the passing direction, and its magnitude denotes passing progress. A trajectory is considered to pass obstacle j if $|\lambda^j| \geq \lambda_{\text{pass}}$, where by default $\lambda_{\text{pass}} = \frac{1}{4\pi}$. Two trajectories are considered distinct if there exists at least one obstacle that the trajectories pass on different sides and equivalent otherwise[7] Computed winding numbers are cached to prevent recomputation.

A.3. UNIVERSAL VISIBILITY DEFORMATION

Universal Visibility Deformation (UVD) was proposed for static obstacle avoidance in $3D$ and therefore does not exactly capture the local optima for collision avoidance in $2D$ dynamic environments. Two trajectories are in the same UVD class if points along the trajectories can be connected without intersecting with obstacles.

Definition 6. [44] *Two trajectories $\boldsymbol{\tau}_1(s), \boldsymbol{\tau}_2(s)$ parameterized by $s \in [0,1]$ and satisfying $\boldsymbol{\tau}_1(0) = \boldsymbol{\tau}_2(0)$, $\boldsymbol{\tau}_1(1) = \boldsymbol{\tau}_2(1)$, belong to the same uniform visibility deformation class if, for all s, line $\overline{\boldsymbol{\tau}_1(s)\boldsymbol{\tau}_2(s)}$ is collision-free.*

In practice, collisions are checked for s at discrete intervals along the trajectories.

A.4. COMPARISON

The homotopy comparison functions are compared in simulation on the scenario of Sec. 6.4.4 over 100 experiments. Table 6.7 indicates that UVD degrades navigation performance, likely because it is not designed for dynamic environments and may lead to duplicate trajectories in practice. The H-signature and winding numbers show similar navigation performance. Winding numbers are computationally more efficient but require a minimum passing angle to be tuned. Since the H-signature generalizes to higher dimensions and both methods are still real-time, the H-signature is used in this paper.

Table 6.7: Comparison between homotopy comparison implementations. Notation follows that of Table 6.2.

	Method	**Dur. [s]**	**Safe (%)**	**Homotopy Comparison Time [ms]**
Homotopy Comparison	Winding Angles	21.2 (0.9)	**93**	**0.3** (0.7)
	H-Signature	21.2 (1.0)	92	2.1 (4.0)
	UVD	**21.1** (0.9)	88	3.7 (8.5)

[7]Future work could also use winding numbers to distinguish between passing and non-passing trajectories.

7

Application to a Self-Driving Vehicle

In theory, theory and practice are the same. In practice, they are not.

Albert Einstein

1. Overview

Various autonomous demonstrator vehicles were reviewed in Sec. 2.4, including demonstration [28] that preceded the experiments in this chapter. These demonstrations generally use deterministic obstacle predictions in the planner and do not account for the uncertainty in human motion predictions. In this chapter, Safe Horizon MPC is applied to a full-scale self-driving car (a Toyota Prius, see Fig. 7.2a) to demonstrate autonomous summoning among Vulnerable Road Users (VRUs) such as pedestrians and cyclists, accounting for uncertainty. This chapter presents the software stack of the demonstrator vehicle, including the proposed motion planner, and presents real-world experiments that feature dynamic collision avoidance with virtual pedestrians.

1.1. Introduction

Automated driving has been making steady progress over recent years. Witness, for example, the introduction of sophisticated autopilot systems for the highway, allowing the driver to take the eyes off the road and perform side activities, until prompted by the system to retake control (conditional automation). Or consider the emergence of driverless vehicles ("robotaxis") providing mobility services in geo-fenced areas of a few US cities (high automation).

Parts of this chapter have been submitted as:
O. de Groot, et al. "Demonstrating Advanced Vehicle Summon in the Presence of Vulnerable Road Users", in *RSS*, Feb. 2024, *under review*.
This dissertation contributed the methodology, writing and experiments for the motion planning in Sec. 2.3.

Vehicle summon is a canonical functionality of automated driving that allows a user to request a vehicle to their location using a mobile phone. For example, Tesla's Summon feature works for driveways and parking lots and requires the user to maintain a clear line of sight to the vehicle and to closely monitor the vehicle and its surroundings at all times.

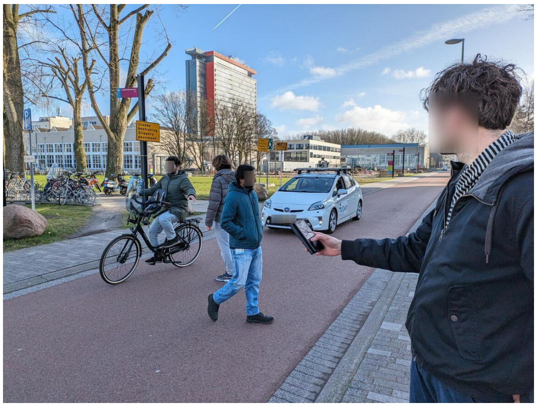


Figure 7.1: Advanced Summon: requesting a vehicle that is able to safely and time-efficiently maneuver around Vulnerable Road Users (VRUs).

This chapter demonstrates an advanced summon functionality that allows a vehicle to arrive to a requested destination autonomously, without such user supervision. Its distinguishing aspect is the ability to both safely and efficiently maneuver around Vulnerable Road Users (VRUs) such as pedestrians and cyclists. See Fig. 7.1. VRUs pose persistent challenges for automated driving, due to their high manoeuvrability, seemingly unpredictable behavior, and their tendency not to always obey the traffic rules. The proposed motion planner considers how the uncertainty of the predicted VRU behavior affects its decisions and considers multiple possible maneuvers to evade the VRUs, in parallel. The proposed demonstration focuses on this motion planner and does not consider a complete mission planner (incl. parking) or the use of an actual mobile phone app to connect with a back-end.
The demonstrated is implemented within Autoware [167], an open-source framework that implements basic building blocks of automated driving. In this demonstration, Autoware's automated valet parking solution is extended for the application of Advanced Vehicle Summon in the presence of VRUs.

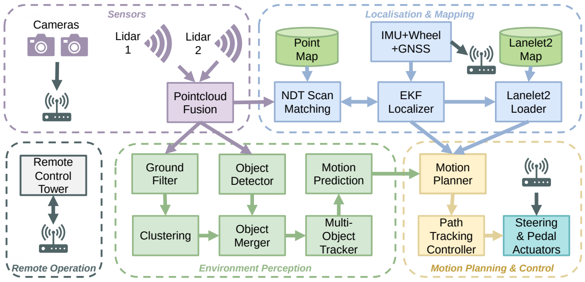
2. DEMO SYSTEM SETUP

The high-level system architecture is shown in Fig. 7.2b. The main components are discussed in the following.

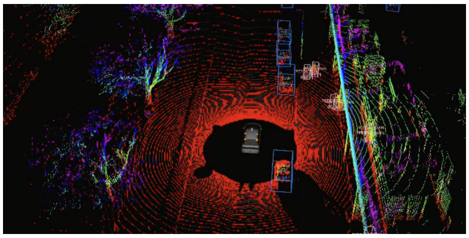
2.1. PROTOTYPE VEHICLE

To perform the demo and the related experiments, a Toyota Prius experimental platform is used. The vehicle has been modified by adding a custom drive-by-wire system called

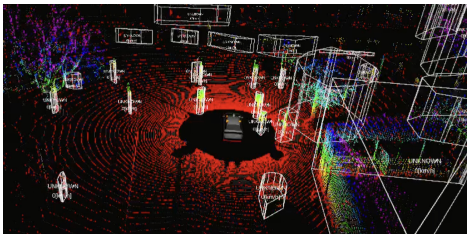
(a) The demonstrator vehicle and its roof-mounted sensors.



(b) Overview of the demonstrator's architecture. The dashed colored regions indicate the main components.



(a) Object Detection



(b) LiDAR Clustering

Figure 7.3: A visualization of the outputs of the object detector and LiDAR clustering. (a) The object detector detects a predefined set of object classes (e.g. car and pedestrian). (b) The LiDAR clustering segments any object protruding from the ground plane, including generic objects.

"Movebox" that allows lateral and longitudinal control of the vehicle. In addition, the vehicle has an on-board computer with two NVIDIA RTX A6000 GPUs, an AMD EPYC 7313 16-Core Processor, and 256 GB RAM.

In terms of sensing capabilities, the vehicle is equipped with two 128-beam rotating LiDARs, a Robosense Ruby and an Ouster OS1-Rev 7 both on the vehicle roof. The Robosense LiDAR, which has a larger detection range, is placed towards the front side of the roof, whereas the Ouster LiDAR, which has a larger vertical Field of View (45°), is placed at an additionally elevated position, more centrally on the roof. It also has eight Surrounding Lucid Triton Cameras and an OXTS3000 Dual antenna GNSS/INS unit. The vehicle and its sensors are shown in Fig. 7.2a.

For the sensors, the manufacturers' drivers are used, while for the *"Movebox"* actuator a custom ROS 2 driver is developed, which translates Autoware control commands (ROS 2 messages) to CAN-BCM messages and sends them to the Electronic Control Unit (ECU) of the vehicle. The *"Movebox"* ROS 2 driver also reads the state of the car (*i.e.* Control Mode, Gear, Steering angle, etc.) from the ECU and translate those signals to high-level ROS 2 messages which are used by Autoware.

2.2. Perception and Localization

The motion planner relies on information about the vehicle state and its environment that are provided by localization and perception subsystems, respectively.

PERCEPTION

The perception subsystem detects VRUs and predicts their future motion. It processes the point clouds from the two LiDARs to detect objects. The objects are tracked across time with a multi-object tracker and for each tracked object, uni-modal motion predictions are obtained with an Extended Kalman Filter (EKF).

LOCALIZATION

The localization stack relies on the vehicle's LiDAR scans, wheel odometry and IMU to determine its position with respect to a geo-localized 3D reference map. The 3D map is created offline prior to driving, and is manually annotated with lane information for the later planning stage. NDT LiDAR scan matching [168] is used for online localization within the map. To produce a smooth trajectory this localization is temporally filtered with an EKF that considers the vehicle kinematics.

2.3. MOTION PLANNING

The motion planner processes the information provided by localization and perception systems and outputs a trajectory to be tracked by the vehicle controller. The objective is to achieve safe, comfortable, and time-efficient automated driving in a complex urban environment while engaging with VRUs such as pedestrians and cyclists.
Fig. 7.4 shows the structure of the motion planner that is based on Topology-driven Model Predictive Control (T-MPC). The guidance planner computes several trajectories that each pass obstacles in distinct ways. In parallel, each local planner optimizes one of the guidance trajectories, ensuring that it is dynamically feasible and collision-free. The lowest-cost feasible trajectory is passed to the vehicle control module.
In the local planners, multi-modal uncertainty associated with the predicted motion of obstacles is incorporated by using Safe Horizon Model Predictive Control (SH-MPC). SH-MPC bounds the probability of collision with all obstacles over the duration of the planned trajectory by a risk ϵ. It solves the following trajectory optimization, online:

$$\begin{aligned}
\min_{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}} \quad & \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) && \text{(7.1a)}\\
\text{s.t.} \quad & \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} && \text{(7.1b)}\\
& \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\dots,N-1 && \text{(7.1c)}\\
& \boldsymbol{p}_k^c \in \mathcal{P}_k^c, \quad \forall k, \forall c. && \text{(7.1d)}
\end{aligned}$$

The objective J includes contouring and lag costs J_c and J_l, respectively, that track the reference path [5] (constructed from lanelet information), a path preview cost J_p [169] to align the vehicle with the reference path at the end of the horizon, a term $||v - v_{\text{ref}}||_2^2$ to track a reference velocity, and penalties $||a||_2^2$ and $||\omega||_2^2$ on the commanded input and steering rate, respectively. Constraints (7.1b) and (7.1c) constrain the initial state and vehicle dynamics. Probabilistic collision avoidance is ensured by enforcing the position of vehicle disc c at time step k to be in a probabilistic safe polygon $\mathcal{P}_k^c$ (see [157]).
The optimization in Eq. 7.1 is solved using the Forces Pro [143] Sequential Quadratic Programming (SQP) solver at 10 Hz with a horizon of $N = 30$ and an integration step of

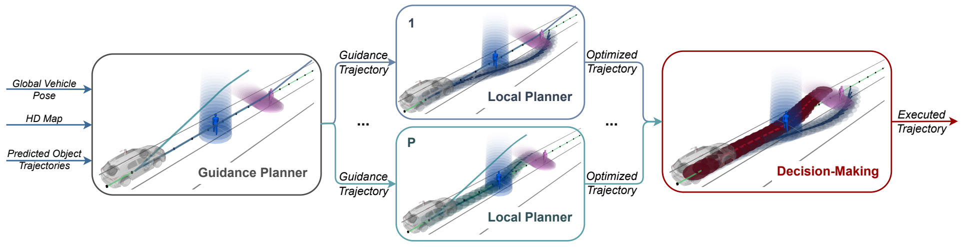


Figure 7.4: An overview of the motion planner (Sec. 2.3). The Guidance Planner plans several guidance trajectories that each pass the dynamic obstacles differently. Each guidance trajectory initializes a Local Planner that, considering the uncertainty associated with the predicted motion of VRUs, locally optimizes the trajectory. The Decision-Making component selects one of the optimized trajectories as output of the motion planner.

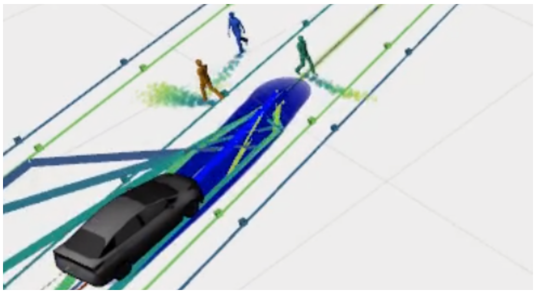


Figure 7.5: Snapshot of the motion planner navigating among virtual VRUs. Sampled VRU predictions and probabilistic safe polytopes are denoted from blue (stage 1) to green (stage 29). Blue circles show the planned trajectory.

0.15 s. The vehicle dynamics follow a kinematic bicycle model and include a physically present delay in the steering input. Fig. 7.5 depicts a snapshot from real-world experiments where the motion planner safely evades virtual pedestrians.
The output trajectory is tracked by the vehicle controller that uses Model Predictive Contouring Control (MPCC), with accurate vehicle dynamics. This controller computes the vehicle control commands at a high rate.

3. DISCUSSION

Several aspects make the transition from theory to practice challenging.

Vehicle Model Usually a simple kinematic vehicle model is assumed for planning to keep computation times tractable. In practice, these models may be insufficient. For example when sensing and actuation delays are not modeled, when the order of the input (e.g., acceleration) is not high enough to achieve smooth behavior (e.g., to impose jerk limits) or when dynamics are ignored (e.g., friction).

Uncertainty Another consideration is that localization, perception, prediction and actuation are all subject to uncertainty that the planner needs to be robust to. Perception and prediction uncertainty were considered through scenario-based planning but a general approach to estimate and account for all uncertainties is still missing.

Horizon Finally, an important practical tuning parameter of MPC is the time horizon (i.e., how far the planner looks ahead in time). On one hand, a longer time horizon can lead to better navigation as long as the underlying prediction models are accurate. On the other hand, it can also lead to planner failure if the growth of uncertainty does not allow for a safe enough plan or, if the computation times become excessive.

All of these aspects can in practice lead to collisions or poor planning performance. Several theoretical possibilities for resolving these issues are discussed in Sec. 8.

8

Conclusions and Future Work

This chapter summarizes the main conclusions of this dissertation and presents topics for future work.

1. Conclusions

The first goal of this dissertation was to accurately incorporate generic uncertainties associated with the predicted motion of dynamic objects. The second goal was to resolve the limitations of optimization-based planners, namely local optimality and infeasibility, with the intention of making these planners more robust and time-efficient in dynamic environments. This dissertation achieved these goals as follows.

Scenario-Based Planning

Chapter 3 presented Scenario-based Model Predictive Contouring Control (S-MPCC) that incorporated the non-Gaussian uncertainty associated with dynamic obstacles in a real-time capable planning framework. The main contribution of this work related to the probability distributions that could describe the motion of dynamic obstacles. Previous work required this distribution to be Gaussian (i.e., uni-modal) and therefore could not account for multiple plausible trajectories that a human may have. In addition, Gaussian distributions could not capture their local motion either, since humans generally follow nonlinear dynamics. This dissertation proposed to evaluate the probability of collision using sampling, which made the planner agnostic to the underlying distribution. Contrary to naive Monte-Carlo sampling methods, which are computationally inefficient when used for planning, the proposed planner used scenario optimization. This scenario optimization sampled possible future positions of the dynamic obstacles (i.e., scenarios) from the distribution that predicted their motion. It then planned a trajectory that avoided collision with all scenarios. By ensuring that the planned trajectory safely avoided the scenarios, the overall probability of collision with all obstacles in each separate time step could be theoretically certified. The feasibility of the trajectory optimization was improved through scenario removal, which could discard outlier samples.

The proposed planner was validated in simulation on a mobile robot navigating among pedestrians that followed Gaussian motion. S-MPCC was compared against baseline methods Local Model Predictive Contouring Control (LMPCC) [27] and Collision Avoidance with Deep Reinforcement Learning (CADRL) [24]. It was shown to compute a safer planning input than the baselines, remaining below the risk specification. At the same time, it reached the goal 6%, 5% and 15% faster than the fastest baseline with 2, 4 and 6 pedestrians respectively. The method was further validated to remain safe when pedestrians followed truncated Gaussian motion. Finally, real-world experiments qualitatively demonstrated the real-time safe navigation capabilities of S-MPCC among humans. While S-MPCC planned probabilistic safe trajectories, it was conservative in the sense that its actual collision probability was only a small fraction (less than 1%) of the specified collision probability. This led to cautious motion that progressed slower than its safety specification required. This conservatism largely originated from the removal of scenarios where the theory required many additional samples to ensure safety. Additionally, S-MPCC constrained the probability of collision in each time step, thereby constraining the probability of collision over the duration of its trajectory conservatively.

Key limitations of S-MPCC were resolved in Chapter 4 that introduced Safe Horizon MPC (SH-MPC). Existing work constrained the marginal probability of collision per time step and/or per obstacle. This ignored two correlation effects. First, if the robot would collide at step k, then collisions after time step k do not contribute to the risk. Second, collisions with more than one obstacle are as unsafe as collisions with one obstacle. To guarantee probabilistic safety, existing methods therefore needed to divide their risk specification by the number of steps in the horizon and the number of obstacles, which significantly increased their conservatism. This dissertation proposed SH-MPC that explicitly constrained the joint probability of collision with all obstacles over the duration of the planned trajectory. SH-MPC followed a similar approach to S-MPCC, but where each scenario represented a trajectory for all obstacles over the time horizon. The conservatism of S-MPCC was reduced through a specialized optimization procedure that tracked those scenarios that impacted the planned trajectory. SH-MPC was compared in simulation to S-MPCC and baseline method Chance Constrained Model Predictive Control (CC-MPC) [84]. The results compared time-efficiency (time to reach the goal) and conservatism (the achieved over the specified collision probability). With four pedestrians and under Gaussian uncertainty SH-MPC was 16% less conservative and 10% faster than S-MPCC. CC-MPC accurately bounds the marginal probability of collision but requires a Gaussian distribution. In the same environment, SH-MPC was 3.5% *more* conservative and 15% *slower* than CC-MPC (where obstacle marginalization was ignored). In the same environment with eight pedestrians and accounting for the obstacle marginalization, SH-MPC was 15% *less* conservative and 13% *faster* than CC-MPC. Similarly, with pedestrian motion following a Gaussian Mixture Model (GMM), SH-MPC was 17% *less* conservative and 13% *faster* than CC-MPC, while CC-MPC additionally took four times longer to compute trajectories. In summary, SH-MPC led to improved navigation performance in crowded environments and enabled safe probabilistic motion planning for any underlying distribution of dynamic obstacle motion. SH-MPC was further demonstrated in the real world on a mobile robot navigating safely among three pedestrians. SH-MPC formally allowed the distribution that predicted obstacle motion, to include interaction between obstacles. These interactions could, for instance, model how humans avoid each other. Interaction between the obstacles and the robot could not yet be handled. Several remaining limitations of the planner were outlined. Decreasing the specified risk of SH-MPC, increases its computation times as more scenarios need to be considered. This makes it difficult to deploy in real-time for risk levels below $5 \cdot 10^{-3}$. It also accounts only for open-loop behavior, ignoring the availability of new information during its planned motion. Finally, its safety guarantees are tied to the predicted probability distribution, which may be inaccurate.

To resolve the last mentioned limitation, Chapter 5 introduced Partitioned Scenario Replay (PSR), a joint data-based framework for prediction and planning. Its main contribution was to provide a real-world safety guarantee by relying on scenario-based planning with real-world data samples. This dissertation showed that observed human trajectories could be replayed to predict human motion during planning. These observed trajectories were partitioned based on the context in which they were observed, as this influences human motion. By using real-world data, a safety guarantee could be provided on the computed trajectories. PSR predictions were validated against state-of-the-art pedestrian prediction algorithms on the ETH/UCY dataset [110]. Using only velocity and acceleration information (no contextual clues), it had a 94% higher Final Displacement Error (FDE) and 138% higher Average Displacement Error (ADE) than the state-of-the-art [121]. However, state-of-the-art methods can provide in the order of 10 samples in real-time. Therefore, the distribution could not be used effectively during planning, even if it was learned accurately. PSR predictions consisted of a look-up operation that enabled large numbers of samples to be drawn instantaneously. Additionally, since the data only needed to be partitioned, it could be trained continuously, online, where it could improve its prediction and planning performance while driving autonomously. This online application was validated on a mobile robot navigating among two pedestrians and resulted in safe navigation, while the framework was not given any prior model of pedestrian behavior. The PSR predictions may still be improved by providing more contextual information, for example, by training an autoencoder to parse complex scene information.

Lastly, SH-MPC was validated on a self-driving vehicle navigating among virtual pedestrians, where predictions followed a Gaussian distribution. The planner resulted in smooth and safe trajectories that avoided the pedestrians in practice, demonstrating the capabilities of the planner for real-world applications.

Topology-Driven Planning

The proposed scenario-based planners S-MPCC and SH-MPC as well as CC-MPC [84], LMPCC [27] and many other local planners rely on a single optimized trajectory to remain safe. In practice, this often leads to infeasibility or poor quality trajectories. Chapter 6 presented Topology-driven Model Predictive Control (T-MPC) that enabled existing local optimization-based planners to compute multiple distinct trajectories in parallel. Similar methods existed for 2D and 3D navigation among static obstacles and for 2D dynamic structured environments. However, the first did not account for dynamic

obstacles, while the second relied on structure in the environment. Both were therefore not useable for mobile robots and self-driving vehicles in urban environments. The main contributions of T-MPC were two-fold. First, a high-level global planner was proposed for 2D unstructured dynamic environments that computed topologically distinct guidance trajectories, in the sense that each trajectory avoided the obstacles in a different way. Second, these guidance trajectories were each passed to a dedicated local optimization-based planner that used the guidance trajectory as its initial guess. It was shown that with this initialization alone, the local planners could still change their passing behavior through optimization. Therefore, homotopy constraints were introduced that linearized the collision avoidance constraints with respect to their guidance trajectory, ensuring that the passing behavior of all optimized trajectories was distinct. Ultimately, the lowest-cost trajectory was executed, discounting the trajectory that was followed in the last planner iteration. T-MPC was validated in simulation against four deterministic baselines: LMPCC [27], guided MPC planner Time Elastic Band (TEB) [70], motion primitives planner [17] and previous conference work [152]. Motion primitives performed worst. In the most crowded environment, T-MPC was 3% safer and faster than LMPCC, but more importantly, its time to reach the goal varied 33% less (i.e., its behavior was more consistent). TEB was 2% safer than T-MPC in one out of three environments, but T-MPC was 9% faster and varied 58% less in the most crowded environment. Additionally, T-MPC was shown to plan smoother trajectories. Under Gaussian uncertainty, T-MPC was deployed with CC-MPC [84] comparing it against regular CC-MPC. T-MPC was 8% safer and 11% faster (varying 65% less). T-MPC (with CC-MPC) was compared in the real world against CC-MPC, navigating for five minutes among five pedestrians. While unaware of the active planner, participants unanimously favored T-MPC as it gave them more space and behaved more predictably.
As T-MPC computed more than one trajectory, it was less sensitive to errors in optimization, such as infeasibility and excessive computation times. T-MPC was infeasible about a quarter as often as LMPCC, which improved all performance metrics. Because redundancy and reduced computation times are key for real-world applications, parallelization may be the key to safely and efficiently deploying optimization-based planners in practice.

Main Contributions

This dissertation has provided two main contributions:

1. **This dissertation showed how multi-modal uncertainties associated with the motion predictions of dynamic obstacles could be addressed in optimization-based planning and how the joint probability of collision, rather than its marginalization could be constrained in real-time.** It established scenario-based planning as a viable alternative to conventional Monte-Carlo methods for evaluating the probability of collision through sampling. These contributions lead to less conservative and more informative motion planning in dynamic uncertain environments.

2. **The proposed parallelization framework with global guidance improved the performance and reliability of optimization-based planners.** Its main advantage

was its applicability: it could directly be applied to existing optimization-based planners and did not rely on a structured environment, making it suitable for urban driving and mobile robotic applications.

2. Future Work

The developed methods proposed a basis that future work can build on to develop automated navigation systems for real-world applications. The following section discusses key remaining challenges in this direction, indicating promising recent developments and potential opportunities for future work.

Reducing Excessive Conservatism in Planning under Uncertainty The results of this dissertation showed that the proposed methods remain conservative with respect to the specified risk. This conservatism can be greatly reduced by running several scenario-based planners in parallel with different risk specifications [9] or with varying slack penalties [170]. The local optimality of the planners can also be overcome through a global guidance strategy, such as the proposed T-MPC framework. However, global guidance planners that can efficiently deal with generic uncertainties (e.g., multi-modal) do not yet exist. Furthermore, when using a model to predict scenarios for the obstacles, the safety certification is tied to the accuracy of the model. Distributionally robust methods, such as [78], [104], account for errors in the predicted distribution and may improve planner safety.

Closed Loop Planning under Uncertainty More generally, probabilistic planners still have to cope with the uncertainty increasing over time. For long time horizons or crowded environments, their plans can be too cautious and occasionally no safe solution can be found. This is in part a fundamental limitation since the future is uncertain. One way to reduce the conservatism in the tail of the trajectory is to assume that specific information on obstacle behavior will be available at a point later in time. Contingency-MPC [32], [171] assumes that the mode of obstacle behavior is only uncertain until an intermediate time (known as the branching time). The planner jointly plans until this time but branches its plans afterwards. The branched plans leverage the information that an obstacle can only follow one mode and are much less conservative than probabilistic methods that evade all modes. These "closed loop" methods are distinct from previously considered "open loop" methods, as they take into account that the planner can make a better decision later. Another notable variant of these approaches is Branch-MPC [34] which branches at several points in time. Although these closed loop methods are promising, there are still severe challenges, including accurate prediction of the modes and the associated uncertainty, scaling to multiple obstacles and determining the branching time.

Interaction-Aware Planning Another limitation of both probabilistic and deterministic planners is that they often assume obstacle behavior to be independent of the action that the robot takes, ignoring interaction. Although this makes it possible to solve the planning problem in real-time, its solution is conservatively avoiding agents, that may themselves try to avoid the robot. There are several ways to include interaction in planning, but all with their difficulties. Social planning methods learn navigation decisions from humans and may solve higher-level interaction problems. GO-MPC learned sub-goals for the MPC from simulations. SHINE [10] built on T-MPC, learning the topology class that humans follow in interactions from a pedestrian dataset. These high-level representations of social behavior were shown to improve social navigation. Still, the underlying MPC relied on non-interactive human motion predictions. Interaction can be accounted for by jointly planning for the robot and the humans, possibly accounting for partial observability, given that agents may not observe all other agents. Most approaches, such as MPC, run into the curse of dimensionality for these problems as the number of states grows with the number of obstacles. One potential way to incorporate interaction is to first resolve the topology of the interaction (similarly to [66]), possibly with a higher-dimensional version of T-MPC's guidance planner. Then, the joint planning problem can be solved by local planners each initialized in a separate interaction class. If the guidance trajectories are sufficiently close to a local optimum, the local planners may converge fast to the solution, making the planner interactive in real-time. Another strategy is to rely on sampling-based MPC solvers, e.g., MPPI [53]. These approaches do not rely on the gradient and the dimension therefore only increases the computation of the roll-outs and evaluation, which scales favorably. Currently, these methods are still local, although several recent works are trying to address this limitation [54], [55]. Applying MPPI within the T-MPC framework may be an interesting avenue to resolve its locality. Another potential future work is to account for non-Gaussian uncertainty in MPPI, which is currently challenging as the probability of collision must be evaluated separately for each sampled trajectory. Nevertheless, sampling-based MPC may have desirable properties for tackling complex problems compared to gradient-based MPC.

Reducing Infeasibility One of the weaknesses of optimization-based planners is that they can fail to produce an action (i.e., when the optimization is infeasible). For example, if a potential collision is detected in the near future that may be incorrectly predicted optimization-based planners may fail to take a safe action in the current step. In contrast, other methods such as Deep Reinforcement Learning methods always produce an action (although it may be unsafe) which in some cases leads to better navigation performance without resulting in collisions. Optimization-based planners could become infeasible less often by relaxing constraints in order of priority [60], [61], making these planners much more robust and allowing them to handle faults. This idea could be pursued for both gradient-based and sampling-based MPC.

Other Applications While this dissertation considered motion planning applications in 2D (flat world) for a single mobile robot, the same methods could be applied in other applications. For instance, for motion planning in 3D spaces. The theory developed for the scenario-based planners holds for this case but the pruning methods that enabled real-time computation in 2D do not trivially extend to 3D. Future work may explore whether another pruning strategy could be applicable in this case. Topology-driven MPC is in principle applicable in 3D dynamic environments (planning in 4D including time)

as homology classes are computable in 4D [7]. These classes behave differently however in 3D. For example, any trajectory that passes a compact obstacle is in the same class, contrary to the 2D case. Alternative topological measures may be of interest for this case, e.g., Universal Visibility Deformation (UVD) [44] and could be applied in the proposed framework to parallelize 3D motion planning in dynamic environments.
The developed methods additionally could be applied to improve the reliability and performance of distributed multi-robot planners. While these systems can use communication to improve their performance, they cannot rely on it to guarantee safety due to delays and drop-outs. Scenario-based planners can account for the lack or presence of communication by representing the associated uncertainty in the other agents' predictions. Another application in this setting is to consider the joint risk of multiple agents. The risk in this setting raises questions on fairness and the responsibility that each agent should take to reduce the risk [172].
Finally, topology-driven planning could have a significant impact on multi-robot planning. By sampling for all agents jointly (i.e., one sample denotes the positions of all agents), low-dimensional topology classes could be used to identify distinct sets of trajectories. A similar approach was explored in [66] using topological braids. Guidance may greatly reduce the computational demand for multi-robot planning thanks to the initial guess that it provides. The proposed parallelization may additionally improve the safety and performance of the resulting solutions.
To conclude this dissertation, state-of-the-art motion planning techniques are becoming advanced enough to be deployed safely and efficiently in applications such as mobile robot planning in non-crowded environments. More crowded and unstructured applications such as automated driving in urban environments, which were the focus of this thesis, remain challenging. More advancements are necessary to endow motion planners with the reasoning and interactive capabilities that are required to seamlessly navigate human environments.

Bibliography

[1] M. Simon, *Inside the Amazon Warehouse Where Humans and Machines Become One*. 2019. [Online]. Available: https://www.wired.com/story/amazon-warehouse-robots/.

[2] J. Walker, *The Self-Driving Car Timeline - Predictions from the Top 11 Global Automakers*. 2019. [Online]. Available: https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/.

[3] MI News Network, *7 Major Developments in Autonomous Shipping in 2018*. 2018. [Online]. Available: https://www.marineinsight.com/know-more/7-major-developments-in-autonomous-shipping-in-2018/.

[4] J. Nocedal and S. J. Wright, *Numerical optimization* (Springer series in operations research), 2nd ed. New York: Springer, 2006.

[5] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model", *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 2994–3008, Sep. 2018.

[6] S. Peng, "Chance constrained problem and its applications", Ph.D. dissertation, Université Paris Saclay (COmUE) ; Xi'an Jiaotong University, Jun. 2019. [Online]. Available: https://theses.hal.science/tel-02303045.

[7] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning", *Auton. Robots*, vol. 33, no. 3, pp. 273–290, Oct. 2012.

[8] A. Tsolakis, D. Benders, O. de Groot, R. R. Negenborn, V. Reppa, and L. Ferranti, "COLREGs-aware Trajectory Optimization for Autonomous Surface Vessels", IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles, vol. 55, no. 31, pp. 269–274, Jan. 2022.

[9] K. A. Mustafa, O. de Groot, X. Wang, J. Kober, and J. Alonso-Mora, "Probabilistic Risk Assessment for Chance-Constrained Collision Avoidance in Uncertain Dynamic Environments", in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 3628–3634. [Online]. Available: https://ieeexplore.ieee.org/document/10161490.

[10] D. Martinez-Baselga, O. de Groot, L. Knoedler, L. Riazuelo, J. Alonso-Mora, and L. Montano, *SHINE: Social Homology Identification for Navigation in Crowded Environments*, Apr. 2024. [Online]. Available: http://arxiv.org/abs/2404.16705.

[12] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window Approach to Collision Avoidance", *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[13] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation", in *IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1928–1935.

[14] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots", in *Distributed Autonomous Robotic Systems*, vol. 83, Berlin, Heidelberg, 2013, pp. 203–216. [Online]. Available: `https://link.springer.com/10.1007/978-3-642-32723-0_15`.

[15] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, Mar. 1986.

[16] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments", *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[17] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame", in *IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 987–993.

[18] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios", in *IEEE Intell. Transp. Syst. Conf.*, Oct. 2019, pp. 3149–3154.

[19] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[20] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces", *IEEE Robot. Autom. Lett.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[21] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-Time Motion Planning With Applications to Autonomous Urban Driving", *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[22] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning", *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, Jun. 2016.

[23] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation", *Int. J. Robot. Res.*, vol. 43, no. 1, pp. 3–33, Jan. 2024.

[24] M. Everett, Y. F. Chen, and J. P. How, "Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning", in *IEEE Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 3052–3059.

[25] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, "Robot navigation in a crowd by integrating deep reinforcement learning and online planning", *Applied Intelligence*, vol. 52, no. 13, pp. 15 600–15 616, Oct. 2022.

[26] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars", *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[27] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments", *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[28] L. Ferranti, B. Brito, E. Pool, *et al.*, "SafeVRU: A Research Platform for the Interaction of Self-Driving Vehicles with Vulnerable Road Users", in *IEEE Intell. Veh. Symp.*, 2019, pp. 1660–1666.

[29] Lyons, L. and Ferranti, L., "Curvature-Aware Model Predictive Contouring Control", in *IEEE Int. Conf. Robot. Autom.*, 2023, pp. 3204–3210.

[30] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design", in *IEEE Intell. Veh. Symp.*, Jun. 2015, pp. 1094–1099.

[31] V. Cardoso, J. Oliveira, T. Teixeira, *et al.*, "A Model-Predictive Motion Planner for the IARA autonomous car", in *IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 225–230. [Online]. Available: http://ieeexplore.ieee.org/document/7989028/.

[32] J. P. Alsterda and J. C. Gerdes, *Contingency Model Predictive Control for Linear Time-Varying Systems*, Feb. 2021. [Online]. Available: http://arxiv.org/abs/2102.12045.

[33] Y. Chen, S. Veer, P. Karkus, and M. Pavone, "Interactive Joint Planning for Autonomous Vehicles", *IEEE Robot. Autom. Lett.*, vol. 9, no. 2, pp. 987–994, Feb. 2024.

[34] Y. Chen, U. Rosolia, W. Ubellacker, N. Csomay-Shanklin, and A. D. Ames, "Interactive multi-modal motion planning with Branch Model Predictive Control", *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, Sep. 2022.

[35] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, *Motion Planning around Obstacles with Convex Optimization*, May 2022. [Online]. Available: http://arxiv.org/abs/2205.04422.

[36] T. Marcucci, J. Umenberger, P. A. Parrilo, and R. Tedrake, *Shortest Paths in Graphs of Convex Sets*, Jul. 2023. [Online]. Available: http://arxiv.org/abs/2101.11565.

[37] V. K. Adajania, A. Sharma, A. Gupta, H. Masnavi, K. M. Krishna, and A. K. Singh, "Multi-Modal Model Predictive Control Through Batch Non-Holonomic Trajectory Optimization: Application to Highway Driving", *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4220–4227, Apr. 2022.

[38] F. Eiras, M. Hawasly, S. V. Albrecht, and S. Ramamoorthy, "A Two-Stage Optimization-Based Motion Planner for Safe Urban Driving", *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 822–834, Apr. 2022.

[39] W. Ding, L. Zhang, J. Chen, and S. Shen, "EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments", *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 1118–1138, Apr. 2022.

[40] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming", *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1101–1115, Oct. 2015.

[41] V. Z. Patterson, F. E. Lewis, and J. C. Gerdes, "Optimal Decision Making for Automated Vehicles Using Homotopy Generation and Nonlinear Model Predictive Control", in *IEEE Intell. Veh. Symp.*, Jul. 2021, pp. 1045–1050.

[42] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model Predictive Trajectory Planning for Automated Driving", *IEEE Trans. Intell. Veh.*, vol. 4, no. 1, pp. 24–38, Mar. 2019.

[43] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles", in *IEEE Conf. Decis. Control.*, Dec. 2017, pp. 2126–2133.

[44] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust Real-time UAV Replanning Using Guided Gradient-based Optimization and Topological Paths", in *IEEE Int. Conf. Robot. Autom.*, May 2020, pp. 1208–1214.

[45] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies", *Robot. Auton. Syst.*, vol. 88, pp. 142–153, Feb. 2017.

[46] R. Penicka and D. Scaramuzza, "Minimum-Time Quadrotor Waypoint Flight in Cluttered Environments", arXiv, Tech. Rep. arXiv:2202.03947, Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.03947.

[47] C. Cao, P. Trautman, and S. Iba, "Dynamic Channel: A Planning Framework for Crowd Navigation", in *IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 5551–5557.

[48] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning", *Int. J. Robot. Res.*, vol. 35, no. 11, pp. 1289–1307, Sep. 2016.

[49] C. Mavrogiannis and R. A. Knepper, "Hamiltonian coordination primitives for decentralized multiagent navigation", *Int. J. Robot. Res.*, vol. 40, no. 10-11, pp. 1234–1254, Sep. 2021.

[50] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning", *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, Jan. 2000.

[51] M. A. Berger, "Topological Invariants in Braid Theory", *Letters in Mathematical Physics*, vol. 55, no. 3, pp. 181–192, Mar. 2001.

[52] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, and S. S. Srinivasa, "Winding Through: Crowd Navigation via Topological Invariance", *IEEE Robot. Autom. Lett.*, vol. 8, no. 1, pp. 121–128, Jan. 2023.

[53] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving", *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[54] E. Trevisan and J. Alonso-Mora, *Biased-MPPI: Informing Sampling-Based Model Predictive Control by Fusing Ancillary Controllers*, Jan. 2024. [Online]. Available: http://arxiv.org/abs/2401.09241.

[55] F. Rastgar, H. Masnavi, B. Sharma, A. Aabloo, J. Swevers, and A. K. Singh, "PRIEST: Projection Guided Sampling-Based Optimization For Autonomous Navigation", *IEEE Robot. Autom. Lett.*, pp. 1–8, 2024.

[56] C. Pek and M. Althoff, “Fail-Safe Motion Planning for Online Verification of Autonomous Vehicles Using Convex Optimization”, *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 798–814, Jun. 2021.

[57] J. P. Alsterda, M. Brown, and J. C. Gerdes, “Contingency Model Predictive Control for Automated Vehicles”, in *IEEE Am. Control Conf.*, Philadelphia, PA, USA, Jul. 2019, pp. 717–722.

[58] D. Saccani, L. Cecchin, and L. Fagiano, “Multitrajectory Model Predictive Control for Safe UAV Navigation in an Unknown Environment”, *IEEE Trans. Control Syst. Technol.*, pp. 1–16, 2022.

[59] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, “FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments”, *arXiv:2001.04420 [cs]*, Jan. 2020. [Online]. Available: http://arxiv.org/abs/2001.04420.

[60] J. Vada, O. Slupphaug, T. A. Johansen, and B. A. Foss, “Linear MPC with optimal prioritized infeasibility handling: Application, computational issues and stability”, *Automatica*, vol. 37, no. 11, pp. 1835–1843, Nov. 2001.

[61] K. P. Wabersich and M. N. Zeilinger, “Predictive Control Barrier Functions: Enhanced Safety Mechanisms for Learning-Based Control”, *IEEE Trans. Autom. Control*, vol. 68, no. 5, pp. 2638–2651, May 2023.

[62] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and Decision-Making for Autonomous Vehicles”, *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, May 2018.

[63] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, “Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data”, *arXiv:2001.03093 [cs]*, Jan. 2021. [Online]. Available: http://arxiv.org/abs/2001.03093.

[64] P. Gupta, D. Isele, D. Lee, and S. Bae, *Interaction-Aware Trajectory Planning for Autonomous Vehicles with Analytic Integration of Neural Networks into Model Predictive Control*, Mar. 2023. [Online]. Available: http://arxiv.org/abs/2301.05393.

[65] L. Streichenberg, E. Trevisan, J. J. Chung, R. Siegwart, and J. Alonso-Mora, “Multi-Agent Path Integral Control for Interaction-Aware Motion Planning in Urban Canals”, in *IEEE Int. Conf. Robot. Autom.*, May 2023, pp. 1379–1385.

[66] C. I. Mavrogiannis and R. A. Knepper, “Multi-agent path topology in support of socially competent navigation planning”, *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 338–356, Mar. 2019.

[67] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, “Hierarchical Game-Theoretic Planning for Autonomous Vehicles”, in *IEEE Int. Conf. Robot. Autom.*, May 2019, pp. 9590–9596.

[68] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, “Where to go Next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments”, *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, Jul. 2021.

[69] L. Knoedler, B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, “Learning a Guidance Policy from Humans for Social Navigation”, [Online]. Available: https://harmony-eu.org/wp-content/uploads/2022/06/knoedler2022icra_ws.pdf.

[70] C. Rösmann, M. Oeljeklaus, F. Hoffmann, and T. Bertram, “Online trajectory prediction and planning for social robot navigation”, in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2017, pp. 1255–1260.

[71] N. Tsoi, A. Xiang, P. Yu, *et al.*, “SEAN 2.0: Formalizing and Generating Social Situations for Robot Navigation”, *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11 047–11 054, Oct. 2022.

[72] L. Kastner, T. Bhuiyan, T. A. Le, *et al.*, “Arena-Bench: A Benchmarking Suite for Obstacle Avoidance Approaches in Highly Dynamic Environments”, *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 9477–9484, Oct. 2022.

[73] A. Biswas, A. Wang, G. Silvera, A. Steinfeld, and H. Admoni, “SocNavBench: A Grounded Simulation Testing Framework for Evaluating Social Navigation”, *ACM Transactions on Human-Robot Interaction*, vol. 11, no. 3, pp. 1–24, Sep. 2022.

[74] A. Francis, C. Pérez-D’Arpino, C. Li, *et al.*, *Principles and Guidelines for Evaluating Social Robot Navigation Algorithms*, Sep. 2023. [Online]. Available: http://arxiv.org/abs/2306.16740.

[75] E. Todorov and Weiwei Li, “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems”, in *American Control Conference*, Jun. 2005, 300–306 vol. 1.

[76] A. Majumdar and M. Pavone, “How Should a Robot Assess Risk? Towards an Axiomatic Theory of Risk in Robotics”, *arXiv:1710.11040 [cs, math]*, Nov. 2017. [Online]. Available: http://arxiv.org/abs/1710.11040.

[77] K. Ren, H. Ahn, and M. Kamgarpour, “Chance-Constrained Trajectory Planning With Multimodal Environmental Uncertainty”, *IEEE Control Systems Letters*, vol. 7, pp. 13–18, 2023.

[78] A. Hakobyan and I. Yang, “Wasserstein Distributionally Robust Motion Control for Collision Avoidance Using Conditional Value-at-Risk”, *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 939–957, Apr. 2022.

[79] J. Yin, Z. Zhang, and P. Tsiotras, “Risk-Aware Model Predictive Path Integral Control Using Conditional Value-at-Risk”, in *IEEE Int. Conf. Robot. Autom.*, London, United Kingdom: IEEE, May 2023, pp. 7937–7943.

[80] A. Ben-Tal and A. Nemirovski, “Robust Convex Optimization”, *Math. Oper. Res.*, vol. 23, no. 4, pp. 769–805, Nov. 1998.

[81] M. Althoff and J. M. Dolan, “Online Verification of Automated Road Vehicles Using Reachability Analysis”, *IEEE Trans. Robot.*, vol. 30, no. 4, pp. 903–918, Aug. 2014.

[82] A. Mesbah, “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research”, *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, Dec. 2016.

[83] B. Kouvaritakis and M. Cannon, *Model Predictive Control: Classical, Robust and Stochastic* (Advanced Textbooks in Control and Signal Processing). Springer International Publishing, 2016.

[84] H. Zhu and J. Alonso-Mora, "Chance-Constrained Collision Avoidance for MAVs in Dynamic Environments", *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 776–783, Apr. 2019.

[85] J. F. Fisac, A. Bajcsy, S. L. Herbert, *et al.*, "Probabilistically Safe Robot Planning with Confidence-Based Human Predictions", *RSS*, Jun. 2018.

[86] L. Blackmore, Hui Li, and B. Williams, "A probabilistic approach to optimal robust path planning with obstacles", in *Proc. Amer. Control Conf.*, Jun. 2006, pp. 2831–2837.

[87] A. Wang, X. Huang, A. Jasour, and B. Williams, "Fast Risk Assessment for Autonomous Vehicles Using Learned Models of Agent Futures", in *RSS*, Jul. 2020.

[88] O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, "Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments", *IEEE Robotics and Automation Letters (RA-L)*, pp. 5389–5396, 2021.

[89] L. Janson, E. Schmerling, and M. Pavone, "Monte Carlo Motion Planning for Robot Trajectory Optimization Under Uncertainty", in *Robotics Research: Volume 2*, ser. Springer Proceedings in Advanced Robotics, Springer International Publishing, 2018, pp. 343–361.

[90] Masahiro Ono and B. C. Williams, "Iterative Risk Allocation: A new approach to robust Model Predictive Control with a joint chance constraint", in *47th IEEE Conf. on Decision and Control*, Dec. 2008, pp. 3427–3432.

[91] B. Luders, M. Kothari, and J. How, "Chance Constrained RRT for Probabilistic Robustness to Environmental Uncertainty", in *Proc. AIAA Guid., Navigat. Control Conf.,*, Aug. 2010.

[92] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns", *Auton. Robots*, vol. 35, no. 1, pp. 51–76, Jul. 2013.

[93] A. Wang, A. Jasour, and B. C. Williams, "Non-Gaussian Chance-Constrained Trajectory Planning for Autonomous Vehicles Under Agent Uncertainty", *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6041–6048, Oct. 2020.

[94] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information", *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 895–913, Jun. 2011.

[95] S. Patil, J. v. d. Berg, and R. Alterovitz, "Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty", in *IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 3238–3244.

[96] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams, "A Probabilistic Particle-Control Approximation of Chance-Constrained Stochastic Predictive Control", *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 502–517, Jun. 2010.

[97] E. Schmerling and M. Pavone, “Evaluating Trajectory Collision Probability through Adaptive Importance Sampling for Safe Motion Planning”, in *RSS*, Jul. 2017.

[98] M. C. Campi, S. Garatti, and F. A. Ramponi, “A General Scenario Theory for Nonconvex Optimization and Decision Making”, *IEEE Trans. Automat. Contr.*, vol. 63, no. 12, pp. 4067–4078, Dec. 2018.

[99] G. Calafiore and M. Campi, “The scenario approach to robust control design”, *IEEE Trans. Automat. Contr.*, vol. 51, no. 5, pp. 742–753, May 2006.

[100] M. C. Campi and S. Garatti, “The Exact Feasibility of Randomized Solutions of Uncertain Convex Programs”, *SIAM J. Optim.*, vol. 19, no. 3, pp. 1211–1230, Jan. 2008.

[101] M. C. Campi and S. Garatti, “A Sampling-and-Discarding Approach to Chance-Constrained Optimization: Feasibility and Optimality”, *J. Optim. Theory Appl.*, vol. 148, no. 2, pp. 257–280, Feb. 2011.

[102] G. Schildbach, L. Fagiano, and M. Morari, “Randomized Solutions to Convex Programs with Multiple Chance Constraints”, *SIAM J. Optim.*, vol. 23, no. 4, pp. 2479–2501, Jan. 2013.

[103] G. Cesari, G. Schildbach, A. Carvalho, and F. Borrelli, “Scenario Model Predictive Control for Lane Change Assistance and Autonomous Driving on Highways”, *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 3, pp. 23–35, 2017.

[104] P. Lathrop, B. Boardman, and S. Martinez, “Distributionally Safe Path Planning: Wasserstein Safe RRT”, *IEEE Robot. Autom. Lett.*, vol. 7, no. 1, pp. 430–437, Jan. 2022.

[105] N. Kaempchen, K. Weiss, M. Schaefer, and K. C. Dietmayer, “IMM object tracking for high dynamic driving maneuvers”, *IEEE Intell. Vehicles Symp.*, pp. 825–830, 2004.

[106] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, “Vehicle Trajectory Prediction by Integrating Physics- and Maneuver-Based Approaches Using Interactive Multiple Models”, *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, Jul. 2018.

[107] B. D. Ziebart, N. Ratliff, G. Gallagher, *et al.*, “Planning-based prediction for pedestrians”, in *IEEE Int. Conf. Intell. Robots. Syst.*, Oct. 2009, pp. 3931–3936.

[108] G. Best and R. Fitch, “Bayesian intention inference for trajectory prediction with an unknown goal destination”, in *IEEE Int. Conf. Intell. Robot. Syst.*, Sep. 2015, pp. 5817–5823.

[109] M. N. Finean, L. Petrović, W. Merkt, I. Marković, and I. Havoutis, “Motion planning in dynamic environments using context-aware human trajectory prediction”, *Robotics and Autonomous Systems*, vol. 166, p. 104 450, Aug. 2023.

[110] A. Ess, B. Leibe, and L. Van Gool, “Depth and Appearance for Mobile Scene Analysis”, in *International Conference on Computer Vision*, Oct. 2007, pp. 1–8.

[111] L. Medsker and L. Jain, *Recurrent Neural Networks: Design and Applications*. CRC Press, 2001.

[112] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social LSTM: Human Trajectory Prediction in Crowded Spaces”, in *IEEE CVPR*, Jun. 2016, pp. 961–971.

[113] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, *Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction*, Mar. 2020. [Online]. Available: http://arxiv.org/abs/2002.11927.

[114] B. Ivanovic and M. Pavone, “Injecting Planning-Awareness into Prediction and Detection Evaluation”, in *IEEE Intell. Veh. Symp.*, Jun. 2022, pp. 821–828.

[115] F. Fang, P. Zhang, B. Zhou, K. Qian, and Y. Gan, “Atten-GAN: Pedestrian Trajectory Prediction with GAN Based on Attention Mechanism”, *Cognitive Computation 2022*, vol. 1, pp. 1–10, Jun. 2022.

[116] K. Sohn, H. Lee, and X. Yan, “Learning Structured Output Representation using Deep Conditional Generative Models”, *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[117] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes”, *ICLR*, Dec. 2013.

[118] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, “A Recurrent Latent Variable Model for Sequential Data”, in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[119] B. Brito, H. Zhu, W. Pan, and J. Alonso-mora, “Social-vrnn: One-shot multi-modal trajectory prediction for interacting pedestrians”, in *CoRL*, 2020.

[120] K. Mangalam, Y. An, H. Girase, and J. Malik, “From Goals, Waypoints & Paths To Long Term Human Trajectory Forecasting”, in *ICCV*, IEEE, Oct. 2021, pp. 15 213–15 222.

[121] J. Yue, D. Manocha, and H. Wang, *Human Trajectory Prediction via Neural Social Physics*, Mar. 2023. [Online]. Available: http://arxiv.org/abs/2207.10435.

[122] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, *Generative Adversarial Networks*, Jun. 2014. [Online]. Available: http://arxiv.org/abs/1406.2661.

[123] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks”, in *IEEE CVPR*, Jun. 2018, pp. 2255–2264.

[124] P. Dendorfer, S. Elflein, and L. Leal-Taixe, “MG-GAN: A Multi-Generator Model Preventing Out-of-Distribution Samples in Pedestrian Trajectory Prediction”, in *IEEE ICCV*, Montreal, QC, Canada, Oct. 2021, pp. 13 138–13 147.

[125] R. Cheng, R. M. Murray, and J. W. Burdick, “Limits of Probabilistic Safety Guarantees when Considering Human Uncertainty”, *arXiv:2103.03388 [cs, eess]*, Mar. 2021. [Online]. Available: http://arxiv.org/abs/2103.03388.

[126] K. Mangalam, H. Girase, S. Agarwal, *et al.*, *It Is Not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction*, Jul. 2020. [Online]. Available: http://arxiv.org/abs/2004.02025.

[127] *Papers with Code - ETH/UCY Benchmark (Trajectory Prediction)*. [Online]. Available: https://paperswithcode.com/sota/trajectory-prediction-on-ethucy.

[128] J. Ziegler, P. Bender, M. Schreiber, *et al.*, "Making Bertha Drive—An Autonomous Journey on a Historic Route", *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, 2014.

[129] C. Badue, R. Guidolini, R. V. Carneiro, *et al.*, "Self-driving cars: A survey", *Expert Systems with Applications*, vol. 165, p. 113 816, 2021.

[130] D. Li and J. Hu, "Mitigating Motion Sickness in Automated Vehicles With Frequency-Shaping Approach to Motion Planning", *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7714–7720, Oct. 2021.

[131] T. Dang, J. Desens, U. Franke, D. Gavrila, L. Schäfers, and W. Ziegler, "Steering and evasion assist", in Jan. 2012, pp. 759–782.

[132] F. Kunz, D. Nuss, J. Wiest, *et al.*, "Autonomous driving at Ulm University: A modular, robust, and sensor-independent fusion approach", in *IEEE Intell. Veh. Symp.*, Jun. 2015, pp. 666–673.

[133] B. Montgomery, "Cruise driverless cars pulled off California roads after safety incidents", *The Guardian*, Oct. 2023. [Online]. Available: https://www.theguardian.com/us-news/2023/oct/24/driverless-car-self-driving-california-cruise-gm.

[134] A. Press, "Cruise recalls all self-driving cars after grisly accident and California ban", *The Guardian*, Nov. 2023. [Online]. Available: https://www.theguardian.com/technology/2023/nov/08/cruise-recall-self-driving-cars-gm.

[135] J. Bhuiyan, "My week navigating the awkward teenage years of self-driving cars", *The Guardian*, May 2023. [Online]. Available: https://www.theguardian.com/us-news/2023/may/13/self-driving-cars-waymo-cruise-san-francisco.

[136] J. F. P. Kooij, F. Flohr, E. A. I. Pool, and D. M. Gavrila, "Context-Based Path Prediction for Targets with Switching Dynamics", *Int. J. of Computer Vision*, vol. 127, no. 3, pp. 239–262, Mar. 2019.

[137] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "MultiPath: Multiple Probabilistic Anchor Trajectory Hypotheses for Behavior Prediction", *arXiv*, Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.05449.

[138] N. Deo and M. M. Trivedi, "Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs", in *IEEE Intell. Veh. Symp.*, Jun. 2018, pp. 1179–1184.

[139] M. E. Muller and G. E. P. Box, "A note on the generation of random numbers", *Ann. Math. Stat.*, vol. 29, pp. 610–11, 1958.

[140] L. Martinet, D. Luengo, and J. Miguez, “Efficient Sampling from Truncated Bivariate Gaussians via the Box-Muller Transformation”, *Electronics Letters*, vol. 48, p. 2, 2012.

[141] NaturalPoint, *Optitrack*, 2021. [Online]. Available: https://optitrack.com/.

[142] D. Helbing and P. Molnár, “Social force model for pedestrian dynamics”, *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, May 1995.

[143] A. Domahidi and J. Jerez, *FORCES Professional*. Embotech AG, Jul. 2014. [Online]. Available: https://embotech.com/FORCES-Pro.

[144] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, 2nd ed. The MIT Press, 2011.

[146] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Scenario-Based Motion Planning with Bounded Probability of Collision*, Jun. 2024. [Online]. Available: https://www.youtube.com/watch?v=vyuNMO1giF0.

[147] P. Billingsley, *Probability and measure* (Wiley series in probability and mathematical statistics), 3rd ed. New York: Wiley, 1995.

[148] S. Garatti and M. C. Campi, “Risk and complexity in scenario optimization”, *Math. Program.*, vol. 191, pp. 243–279, Nov. 2019.

[149] A. D. Morgan, *Formal Logic: Or, The Calculus of Inference, Necessary and Probable*. Taylor and Walton, 1847.

[150] O. de Groot, *Jupyter Notebook for Scenario Dimensioning*, Mar. 2024. [Online]. Available: https://colab.research.google.com/drive/1Xye7960CZAlt4gHpoH6HOvok4-YuLFbD?usp=sharing.

[151] F. J. Aragón Artacho, R. Campoy, and M. K. Tam, “The Douglas–Rachford algorithm for convex and nonconvex feasibility problems”, *Math Meth Oper Res*, vol. 91, no. 2, pp. 201–240, Apr. 2020. [Online]. Available: https://doi.org/10.1007/s00186-019-00691-9.

[153] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator”, in *Conf. on Robot Learning*, Oct. 2017, pp. 1–16.

[154] K. Frey, T. Steiner, and J. How, “Collision Probabilities for Continuous-Time Systems Without Sampling”, in *RSS*, Jul. 2020.

[155] H. Sartipizadeh and B. Açikmeşe, “Approximate convex hull based scenario truncation for chance constrained trajectory optimization”, *Automatica*, 2020.

[156] O. de Groot, A. Sridharan, J. Alonso-Mora, and L. Ferranti, “Probabilistic motion planning and prediction via partitioned scenario replay”, in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 7546–7552.

[158] P. Billingsley, *Probability and Measure*, 3rd. 1995.

[159] O. de Groot, *Jupyter Notebook for Scenario Optimization*, Aug. 2023. [Online]. Available: https://colab.research.google.com/drive/1Ur20Oqx4WzUT3rNU-Gm1g0QoZ5ChufR_?usp=sharing.

[160] X. Jin and J. Han, "K-Means Clustering", in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Boston, MA: Springer US, 2010, pp. 563–564. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_425.

[161] J. Levy-Kramer, *K-means-constrained*, Aug. 2023. [Online]. Available: https://github.com/joshlk/k-means-constrained.

[163] O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, *Video Topology-Driven Parallel Trajectory Optimization in Dynamic Environments*, Sep. 2024. [Online]. Available: https://youtu.be/kXUAldQXrNk.

[164] C. Gloor, "Pedsim: Pedestrian crowd simulation", 2016. [Online]. Available: https://github.com/chgloor/pedsim.

[165] C. Rösmann, *ROS Package teb_local_planner*, Nov. 2023. [Online]. Available: https://github.com/rst-tu-dortmund/teb_local_planner.

[166] M. Galassi, J. Davies, J. Theiler, *et al.*, *GNU Scientific Library Reference Manual (3rd Ed.)* Aug. 2019.

[167] S. Kato, S. Tokunaga, Y. Maruyama, *et al.*, "Autoware on board: Enabling autonomous vehicles with embedded systems", in *ICCPS*, 2018, pp. 287–296.

[168] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching", in *IROS*, vol. 3, 2003, pp. 2743–2748.

[169] G. Chen, J. Yao, H. Hu, Z. Gao, L. He, and X. Zheng, "Design and experimental evaluation of an efficient MPC-based lateral motion controller considering path preview for autonomous vehicles", *Control Engineering Practice*, vol. 123, p. 105 164, 2022.

[170] S. Garatti and M. C. Campi, "Scenario Optimization with Constraint Relaxation in a Non-Convex Setup: A Flexible and General Framework for Data-Driven Design", in *IEEE Conf. Decis. Control*, Singapore, Singapore, Dec. 2023, pp. 26–31.

[171] L. Peters, A. Bajcsy, C.-Y. Chiu, *et al.*, *Contingency Games for Multi-Agent Interaction*, Dec. 2023. [Online]. Available: http://arxiv.org/abs/2304.05483.

[172] M. Geisslinger, F. Poszler, and M. Lienkamp, "An ethical trajectory planning algorithm for autonomous vehicles", *Nature Machine Intelligence*, vol. 5, no. 2, pp. 137–144, Feb. 2023.

Curriculum Vitæ

Oscar de Groot received both the B.Sc. degree in electrical engineering, in 2016, and the M.Sc. degree in systems & control, in 2019, from the Delft University of Technology, Delft, The Netherlands. In March 2020, he started a Ph.D. at the department of Cognitive Robotics at the Delft University of Technology. In his Ph.D., he worked on probabilistic safe motion planning algorithms for mobile robots and self-driving vehicles under primary supervision of Dr. Laura Ferranti and Dr. Javier Alonso-Mora. During his Ph.D., he collaborated with the Intelligent Vehicles (IV) group at the Delft University of Technology on motion planning techniques for self-driving vehicles. From March 2024, he is a Postdoctoral researcher at the Intelligent Vehicles group.

His research interests include robot motion planning, model predictive control, scenario optimization, social motion planning and planning for self-driving vehicles.

List of Publications

Publications

1. O. de Groot, B. Brito, L. Ferranti, D. Gavrila, and J. Alonso-Mora, “Scenario-Based Trajectory Optimization in Uncertain Dynamic Environments”, *IEEE Robotics and Automation Letters (RA-L)*, pp. 5389–5396, 2021

2. O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso–Mora, “Globally Guided Trajectory Planning in Dynamic Environments”, in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 10 118–10 124

3. O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, “Scenario-Based Motion Planning with Bounded Probability of Collision”, Jul. 2023, *under review*

4. O. de Groot, L. Ferranti, D. Gavrila, and J. Alonso-Mora, “Topology-Driven Parallel Trajectory Optimization in Dynamic Environments”, Sep. 2024, *accepted for publication in IEEE Transactions on Robotics (T-RO)*

5. O. de Groot, A. Sridharan, J. Alonso-Mora, and L. Ferranti, “Probabilistic motion planning and prediction via partitioned scenario replay”, in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 7546–7552

Co-authored Publications

1. R. J. Perez Dattari, B. F. Ferreira de Brito, O. de Groot, M., J. Kober, and J. Alonso Mora, “Visually-guided motion planning for autonomous driving from interactive demonstrations”, *Engineering Applications of Artificial Intelligence*, vol. 116, 2022

2. A. Tsolakis, D. Benders, O. de Groot, R. R. Negenborn, V. Reppa, and L. Ferranti, “COLREGs-aware Trajectory Optimization for Autonomous Surface Vessels”, IFAC Conference on Control Applications in Marine Systems, Robotics and Vehicles, vol. 55, no. 31, pp. 269–274, Jan. 2022

3. K. A. Mustafa, O. de Groot, X. Wang, J. Kober, and J. Alonso-Mora, “Probabilistic Risk Assessment for Chance-Constrained Collision Avoidance in Uncertain Dynamic Environments”, in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 3628–3634

4. D. Martinez-Baselga, O. de Groot, L. Knoedler, L. Riazuelo, J. Alonso-Mora, and L. Montano, *SHINE: Social Homology Identification for Navigation in Crowded Environments*, Apr. 2024, *under review*

PUBLICATIONS OUTSIDE OF THIS DISSERTATION

1. O. de Groot and T. Keviczky, “Cooperative r-Passivity Based Control for Mechanical Systems”, *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 3476–3481, 2020

2. O. de Groot, L. Valk, and T. Keviczky, “Cooperative Passivity-Based Control of Nonlinear Mechanical Systems”, *MDPI Robotics*, vol. 12, no. 5, p. 142, Oct. 2023