Scenario-based motion planning with bounded probability of collision

de Groot, Oscar; Ferranti, Laura; Gavrila, Dariu M.; Alonso-Mora, Javier

DOI
10.1177/02783649251315203

Publication date
2025

Document Version
Final published version

Published in
International Journal of Robotics Research

Original Article

The International Journal of Robotics Research
2025, Vol. 0(0) 1–19
© The Author(s) 2025
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/02783649251315203
journals.sagepub.com/home/ijr
Sage

Scenario-based motion planning with bounded probability of collision

Oscar de Groot, Laura Ferranti, Dariu M. Gavrila and Javier Alonso-Mora

Abstract
Robots will increasingly operate near humans that introduce uncertainties in the motion planning problem due to their complex nature. Optimization-based planners typically avoid humans through collision avoidance chance constraints. This allows the planner to optimize performance while guaranteeing probabilistic safety. However, existing real-time methods do not consider the actual probability of collision for the planned trajectory but rather its marginalization, that is, the independent collision probabilities for each planning step and/or dynamic obstacle, resulting in conservative trajectories. To address this issue, we introduce a novel real-time capable method termed Safe Horizon MPC that explicitly constrains the joint probability of collision with all obstacles over the duration of the motion plan. This is achieved by reformulating the chance-constrained planning problem using scenario optimization and predictive control. Out of sampled realizations of human motion, we identify which cases affect the optimization. This allows us to certify the planned trajectory in real-time. Our method is less conservative than state-of-the-art approaches, applicable to arbitrary probability distributions of the obstacles' trajectories, computationally tractable and scalable. We demonstrate our proposed approach using a mobile robot and an autonomous vehicle in an environment shared with humans.

Keywords
Motion and path planning, optimization and optimal control, collision avoidance, probability and statistical methods

Received 29 February 2024; Revised 5 November 2024; Accepted 23 December 2024

1. Introduction

Mobile robots can improve our quality of life, from transporting goods in warehouses (Simon, 2019) to helping us commute more efficiently and safely using self-driving vehicles (Walker, 2019). In most applications where robots are currently deployed, the operating domain does not allow the robots to operate near humans or they do so in a conservative manner to simplify the robot navigation task. To deploy mobile robots in real-world environments, such as our cities, they need to be capable of navigating among humans, which remains challenging.

In crowded environments, the robot needs to understand and infer the motion of humans in order to move safely and efficiently. Unfortunately, human behavior is hard to predict and varies per person. In addition, human intentions cannot be explicitly communicated to the robot. This inherently makes the motion prediction of humans nondeterministic. Recent perception methods, such as Kooij et al. (2019), Salzmann et al. (2021), and Yue et al. (2023), infer human intentions, returning probabilistic information concerning humans. These probabilistic predictions need to be considered and exploited by the motion planner.

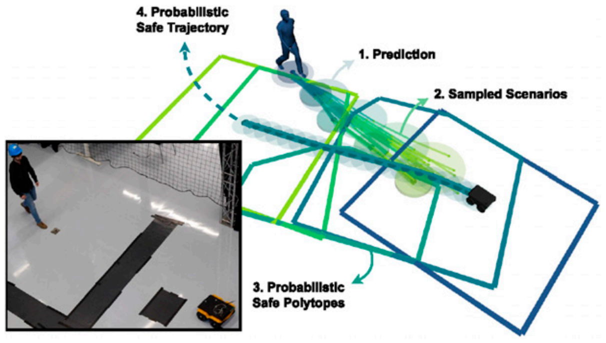

4. Probabilistic Safe Trajectory
1. Prediction
2. Sampled Scenarios
3. Probabilistic Safe Polytopes

Figure 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.

Department of Cognitive Robotics, TU Delft, Delft, The Netherlands

Corresponding author:
Oscar de Groot, Department of Cognitive Robotics, TU Delft, Mekelweg 2, Delft, 2628 CD, The Netherlands.
Emails: o.m.degroot@tudelft.nl; j.alonsomora@tudelft.nl

If we look at the motion planning problem from a probabilistic perspective, the occurrence of a collision is a *probabilistic* event. Our goal is to assess and bound the *Collision Probability* (CP) of the planned trajectory. This is challenging because of the non-Gaussian and multi-modal, that is, multiple paths are possible, distributions involved when predicting human behavior. In addition, when the collision probability spans a duration, for example, over the planned trajectory, then it must consider the correlation over time. The time correlation exists because the first collision in a trajectory renders it unsafe, such that all subsequent collisions can be ignored. Almost all of previous works, such as Blackmore et al. (2006), Luders et al. (2010), van Den Berg et al. (2011), Zhu and Alonso-Mora (2019), Wang et al. (2020a, b), and De Groot et al. (2021), do not account for this correlation, which leads to conservative trajectories. Similarly, the collision probability in these works is computed per obstacle, which degrades performance when more obstacles influence the plan, that is, in crowded environments. By accounting for the correlation in time and by considering all obstacles, one would be able to plan more efficient trajectories without compromising safety.

We achieve this goal by devising a novel probabilistic safe motion planner based on scenario optimization and nonlinear Model Predictive Control (MPC). Nonconvex scenario optimization (Campi et al., 2018) is a sampling-based approach for assessing the probability that a constraint holds under uncertainty. In the context of this paper, the imposed constraint probabilistically enforces collision avoidance between the controlled robot and the dynamic obstacles over the duration of the motion plan (see Figure 1). Scenario optimization has found limited applications in robotics. Its a posteriori safety verification is computationally demanding and it cannot directly be applied to distributions with unbounded support (e.g., Gaussians). We address these limitations in this work for our target application of motion planning in dynamic 2D environments, enabling its use in similar robotics applications.

In practice, the uncertainty associated with the motion of detected obstacles is predicted forward in time (Step 1). We sample scenarios from these predictions that describe the trajectories of all dynamic obstacles during the planning horizon (Step 2) and construct collision avoidance constraints around each of the scenarios (Step 3). The robot trajectory is optimized with respect to the constraints (Step 4), which provides probabilistic collision avoidance. This is, to our knowledge, the first real-time capable method that bounds the CP of the planned trajectory jointly, in contrast with the existing state-of-the-art where the same quantity is conservatively approximated through its marginals. Our method therefore provides more consistent real-time guarantees on the CP, irrespective of the number of time steps and obstacles, and the underlying probability distribution that describes obstacle motion. This ultimately enables our method to trade-off safety and planning performance more accurately. We refer to our novel probabilistic safe motion planner as *Safe Horizon MPC* (SH-MPC).

2. Related work and contribution

A sizable body of literature exists on motion planning for safe, autonomous navigation. In this section, we review some of the existing work on trajectory optimization with collision avoidance under uncertainty. For a general overview of autonomous navigation, we refer the reader to Paden et al. (2016) and Schwarting et al. (2018b).

2.1. Collision avoidance under uncertainty

An important problem in autonomous navigation is to prevent collisions with dynamic obstacles. In trajectory optimization (Brito et al., 2019; Schwarting et al., 2018a), the navigation problem is formulated as an optimization problem where performance criteria are optimized (e.g., lane following and progress towards the goal) under constraints (e.g., collision avoidance and actuator limits). Due to large and multi-modal uncertainties in human motion prediction, collision avoidance in the mean or nominal case (e.g., constant velocity) may lead to collisions in practice. Many works therefore consider how to address this uncertainty.

One can consider the collision avoidance problem as a special case of optimization under uncertainty, for which two common approaches exist. *Robust optimization* (Ben-Tal and Nemirovski, 1998) requires the constraints to be satisfied for all possible realizations of the uncertainty, while *stochastic optimization* (see Mesbah (2016) 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, we focus in the remainder of this section on stochastic optimization. We refer to Kouvaritakis and Cannon (2016) for more details on both methods in the context of Model Predictive Control (MPC).

2.2. Collision-avoidance chance constraints

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. These can be divided into discrete-time and continuous-time methods. *Discrete-time* methods (e.g., Blackmore et al. (2006)) consider the CP at a fixed set of time intervals, usually at the discretization time of the robot dynamics. *Continuous-time* methods (e.g., Frey et al. (2020)) instead consider the CP over a continuous time interval, resulting in more accurate approximations but usually at a high computational cost. For our real-time application, we focus on the discrete-time methods. Existing discrete-time methods usually approximate chance constraints through additional assumptions on the probability distribution associated with the uncertainty, for example, by assuming Gaussian distributions in Zhu and Alonso-Mora (2019) and in Fisac et al. (2018) or through additional assumptions on the controlled robot, for example, by assuming linear dynamics in Blackmore et al. (2006). The recent works (Wang et al., 2020a; De Groot et al., 2021) 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, that is, the timed sequence of planned positions, but rather on each of its individual positions over the planning horizon. This fails to accurately bound the probability of colliding at *any* time. Similarly, chance constraints imposed per obstacle fail to estimate the probability of colliding with *any* obstacle. In this regard, three types of chance constraint formulations have been considered in previous work: *Marginal*, *Conditional*, and *Joint*.

2.2.1. 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 we assume instead that this event is independent for all states ($\widetilde{\mathbb{P}}(A) \approx \prod_k \mathbb{P}(A_k)$). In Janson et al. (2018), 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, Blackmore et al. (2006) reformulated the constraints as an analytical constraint on the 1D Cumulative Density Function (CDF). The same idea is used in Zhu and Alonso-Mora (2019) in an MPC framework to prevent collision between robot and obstacle volumes. In Masahiro Ono and Williams (2008), 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 Luders et al. (2010) and Aoude et al. (2013), 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 Wang et al. (2020b, a), 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 Ren et al. (2023) where the conditional Variance-at-Risk (cVaR) is used to minimize constraint violation. In our previous work (De Groot et al., 2021), we ensure safety for a finite set of sampled chance constraints and generalize the associated safety properties using the scenario approach (Campi et al., 2018).

The *multiplicative* formulation explicitly constrains the product of the marginal probabilities. It was applied in van Den Berg et al. (2011) to plan motion under sensing and actuation uncertainty. An alternative marginal formulation is proposed in Fisac et al. (2018), which bounds the largest marginal constraint violation.

The limitation of marginal approaches is that the bound on the CP of the trajectory is inaccurate, as noted by Patil et al. (2012) and Janson et al. (2018). It is shown in Janson et al. (2018) 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, Patil et al. (2012) 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.

2.2.2. 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 (Janson et al., 2018). In particular, Blackmore et al. (2010), Janson et al. (2018), and Schmerling and Pavone (2017) 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 over trajectories and evaluating the constraint for each sample. Since MC methods approximate the CP and its gradient with sampling, they can be computationally expensive to apply in trajectory optimization without further alterations. An alternative is to formulate a mixed-integer problem (Blackmore et al., 2010) to decide which samples may be violated, but these problems are hard to solve in real-time.

2.3. Scenario optimization

Rather than constraining each of the marginal CPs along the planned trajectory, we constrain the joint CP of the trajectory. We solve the resulting problem with Nonconvex Scenario Optimization (NSO) (Campi et al., 2018) under an explicit chance constraint on the joint CP. Scenario optimization is well established for convex optimization under uncertainty (Calafiore and Campi, 2006; Campi and Garatti, 2008, 2011;

Schildbach et al., 2013) and has recently been extended to the general nonconvex case (Campi et al., 2018; Garatti and Campi, 2023). NSO solves the optimization problem under uncertainty by drawing a large set of samples from the uncertainty, that is, possible future motions of all obstacles, and formulating a deterministic constraint, for example, collision avoidance, for each sample. Our approach leverages a constraint formulation where many of the sampled constraints can be pruned, resulting in a deterministic program that can efficiently be solved online.

The NSO framework (Campi et al., 2018) cannot directly be applied to reformulate the joint collision avoidance chance constraint. To assess probabilistic safety, it traditionally needs to repeat the optimization problem once for each scenario and thus cannot be applied in real-time. In addition, unbounded distributions (e.g., Gaussian distributions) cannot be readily incorporated. We address these limitations in this work, including distributions with unbounded support (Sec. 5.1) and significantly reducing the computational complexity of the online safety assessment (Sec. 5.3) for our application to motion planning in dynamic environments. This allows us to apply the framework to online motion planning in real-time.

2.4. Contribution

The contributions of this paper 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 our work from previous work, where the collision probability is constrained per planning time instance. The idea is that each sample we draw from the distribution of the uncertain obstacle trajectories corresponds to a single collision avoidance constraint over the horizon. We want to stress that each sample represents a possible trajectory for all obstacles. The probability of collision is controlled through the number of samples drawn. By relying on sampling, our 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 Campi et al. (2018) where the support is computed after optimization. We leverage this information to certify the motion plan online.

We compare our approach in two simulation environments, on a mobile robot and a self-driving vehicle, with pedestrians against marginal baselines and show that our method achieves shorter task durations while maintaining the probabilistic safety specification. In addition, we show that the computation times scale well with respect to the number of obstacles and the type of distribution. Finally, we deploy our method on a mobile robot navigating among pedestrians. Our planner is implemented in C++/ROS and will be released open source.

2.5. Notation

Vectors and matrices are expressed in bold and capital bold notation, respectively. We use $\bigwedge$ to denote the "and" operation and $\bigvee$ to denote the "or" operation. The notation $\mathcal{C}\backslash C_i$ refers to the set $\mathcal{C}$ from which the element C_i is removed.

3. Problem formulation

We consider a controlled robot with nonlinear discrete-time dynamics

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

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 state is assumed to contain its x-y position $\boldsymbol{p} = [x, y] \in \mathbb{R}^2 \subseteq \mathbb{R}^{n_x}$. Humans in the environment of the robot pose constraints on the navigation envelope. We model the robot area with the union of n_d discs and the area of each obstacle with a single disc. The position of the robot disc with index c at time step k is denoted $\boldsymbol{p}_k^c$. We model each obstacle region with a single disc for simplicity, our method trivially extends to the case of multiple discs. We consider that the motion prediction of the dynamic obstacles is uncertain by modeling the positions of at most M obstacles as random variables. In particular, we denote the uncertain position of Obstacle j at time step k as $\boldsymbol{\delta}_{k,j} \subseteq \boldsymbol{\delta}_k$, where $\boldsymbol{\delta}_k$ contains all obstacle positions at time k. The *joint uncertainty* $\boldsymbol{\delta} = \left[\boldsymbol{\delta}_1^\top, \ldots, \boldsymbol{\delta}_N^\top\right]^\top \in \Delta$ stacks the uncertainty over all N time steps. The probability space[1] of the joint uncertainty $\Delta = \mathbb{R}^{2MN}$ therefore captures the future N positions of M obstacles and is endowed with a σ-algebra $\mathcal{D}$ and a probability measure $\mathbb{P}$.

3.1. Chance constrained planning problem

The planning problem is visualized in Figure 2. The dynamic uncertainty of other road users affects the navigation envelope of the robot. To constrain the probability of a collision with *any* obstacle along the horizon, we formulate a single chance constraint for collision avoidance. This leads to the following Chance Constrained Problem (CCP)
Problem 1. CCP.

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \sum_{k=0}^{N} J(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{2b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \ k = 0, \ldots, N-1 \tag{2c}$$

$$\mathbb{P}\left[\bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\bigwedge_{j=0}^{M}\left(\|\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}\|_2 \geq r\right)\right] \geq 1-\epsilon \tag{2d}$$

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, (2d) is the chance constraint for

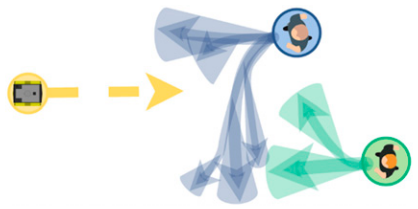


Figure 2. The chance constrained motion planning problem considered in this paper with the robot (yellow) navigating under probabilistic motion predictions of obstacles (blue/green). The distribution of the motion predictions can take any form but is visualized here with several modes and their variance (arrows/shaded regions).

avoiding collisions in every time step with all obstacles and r is the summed radius of the robot and obstacle discs (for simplicity assumed the same for all obstacle and robot discs). This problem is in general nonconvex due to the nonlinear dynamics and collision avoidance chance constraint. Our 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 $\widehat{\mathbb{P}}$ of $\mathbb{P}$ and the chance constraint relates to the estimate of the probability distribution.

3.2. Scenario-based planning problem

Directly evaluating chance constraint (2d) is not computationally feasible in closed loop. Our goal is to formulate a sampled deterministic version of the CCP, known as a Scenario Program (SP) (Campi et al., 2018). 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.

3.3. Paper organization

In the following, we first consider a more general CCP formulation that can be solved by the proposed framework. We provide a brief summary of the nonconvex scenario optimization framework of Campi et al. (2018) to show how this general class of CCPs can be solved via its associated SP. We then present our main results in Sec. 5, which shows how this SP can be solved in closed loop as MPC. Finally, we apply our main results to probabilistic motion planning in dynamic environments, in Sec. 6.

4. Nonconvex scenario optimization

In the following, we summarize the main results of the NSO framework of Campi et al. (2018) that we use to build our motion planning framework. To this end, consider the following generalization of Problem 1,

Problem 2. General CCP.

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{3a}$$

$$\text{s.t.}\quad \boldsymbol{x}_0=\boldsymbol{x}_{\text{init}} \tag{3b}$$

$$\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\dots,N-1 \tag{3c}$$

$$\mathbb{P}[g(\boldsymbol{x},\boldsymbol{\delta})\le 0]\ge 1-\epsilon,\ \boldsymbol{\delta}\in\Delta. \tag{3d}$$

The nominal constraint $g(\boldsymbol{x},\boldsymbol{\delta})\le 0$ 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 each scenario is independently extracted from $\mathbb{P}$ (Δ^S represents the S-fold Cartesian product of Δ associated with drawing S random samples). 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.

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U}\\ \boldsymbol{x}\in\mathbb{X}}} \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{4a}$$

$$\text{s.t.}\quad \boldsymbol{x}_0=\boldsymbol{x}_{\text{init}} \tag{4b}$$

$$\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\dots,N-1 \tag{4c}$$

$$g(\boldsymbol{x},\boldsymbol{\delta}^{(i)})\le 0,\ \boldsymbol{\delta}^{(i)}\in\Omega,\ i=1,\dots,S, \tag{4d}$$

where chance constraint (3d) has been replaced by the deterministic constraints for each of the scenarios in (4d). To simplify notation, we define a *decision* as $\boldsymbol{\theta}:=\left[\boldsymbol{x}^\top,\boldsymbol{u}^\top\right]^\top\in\Theta$, where $\Theta=\mathbb{X}\times\mathbb{U}$. Each scenario $\boldsymbol{\delta}^{(i)}$ imposes a constraint on the decision, denoted $\boldsymbol{\theta}\in\Theta_{\boldsymbol{\delta}^{(i)}}$, $\Theta_{\boldsymbol{\delta}^{(i)}}=\{\boldsymbol{\theta}\in\Theta: g(\boldsymbol{x},\boldsymbol{\delta}^{(i)})\le 0\}$. Formally, to make a decision based on the scenarios, the *decision algorithm* $\mathcal{A}$ is defined, that maps the scenarios to a decision (i.e., solving Problem 3). 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 1. For any finite $S=1,2,\dots$ and for any set of scenarios $\Omega\in\Delta^S$, it holds that

$$\mathcal{A}(\Omega)\in\Theta_{\boldsymbol{\delta}^{(i)}},\ i=1,\dots,S. \tag{5}$$

We focus our attention on the planning problem by considering only finite sample sizes S. Assumption 1 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). We will consider a relaxed constraint formulation in Sec. 5.1 to address this limitation.

Since the scenario decision accounts for all sampled scenarios, each additional scenario makes the scenario decision more robust. Consequently, the more scenarios that are 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

$$V(\boldsymbol{\theta}) := \mathbb{P}[\boldsymbol{\delta} \in \Delta : \boldsymbol{\theta} \notin \Theta_{\boldsymbol{\delta}}], \tag{6}$$

defines the probability that a decision $\boldsymbol{\theta}$ violates a newly observed scenario. We 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). We are 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.

Definition 1. Campi et al. (2018): Given a set of scenarios $\Omega = \{\boldsymbol{\delta}^{(1)}, \ldots, \boldsymbol{\delta}^{(S)}\}$, a *support subsample* $\mathcal{C} = \{\boldsymbol{\delta}^{(i_1)}, \ldots, \boldsymbol{\delta}^{(i_n)}\}$ is a tuple of n elements extracted from the set of scenarios with $i_1 < i_2 < \ldots < i_n$, which gives the same solution as with all scenarios in place, that is,

$$\mathcal{A}(\boldsymbol{\delta}^{(i_1)}, \ldots, \boldsymbol{\delta}^{(i_n)}) = \mathcal{A}(\boldsymbol{\delta}^{(1)}, \ldots, \boldsymbol{\delta}^{(S)}). \tag{7}$$

The cardinality of the support subsample is referred to as the *support size*, that is, $n := |\mathcal{C}|$. In the context of this paper, 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. For a confidence $1 - \beta$, Theorem 1 in Campi et al. (2018) provides the bound

$$\mathbb{P}^S[V(\boldsymbol{\theta}^*) > \epsilon(n)] \leq \beta, \tag{8}$$

where the risk is defined as a function of the support size, $\epsilon(n) : \{0, \ldots, S\} \to [0, 1]$, that must satisfy

$$\sum_{v=0}^{S-1} \binom{S}{v} [1 - \epsilon(v)]^{S-v} = \beta, \quad \epsilon(S) = 1. \tag{9}$$

Equation (8) therefore upper limits the probability that the scenario decision $\boldsymbol{\theta}^*$ exceeds the acceptable risk $\epsilon(n)$ by β. The risk is divided equally over all support values by the mapping

$$\epsilon(n) = \begin{cases} 1, & n = S, \\ 1 - \left(\dfrac{\beta}{S\binom{S}{n}}\right)^{\frac{1}{S-n}}, & \text{otherwise.} \end{cases} \tag{10}$$

This mapping certifies the solution without further information on the problem. Other mappings are possible; for more details, see Campi et al. (2018). The interpretation of equation (8) is that when an algorithm computes a decision $\boldsymbol{\theta}^*$ along with the support size n^*, then the risk is certified to be no larger than $\epsilon(n^*)$ with confidence $1 - \beta$. For example, with $N = 1000$ and $1 - \beta = 1 - 10^{-6}$, when the support size is $n^* = 6$, then equation (10) gives $\epsilon(6) = 5.4\%$ and we can certify that the risk is bounded by 5.4% with confidence $1 - 10^{-6}$.

A general algorithm to determine the support is the *greedy* algorithm of Campi et al. (2018). 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.

4.1. Limitations of NSO for motion planning

An SP can be constructed from the CCP in Problem 1 by sampling scenarios from the distribution over obstacle trajectories and formulating collision avoidance constraints for each scenario. However, this strategy is not directly applicable for the motion planning problem because of two limitations.

Limitation 1. Assumption 1 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), there is always a nonzero probability of drawing such a sample.

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.

Our goal is to address these limitations so that we can solve Problem 1 with an SP to bound the joint CP.

5. Safe horizon model predictive control

This section introduces our safe motion planning framework. First, we reformulate the SP to obtain a real-time solvable problem that satisfies Assumption 1. Then, we show how the support can be estimated during optimization. Finally, we derive the sample size of the SP.

5.1. 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 Figure 3(a)). Using these scenarios, we can construct the following SP from the CCP in Problem 1

Problem 4. Motion Planning SP.

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}\\ d\in\mathbb{R}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) + w_d\|d\|_2^2 \tag{11a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{11b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\dots,N-1 \tag{11c}$$

$$\bigwedge_{i=0}^{S}\bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\bigwedge_{j=0}^{M}\left(\|\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}^{(i)}\|_2 \geq r+d\right), \tag{11d}$$

where (11d) represents the scenario constraints (bundled with the "and" operator) and $w_d > 0, w_d \in \mathbb{R}$ weighs the slack variable. 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 1 is satisfied (addressing Limitation 1). We utilize a single slack variable to allow the scenario constraints to be pruned for real-time performance. Less conservative trajectories may be obtained with S slack variables as proposed in Garatti and Campi (2023), but come at a high computational cost. The violation probability associated with Problem 4 is

$$\mathbb{P}\left[\boldsymbol{\delta}\in\Delta \,\middle|\, \bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\bigwedge_{j=0}^{M}\left(\|\boldsymbol{p}_k^c - \boldsymbol{\delta}_{k,j}\|_2 \geq r+d\right)\right] \geq 1-\epsilon. \tag{12}$$

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).

5.2. Improved computational efficiency

While Problem 4 can be solved with off-the-shelf nonlinear program solvers, its problem statement is computationally inefficient. The collision-free space described by (11d) is nonconvex and can lead to high support (i.e., requiring many samples). We consider a more efficient over-approximation of the collision avoidance constraints with lower support.

We linearize the collision regions with respect to the previously planned robot trajectory (denoted $\widehat{\boldsymbol{p}}$). After linearization, each scenario is associated with a linear constraint (depicted in Figure 3(b)). For a previous robot position $\widehat{\boldsymbol{p}}_k$ and obstacle position $\boldsymbol{\delta}_k$, the constraints are given by

$$\mathcal{H}(\widehat{\boldsymbol{p}}_k,\boldsymbol{\delta}_k,d) = \{\boldsymbol{p}_k | \boldsymbol{A}(\widehat{\boldsymbol{p}}_k,\boldsymbol{\delta}_k)^\top \boldsymbol{p}_k \leq b(\widehat{\boldsymbol{p}}_k,\boldsymbol{\delta}_k,d)\}, \tag{13}$$

where

$$\boldsymbol{A}(\widehat{\boldsymbol{p}}_k,\boldsymbol{\delta}_k) = \frac{\boldsymbol{\delta}_k - \widehat{\boldsymbol{p}}_k}{\|\boldsymbol{\delta}_k - \widehat{\boldsymbol{p}}_k\|}, b(\widehat{\boldsymbol{p}}_k,\boldsymbol{\delta}_k,d) = \boldsymbol{A}^\top\boldsymbol{\delta}_k - (r+d). \tag{14}$$

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 tight approximation of the original collision regions. For the linearized constraints, we obtain the following SP:

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}\\ d\in\mathbb{R}}} \quad \sum_{k=0}^{N} J(\boldsymbol{x}_k,\boldsymbol{u}_k) + w_d\|d\|_2^2 \tag{15a}$$

$$\text{s.t.} \quad \boldsymbol{x}_0 = \boldsymbol{x}_{\text{init}} \tag{15b}$$

$$\boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\dots,N-1 \tag{15c}$$

$$\bigwedge_{i=0}^{S}\bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\bigwedge_{j=0}^{M}\left(\boldsymbol{p}_k^c \in \mathcal{H}\left(\widehat{\boldsymbol{p}}_k^c, \boldsymbol{\delta}_{k,j}^{(i)}, d\right)\right). \tag{15d}$$

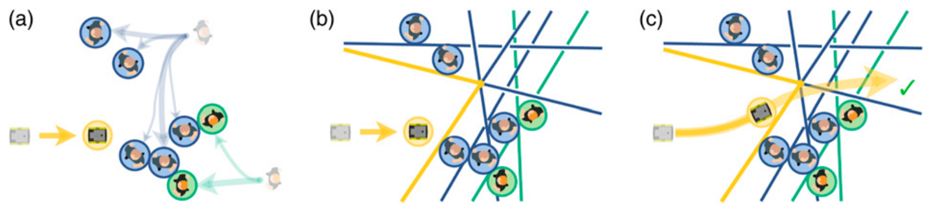


Figure 3. Schematic illustration of Safe Horizon MPC applied to a mobile robot. (a) Scenarios are sampled from the trajectory distributions. Each time instance of SP (Equation (4)) is associated with a set of sampled obstacle positions as visualized by the green and blue circled pedestrians. (b) Halfspace constraints are constructed between sampled obstacles and the robot, and are reduced to a probabilistic safe polytope for each time instance and robot disc, depicted by the yellow lines. (c) Problem 5 is solved via Algorithm 1. The resulting trajectory is certified up to a probabilistic bound.

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. We provide more details in Appendix A.

To drastically reduce the computational demand of this formulation, we reorder Constraints (15d) as

$$\bigwedge_{k=1}^{N}\bigwedge_{c=0}^{n_d}\left[\bigwedge_{i=0}^{S}\bigwedge_{j=0}^{M}\left(\boldsymbol{p}_k^c\in\mathcal{H}\left(\widehat{\boldsymbol{p}}_k^c,\boldsymbol{\delta}_{k,j}^{(i)},d\right)\right)\right], \tag{16}$$

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 Figure 3(b)). The constraints (15d) can, therefore, be reduced to free-space polytopes before optimization, which significantly reduces computation times. We denote these polytopes, for disc c and time step k as

$$\mathcal{P}_k^c(0)=\left\{\boldsymbol{p}_k^c \mid \bigwedge_{(i,j)\in\mathcal{I}_k^c}\boldsymbol{p}_k^c\in\mathcal{H}\left(\widehat{\boldsymbol{p}}_k^c,\boldsymbol{\delta}_{k,j}^{(i)},0\right)\right\}, \tag{17}$$

where the indices of scenarios that form the boundary of the polytope are collected in the set $\mathcal{I}_k^c$. For $d>0$, the polytope is denoted $\mathcal{P}_k^c(d)$ and consists of the same linear constraints relaxed with a distance d. We derived an algorithm based on recursive search, detailed in Appendix B, that performs this pruning in less than 100 μs, while it typically reduces the number of constraints by a factor 10^2-10^3. We verified, for an example in our simulations, that with 10,816 samples, the final polytope consists of 11 constraints on average.

The final SP that is solved online, after pruning, is given by

Problem 5. Safe Horizon MPC.

$$\min_{\substack{\boldsymbol{u}\in\mathbb{U},\boldsymbol{x}\in\mathbb{X}\\ d\in\mathbb{R}}}\quad \sum_{k=0}^{N}J(\boldsymbol{x}_k,\boldsymbol{u}_k)+w_d\|d\|_2^2 \tag{18a}$$

$$\text{s.t.}\quad \boldsymbol{x}_0=\boldsymbol{x}_{\text{init}} \tag{18b}$$

$$\boldsymbol{x}_{k+1}=f(\boldsymbol{x}_k,\boldsymbol{u}_k),\ k=0,\ldots,N-1 \tag{18c}$$

$$\boldsymbol{p}_k^c\in\mathcal{P}_k^c(d),\quad \forall k,\forall c. \tag{18d}$$

The SH-MPC problem can be solved in real-time while bounding the joint CP of its trajectory (see Figure 3(c)).

5.3. *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, we propose an estimate of the support $\widehat{n}\geq n$ that can efficiently be computed during optimization.

For a convex SP, the support can easily be computed as its support constraints are active (Campi and Garatti, 2008; Garatti and Campi, 2019). Computing the support in the nonconvex case is much harder as this property does not hold. We show 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 $\mathcal{A}^l:\Theta\times\Delta^S\rightarrow\Theta,\ l\in 0,\ldots,L$ as

$$\mathcal{A}=\mathcal{A}^L(\ldots\mathcal{A}^1(\mathcal{A}^0(\boldsymbol{\theta}^0,\Omega),\Omega)\ldots,\Omega). \tag{19}$$

We define the support of an iteration as a set of scenarios that results in the same output as the output where all scenarios are considered. 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 (19). Its support $\mathcal{C}$ satisfies

$$\mathcal{C}\subseteq\Omega\backslash\Omega_{\text{ns}},\quad \Omega_{\text{ns}}=\bigcap_{l=0}^{L}\Omega_{\text{ns}}^l, \tag{20}$$

where

$$\Omega_{\text{ns}}^l=\left\{\boldsymbol{\delta}^{(i)}\in\Omega \mid \mathcal{A}^l(\boldsymbol{\theta}^l,\Omega)=\mathcal{A}^l(\boldsymbol{\theta}^l,\Omega\backslash\boldsymbol{\delta}^{(i)})\right\}, \tag{21}$$

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 (20) and (21),

$$\mathcal{A}(\boldsymbol{\theta}^0,\Omega\backslash\boldsymbol{\delta}^{(i)})=\mathcal{A}(\boldsymbol{\theta}^0,\Omega)=\boldsymbol{\theta}^*, \tag{22}$$

for all $\boldsymbol{\delta}^{(i)}\in\Omega_{\text{ns}}$. 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.

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, we note 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 for which the support constraints are active (Campi and Garatti, 2008). For example, in Sequential Quadratic Programming (SQP), each iteration $\mathcal{A}^l$ refers to the inner QPs. We explicitly impose this assumption on the solver.

Assumption 2. 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. We therefore require the additional assumption that the constraints of the convex problem are satisfied in each iteration.

Assumption 3. The solution computed by each iteration $\mathcal{A}^l$ is feasible with respect to its inequality constraints.

The scenario constraints are in general nonlinear due to the robot dynamics. In each iteration, and as commonly done to solve nonlinear problems, the solver makes the problem convex, deriving constraints from the locally linearized dynamics. We only require that these constraints are satisfied. Using these assumptions, we propose the following support estimation.

Theorem 1. If iterations $\mathcal{A}^l$ of the decision algorithm satisfy Assumptions 2 and 3, then the support of Problem 3 satisfies

$$\mathcal{C} \subseteq \bigcup_{j=0}^{L} \Omega^j_{\text{active}} = \widehat{\mathcal{C}}, \quad \hat{n} := |\widehat{\mathcal{C}}|, \tag{23}$$

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

$$\Omega^l_{\text{active}} = \{\boldsymbol{\delta}^{(i)} \in \Omega \mid \exists k \; g^l(\boldsymbol{x}^l_k, \boldsymbol{\delta}^{(i)}_k, d^l) = 0\}, \tag{24}$$

denote the active constraints in iteration l.

Proof. Under Assumption 2, the support constraints of iteration l are in the set Ω^l_{active} (all constraints are satisfied by Assumption 3). Under convexity, Ω^l_{active} is therefore the complement of the set Ω^l_{ns}. Invoking Lemma 1 and using De Morgans Law (Morgan, 1847), we obtain (23).

The support of SH-MPC can therefore be estimated by the aggregated set of active scenarios over all iterations, addressing Limitation 2. In practice, we use Sequential Quadratic Programming (SQP) (Nocedal and Wright, 2006) Chapter 18 to solve Problem 5, which satisfies Assumption 2 (convexity of iterates) and Assumption 3 (feasibility of intermediate iterates).

5.4. Determining the sample size

With the support of SH-MPC estimated through Theorem 1 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. We therefore propose to find a sufficiently high S that certifies the joint CP almost always. To that end, we define a support limit $\overline{n}$ describing a support size that we expect not to exceed and use it to certify the optimized trajectory in practice.

Theorem 2. SH-MPC (Problem 5) with sample size S, computed from (10) with support limit $\overline{n}$, does not exceed a specified risk of ϵ (with specified confidence $1-\beta$) if its support is lower than the support limit, that is, if $n \leq \overline{n}$.

Proof. SH-MPC satisfies Assumption 1 and with $\epsilon(n)$ as in (10), Theorem 1 in Campi et al. (2018) ensures that (8) holds. Since $\epsilon(n)$ in (10) is monotonically increasing in n (i.e., $\epsilon(n+1) > \epsilon(n), \forall n < S$) and given that the computed S ensures that $\epsilon(\overline{n}) \leq \epsilon$, we have that $\epsilon(n) \leq \epsilon, \forall n \leq \overline{n}$.

Theorem 2 shows that SH-MPC solves the CCP in Problem 1 if its support is lower than the support limit (and if $d = 0$).

In practice, the support limit can be estimated by deploying the planner and collecting the support estimation of Theorem 1. In our approach, we keep track of the largest estimated support over a large number of iterations and use this empirical worst-case support as the support limit. An alternative could be to start with a large support limit, lowering it when it is not passed for many iterations. Since the support limit is likely higher than necessary for many iterations, the trajectories computed by SH-MPC become conservative. The support limit however allows the planner to incorporate the joint collision avoidance chance constraint in *real-time*. Conservatism can be reduced by running several instances of SH-MPC in parallel (along the lines of Mustafa et al. (2023)), which can lead to lower and more consistent support. Alternatively, the support limit could be continuously adapted based on its online estimation. We consider these future work.

5.5. Algorithm outline

Algorithm 1 summarizes the SH-MPC framework. Offline, we compute the sample size S based on the joint CP ϵ, confidence $1-\beta$ and support limit $\overline{n}$ (Line 2). We provide a Jupyter notebook with this paper (De Groot, 2024) that performs this computation using a bisection of (10). Online, the predicted probability distribution is received from the perception module and scenarios are sampled from it (Line $4-5$). We solve 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 $10-12$). Otherwise, the robot executes a fallback action (e.g., braking).

Algorithm 1: Safe Horizon MPC

```
1  Input: $\epsilon$, $\beta$, $\bar{n}$ and $L$
2  $S$ ← Bisection of Eq. 10 using $\epsilon, \beta, \bar{n}$
3  while True do
4      $\mathbb{P}$ ← Motion prediction received from perception
5      $\Omega$ ← Draw $S$ samples from $\mathbb{P}$
6      for $l = 1, \dots, L$ do
7          $\boldsymbol{x}^l, \boldsymbol{u}^l, d^l$ ← Solve an iteration of Problem 5
8          $\Omega^l_{\text{active}}$ ← Determine active scenarios (Eq. 24)
9          $\hat{n}^l$ ← Aggregate the support (Eq. 23)
10     if $d^L = 0$ and $\hat{n}^L \leq \bar{n}$ then
11         Trajectory attains a collision risk of at most $\epsilon$
12         Actuate $\boldsymbol{u}^L_0$
13     else
14         Decelerate the robot with $1.0\text{m/s}^2$
```

6. Results

We evaluate SH-MPC in simulation, comparing against two MPC baselines, and in real-world experiments on a mobile robot among pedestrians. A video of the results accompanies this paper (De Groot et al., 2024a).

6.1. Simulation setup

We consider a mobile robot moving through an environment with pedestrians (see Figure 4) in which the robot is modeled by a kinematic unicycle model (Siegwart and Nourbakhsh, 2011). We assume throughout that the distribution of pedestrian motion is known in order to evaluate the performance of the planner in isolation (i.e., without prediction errors). We first validate on a Gaussian case, 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

$$\boldsymbol{\delta}_{k+1} = \boldsymbol{\delta}_k + (\boldsymbol{v} + \boldsymbol{\delta}_{w,k})dt, \quad \boldsymbol{\delta}_{w,k} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_w), \tag{25}$$

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 0.3 m and 0.325 m, respectively. Throughout our simulations and experiments, we set the maximum open-loop joint CP as $\epsilon = 0.05$. This value can be handled computationally by the considered methods and is typically sufficient to prevent collisions in closed-loop without excessively reducing the probabilistic collision-free space.

6.2. Implementation of SH-MPC

All baselines and SH-MPC use the Model Predictive Contouring Control formulation in Brito et al. (2019), which tracks a reference path and reference velocity while penalizing robot inputs. We solve Problem 5 using the Forces Pro SQP solver (Domahidi and Jerez, 2014) with at most 12 iterations. We empirically selected the support limit $\overline{n} = 10$. With $\epsilon = 0.05$ and setting a confidence of $1 - \beta = 0.99$, the sample size is $S = 1351$. If the support limit is exceeded or the slack is nonzero, we slow down the robot. The linearization of the constraints requires the previous plan to be feasible. We use of 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, we solve a feasibility program using Douglas–Rachford Splitting (Aragón Artacho et al., 2020). SH-MPC is implemented in C++/ROS and will be released open source.

6.3. Baselines

We compare SH-MPC against a deterministic baseline and two probabilistic baselines that constrain the marginal CP, that is, the independent CP per time instance and/or obstacle.

(1) **Deterministic MPC**: Avoids the mean of each mode as a deterministic circular obstacle.
(2) **S-MPCC** (De Groot et al., 2021): Marginal CP per time instance.
(3) **CC-MPC** (Zhu and Alonso-Mora, 2019): Marginal CP per time instance 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

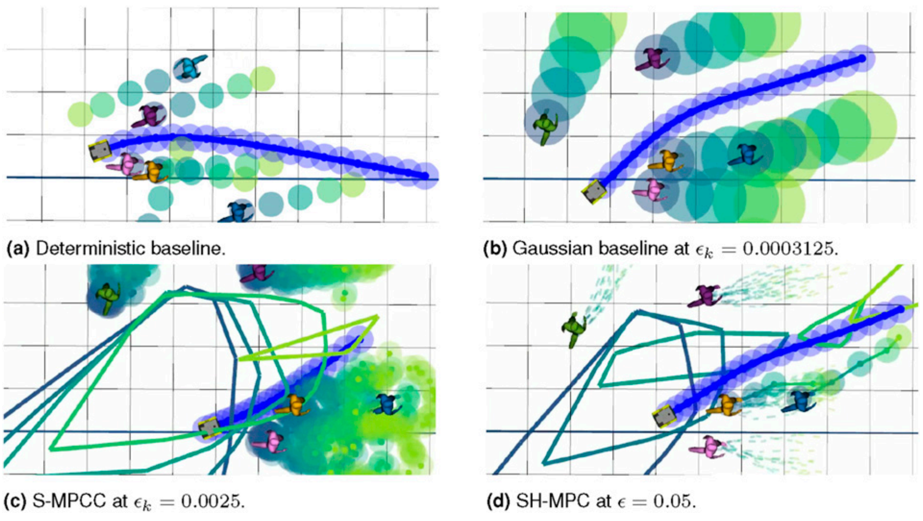


Figure 4. Planned trajectories for the different methods in the Gaussian simulation with eight pedestrians. The robot plan and associated collision areas are drawn in blue. For all methods, visualizations are shown in blue to green colors for stages 0, 5, 10, 15, and 19, respectively. (a) Circles show the predicted areas occupied by the obstacles following the mean of the Gaussian distribution. (b) Circles show the level sets of the Gaussian distribution at the specified ϵ. (c) 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. (d) Similar to (c) but depicting sampled *trajectories* as dashed lines and highlighting support constraints. (a) Deterministic baseline. (b) Gaussian baseline at $\epsilon_k = 0.0003125$. (c) S-MPCC at $\epsilon_k = 0.0025$. (d) SH-MPC at $\epsilon = 0.05$.

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 requires $S = 75946$ for instance when $\epsilon_k = 0.0025$. Samples in the center of the distribution are pruned in the Gaussian case to reduce the number of samples considered online.

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

$$\mathbb{P}\left[\boldsymbol{a}_{k,j}^\top(\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}) \geq r\right] \geq 1 - \epsilon_k, \quad \boldsymbol{a}_{k,j} = \frac{\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}}{\|\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}\|}, \tag{26}$$

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

$$\boldsymbol{a}_{k,j}^\top(\boldsymbol{p}_k - \boldsymbol{\delta}_{k,j}) - r \geq \text{erf}^{-1}(1 - 2\epsilon_k)\sqrt{2\boldsymbol{a}_{k,j}^\top \boldsymbol{\Sigma} \boldsymbol{a}_{k,j}}, \tag{27}$$

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.

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, we consider three versions of the baselines. The first sets $\epsilon_k = \epsilon$, which is not provably safe but relies on updates of the controller to remain safe. The second version sets $\epsilon_k = \epsilon/N$, accounting for the marginal approximation over time since $\sum_k \epsilon_k = \epsilon$, but ignoring marginalization per obstacle. As S-MPCC inherently accounts for the obstacle marginalization, it matches SH-MPC's safety specification (joint CP of ϵ) with this version. The third version accounts for both marginalizations, setting $\epsilon_k = \epsilon/NM$ such that $M\sum_k \epsilon_k = \epsilon$. CC-MPC only attains the same safety specification as SH-MPC using this last version.

Table 1. Weights of the MPC problem (see Brito et al. (2019)).

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

6.4. Weights and parameters

The only difference between the planners is their collision avoidance constraints. We use the same solver, weights and cost function for all methods. Weights of the MPC problem are given in Table 1. We define a horizon of $N = 20$ steps, with a discretization step of 0.2s, giving a time horizon of 4.0s. The control rate is 20 Hz, corresponding to a sampling time of 50 ms. The computer running the simulations is equipped with an Intel i9 CPU@2.4GHz.

6.5. Baseline comparison—Gaussian uncertainties

We consider an environment with multiple dynamic pedestrians following the dynamics in equation (25) where $\boldsymbol{v}$ describes a constant velocity. We validate the actual CP of the motion plan offline after the experiments through Monte Carlo sampling for all methods, where the dynamics in equation (25) are used to generate the samples. We compute 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. We validate in a scenario with 8 pedestrians.

Figure 4 depicts snapshots of the simulations. Quantitative results are summarized in Table 2. The deterministic baseline completes the task faster than the other methods but attains a high CP and consequently collides in total in 102 cases, whereas the other methods do not collide. S-MPCC is more conservative than the other methods because of its conservatively large sample size. CC-MPC, that uses the Gaussian CDF, bounds the marginal CP (CP$_k$) per obstacle accurately. However, when $\epsilon_k = 0.05$, its maximum joint CP is 0.2264, which exceeds 0.05. The trajectories are safe

Table 2. Statistical results for the eight pedestrian environment under Gaussian uncertainty (sec. 6.5). Results compare the marginal CP ("CP$_k$") and joint CP ("CP"), the task duration, minimum distance to the pedestrians and computation times over 100 experiments. 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. The divider separates methods that do not bound the joint risk at $\epsilon = 0.05$ (above) and methods that do (below). Our method (SH-MPC) is less conservative than the baselines with similar safety specifications.

Method	**Max CP$_k$/Spec. (%)**	**Max CP/Spec. (%)**	**Dur. [s]**	**Min Dist. [m]**	**Runtime (Max) [ms]**
Deterministic MPC	0.9387/- (-)	0.9989/- (-)	10.1 (0.9)	-0.03 (0.07)	13 (76)
S-MPCC ($\epsilon_k = 0.05$)	0.0037/0.0500 (7)	0.0049/- (-)	15.6 (2.6)	0.36 (0.06)	47 (137)
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)
S-MPCC ($\epsilon_k = 0.0025$)	0.0002/0.0025 (7)	0.0002/- (-)	19.2 (3.2)	0.55 (0.08)	83 (286)
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)

under $\epsilon_k = 0.0025$ in practice with a maximum overall CP of 0.0159, although it is not theoretically accounting for the obstacle marginalization. This is evident in the marginal CP that, with 0.0047, exceeds its specification of 0.0025. For clarification, Figure 6 illustrates this error made by obstacle marginalization on a 1-dimensional example. The version of CC-MPC with the same safety specification as SH-MPC, constrains $\epsilon_k = 0.0025/8 = 0.0003125$. The results indicate that SH-MPC moves through the environment faster than the safe baselines with the same safety specification as its joint CP (0.0092) is less conservative than that of the baselines (e.g., 0.0019 for CC-MPC).

6.6. Baseline comparison: Gaussian mixture

We now modify the distribution to incorporate a probability that the pedestrians will cross. We encode this scenario with a Markov Chain (see Figure 5) that changes pedestrian movement from horizontal to diagonal in addition to the Gaussian process noise of the previous simulations. We define the pedestrian dynamics as

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + (B\boldsymbol{v} + \boldsymbol{\delta}_{w,k})dt, \quad \boldsymbol{\delta}_{w,k} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{w,k}), \qquad (28)$$

where B is either $B_h = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ or $B_d = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}^\top$ 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 $p_2 = (1 - 0.025)0.025$. We apply the

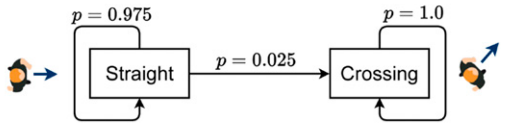


Figure 5. Markov chain modeling of a crossing pedestrian.

deterministic baseline to the GMM distribution by avoiding the mean of each mode. For CC-MPC, we formulate the constraints at a risk of ϵ_k for all modes, which can lead to conservatism when multiple modes influence the plan. We validate in this environment with eight pedestrians.

Results are summarized in Table 3. SH-MPC outperforms the baselines on almost all metrics. We observed in total 25 collisions for the deterministic baseline. The computation times of CC-MPC become excessive due to the many modes to be considered, while the computation times of SH-MPC (and S-MPCC) are unaffected.

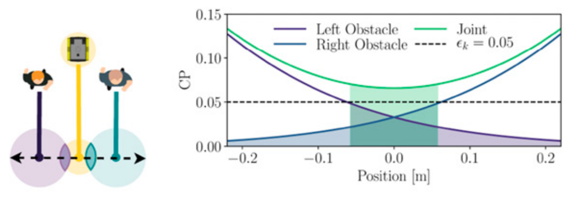
6.7. Empirical analysis and sensitivity

6.7.1. Support estimation. We compare our proposed support estimation in equation (23) with the default Greedy algorithm of Campi et al. (2018) in a scenario with one static pedestrian. We manually vary the sample size between 100 and 1000 and collect, for 100 iterations of each sample size, the runtime of the optimization with support estimation and the estimated support size. Figure 7 shows that, while the runtime of our support estimation is negligible compared to the optimization, the greedy algorithm takes roughly $S + 1$ times as long to solve.

6.7.2. Empirical risk distribution. We visualize the empirical distribution of the joint CP for each value of the support estimate in Figure 8 computed for the simulations of Sec. 6.5. In line with the theory, we observe that the trajectory CP is on average higher for higher support values. Our support limit ($n = 10$) is reached in only 2 out of 33564 cases and is never exceeded. In the other cases, our support limit is conservative. In the simulations of Sec. 6.6, the support limit is exceeded six times in total. This usually indicates that the planner's intended trajectory is becoming increasingly risky and is a valid cause for slowing the robot down. The same situation can result in $d > 0$ (i.e., infeasible optimization problem) of which we detected 50 occurrences in the same simulations.

Table 3. Statistical results for the eight pedestrian environment under multimodal uncertainty (sec. 6.5). Displayed results follow the notation in Table 2. Our method (SH-MPC) maintains a similar joint CP and similar computation times 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]**
Deterministic MPC	0.9426/- (-)	1.0000/- (-)	11.7 (1.8)	0.12 (0.20)	204 (483)
S-MPCC ($\epsilon_k = 0.05$)	0.0042/0.0500 (8)	0.0059/- (-)	17.6 (4.3)	0.41 (0.16)	44 (123)
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)
S-MPCC ($\epsilon_k = 0.0025$)	0.0002/0.0025 (6)	0.0002/- (-)	19.6 (3.0)	0.42 (0.22)	73 (131)
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 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. (a) Example case. (b) Joint and marginalized risk in the direction of the black dashed arrow in (a).

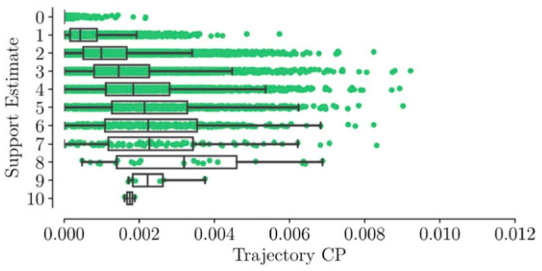


Figure 8. The empirical distribution of the joint CP for SH-MPC for each estimated support over 33,564 planner iterations (in simulations of Sec. 6.5). Dots denote individual planner iterations. Boxplots denote the statistics per support value.

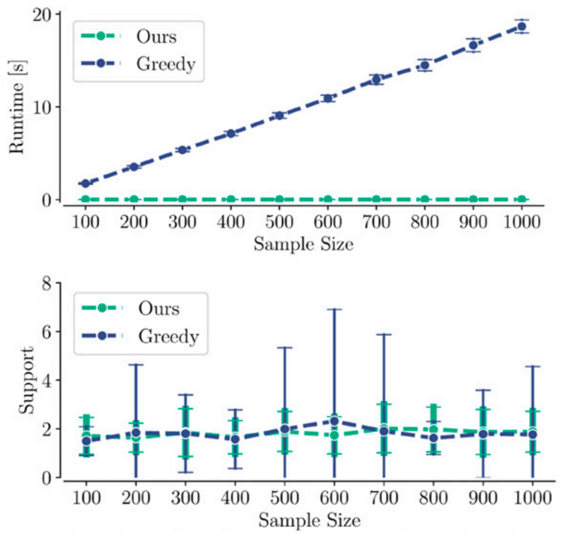


Figure 7. The mean and standard deviation of the runtime (top) and estimated support size (bottom).

6.7.3. Sensitivity to ϵ *and* N. We validate the sensitivity of SH-MPC to varying risk specifications (ϵ) and horizon lengths (N) in the scenario of Sec. 6.5. Figure 9(a) 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, Figure 9(b) 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.

6.8. Real-world experiments

We validate 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 three pedestrians in a 7 m × 9 m space. We detect the positions of all agents with a motion capture system. The robot is given a reference path along the center of the space and a reference velocity of 1.5 m/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 20 Hz by SH-MPC that models the dynamics as a second-order unicycle. We specify an acceptable risk of $\epsilon = 0.05$ over a 20 step and 4s horizon with confidence $1 - \beta = 0.99$ and support limit $\overline{n} = 6$. The pedestrian motion predictions follow Gaussian constant velocity dynamics (Equation (25)), where an Extended Kalman Filter (EKF) is used to estimate the velocity.

Figure 10 depicts three experiments with one 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 Figure 10(a) and 10(b), it keeps sufficient lateral distance when passing, while in Figure 10(c), it reacts to the pedestrian speeding up its pace. Figure 11 shows three experiments with three pedestrians. The robot evades the pedestrians smoothly in this more crowded environment. In Figure 11(a) and 11(b), the robot smoothly evades multiple pedestrians. In Figure 11(c), the planner first evades two pedestrians, then reverses to ensure the safety of the third, fast-moving pedestrian.

With 1 pedestrian, the support limit was never exceeded. With three pedestrians, it was exceeded 30 times in total. This could be resolved by increasing the support limit when it is exceeded. In our case, slowing down led to safe behavior. We did not observe collisions 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 De Groot et al. (2023)) but this is outside of the scope of this paper.

6.9. Autonomous navigation in an urban environment

SH-MPC can be applied to different robot morphologies and scenarios. To demonstrate this, we deploy our approach on a simulated self-driving vehicle in Carla simulator (Dosovitskiy et al., 2017). The dynamics are modeled with a second-order bicycle model (Kong et al., 2015) and the collision region consists of three discs. We construct a collision-free polytope for each of the discs. The pedestrians are programmed to follow the same dynamics as in Sec. 6.6, that is, a GMM

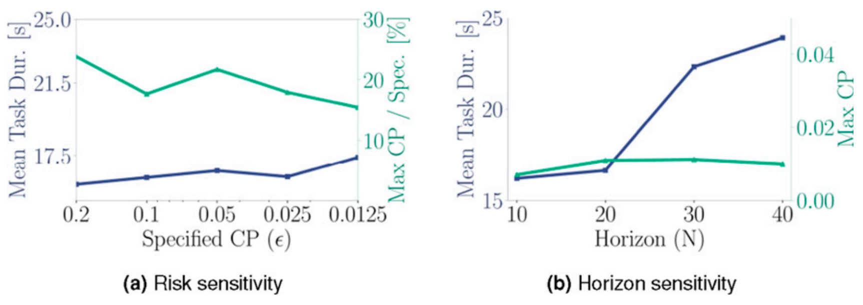


Figure 9. Sensitivities of the empirical CP and task duration with respect to (a) the specified CP with constant $N = 20$ and (b) the horizon length with constant specified CP $\epsilon = 0.05$, evaluated over 25 experiments in the setting of Sec. 6.5. (a) Risk sensitivity. (b) Horizon sensitivity.

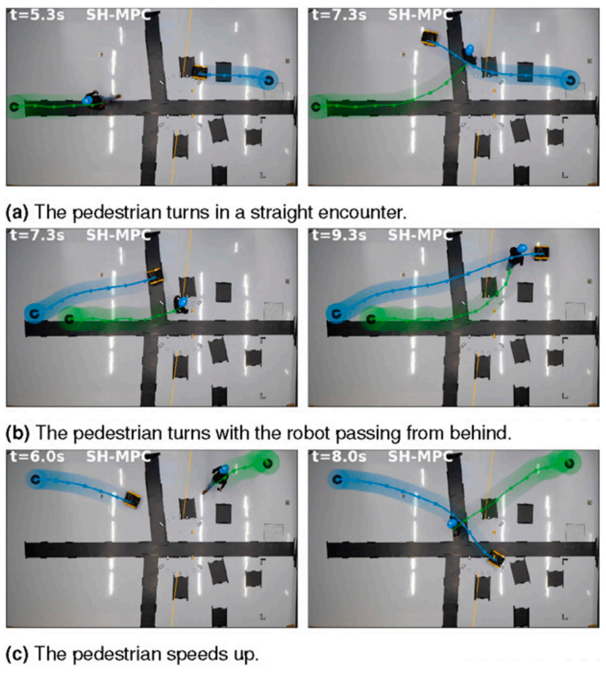

t=5.3s SH-MPC
t=7.3s SH-MPC
(a) The pedestrian turns in a straight encounter.
t=7.3s SH-MPC
t=9.3s SH-MPC
(b) The pedestrian turns with the robot passing from behind.
t=6.0s SH-MPC
t=8.0s SH-MPC
(c) The pedestrian speeds up.

Figure 10. Past 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. (a) The pedestrian turns in a straight encounter. (b) The pedestrian turns with the robot passing from behind. (c) The pedestrian speeds up.

modeling crossing behavior. We do not model the interaction between the vehicle and the pedestrians. The control frequency is 10 Hz. We measured the computation time to be 88 ms on average and 135 ms maximum. Figure 12 visualizes snapshots of the simulations. In Case A (see Figure 12(b)), the planner keeps enough distance to let the pedestrians cross while driving as close to the path as is safe. In Case B (see Figure 12(c)), the vehicle passes behind the pedestrians while keeping its distance from the pedestrian that is not crossing.

7. Discussion

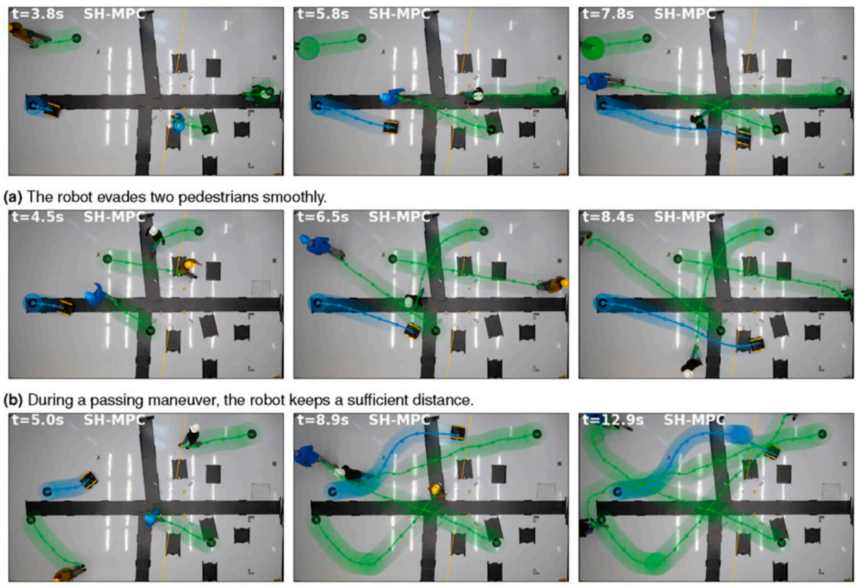
As expected from the theoretical analysis, our experiments showed that SH-MPC can consistently bound the CP of the overall trajectory. Where methods that impose marginal constraints need to decide between performance ($\epsilon_k = \epsilon$) and safety ($\epsilon_k = \epsilon/NM$), SH-MPC makes this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

7.1. Reducing conservatism

Planning performance could be improved by reducing the gap between the guaranteed and obtained risk, for example, by assuming some knowledge of the distribution or by running multiple scenario programs in parallel as considered in Mustafa et al. (2023). Over-approximating Monte Carlo sampling may be less conservative than NSO and may reduce conservatism while maintaining safety. Recent work (Brudermüller et al., 2024) proposed such an approximation. Their method is less conservative, but also computationally more demanding than SH-MPC, achieving about an order of magnitude slower computation times than SH-MPC without considering non-holonomic constraints.

The guarantees provided in this paper 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 (see De Groot et al. (2024b)), in line with the typical scenario approach, to reduce

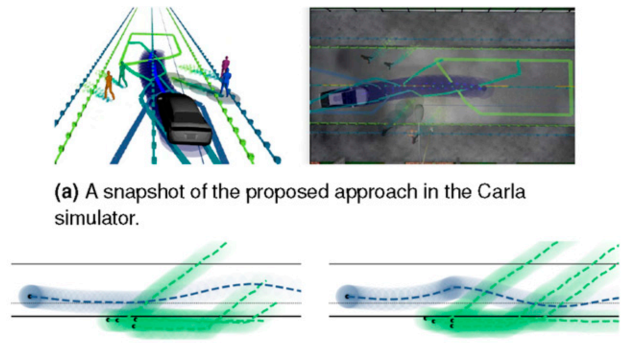


(a) The robot evades two pedestrians smoothly.

(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 11. Past trajectories of the robot and three pedestrians overlayed with top-view camera images. The robot consistently avoids collisions with pedestrians. (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.



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

(b) Case A: Safe overtake **(c)** Case B: Passing behind

Figure 12. (a) Snapshot from Carla simulations. Visualization follows Figure 4(d) 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. (a) A snapshot of the proposed approach in the Carla simulator. (b) Case A: Safe overtake. (c) Case B: Passing behind.

modeling errors and provide formal guarantees with respect to the true motion of the obstacles.

7.2. *Computational efficiency*

We showed that our approach 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, De Groot et al. (2021)), or by solving an approximate scenario optimization (Sartipizadeh and Açikmeşe, 2020). In addition, most computations of SH-MPC are parallel linear computations (for each sample), which potentially leave room for further optimization, for example, by delegating computations to a Graphical Processing Unit (GPU).

The computational efficiency of our approach derives from the merging of linear constraints which is facilitated by modeling collision regions as discs. While this modeling can be used to represent several obstacles, we currently cannot efficiently handle more complex shaped obstacles, such as objects with sharp corners. Future work could explore if other linear constraints (e.g., towards vertices of obstacles) could be merged safely. The same comment holds for extension to planning problems in higher dimensions. Directly merging halfspaces in 3D may, for example, be more complicated than in the 2D case and may require constraints of different shapes to be merged effectively.

7.3. *Support limit*

We showed that the support limit could be estimated online by using a solver that convexifies the problem in each iteration, such as SQP. Although this limits the generality of the support estimation, SQP solvers are a common choice for robotics applications. SH-MPC furthermore requires the user to estimate a support limit online because the support depends on the complexity of the problem to be solved. We do provide the online estimation of this single parameter, so that the user can bound it conservatively based on real-world data, enabling real-world robotics applications.

Future work could explore if the support limit could be known before optimization. For example, by adapting the constraint formulation so that the support is fixed by construction (see Appendix A in Campi et al. (2018)).

8. Conclusions

Safe Horizon Model Predictive Control (SH-MPC) enabled mobile robots to constrain the joint open-loop probability of collision over the planned trajectory and with respect to all obstacles *in real-time* for 2D dynamic environments. It does so without making simplifying assumptions on the robot dynamics or underlying probability distribution describing the motion of the 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, under our approach, could be computed before deploying the controller.

Our results, 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 we varied the underlying distribution. The overall risk of SH-MPC remained less conservative and more consistent between different environments and distributions, which resulted in faster trajectories. In addition, we showed excellent scaling of the computation time with respect to the number of obstacles and under varying distributions.

The real-time performance of SH-MPC is currently tied to a 2D environment with disc-shaped collision regions, while the probabilistic safety certification poses assumptions on the solver (satisfied by SQP) and requires the support limit to be estimated online. Our future work will try to alleviate these assumptions to further generalize our approach.

Acknowledgments

The authors would like to thank Bruno Brito for assisting in the simulated experiments.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

This work received support from the Dutch Science Foundation NWO-TTW, within the SafeVRU project (14667) and Veni project HARMONIA (18165), and the European Union, within the ERC Starting Grant INTERACT (101041863). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

ORCID iD

Oscar de Groot https://orcid.org/0000-0001-6527-7367

Supplemental Material

Supplemental material for this article is available online.

Note

1. See, for example, Billingsley (1995) for more details.

References

Aoude GS, Luders BD, Joseph JM, et al. (2013) Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Auton. Robots* 35(1): 51–76.

Aragón Artacho FJ, Campoy R and Tam MK (2020) The Douglas–Rachford algorithm for convex and nonconvex feasibility problems. *Math. Meth. Oper. Res* 91(2): 201–240.

Ben-Tal A and Nemirovski A (1998) Robust convex optimization. *Mathematics of Operations Research* 23(4): 769–805.

Billingsley P (1995) Probability and measure. In: *Wiley Series in Probability and Mathematical Statistics*. 3rd edition edition. New York: Wiley.

Blackmore L, Li H and Williams B (2006) A probabilistic approach to optimal robust path planning with obstacles. In *Proc. Amer. Control Conf*, 2831–2837.

Blackmore L, Ono M, Bektassov A, et al. (2010) A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics* 26(3): 502–517.

Brito B, Floor B, Ferranti L, et al. (2019) Model predictive contouring control for collision avoidance in unstructured dynamic environments. *IEEE Robot. Autom. Lett* 4(4): 4459–4466.

Brudermüller L, Berger G, Jankowski J, et al. (2024) CC-VPSTO: chance-constrained via-point-based stochastic trajectory optimisation for safe and efficient online robot motion planning. arXivURL. https://arxiv.org/abs/2402.01370.

Calafiore G and Campi M (2006) The scenario approach to robust control design. *IEEE Trans. Automat. Contr* 51(5): 742–753.

Campi MC and Garatti S (2008) The exact feasibility of randomized solutions of uncertain convex programs. *SIAM Journal on Optimization* 19(3): 1211–1230.

Campi MC and Garatti S (2011) A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality. *J. Optim. Theory Appl* 148(2): 257–280.

Campi MC, Garatti S and Ramponi FA (2018) A general scenario theory for nonconvex optimization and decision making. *IEEE Trans. Automat. Contr* 63(12): 4067–4078.

de Groot O (2024) Jupyter notebook for scenario dimensioning. URL. https://colab.research.google.com/drive/1Xye7960CZAlt4gHpoH6HOvok4-YuLFbD?usp=sharing.

de Groot O, Brito B, Ferranti L, et al. (2021) Scenario-based trajectory optimization in uncertain dynamic environments. *IEEE Robot. Autom. Lett* 6: 5389–5396.

de Groot O, Ferranti L, Gavrila D, et al. (2023) Globally guided trajectory planning in dynamic environments. In *IEEE Int. Conf. Robot. Autom*, 10118–10124.

de Groot O, Ferranti L, Gavrila D, et al. (2024) Scenario-based motion planning with bounded probability of collision. URL. https://www.youtube.com/watch?v=vyuNMO1giF0.

de Groot O, Sridharan A, Alons-Mora J, et al. (2024) *Probabilistic Motion Planning and Prediction via Partitioned Scenario Replay*. In IEEE International Conference on Robotics and Automation (ICRA).

Domahidi A and Jerez J (2014) *FORCES Professional*. Embotech AG. URL. https://embotech.com/FORCES-Pro.

Dosovitskiy A, Ros G, Codevilla F, et al. (2017) CARLA: an open urban driving simulator. In *Conf. On Robot Learning*, 1–16.

Fisac JF, Bajcsy A, Herbert SL, et al. (2018) *Probabilistically Safe Robot Planning with Confidence-Based Human Predictions*. RSS.

Frey K, Steiner T and How J (2020) Collision probabilities for continuous-time systems without sampling. In *RSS*.

Garatti S and Campi MC (2019) Risk and complexity in scenario optimization. *Mathematical Programming* 191: 243–279.

Garatti S and Campi MC (2023) 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, 26–31. URL. https://ieeexplore.ieee.org/document/10383474/. DOI: 10.1109/CDC49753.2023.10383474.

Janson L, Schmerling E and Pavone M (2018) Monte Carlo motion planning for robot trajectory optimization under uncertainty. In *Robotics Research: Volume 2, Springer Proceedings in Advanced Robotics*, 343–361.

Kong J, Pfeiffer M, Schildbach G, et al. (2015) Kinematic and dynamic vehicle models for autonomous driving control design. In *RSS*, 1094–1099.

Kooij JFP, Flohr F, Pool EAI, et al. (2019) Context-based path prediction for targets with switching dynamics. *Int. J. of Computer Vision* 127(3): 239–262.

Kouvaritakis B and Cannon M (2016) Model predictive control: Classical, robust and stochastic. In: *Advanced Textbooks in Control and Signal Processing*. Springer International Publishing.

Luders B, Kothari M and How J (2010) Chance constrained RRT for probabilistic robustness to environmental uncertainty. In *Proc. AIAA Guid., Navigat. Control Conf.*

Mesbah A (2016) Stochastic model predictive control: an overview and perspectives for future Research. *IEEE Control Systems Magazine* 36(6): 30–44.

Morgan AD (1847) *Formal Logic: or, the Calculus of Inference, Necessary and Probable*. Taylor and Walton.

Mustafa KA, de Groot O, Wang X, et al. (2023) Probabilistic risk assessment for chance-constrained collision avoidance in uncertain dynamic environments. *2023 IEEE International Conference on Robotics and Automation ICRA*. URL. https://arxiv.org/abs/2302.10846.

Nocedal J and Wright SJ (2006) Numerical optimization. In: *Springer Series in Operations Research*. 2nd edition edition. New York: Springer.

Ono M and Williams BC (2008) Iterative risk allocation: a new approach to robust model predictive control with a joint chance constraint. In 47th IEEE Conf. on Decision and Control, 3427–3432.

Paden B, Čáp M, Yong SZ, et al. (2016) A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. on Intelligent Vehicles* 1(1): 33–55.

Patil S, Berg J and Alterovitz R (2012) Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In IEEE Int. Conf. Robot. Autom, 3238–3244.

Ren K, Ahn H and Kamgarpour M (2023) Chance-constrained trajectory planning with multimodal environmental uncertainty. *IEEE Control Systems Letters* 7: 13–18.

Salzmann T, Ivanovic B, Chakravarty P, et al. (2021) Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. *Computer Vision – ECCV 2020*. Springer International Publishing.Vedaldi, Andrea and Bischof, Horst and Brox, Thomas and Frahm, Jan-Michael. 683-700 URL. https://arxiv.org/abs/2001.03093.

Sartipizadeh H and Açikmeşe B (2020) *Approximate convex hull based scenario truncation for chance constrained trajectory optimization*. Automatica.

Schildbach G, Fagiano L and Morari M (2013) Randomized solutions to convex programs with multiple chance constraints. *SIAM Journal on Optimization* 23(4): 2479–2501.

Schmerling E and Pavone M (2017) Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. In *RSS*.

Schwarting W, Alonso-Mora J, Paull L, et al. (2018a) Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Trans. Intell. Transp. Syst* 19(9): 2994–3008.

Schwarting W, Alonso-Mora J and Rus D (2018b) Planning and decision-making for autonomous vehicles. *Annu. Rev. Control Robot. Auton. Syst* 1(1): 187–210.

Siegwart R and Nourbakhsh IR (2011) *Introduction to Autonomous Mobile Robots*. 2 edition. The MIT Press.

Simon M (2019) Inside the Amazon warehouse where humans and machines become one. URL. https://www.wired.com/story/amazon-warehouse-robots/.

van den Berg J, Abbeel P and Goldberg K (2011) LQG-MP: optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research* 30(7): 895–913.

Walker J (2019) The self-driving car timeline - predictions from the top 11 global automakers. URL. https://emerj.com/ai-adoption-timelines/self-driving-car-timeline-themselves-top-11-automakers/.

Wang A, Huang X, Jasour A, et al. (2020) Fast risk assessment for autonomous vehicles using learned models of agent futures. In *RSS*.

Wang A, Jasour A and Williams BC (2020) Non-Gaussian chance-constrained trajectory planning for autonomous vehicles under agent uncertainty. *IEEE Robot. Autom. Lett* 5(4): 6041–6048.

Yue J, Manocha D and Wang H (2023) Human trajectory prediction via neural social physics. URL. https://arxiv.org/abs/2207.10435.

Zhu H and Alonso-Mora J (2019) Chance-constrained collision avoidance for MAVs in dynamic environments. *IEEE Robot. Autom. Lett* arXiv. 4(2): 776–783.

A Appendix

In practice, linearization of the collision avoidance constraints results in a support that consists of a small subset of the scenarios. To support this observation, we consider here how the constraint formulation impacts the support.

Definition 2. The *shadow* of a scenario $\boldsymbol{\delta}^{(i)}$ under the constraints $\boldsymbol{\theta} \in \Theta_{\boldsymbol{\delta}^{(i)}}$ is a region $S_{\boldsymbol{\delta}^{(i)}} \subseteq \Delta$ such that if another scenario satisfies $\boldsymbol{\delta} \in S_{\boldsymbol{\delta}^{(i)}}$, then either $\boldsymbol{\delta}^{(i)}$ or $\boldsymbol{\delta}$ is redundant. That is,

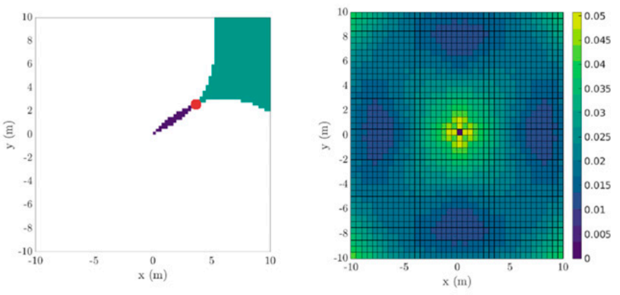
$$S_{\boldsymbol{\delta}^{(i)}} = \{\boldsymbol{\delta} \in \Delta \mid \Theta_{\boldsymbol{\delta}^{(i)}} \subset \Theta_{\boldsymbol{\delta}}\} \cup \{\boldsymbol{\delta} \in \Delta \mid \Theta_{\boldsymbol{\delta}} \subset \Theta_{\boldsymbol{\delta}^{(i)}}\}. \qquad (29)$$

An example shadow is visualized in Figure 13(a) for the proposed constraints. Note that in Figure 13(b), the shadow always occupies a non-zero area. This is a result of the linearization and a set of box constraints on the position that limit the range in which constraints are considered. In this case we can prove the following.

Theorem 3. Suppose that the shadow is non-empty for all possible samples in the support (i.e., $S_{\boldsymbol{\delta}} \neq \emptyset, \forall \boldsymbol{\delta} \in \Delta$), then the probability that out of S samples, none of the samples are redundant goes to 0 exponentially, for $S \to \infty$.

Proof. Given S samples, a new sample $\boldsymbol{\delta}$ is not redundant if it falls outside of the aggregated shadow, that is, if $\boldsymbol{\delta} \notin \cup_{i=1}^{S} S_{\boldsymbol{\delta}^{(i)}}$. The associated probability is

$$P_S = \mathbb{P}\left[\boldsymbol{\delta} \in \Delta \mid \boldsymbol{\delta} \notin \bigcup_{i=1}^{S} S_{\boldsymbol{\delta}^{(i)}}\right]. \qquad (30)$$



(a) A visualization for one 2D position in the trajectory of the shadow cast by the scenario positioned at $(2.5, 3.5)$.

(b) Area occupied by the shadow.

Figure 13. Empirical 2D visualizations of the shadow region for linearized constraints (14) considering a single 2D position in the trajectory. (a) Given a single scenario (red dot) positioned at (2.5, 3.5), shows what scenarios would dominate it (blue region) and what scenarios it dominates (green region). Regions were computed by sampling a new scenario over a grid and validating whether the two scenarios are active for any point in the box-constraints. (b) The area occupied by the shadow for each scenario within the box-constraints. (a) A visualization for one 2D position in the trajectory of the shadow cast by the scenario positioned at (2.5, 3.5). (b) Area occupied by the shadow.

Since $S_{\boldsymbol{\delta}} \neq \emptyset$, $\forall \boldsymbol{\delta} \in \Delta$, the aggregated shadow grows when sample $S+1$ is not redundant, that is, $\cup_{i=1}^{S} S_{\boldsymbol{\delta}^{(i)}} \subset \cup_{i=1}^{S+1} S_{\boldsymbol{\delta}^{(i)}}$. Therefore, $P_{S+1} < P_S$ and additionally $P_i < 1$ for $i > 0$. We obtain the following result,

$$\lim_{S \to \infty} \mathbb{P}^S[\text{No redundant scenarios}] = \lim_{S \to \infty} \prod_{i=1}^{S} P_i = 0, \qquad (31)$$

implying that the probability of sampling no redundant scenarios goes to zero. This probability is upper bounded by $(P_1)^S$, that is, it converges at least exponentially.

Although this does not prove that the support is bounded for finite S, it shows that it is very likely that redundant scenarios are sampled, which are not of support.

B Halfspace merging algorithm

This appendix details our algorithm for merging the set of halfspace constraints in equation (16) into a polygon with a minimal number of constraints. The algorithm is tailored to scale well to a large set of constraints. We make the following assumptions. First, we assume that box constraints are defined around the robot with finite side lengths $2R$, $R \in \mathbb{R}_{\geq 0}$. We further assume that the polygon is non-empty and that the robot position lies in its interior (which is checked before). Due to the convexity of the polygon, it is possible to divide halfspaces in those above and below the robot as viewed from above the robot at an arbitrary yaw angle. We describe here how to find the halfspaces that span the bottom half of the polygon, the same discussion holds for the top half. The halves are merged to obtain the final polygon. For simplicity, we assume that the robot is at the origin.

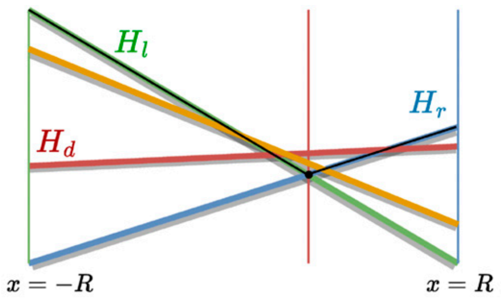
Algorithm 2: Constraint Merging Algorithm

```
1  Input: All halfspaces $\mathcal{H}$, initial halfspaces $H_l, H_r$
2  Function RecursiveSearch($H_i, H_j, \mathcal{H}$):
3      $(x_{ij}, y_{ij}) \leftarrow$ intersection of $H_i$ and $H_j$
4      $\hat{\mathcal{H}} \leftarrow \{H_h \in \mathcal{H} : H_h(x_{ij}) > y_{ij}\}$
5      if $\hat{\mathcal{H}} = \emptyset$ then
6          return $[H_i, H_j]$
7      $d \leftarrow \text{argmax}_h H_h(x_{ij}), \forall H_h \in \hat{\mathcal{H}}$
8      $\mathcal{H}^L_{\text{out}} \leftarrow$ RecursiveSearch($H_i, H_d, \hat{\mathcal{H}}$)
9      $\mathcal{H}^R_{\text{out}} \leftarrow$ RecursiveSearch($H_d, H_j, \hat{\mathcal{H}}$)
10     return $[H_i, \mathcal{H}^L_{\text{out}}, \mathcal{H}^R_{\text{out}}, H_j]$
```

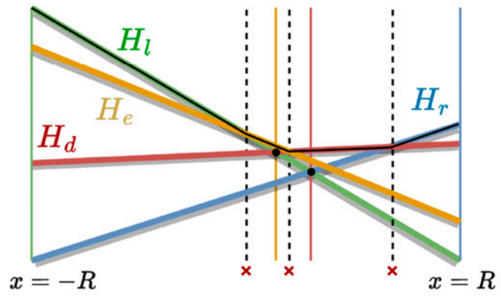
Algorithm 2 describes the search and Figure 14 gives an example. The main idea is to start with two halfspaces H_l, H_r that are in the polygon at the left and right extremes of the search domain, respectively. With the y-value of halfspace h at x denoted as $H_h(x)$, these are indexed by

$$l = \text{argmax}_h H_h(-R), \quad r = \text{argmax}_h H_h(R). \qquad (32)$$

The remainder of the procedure is applied recursively for any two halfspaces H_i, H_j that are in the polygon at the boundaries of a particular search domain. A key observation is that, for another halfspace to be larger at any considered x



(a) Search after one recursion (up until Line 8 of Algorithm 2). Halfspaces H_i and H_j are identified first. At their intersection, H_d has the largest y-value, splitting the search domain in two.



(b) After H_e is added on the left and the domain is split once more, all branches terminate and the final polygon (black solid lines) is returned.

Figure 14. Illustrative example of the halfspace intersection algorithm with four halfspaces (lines with feasible side colored). Splits in the search domain are denoted by solid lines colored with the largest halfspace at that point, dashed vertical lines denote terminated recursions. (a) Search after one recursion (up until Line 8 of Algorithm 2). Halfspaces H_i and H_j are identified first. At their intersection, H_d has the largest y-value, splitting the search domain in two. (b) After H_e is added on the left and the domain is split once more, all branches terminate and the final polygon (black solid lines) is returned.

than these two halfspaces, it must be larger at their intersection (a more formal statement will follow). We therefore compute the intersection of the given halfspaces (Line 3) and identify the largest halfspace H_d at x_{ij} (Line 7). If $d = i$ or $d = j$, no halfspace is larger in the search domain and the algorithm terminates. Otherwise, we obtain two new search domains, consisting of the pairs H_i, H_d and H_d, H_j (see Figure 14(a)). We repeat this process recursively (Lines 8 and 9) until all recursions are terminated (see Figure 14(b)). The returned aggregated set of halfspaces (excluding duplicates) makes up half the polygon.

To formalize that checking halfspaces at intersections is sufficient, we provide the following result.

Theorem 4. Given a set of halfspaces $\mathcal{H}$ and the halfspaces H_i and H_j with the highest y-values at the left and right extremes of x-domain $[x_i, x_j]$, respectively. If for any non-empty subdomain $x \in \mathcal{X}_d \subset [x_i, x_j]$ there exists a halfspace $H_d \in \mathcal{H}$ for which $H_d(x) > H_i(x)$ and $H_d(x) > H_j(x)$ then the intersection of H_i and H_j is in that set, that is, $x_{ij} \in \mathcal{X}_d$.

Proof. Assume that the theorem does not hold, such that $H_d(x_{ij}) < H_i$. To be largest on the left side of the intersection, H_d has to intersect H_i for some $x \in [-R, x_{ij}]$. Left of that intersection, $H_d(x) > H_i(x)$, which implies $H_d(-R) > H_i(-R)$. But since $H_i(-R) > H_d(-R)$ by definition we have contradiction. A similar conclusion is obtained for H_j. Therefore, H_d must be larger at the intersection and the result holds.

The computational efficiency of this algorithm comes from an additional observation. When the search domain is split, only halfspaces whose y-values are between the evaluated intersection point (e.g., y_{ij}) and the largest constraint there (e.g., $H_d(x_{ij})$) remain relevant in ongoing searches. All other constraints can be pruned (Lines 4, 8 and 9). Consequently, only a few halfspaces have to be checked in each recursion after the first steps.