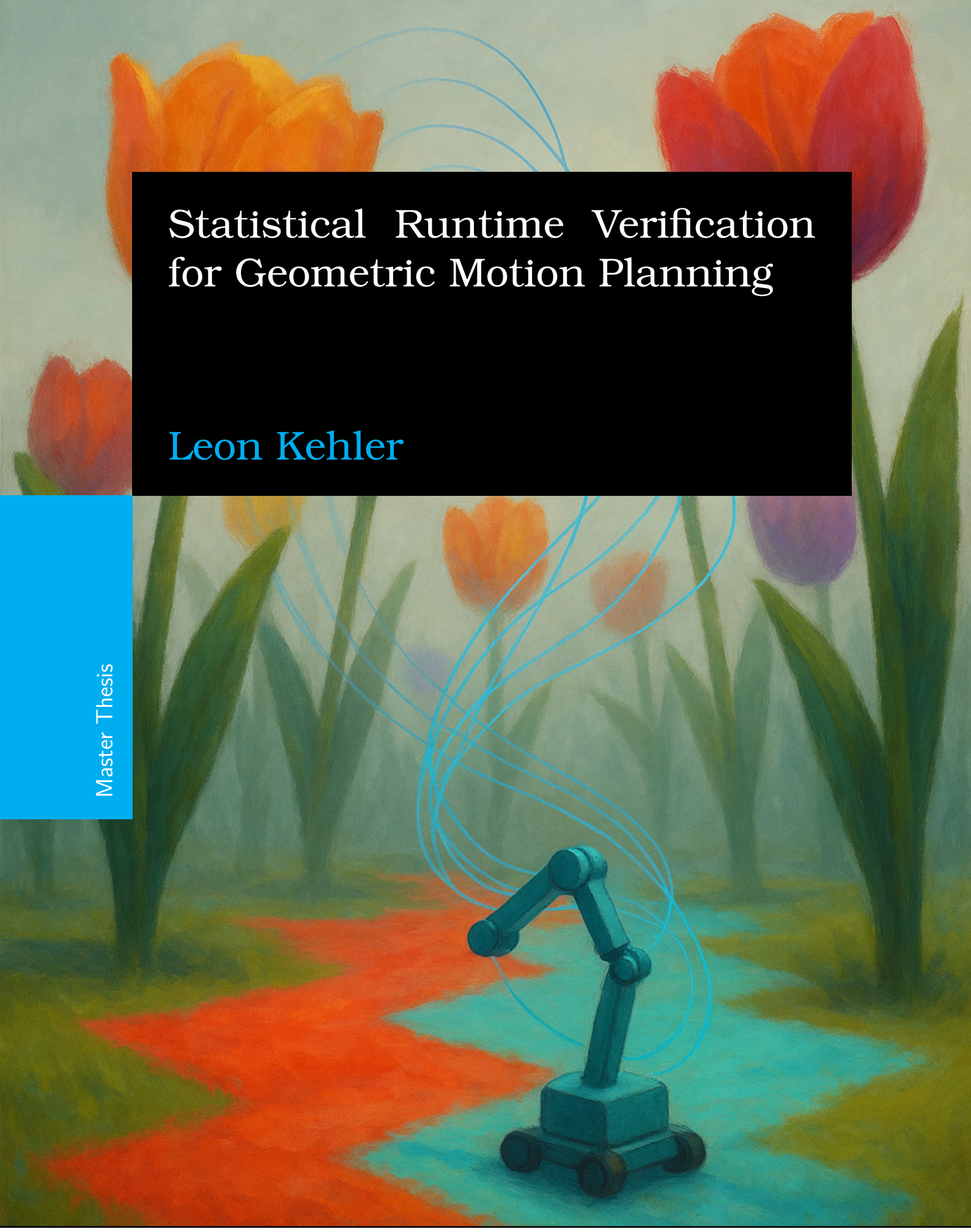
Statistical Runtime Verification for Geometric Motion Planning

Leon Kehler

Master Thesis



TU Delft Delft University of Technology

Cognitive Robotics, and
Delft Center for Systems and Control

Statistical Runtime Verification for Geometric Motion Planning

Master Thesis

Leon Kehler

December 17, 2025

Faculty of Mechanical Engineering · Delft University of Technology

Master thesis supervised by:
Saray Bakker (PhD candidate at CoR)
Prof. Dr. Javier Alonso-Mora (Full Professor at CoR)
Dr. Manuel Mazo (Associate Professor at DCSC)

Delft University of Technology

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical Engineering for acceptance a thesis entitled

Statistical Runtime Verification for Geometric Motion Planning

by

Leon Kehler

in partial fulfillment of the requirements for the degrees of

Master of Science in Robotics

and

Master of Science in Systems and Control.

Dated: December 17, 2025

Supervisors:

Prof. Dr. Javier Alonso-Mora

Dr. Manuel Mazo Jr.

Saray Bakker, M.Sc.

Other Committee Members:

Dr. Laura Ferranti

Dr. Luca Laurenti

Dr. Raj T. Rajan

Abstract

As robots increasingly operate in human environments, their controllers must ensure safe and reliable behavior under real-time constraints. Although optimization-based motion planners can enforce hard safety constraints, their computational demands limit their use on complex robotic platforms. Geometric motion planning offers a real-time alternative through optimization-free, closed-form control laws with reach–avoid guarantees. However, these guarantees rely on assumptions about obstacle representations that are often violated in realistic settings. When such assumptions fail, the planner's dynamical system may preserve invariance of the safe set but lose global attractivity, jeopardizing goal reachability.

This thesis introduces a runtime verification algorithm, *Scenario-Shield*, that adapts the geometric planner's underlying dynamical system to expand its finite-time region of attraction. The method periodically samples nearby robot configurations and performs forward simulations to approximate this region. To accelerate this process, the approach is extended by incorporating statistical uncertainty quantification: conformal prediction is used to calibrate a fast membership test for candidate states, and the scenario approach provides a principled approximation of an uncountably infinite subset of the region of attraction.

To maintain computational efficiency, the algorithm is implemented using parallel computing and integrated into a geometric motion planning toolbox with robot operating system (ROS). The proposed method is validated in simulation on both a holonomic ground robot and a mobile manipulator, demonstrating improved reliability over baseline geometric fabrics controllers.

Table of Contents

Acknowledgements

Thank you to the wonderful friends who made this journey far more enjoyable. In particular, thank you, David, for our dramatic journey through the MPC course. Thank you, Maria, for becoming my friend through a bad peer review. Thank you, Seppe, for making our "Mars" rover work. And thank you, Srujan, for introducing me to the random strategy in Catan. Also, of course, thank you to all my friends back home.

Thank you to Emilia for making me laugh every day and supporting me despite how much I enjoyed studying. Thank you to my family. Also, thank you, Mr. Kaser; without you, I probably would not have chosen this path for my studies.

I would like to thank my daily supervisor, Saray Bakker, for all the insightful discussions and guidance throughout this project. Furthermore, I want to thank my main supervisors, Prof. Dr. Alonso-Mora and Dr. Mazo, who taught me a lot and allowed me to truly enjoy my studies. Finally, I would like to thank everyone involved in the study trips to the U.S., which helped connect the topics of robotics, systems, and control in new ways.

Delft, University of Technology — Leon Kehler
December 17, 2025

Chapter 1

Introduction

In recent years, robots have increasingly entered human-shared environments. Whereas traditionally confined to structured factory settings, they now appear as autonomous vehicles, delivery robots, or household assistants, such as vacuum cleaning robots. Such robots typically either operate under well-defined safety rules, e.g., traffic laws for autonomous vehicles, or, in the example of vacuum robots, involve low-risk interactions with their environment.

However, emerging robotic applications, such as humanoids or mobile manipulators, may operate in environments that lack clearly defined rules, yet are safety-critical, such as an operating room or a professional kitchen. In these environments, humans and robots must collaborate closely, often in unstructured and dynamic settings. While these developments offer desperately needed productivity gains, in light of our demographic trends, they simultaneously demand stringent safety guarantees. Safety through conservatism, however, can impair a robot's utility in practice. In a highly dynamic professional kitchen, overly cautious behavior may cause the robot to stall or block workflows, reducing overall efficiency. Conversely, in safety-critical contexts like operating rooms, insufficient responsiveness may compromise procedural outcomes.

To optimally trade-off between performance and safety, engineers often formulate the motion planning problem as a constrained optimization problem. This typically enforces hard safety constraints while optimizing a performance-related cost function. Due to the large decision variable space, this is commonly solved online in a receding horizon fashion known as model predictive control (MPC). However, the original problem is often intractable to solve at a sufficient frequency online. Hence, in practice, simplifications are made, e.g., the fidelity of the predictive model is reduced, or the representation of obstacles is simplified, yielding easier-to-check constraints, but at the expense of overapproximating obstacles, such that feasibility may be lost. On the other hand, sticking to a precise formulation of the robot dynamics and a tight representation of constraints increases the computational cost of the optimization problem. This will

render a significant lower bound on the allowable discrete time step; effectively abstracting the true robot environment. Constraints will only be enforced at the discrete-time steps, but the robot's true state evolves in continuous time, such that continuous-time feasibility is not guaranteed.

In explicit MPC, the optimization problem is no longer solved online, but the solution of the constrained optimization problem is pre-computed in a quantized state space [1]. However, to be tractable in terms of storage, a compromise must be made with the resolution of the quantization, effectively limiting the ability to parameterize the problem. In the above-mentioned scenarios, the motion planning problem must deal with a varying number and locations of obstacles, as well as different goal configurations. An alternative approach to obtaining an optimization-free, closed-form control law is to leverage the problem's geometry. This was initially pioneered in navigation functions [2], artificial potential fields [3], and later generalized in feedback motion planning [4]. Effectively, these methods find an artificial dynamical system whose trajectories solve the motion planning problem; with the advantage that their solution can be queried from anywhere in the state space and their closed-form allows for fast evaluations, thus, reactive planning.

However, there is, of course, no free lunch: reaching the goal and avoiding collisions are often competing objectives. First, these geometric approaches do not incorporate an explicit cost function and therefore cannot guarantee optimality with respect to a chosen performance metric. Nonetheless, recent work such as [5] has shown that their structure is particularly suited for imitation-learning formulations, partially recovering control over the resulting behavior. Second, motion-planning problems commonly require reach–avoid guarantees, which cannot be satisfied globally by such geometric planners. In particular, the synthesized artificial dynamical systems are only almost[1] globally asymptotically stable under assumptions on the environment that often do not hold in practice. Some methods require obstacles to be star-shaped [6], while others rely on finding a smooth obstacle enclosing function [7]. Moreover, geometric fabrics [8], which have gained traction especially for mobile manipulators, require obstacle-repulsive forces to vanish at sufficiently low velocities. As a consequence, they cannot simultaneously guarantee both reaching the goal and avoiding collisions.

When these assumptions cannot be satisfied in practice, the resulting feedback motion planner can guarantee only local asymptotic stability, because the set of initial conditions that do not flow to the global minimum is of positive measure. As a consequence, the reactive planner may get stuck while attempting to balance safety and performance, and some trajectories may fail to converge to the goal. In practice, global planners can guide these local geometric planners [9]. However, a global planner cannot reliably guide the system without explicit knowledge of the local planner's feasible domain, which, in a dynamical-system representation, corresponds to the region of attraction. This prevents establishing a meaningful assume–guarantee relationship, which is required for a principled global–local decomposition.

[1] Almost refers to attractivity of an equilibrium from almost all initial conditions except those on a set of Lebesgue measure zero. Sampling an initial state from a measure-zero set has probability zero; hence, for such systems, a random initial state will reach the equilibrium with almost certainty.

Motivated by these limitations caused by the local nature of geometric planners, this thesis explores how to accommodate the fading of reach-avoid guarantees of such planners while maintaining their reactive properties. The core idea is to equip a high-frequency, reactive planning loop with a parallel runtime verification loop that estimates the geometric planner's region of attraction and, when needed, adapts the underlying artificial dynamical system to keep the system within it.

This framework is instantiated using geometric fabrics [10], which provide a systematic way to compose artificial dynamical systems for high-dimensional, nonlinear robotic systems such as mobile manipulators. Geometric fabrics naturally support the composition of multiple sub-behaviors, such as goal reaching, obstacle avoidance, joint-limit avoidance, and posturing, across different task dimensions.

However, while the synthesis of geometric fabrics is interpretable and intuitive, the resulting controller dynamics are highly nonlinear, and no analytic characterization of their region of attraction is available in general. This observation shapes the proposed runtime-verification approach: the algorithm relies on sampled rollouts to estimate the region of attraction with a probabilistic correctness guarantee. This sampling and rollout procedure exploits the inherent parallelizability of autonomous dynamical system rollouts, an aspect that previous approaches have not leveraged.

Contributions

This thesis addresses a fundamental limitation of geometric motion planners: their reach–avoid guarantees typically rely on stringent assumptions about the environment that rarely hold in practice. Consequently, geometric planners may fail to reach the goal when deployed in realistic settings. To address this issue, this thesis introduces a runtime verification algorithm called *Scenario-Shield*. The algorithm periodically adapts the geometric planner's underlying dynamical system using estimates of its finite-time region of attraction, drawing on statistical uncertainty quantification, namely, conformal prediction and the scenario approach.

Beyond the theoretical contributions, the computational demands of the algorithm were reduced by implementing batched rollouts in `JAX`, and the full system was integrated into the Autonomous Multi-Robots Lab's geometric fabrics toolbox using robot operating system (ROS). This was, moreover, validated in simulation for a holonomic ground robot and a mobile manipulator robot.

1. Proposal of the runtime verification algorithm *Scenario-Shield* to reason about reachability of geometric motion planning through statistical methods and adapting the geometric planner's underlying dynamical system.
2. Applying conformal prediction to faster compute whether a sampled state belongs to the finite-time region of attraction.
3. Using the scenario approach to compute an uncountably infinite set approximation of the region of attraction from finite trajectory samples.

4. Fast batch rollouts implemented into the fabrics toolbox with `JAX`.
5. ROS implementation of the *Scenario-Shield* algorithm and integration into the fabrics toolbox.
6. Simulation experiments to compare the vanilla geometric fabrics implementation for a holonomic ground robot and a mobile manipulator.

Outline

This thesis comprises six chapters, including this introduction. Chapter 2 briefly introduces feedback motion planning, specifies its difficult-to-satisfy assumptions, and reviews concepts of statistical uncertainty quantification.

Next, Section 3-1 foreshadows the main ideas of the proposed verification algorithm, which are then motivated and developed in the main body of Chapter 3. Chapter 4 explains how the algorithm was implemented using parallel computing and ROS. Chapter 5 shows simulation experiment results from applications to a ground robot and a mobile manipulator. Lastly, Chapter 6 concludes the thesis, discusses its limitations, and gives an outlook into future work directions.

Chapter 2

Related work

This chapter begins by briefly introducing notation for feedback motion planning in Section 2-1. Subsequently, subsection 2-1-1 reviews constructive geometric methods that formulate the feedback law as a dynamical system. Building on this, subsection 2-1-2 presents geometric fabrics, a specific feedback motion planner that serves as the baseline method in this thesis and, moreover, as the nominal geometric planner wrapped by the runtime verification loop. Finally, because the verification draws on statistical uncertainty quantification, Section 2-2 introduces the required preliminaries, namely conformal prediction and the scenario approach.

2-1 Feedback motion planning

The robotic motion planning problem seeks to compute a feasible trajectory from an initial configuration to a goal configuration while avoiding obstacles. As outlined in Chapter 1, safety-critical robotic applications in dynamic environments often benefit from closed-form solutions. [4] introduces the class of feedback motion planners, which do not explicitly generate a full trajectory but instead provide a feedback law that, under some assumptions, upon integration produces a trajectory that solves the motion planning problem. Moreover, the feedback law serves as a local motion reference, which can be tracked by the robot. Because this feedback law is a closed-form function of the robot's state, possibly parameterized by the environment, it enables high-frequency replanning and thereby reactive behavior even in dynamic settings.

Following the notation of [4], feedback motion planning seeks a feedback law $\boldsymbol{u} = k(\boldsymbol{x}) \in \mathcal{U}$ for every state $\boldsymbol{x} \in \mathcal{X}$, which drives the dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{u}$. Importantly, the space $\mathcal{X}$ need not coincide with the state space of the underlying physical system. The challenge lies in designing the feedback law k, such that $\mathcal{X}$ can be constrained, e.g., by setting it equal to the free configuration space $\mathcal{C}_{\text{free}}$, while simultaneously ensuring that

all trajectory eventually reaches the goal set $\mathcal{X}_G \subset \mathcal{X}$. The synthesis of such feedback laws is the focus of subsection 2-1-1, where selected approaches are reviewed.

Even before imposing additional assumptions required for specific synthesis methods, the feedback-motion-planning formulation relies on idealizations that are rarely satisfied in practice:

- The control input space $\mathcal{U}$ is assumed to contain the zero vector $\mathbf{0}$. However, systems with inertia cannot instantaneously stop, introducing a tracking error for most physical systems.
- Although a trajectory $\boldsymbol{x}(t)$, for $t \in [0, T]$, generated by the continuous-time system $\dot{\boldsymbol{x}} = k(\boldsymbol{x})$ may theoretically remain in $\mathcal{C}_{\text{free}}$ and reach $\mathcal{X}_G$, in practice trajectories must be computed numerically. This limitation arises because discrete-time closed-form solutions are unavailable, and because sensing and perception modules, such as frame-based vision processing, typically update discretely in time.

2-1-1 Geometric motion planning

A substantial body of work investigated how to design $\dot{\boldsymbol{x}} = k(\boldsymbol{x})$, such that the resulting trajectories satisfy a desired specification. Typically, each subformula of such a specification falls into either a reach or a viability category. In robotics, reaching concerns completeness, i.e., the method reaches the goal if possible, while viability refers to maintaining constraint satisfaction, such as collision and joint-limit avoidance. This section focuses on approaches that leverage the geometry of the environment to construct artificial dynamical systems, as opposed to modeling physical dynamical systems from first principles.

Artificial potential fields [3] approach the feedback law synthesis by constructing an artificial potential function $\Phi : \mathcal{X} \to \mathbb{R}_{\geq 0}$, typically formed as the sum of repulsive and attractive potentials for obstacle avoidance and goal reaching, respectively. The corresponding feedback law is then given by the negative gradient $k(\boldsymbol{x}) = -\nabla\Phi(\boldsymbol{x})$. A well-known drawback of this method is that such additive compositions induce a topology with many local minima into which trajectories, following the steepest gradient descent, may flow. Randomized potential fields attempt to escape such local minima by temporary random walks [11]. Subsequent work, however, sought to reduce the number of local minima by construction.

Navigation functions, initially introduced by [2], addressed the problem of obtaining feedback laws which merely yield local attractivity to the goal set $\mathcal{X}_G$. Their result, under certain assumptions, a vector field can be constructed which is almost globally attractive, that is, for all initial states $\boldsymbol{x}_0 \in \mathcal{X}$, except those on a set of Lebesgue measure zero. Hence, a randomly chosen initial condition flows into the goal with almost certainty. This result holds for general sphere worlds under the assumption that the obstacles' closure is disjoint [12, p.507]. This assumption limits its applicability, because complex obstacles can no longer be approximated as the union of multiple intersecting spheres, but instead, must be overapproximated as large, disjoint spheres.

Concrete implementations of navigation functions, address nonholonomic dynamics [13], dynamic environments [14], and the multi-agent setting [15]. However, in order to attain guarantees for at least almost global attractivity, certain assumptions are necessary. Specifically, [6] shows how to generalize the initial navigation functions to environments with convex and specific concave obstacles, which must be star-shaped[1]. However, a robot may encounter obstacles that are not star-shaped, and hence, must be overapproximated. [7] retains the guarantees of [6] while allowing the obstacles to be an algebraic set to address non-star-shaped obstacles. Furthermore, the authors achieve global asymptotic convergence of the flow to the goal by employing a switching mechanism. However, this method requires finding a smooth function for each obstacle to enclose it within a level set, which may not always be possible or may lead to conversivism; in particular, when the environment is not immediately observable.

Lastly, the above methods primarily apply to holonomic point-mass robots, for which the workspace and configuration space are equivalent. The work of geometric fabrics [10] addresses the geometric motion planning problem for robots with nonlinear kinematics, such as manipulators. However, geometric fabrics also make an assumption that repulsive forces of obstacles must vanish at low velocities in order to uphold a guarantee of attractivity. Again, an assumption that is not in agreement with the objectives of safety-critical control in human-shared environments as outlined in Chapter 1.

Therefore, this thesis addresses how to verify geometric motion planning methods, which enable the construction of (almost) globally attractive feedback laws under strict assumptions, and what can be retained when these assumptions are violated. This is furthermore applied to a mobile manipulator robot; hence, the line of geometric fabrics is introduced in more detail in the next subsection.

2-1-2 Geometric fabrics for local motion planning

Geometric fabrics, as introduced in [10], are second-order differential equations of the following shape

$$\boldsymbol{M}(\boldsymbol{q},\dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \boldsymbol{\xi}(\boldsymbol{q},\dot{\boldsymbol{q}}) = -\partial\psi(\boldsymbol{q}) - \boldsymbol{B}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \tag{2-1}$$

$$\Leftrightarrow \ddot{\boldsymbol{q}} = -(\underbrace{\boldsymbol{M}^{-1}\boldsymbol{\xi}}_{-\tilde{h}(\boldsymbol{q},\dot{\boldsymbol{q}})} + \underbrace{\boldsymbol{M}^{-1}(\partial_{\boldsymbol{q}}\psi + \boldsymbol{B}\dot{\boldsymbol{q}})}_{-f(\boldsymbol{q},\dot{\boldsymbol{q}})}), \tag{2-2}$$

where $\boldsymbol{q} \in \mathcal{C}$ is the configuration vector with first and second time derivative $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$, respectively. The symmetric and invertible priority matrix $\boldsymbol{M}(\boldsymbol{q},\dot{\boldsymbol{q}})$, the damping matrix $\boldsymbol{B}(\boldsymbol{q},\dot{\boldsymbol{q}})$ depend on both the configuration and its velocity. $\psi(\boldsymbol{q})$ is a potential function. The acceleration signal is a sum of a energy-preserving geometry $\tilde{h}(\boldsymbol{q},\dot{\boldsymbol{q}})$, which determines the unforced flow of the system, and the forcing term $f(\boldsymbol{q},\dot{\boldsymbol{q}})$, which can be seen as an energy regulating term that dissipates energy from the system until the trajectory

[1] A star-shaped obstacle $S \subset \mathcal{X}$ has a kernel point $\boldsymbol{s}_0 \in S$, such that the line-segment between any point $\boldsymbol{s} \in S$ and the kernel point lies in S, i.e. $\exists \boldsymbol{s}_0$ such that $\{(1-\lambda)\boldsymbol{s}_0 + \lambda\boldsymbol{s} \mid \lambda \in [0,1], \forall \boldsymbol{s} \in S\} = S$.

has flown into a minimum of the potential function $\psi(\boldsymbol{q})$, e.g., $\psi(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{q}_G||_2^2$, where $\boldsymbol{q}_G$ is a goal configuration. Next follows the constructive mechanism to arrive at such a differential equation, which, according to [10], exhibits great expressivity for mobile manipulation planning.

A geometry, as later introduced for different sub-behaviors, is not necessarily energy preserving, but can be made to conserve the Finsler energy $\mathcal{L}$, rendering the energy conserving geometry, also known as a fabric $\tilde{h}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ through the energize operation:

$$\tilde{h} = \text{energize}_{\mathcal{L}}(h(\boldsymbol{q}, \dot{\boldsymbol{q}})) = h + \alpha \dot{\boldsymbol{q}}, \qquad \text{where } \alpha = -\frac{\dot{\boldsymbol{q}}^\top (\boldsymbol{M}_{\mathcal{L}} h(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \xi_{\mathcal{L}})}{\dot{\boldsymbol{q}}^\top \boldsymbol{M}_{\mathcal{L}} \dot{\boldsymbol{q}}}. \tag{2-3}$$

The forcing term $f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ simultaneously navigates the potential $\psi(\boldsymbol{q})$ via gradient descent and dissipates energy through the damping term $\boldsymbol{B}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ while obeying the priority metric $\boldsymbol{M}(\boldsymbol{q}, \dot{\boldsymbol{q}})$. For example, a holonomic ground robot with $\boldsymbol{q} \in \mathbb{R}^2$ may have a baseline fabric $\tilde{h}_b(\boldsymbol{q}, \dot{\boldsymbol{q}})$ of

$$\tilde{h}_b(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \text{energize}_{\mathcal{L}_b}(h_b), \qquad \text{where } h_b = \boldsymbol{0}, \quad \mathcal{L}_b = \text{Lagrangian}\left(\frac{1}{2}||\dot{\boldsymbol{q}}||_2^2\right), \tag{2-4}$$

where the Langrangian is obtained via [16]. For circular obstacle avoidance, we may employ a barrier-like geometry that ensures that the robot's Euclidean distance to the obstacle border is positive. Note that it does not live in the same space as the $\boldsymbol{q} \in \mathcal{R}^2$, but instead in a space more convenient for defining collision geometries in the single dimension indicating the Euclidean distance to the obstacle border, i.e., $\boldsymbol{x} \in \mathbb{R}_{\geq} 0$:

$$\tilde{h}_o(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \text{energize}_{\mathcal{L}_o}(h_o(\boldsymbol{x}, \dot{\boldsymbol{x}})), \tag{2-5}$$

$$\text{where } h_o(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \frac{\dot{\boldsymbol{x}}^2}{\boldsymbol{x}^2 + \epsilon} \text{sign}(\dot{\boldsymbol{x}} - 1), \quad \mathcal{L}_o = \text{Lagrangian}\left(\frac{2\dot{\boldsymbol{x}}^2}{\boldsymbol{x}^2 + \epsilon}\right), \tag{2-6}$$

with ϵ being a small positive constant to avoid division by zero. Differential equations as shown in Equation 2-1, are known as semi-spectral sprays, which are endowed with a so-called spec algebra. This algebra includes the operations of summation and pullback. The former can be used to combine geometries (weighted by their priority metrics), however, in this example, $\tilde{h}_b(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\tilde{h}_o(\boldsymbol{x}, \dot{\boldsymbol{x}})$ live in different spaces. The pullback operation, as introduced in [17], allows bringing both fabrics onto the same manifold, provided that a differentiable map exists. Then, summation can be applied to combine the behaviors. Then for any fabric defined on $\mathcal{X}$, namely $(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}}$, and given a differentiable map $\phi : \mathcal{Q} \to \mathcal{X}$ and its Jacobian $\boldsymbol{J} = \frac{\partial \phi}{\partial \boldsymbol{q}}$ can be pulled onto $\mathcal{Q}$ via

$$\text{pull}_{\phi}(\boldsymbol{M}, \boldsymbol{f})_{\mathcal{X}} = \left(\boldsymbol{J}^T \boldsymbol{M} \boldsymbol{J}, \boldsymbol{J}^T (\boldsymbol{f} + \dot{\boldsymbol{J}} \dot{\boldsymbol{q}})\right)_{\mathcal{Q}}, \tag{2-7}$$

where $\dot{J} = \frac{\partial}{\partial t} \boldsymbol{J} = \frac{\partial \boldsymbol{J}}{\partial \boldsymbol{q}} \cdot \frac{d\boldsymbol{q}}{dt}$ and any dependency on $\boldsymbol{x}$ on the right hand side is substituted by $\boldsymbol{x} = \phi(\boldsymbol{q})$. Applied to the obstacle fabric $\tilde{h}_o(\boldsymbol{x}, \dot{\boldsymbol{x}})$, the following pullback operation could be performed, where the differentiable map follows from the Euclidean distance between the configuration $\boldsymbol{q} = \begin{bmatrix} q_x & q_y \end{bmatrix}^\top$ and the obstacle position $\boldsymbol{p}_o = \begin{bmatrix} o_x & o_y \end{bmatrix}^\top$, as

$$\phi(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{p}_o||_2^2 \qquad \boldsymbol{J} = \frac{\partial \phi}{\partial q} = \left[\frac{q_x - o_x}{||\boldsymbol{q}-\boldsymbol{p}_o||_2^2} \quad \frac{q_y - o_y}{||\boldsymbol{q}-\boldsymbol{p}_o||_2^2}.\right] \tag{2-8}$$

This summarizes the underlying geometric motion planning policy employed for the remainder of the thesis. However, the resulting dynamical system is not globally attractive. To attempt to adapt the geometric fabrics during runtime, we will need to reason about their local region of attraction. Due to the possibly high nonlinearity and absence of explicit characterizations of such regions, a sampling-based approach will be paired with tools from uncertainty quantification. The preliminaries of these methods are introduced below.

2-2 Preliminaries

The proposed runtime verification algorithm, *Scenario-Shield*, draws on tools from uncertainty quantification, specifically conformal prediction and the scenario approach, which are introduced in this section.

2-2-1 Conformal prediction

Conformal prediction (CP), introduced by [18], is a distribution-free approach to quantify the uncertainty of a predictor, which may be any model or algorithm producing a point prediction. CP then quantifies the uncertainty by quantifying a coverage region around the output, within which the ground truth lies with high probability.

Given K calibration data points $R^{(1)}, \ldots, R^{(K)}$, an exchangeable test point $R^{(0)}$ lies within the $1-\delta$ quantile of the calibration data C with probability $1-\delta$, i.e.,

$$\mathbb{P}\left(R^{(0)} \leq C(R^{(1)}, \ldots, R^{(K)}, \infty)\right) \geq 1 - \delta, \tag{2-9}$$

where the added ∞ ensures that the guarantee holds even for finite samples; this works by putting a lower bound on the number of calibration samples K to render a useful coverage region, i.e., $K > \lceil (K+1)(1-\delta) \rceil$, as otherwise trivially $C = \infty$.

The data points are random variables which follow a pushforward measure of the predictor's output $\hat{y}$ and ground truth data y through a chosen non-conformity score function. An example of a non-conformity score is the absolute prediction error,

$$R^{(i)} = \mid y - \hat{y} \mid . \tag{2-10}$$

While CP is distribution-free, in the sense that it does not assume a specific probabilistic model as in Bayesian methods, it relies on the assumption that the nonconformity scores of the calibration and test data are exchangeable. Exchangeability implies identically

distributed, hence, a concern of CP is when the nonconformity scores follow different distributions between calibration and test time.

Recent applications of CP include the conformal Kalman filter, which leverages conformal methods to obtain probably correct state estimates despite violations of the Kalman filter's Gaussian noise assumptions [19]. In Section 3-2 CP will be used to conformalize a predictor, i.e., obtain a coverage region within which the true value will lie with high probability, that classifies the $K-$step attractivity of a state x based on a rollout of length k, with $k \ll K$.

2-2-2 Scenario approach

The mentioned sampling-based approximation of a region of attraction renders a finite subset. However, the underlying region of attraction may be only characterizable by infinite sets. When approached from an optimization-based perspective, which involves finding the region of attraction by optimizing a parameterized set, a related problem arises. Correctly constraining the optimization problem may result in a semi-infinite optimization problem, i.e., a finite number of decision variables but an infinite number of constraints.

This, however, is computationally intractable to solve. The scenario approach, as introduced by [20], addresses this by sampling a finite number of constraints, known as scenarios, and reasoning about the uncertainty of the obtained solution for new scenarios. The underlying idea is that, under assumptions, an optimal solution obtained for a sufficiently large number of sampled constraints will remain feasible for newly sampled constraints. Specifically, the semi-infinite problem

$$x^* = \underset{x \in \mathcal{X} \subseteq \mathbb{R}^d}{\arg\min} \quad f(x) \tag{2-11}$$

$$\text{s.t.} \quad g(x, \delta) \leq 0, \quad \forall \delta \in \Delta, \tag{2-12}$$

where f is convex in x, the number of decision variables is finite, i.e., $d \in \mathbb{N}$, and the feasible set is convex. This problem can be converted into a scenario program, by sampling N scenarios $\delta^{(1)}, \ldots, \delta^{(N)}$ from Δ

$$x_N^* = \underset{x \in \mathcal{X} \subseteq \mathbb{R}^d}{\arg\min} \quad f(x) \tag{2-13}$$

$$\text{s.t.} \quad g(x, \delta^{(i)}) \leq 0, \quad \forall i \in \{1, \ldots, N\}. \tag{2-14}$$

Then, if the problem is convex, the solution is unique, and the scenarios are independent and identically distributed (i.i.d.), the probability that any newly sampled scenario $\delta^{(N+1)}$ constraint pushes x_N^* into infeasibility is bounded by

$$\mathbb{P}\left(g(x_N^*, \delta^{(N+1)}) \not\leq 0\right) \leq \epsilon, \tag{2-15}$$

with confidence $1 - \beta$ over the previously drawn scenarios. Thus, Equation 2-15 can be rewritten into a probably approximately correct (PAC) guarantee

$$\mathbb{P}\left(\mathbb{P}\left(g(x_N^*, \delta^{(N+1)}) \not\leq 0\right) \leq \epsilon\right) \geq 1 - \beta.$$

Finally, [21] established a relation between the risk ϵ, the confidence $1-\beta$, the number of decision variables d, and the number of sampled scenarios N, as

$$\sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta. \tag{2-16}$$

Moreover, this expression can be rewritten into a lower bound on N as a function of $\epsilon, \beta,$ and d, thus given the number of decision variables, the user can specify a desired risk and confidence, informing us how many scenarios we must draw.

This result will be used in subsection 3-3-3 to optimize a semialgebraic set description of the region of attraction of a feedback planner's underlying dynamical system.

Chapter 3

Methodology

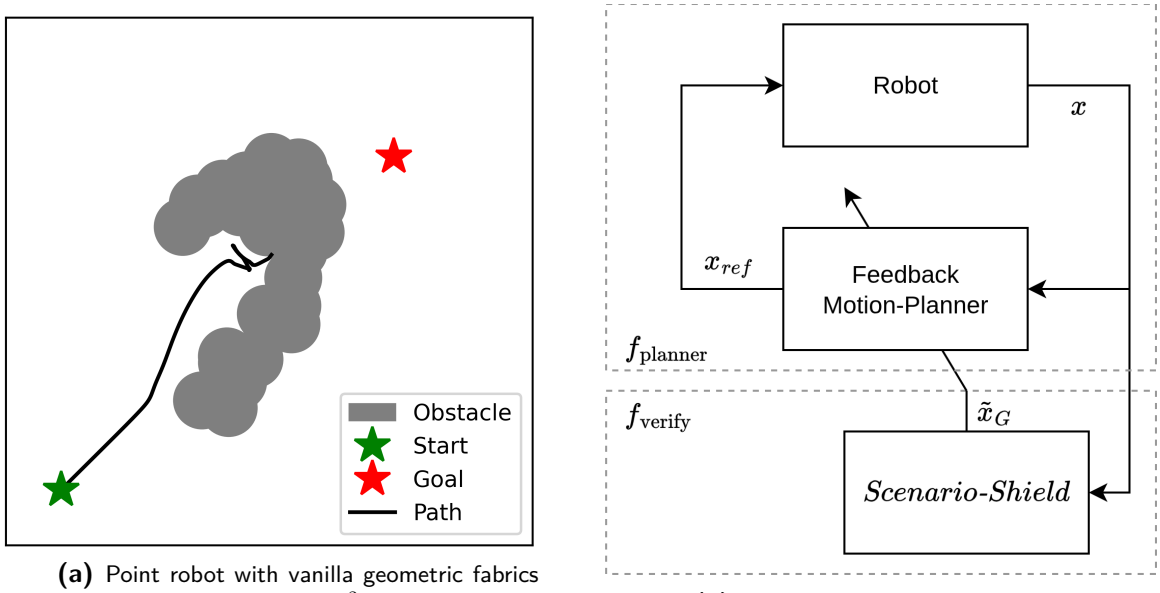
In the previous chapters, it was identified that geometric motion planners may not uphold their attractivity guarantees in environments encountered by robotic systems in practice. This chapter develops the main methodology to design a runtime verification algorithm that adapts the planner's underlying dynamical system to escape regions of no attraction.

First, Section 3-1 introduces the main idea behind the verification algorithm coined *Scenario-Shield*. Since the algorithm relies on numerous forward simulations of sampled robot configurations, the computational cost is a bottleneck. This is addressed in Section 3-2, where conformal prediction (CP) is used to calibrate a heuristic to render computationally cheaper rollouts. Moreover, the sample-based method only allows for constructing a finite set approximation of the region of attraction, while the underlying region may be continuous, i.e., an infinite set description may be necessary. How this can be obtained first geometrically and next via the scenario approach is the topic of Section 3-3. Finally, the two improvements are cast into an extension of the *Scenario-Shield* algorithm, which is presented in pseudo code in Section 3-4.

3-1 Verification guided geometric motion planning

This chapter briefly foreshadows the main components of the *Scenario-Shield* algorithm, which runs in parallel to the geometric fabrics motion planner. The main idea to address the local minimum problem of geometric motion planning stems from multi-run non-convex optimization. There, in order to improve a local optimum, other initial conditions are sampled, and the optimization is repeated. Geometric motion planning draws a similar picture, while the current robot configuration may not lie in the underlying region of attraction, configurations in close proximity may be, this is further illustrated in Section 3-3.

By locally sampling other configurations and forward simulating them, we can find a subset of the region of attraction. Then, the robot can attempt to reach this subset, which turns out to be easier than reaching the final goal directly. Effectively, this requires changing the goal of the geometric fabrics. Figure 3-1a depicts how a point robot employing geometric fabrics fails to reach the goal. Figure 3-1b suggests how to adapt the planning loop, i.e., only by changing the geometric fabrics' goal configuration.



(a) Point robot with vanilla geometric fabrics not reaching the goal in a $\mathbb{R}^2$ workspace.

(b) Proposed verification architecture.

Figure 3-1: Problem motivation and solution approach.

The regular feedback motion planning loop runs at a high frequency f_{planner}, to allow for safety-critical, reactive maneuvers. In parallel, but at a lower frequency f_{verify}, the block *Scenario-Shield* adapts the feedback planner's underlying artificial dynamical system. Specifically, it adapts the reference state, temporarily overwriting it with subgoals to steer the state into the planner's region of attraction. The subgoals are obtained by forward simulating samples obtained locally around the robot's current configurations, i.e., from the sampling space $\Omega(\boldsymbol{q_0})$.

This is illustrated in the motion lapse in Figure 3-2, where a holonomic ground robot navigates around the same obstacle, as in Figure 3-1a. However, with the *Scenario-Shield* algorithm, which adapts the goal during runtime, the robot is able to reach the goal. The sampled states are classified to lie in the region of attraction $\mathcal{A}$ or its complement $\mathcal{A}^c$, based on the terminal state of a rollout being close to the goal. An approximation of the underlying continuous regions $\hat{\mathcal{A}}$ and $\hat{\mathcal{A}}^c$ is furthermore plotted for orientation, but actually later statistically derived in Section 3-3.

In the first snapshot, configurations within the local sampling space Ω are sampled and rolled out. From the first iteration, it is clear that the trajectories contract into a few discrete modes. Moreover, the current configuration $\boldsymbol{q_0}$ (tip of the black path) will not get attracted. However, by picking a subgoal, the original region of attraction can eventually be reached. Note that in the second time frame, the region of attraction is

computed with respect to the subgoal and not the original goal. Once the subgoal is passed, the targeted goal $\tilde{\boldsymbol{x}}_G$ can switch back to the originally specified one. Finally, the robot reaches the goal configuration.

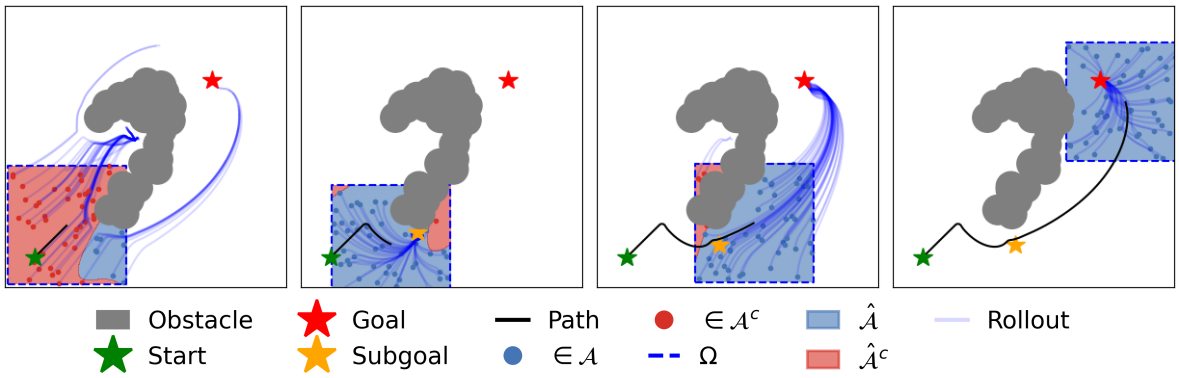


Figure 3-2: Paths of a point-robot in its $\mathbb{R}^2$ workspace.

This is furthermore depicted in the pseudo code algorithm 1. There, t is the incoming continuous time signal, $\boldsymbol{x} \in \mathcal{X}$ the current state, $\boldsymbol{x}_G \in \mathcal{X}$ denotes the final goal, $\tilde{\boldsymbol{x}}_G$ is the currently targeted goal for the feedback planner to track, f_{verify} is the verification frequency. $\mathbb{P}$ is the distribution from which samples are drawn with support, denoted as $\text{supp}(\mathbb{P})$, on the sampling space $\Omega(\boldsymbol{x_0})$, which moves with the robot. In the example above, Ω is an infinity norm ball around the current state $\boldsymbol{x}_0$, and $\mathbb{P}$ is the uniform distribution. Any state can be forward simulated over K timesteps by applying the discrete autonomous system dynamics $f : \mathcal{X} \to \mathcal{X}$ K-times, i.e. the final state $\boldsymbol{x}_K = f^{(K)}(\boldsymbol{x}_0)$. $A \bowtie B$ is the zipped set between sets of equivalent cardinality, i.e., $\{(a_i, b_i) \in A \times B \mid i = 1, \ldots, |A|\}$. Finally, the closed ε-ball is denoted as $B_\varepsilon[x] = \{x_i \in \mathcal{X} \mid d(\boldsymbol{x}, \boldsymbol{x}_i) \leq \varepsilon\}$ for some metric d. The queue of goal configurations returns or drops its last element by calling Q.last() or Q.pop(), respectively. This is needed to obtain the most recently selected subgoal and drop it upon reaching it.

Algorithm 1: *Scenario-Shield*

Input: $t, \boldsymbol{x}$
Output: $\tilde{\boldsymbol{x}}_G$

Initialization (once at startup):
  $Q \leftarrow [\boldsymbol{x}_G]$; `// queue of goal configurations`
do in parallel
  for $t = 0$ *until* T, *if* t *mod* $f_{verify}^{-1} = 0$ **do**
    $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
    $S \leftarrow \{\boldsymbol{x}_i\}_{i=1}^{N}$, $\boldsymbol{x}_i \sim \mathbb{P}$, with $\text{supp}(\mathbb{P}) \subseteq \Omega(\boldsymbol{x}_0)$ `// Locally sample states`
    if $f^{(K)}(\boldsymbol{x}_0) \notin B_\varepsilon\left[Q.last()\right]$ **then**
      $S_K \leftarrow \{f^{(K)}(\boldsymbol{x}_i) : \boldsymbol{x}_i \in S\}$ `// Compute rollouts`
      if $\exists(\boldsymbol{x}_{i,0}, \boldsymbol{x}_{i,K}) \in S \bowtie S_K$ *such that* $d(\boldsymbol{x}_{i,K}, Q.last()) \leq \varepsilon$ **then**
        Q.append$(\boldsymbol{x}_{i,0})$
  while *true* **do**
    `// Real-time execution loop`
    if $d(\boldsymbol{x}, Q.last()) \leq \varepsilon$ **then**
      Q.pop()
    $\tilde{\boldsymbol{x}}_G \leftarrow Q$.last()
    return $\tilde{\boldsymbol{x}}_G$

The basic *Scenario-Shield* algorithm has two disadvantages, which are addressed in the following methodology. Firstly, rollouts are computationally expensive. Thus, a verification cycle can only occur at a low frequency, which introduces a delay in how quickly *Scenario-Shield* can adapt to the subgoal. Thus, the faster the rollouts, the faster the adaptation, the lower the time to goal. This issue is addressed in Section 3-2.

Secondly, in contrast to Bellman's principle of optimality, since geometric fabrics are not an optimal control method, the trajectory segment from the initial state to a subgoal is not necessarily contained in the optimal trajectory from the initial state to the final goal. Thus, the resulting paths with intermediate goals may appear to make detours. To mitigate the burden of detours, intermediate goals can be skipped once the goal, or another queued subgoal, is viable again. Computing when a subgoal can be skipped directly translates into computing a continuous estimate of the underlying region of attraction, as opposed to the discrete underapproximation obtained from the finite number of rollouts. Section 3-3 addresses this first from a geometric perspective, before subsection 3-3-3 addresses it through scenario optimization, i.e., reasoning from the finite set of samples to an infinite set characterization in a probabilistically sound manner.

3-2 Conformal partial trajectory rollouts

Section 3-1 introduced the verification's reliance on trajectory rollouts. However, they are computationally expensive. This section proposes to employ a heuristic which is calibrated with conformal prediction to classify correctly with high probability whether a sample shall belong to the region of attraction $\mathcal{A}$, or its complement $\mathcal{A}^c$. The idea is that this heuristic only depends on k, with $k \ll K$, steps; thus, a classification can be done in a shorter time.

3-2-1 Classifying a sampled state

Since the feedback motion planner is represented through an autonomous dynamical system, a rollout of K steps can be obtained by applying the system dynamics $f : \mathcal{X} \to \mathcal{X}$ K times, i.e., the terminal state of the rollout $\boldsymbol{x}_K \in \mathcal{X}$ can be computed from the initial state $\boldsymbol{x}_0$ via $f^{(K)}(\boldsymbol{x}_0) = f \circ f \circ \cdots \circ f(\boldsymbol{x}_0)$.

The states that flow into the equilibrium goal state $\boldsymbol{x}_G$ belong to the attractive set $\mathcal{A}$. For each sampled state $\boldsymbol{x}_{i,0}$, the labeling function $y : \mathcal{X} \to \{-1, 1\}$ can be employed to classify whether the sample shall belong to the region of attraction $\mathcal{A}$ ($y(\boldsymbol{x}_{i,0}) = 1$) or its complement $\mathcal{A}^c$ ($y(\boldsymbol{x}_{i,0}) = -1$):

$$y(\boldsymbol{x}_0) = \begin{cases} 1, & f^{(K)}(\boldsymbol{x}_0) \in B_\varepsilon(\boldsymbol{x}_G) \\ -1, & \text{otherwise,} \end{cases} \tag{3-1}$$

where $B_\varepsilon(z)$ is the ε-ball around z.

Although this binary representation is only an intermediate step, it provides a useful foundation to construct a continuous prediction heuristic, i.e., with an output on the interval $[-1, 1]$, and use conformal prediction to introduce coverage regions around the nominal value 1 and -1 for $\mathcal{A}$ and $\mathcal{A}^c$, respectively.

Since these rollouts are performed during runtime, it is of interest to obtain the trajectory rollouts quickly. Sequentially computing the rollouts of N initial conditions over a horizon of K timesteps has a time complexity of $\mathcal{O}(N \cdot K)$. However, we can vectorize over the initial conditions, which, if resources permit, will allow us to reduce the complexity to $\mathcal{O}(K)$. Moreover, a trivial method to compute a rollout faster is to reduce the horizon K. However, when deciding whether an initial state $\boldsymbol{x}_0$ lies in the backward reachable set of the goal set $\mathcal{X}_g$ within K timesteps, it generally requires a rollout of K timesteps. Note that we do not aim to compute the entirety of the backward reachable set, but instead, need to classify each sample in order to pick a suitable intermediate goal.

In specific cases, reachability may be decided earlier. For example for an initial state $\boldsymbol{x}_0$, we want to check whether $\boldsymbol{x}_K = f^{(K)}(\boldsymbol{x}_0) \in \mathcal{X}_G$, but when a trajectory enters an invariant set $\mathcal{I}$ at time step $k < K$ and $\mathcal{I} \cap \mathcal{X}_G = \emptyset$, we can decide that $\boldsymbol{x}_0$ does not lie in the backward reachable set of $\mathcal{X}_\mathcal{G}$. However, this may lead to replacing a

computationally expensive rollout with a more expensive invariant set computation; even though there are specific, computationally cheap cases, such as where $\boldsymbol{x}_t$ is an equilibrium point.

Instead, the rollout length can be reduced by using a heuristic, which is introduced in Equation 3-2. This heuristic predicts on the continuous interval $[-1, 1]$. Then, the final classification can be done by calibrating thresholds with conformal prediction. These decisions will not be correct with certainty, but instead only with high probability. Hence, we trade-in certainty for computing resources, but importantly, retain a high probability of correctly classifying the samples.

While the prediction heuristic can be anything, e.g., a neural network, or, as in this case, a heuristic that rewards progress towards the goal, normalized by the distance of the sampled initial condition to the goal. The proposed heuristic score function $H : \mathcal{X} \times \mathcal{X} \times \mathcal{X} \to [-1, 1]$ is

$$H(\boldsymbol{x}_0, \boldsymbol{x}_k, \boldsymbol{x}_g) = -\min\left(\frac{d(\boldsymbol{x}_k, \boldsymbol{x}_g)}{d(\boldsymbol{x}_k, \boldsymbol{x}_0) + \epsilon} - 1, 1\right), \tag{3-2}$$

where d(.,.) is a metric, such as the Euclidean distance, and ϵ is a small number. Note that H is upper-bounded by 1 due to metrics being nonnegative. Hence, as k approaches K, H approaches y, the ground truth label, which can be obtained with certainty with a K-step rollout.

Next, a threshold C must be set to go from the continuous domain of the heuristic to the binary classification. This will lead to an estimated label $\hat{y}(\boldsymbol{x}_0)$, note that this is only informative if $C \in [0, 1)$.

$$\hat{y}(\boldsymbol{x}_0) = \begin{cases} 1, & \text{if } H(\boldsymbol{x}_0, f^{(k)}(\boldsymbol{x}_0), \boldsymbol{x}_g) \geq 1 - C \\ -1, & \text{if } H(\boldsymbol{x}_0, f^{(k)}(\boldsymbol{x}_0), \boldsymbol{x}_g) \leq C - 1 \\ \mathcal{U}(\{-1, 1\}) & \text{otherwise,} \end{cases} \tag{3-3}$$

where the random tie-breaker ensures that a decision can be made for all values of H.

3-2-2 Conformalizing the prediction

The aim is to set k and C, such that the probability of falsely classifying $\boldsymbol{x}_0$ based on the short rollout of k steps, with respect to the long horizon of K steps, is below a user specified threshold $\delta \in [0, 1]$, i.e., $\mathbb{P}(y = \hat{y}) \geq 1 - \delta$. This problem can be cast into a coverage problem, in which we estimate the coverage interval of H in the $1 - \delta$ quantile of predictions. This is exactly what we can obtain through conformal prediction. The first step is to choose a non-conformity score, e.g., the norm of the difference between the true label, obtained with the entire horizon, and the heuristic value:

$$R(y, H) = |y - H|. \tag{3-4}$$

By collecting $K_{cal} \in \mathbb{N}_+$ calibration data points, i.e., pairs of ground truth, corresponding heuristic value, and resulting non-conformity score, we can make the statistical

argument that a test point $s^{(0)}$ lies within the $1-\delta$ quantile spanned by the calibration data, i.e.,

$$\mathbb{P}\left(R^{(0)} \leq C(R^{(1)}, \ldots, R^{(K_{cal})}, \infty)\right) \geq 1-\delta, \tag{3-5}$$

under the assumption that the test and calibration data are exchangeable. Moreover, it should be noted that this will only render a useful result when using a sufficiently large calibration set, specifically, $K_{cal} > \lceil (K_{cal}+1)(1-\delta) \rceil$ [19], otherwise the $1-\delta$-quantile C will take on the correct, but trivial value of ∞. Furthermore, if the predictor is poor, again the resulting C will be uninformative; imagine the predictor classifies inversely to the ground truth, this would render $C = 2$ at best, hence, making the classification rule in Equation 3-3 ambiguous.

For the application of using partial rollouts to classify whether initial states should belong to the attractive set $\mathcal{A}$ or its complement $\mathcal{A}^c$, we may want to analyze the probabilistic coverage per class, in case there is a class imbalance. A class imbalance could arise when a small fraction of samples belongs to $\mathcal{A}^c$, then a primitive predictor that always predicts $\mathcal{A}$ may score a high accuracy over all classes, while performing poorly for the minority class.

This can be simply done by introducing two non-conformity scores corresponding:

$$R_{\mathcal{A}}(H) = |1-H| \text{ if } y = 1 \qquad R_{\mathcal{A}^c}(H) = |-1-H| \text{ if } y = -1 \tag{3-6}$$

Then, one obtains one coverage interval per label:

$$\mathbb{P}\left(R^{(0)} \leq C_{\mathcal{A}}(R_{\mathcal{A}}^{(1)}, \ldots, R_{\mathcal{A}}^{(K)}, \infty)\right) \geq 1-\delta_{\mathcal{A}}, \tag{3-7}$$

$$\mathbb{P}\left(R^{(0)} \leq C_{\mathcal{A}^c}(R_{\mathcal{A}^c}^{(1)}, \ldots, R_{\mathcal{A}^c}^{(K)}, \infty)\right) \geq 1-\delta_{\mathcal{A}^c}, \tag{3-8}$$

To conformalize the heuristic-based predictor presented in Equation 3-3, it can be rewritten as

$$\hat{y} = \begin{cases} 1, & \text{if } H \geq 1 - C_{\mathcal{A}} \\ -1, & \text{if } H \leq -1 + C_{\mathcal{A}^c} \quad \text{for } C_{\mathcal{A}} + C_{\mathcal{A}^c} < 2 \\ \mathcal{U}(\{-1, 1\}) & \text{otherwise.} \end{cases} \tag{3-9}$$

Depending on the application, it may be more sensible to replace the tiebreaker with a conservative classification. For example, in the case of *Scenario-Shield*, it is better to falsely classify a sample as belonging to the complement of the region of attraction. Upper-bounding the sum of the prediction intervals ensures that the decision is not ambiguous.

Then, to analyze the performance of the heuristic as a function of the partial rollout length k, we can plot $C_{\mathcal{A}}$ and $C_{\mathcal{A}^c}$ for different values of k, where the ground truth K remains fixed. Figure 3-3 depicts different coverage regions for different choices of δ (for simplicity, $\delta_{\mathcal{A}} = \delta_{\mathcal{A}^c}$).

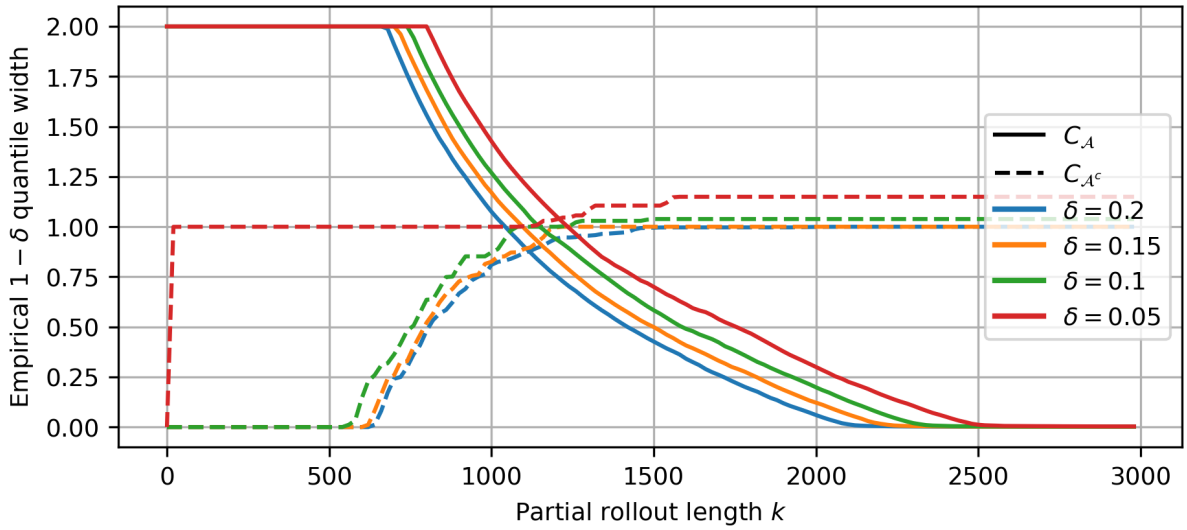


Figure 3-3: Empirical quantile width against the partial rollout length k for different failure probabilities δ. The full length rollout is $K = 3000$.

It becomes apparent that for very short partial rollouts, most initial states do not make any progress toward the goal set and hence get labels to belong to $\mathcal{A}^c$. Hence, the heuristic rarely misclassifies membership of $\mathcal{A}^c$ trivially.

Exemplary, for a short partial rollout length of $k = 800$, and $\delta = 0.1$, the data in Figure 3-3 suggests to use $C_{\mathcal{A}} \approx 1.8$ and $C_{\mathcal{A}^c} \approx 0.6$, which means their sum exceeds the allowable value and a conformal prediction cannot be made according to Equation 3-9. Therefore, the partial rollout length may be constrained. The allowable choice for k for $\delta = 0.1$ is hence $k \in [0, 540] \cup [1150, 3000]$; it makes sense to choose t such that the true positive rates of both classes are balanced, furthermore the lower k, the faster the rollout computation, hence, it would make sense to choose $k = 1150$. Then, $C_{\mathcal{A}}$ and $C_{\mathcal{A}^c}$ take values 0.98 and 1.01, respectively. Newly deciding attractivity of a test point is then probably correct with probability $1 - \delta$ under the assumption that the test point is exchangeable with the calibration data.

Finally, it is left to discuss how an appropriate calibration set is selected. In this thesis, it is assumed that sampled trajectories are exchangeable with those encountered during test time. Thus, the calibration set is appropriate by assumption. While not generally reasonable, in this case, the distribution of non-conformity scores is driven by changes in the environment, specifically the distribution of obstacles, which is controlled in the later experiment. This can be seen as a covariate shift, while the concept distribution, i.e., the underlying true classification given its inputs, remains unchanged. Later in the experiments, the obstacles across experiments are sampled independently and follow an identical distribution; independent and identically distributed (i.i.d.) implies exchangeability.

After making the rollouts faster, the next section deals with using them to approximate a continuous region of attraction, as opposed to the finite set one obtained through rollouts alone.

3-3 Computing a continuous approximation of the region of attraction

The previous rollouts allowed obtaining finite sets as subsets of the underlying region of attraction $\mathcal{A}$ and its complement $\mathcal{A}^c$. However, a subgoal really is only a proxy to steer the state into the underlying region of attraction. If the subgoal can be dropped once the region of attraction is entered, we could avoid detours. However, since the samples compose a finite set, it is unlikely that the robot's trajectory will intersect it before reaching a subgoal. This section addresses this problem by computing a continuous approximation of $\mathcal{A}$; hence, this approximation will be characterized as an uncountably infinite set.

Due to the complex dynamics of a geometric fabrics' expression, the previously collected samples will be paired with a scenario optimization program, as outlined in subsection 3-3-3. However, there are valuable insights for simple cases; hence, this section is built up in increasing complexity: subsection 3-3-1 computes $\mathcal{A}$ for a holonomic point robot in $\mathbb{R}^2$ with a single circular obstacle, subsection 3-3-2 extends this to $\mathbb{R}^2$ with multiple obstacles, and finally, the scenario program is employed to compute it for a general geometric planner in a general environment.

3-3-1 Geometrically in $\mathbb{R}^2$ with one obstacle

According to *Morse theory*, and specifically the *hairy ball theorem* [22], for a general environment with at least one obstacle, the topology prohibits the construction of a smooth, globally attractive vector field [2]. This is precisely because the obstacle introduces a saddle point at which the gradient vanishes. Then, using a gradient descent-based trajectory generation, any initial state flows into the set of zero gradient. However, due to the saddle point, this set is not merely the global minimum, but instead the disjoint union of the global minimum and the saddle point. Hence, at best, one can construct a vector field that is almost globally attractive, which means it is attractive for all initial conditions except for those lying in $\mathcal{A}^c$, which is the complement of the region of attraction to the goal, which in this case contains all points flowing into the saddle point.

A navigation function in a two-dimensional workspace is a potential composed of goal-attracting and obstacle-repelling sub-potentials; derivations can be found in [12]. As illustrated in Figure 3-4, the initial conditions that flow into the saddle point are those in the set $\mathcal{A}^c := \{\boldsymbol{x} \in \mathbb{R}^2 : \boldsymbol{x} = \boldsymbol{o} + \rho(\boldsymbol{o} - \boldsymbol{g}), \rho \in [o_r, \infty)\}$, where o is the obstacle center, o_r its radius, and $\boldsymbol{g}$ is the goal location. In geometry, this is the definition of a ray, and hence, $\mathcal{A}^c$ has a Lebesgue measure of zero in $\mathbb{R}^2$. Therefore, the probability of randomly choosing an initial condition that lies in $\mathcal{A}^c$ is zero. And thus, the probability of not flowing into the goal from a random initial condition is zero. Or complementarily stated, the vector field is almost globally attractive; where the word *almost* refers to the fact that this is true for all initial conditions except those within a set of measure zero, rendering the probability of convergence from a random initial condition equal to 1, hence, achieving convergence with almost certainty.

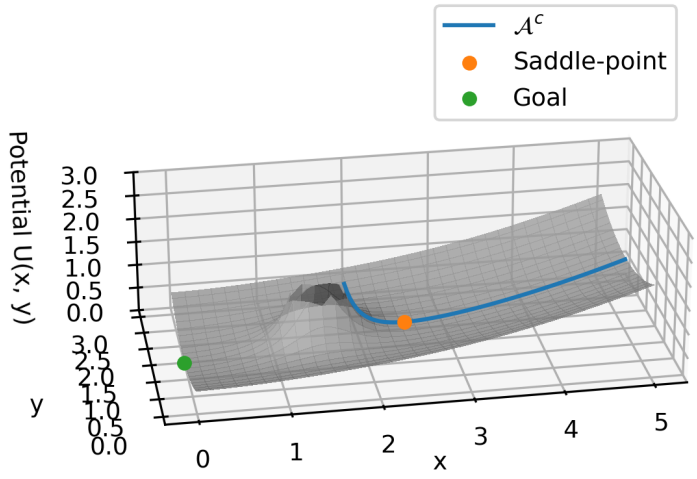


Figure 3-4: Topological implications of Morse theory in $\mathbb{R}^2$ with one obstacle

3-3-2 Geometrically in $\mathbb{R}^2$ with countably many obstacles

When multiple obstacles are present, the measure of $\mathcal{A}^c$ depends on the spacing between obstacles. To simplify this discussion in $\mathbb{R}^2$ it is assumed that the robot can be overapproximated with a circle, and that all circle obstacles are expanded by the radius of the robot's collision circle, such that collision checking can be done by checking whether the robot's center is within any of the expanded, circular obstacles.

Let the i^{th} obstacle be defined by the set $O_i(o_x, o_y, o_r) = \{(\boldsymbol{x} \in \mathbb{R}^2 : ||\boldsymbol{x} - (o_x, o_y)||_2^2 \leq o_r^2\}$, where (o_x, o_y) is the obstacle's center and o_r is its radius. Then, if no obstacles touch, that is for any distinct i^{th} and j^{th} obstacle $O_i \cap O_j = \{\emptyset\}$, $\mathcal{A}^c$ is the union of all the rays produced by each obstacle, as computed in the subsection 3-3-1. Thus, $\mathcal{A}^c$ remains of measure zero, as the union of countably many sets of measure zero is a set of measure zero. This assumption is commonly found in navigation function planning, as seen in [15].

This assumption was relaxed in [6] by allowing disjoint unions of star-shaped obstacles. In practice, however, this is still restrictive: after overapproximating obstacles by the robot's radius, physically non-colliding obstacles may appear to intersect, and complex obstacles represented as unions of spheres cannot intersect densely enough to capture their shape without violating the star-shaped assumption.

Therefore, we are interested in computing $\mathcal{A}^c$ for the setting of closely positioned obstacles. In this case, the set will no longer be of measure zero, but instead, will result in an unbounded polygonal domain with two finite vertices and two more infinitely far for any pair of intersecting obstacles. This can be represented through an intersection of three halfplanes.

The entirety of $\mathcal{A}^c$ will be composed of the union of local ones, one of measure zero per obstacle, and additionally, those of positive measure spanned by any two obstacles that

intersect. The latter case can be computed by first detecting the intersections, for that one can utilize the pairwise distance matrix whose entry (i, j) is the minimum distance, i.e., between the closest points on the contour, of the i^{th} and j^{th} obstacle unless $i = j$:

$$D = (D_{ij})_{i,j=1}^{o} \quad , \text{ where } D_{ij} = \begin{cases} ||(o_x^i, o_y^i) - (o_x^j, o_y^j)||_2^2 - o_r^i - o_r^j, & i \neq j, \\ 0, & i = j. \end{cases} \tag{3-10}$$

Negative entries of D indicate an intersection of the corresponding obstacles. For those entries, the region that does not attract to the goal is geometrically defined as an unbounded polygonal domain, with two vertices at the obstacle centers and two infinitely far away in the direction from the goal to the obstacle. Mathematically, this can be expressed as the intersection of three halfplanes. Specifically, let's construct this for the obstacles with indices i and j, noting that $D_{i,j} < 0$. Then, for the obstacle locations o^i, o^j and the goal location $\boldsymbol{x}_G$, we have:

$$a^c = \{\boldsymbol{x} \in \mathbb{R}^2 \mid A\boldsymbol{x} \leq b\}, A = \begin{bmatrix} -(\boldsymbol{o}^i - \boldsymbol{x}_G) \\ \boldsymbol{o}^j - \boldsymbol{o}^i \\ \boldsymbol{o}^j - \boldsymbol{x}_G \end{bmatrix}, \qquad b = \begin{bmatrix} -(\boldsymbol{o}^i - \boldsymbol{x}_G)^\top \boldsymbol{o}^i \\ -(\boldsymbol{o}^j - \boldsymbol{o}^i)^\top \boldsymbol{o}^i \\ -(\boldsymbol{o}^j - \boldsymbol{x}_G)^\top \boldsymbol{o}^i \end{bmatrix} \tag{3-11}$$

Finally, the entire $\mathcal{A}^c$ is the union of the local regions which may be a disjoint set:

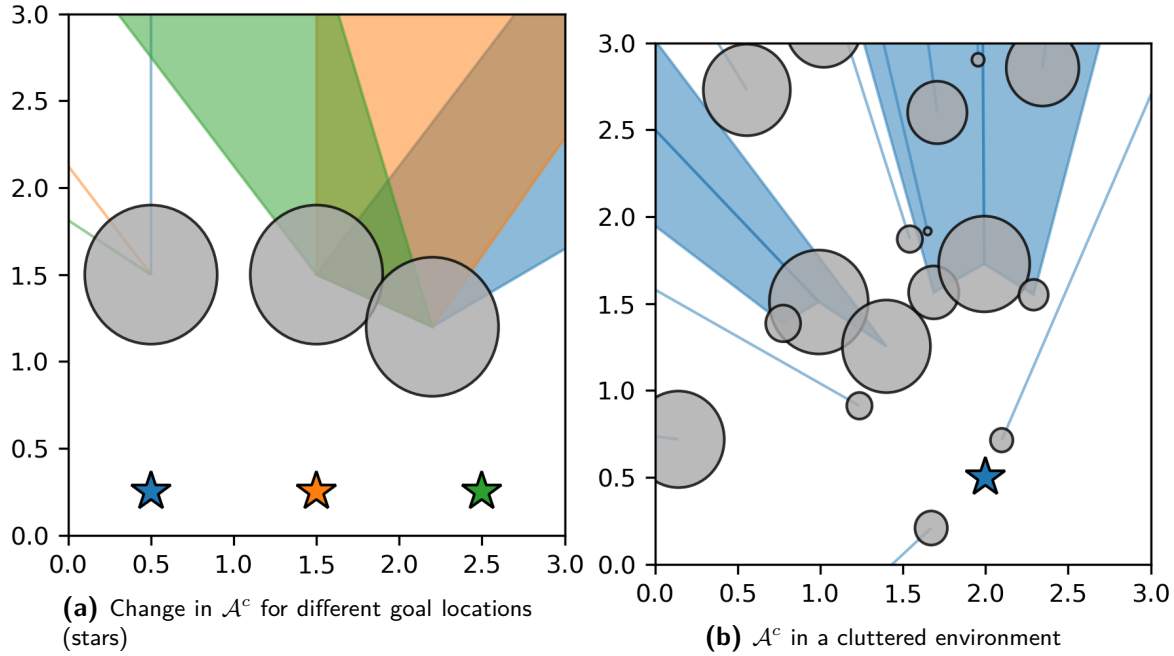
$$\mathcal{A}^c = \bigcup a^c. \tag{3-12}$$

Now, if the Lebesgue measure $\lambda(.)$ of any of the local a^c sets is positive, this propagates into the union, i.e. $\exists a^c : \lambda(a^c) > 0 \Rightarrow \lambda(\mathcal{A}^c) > 0$. This has an important implication: while previously the feedback planner could be designed to be almost globally attractive, this is no longer the case.

Furthermore, there is one insight which further motivates the previous approach outlined in Section 3-1: when multiple obstacles intersect, the region of attraction depends on the goal location. As shown in Figure 3-5a, when initially attempting to reach the blue goal but starting from the top right corner, $(x, y) = (3, 3)$, the underlying navigation function is not attractive. However, by first going to the green goal, the region of attraction to the blue goal can be entered. Furthermore, Figure 3-5b illustrated that for cluttered environments, the complement of the region of attraction can span a significant fraction of the free space.

3-3-3 From finite samples in $\mathbb{R}^n$ with countably many obstacles

In some applications, such as mobile manipulator control, practically useful artificial dynamical systems are highly nonlinear and have a large number of dimensions. In such cases, the attracting potential may not be spherically symmetric, hence, not allowing for the above geometric argument. Thus, finding $\mathcal{A}^c$ explicitly becomes infeasible.



(a) Change in $\mathcal{A}^c$ for different goal locations (stars)

(b) $\mathcal{A}^c$ in a cluttered environment

Figure 3-5: $\mathcal{A}^c$ in $\mathbb{R}^2$ environments with multiple obstacles

However, through the sampled initial states and their classification through rollouts, as outlined in Section 3-1, we have already obtained a finite subsets of the underlying $\mathcal{A}$ and $\mathcal{A}^c$. Now, can we extend this to a possibly infinite set by reasoning from the finite subset and the sampling strategy? The answer is yes; [23] demonstrated a similar approach to employing a stochastically estimated reachable set to verify the feasibility of a model predictive control feedback law. Hence, similarly here, a parameterized infinite set can be constrained according to the sampled states and optimized to render an estimated $\mathcal{A}_\theta$, where θ is the parameterization vector. This ties in well with the previous *Scenario-Shield* algorithm, as the necessary data for the optimization problem is already collected. These samples and their classification according to the full rollouts or partial rollouts, as introduced in Section 3-2.

In general, the region of attraction considers all states that flow into a specific equilibrium. However, in the case of reasoning from K-step finite-time rollouts, the approximated region will be a subset, namely, the finite-time region of attraction in K steps. Yet, there are still uncountably many initial conditions, and hence uncountably many trajectories required to properly constrain the optimization problem. When determining $\hat{\mathcal{A}}$ through an optimization problem, even when choosing a finite parameter vector, the problem will be semi-infinite due to an infinite number of constraints, and hence, computationally not tractable in its original form.

The scenario approach [24] recasts a semi-infinite problem into a chance-constrained one with a finite number of sampled constraints. Under convexity, the obtained solution is probably optimal even for newly i.i.d. sampled constraints. However, the details for this will be introduced later, let's start by expressing the complement of the region of attraction $\mathcal{A}^c$ as an optimization problem. This will be done in two steps: first, a

parameterized set description $\mathcal{A}_\theta^c$ must be chosen, then, the constraints are constructed.

Figure 3-5b illustrates that $\mathcal{A}^c$ may be a disjoint set, which can be modeled by a parameterized set-estimator $\hat{\mathcal{A}}_\theta$, such as a basic semialgebraic set composed of a linear combination of monomials up to degree p:

$$\hat{\mathcal{A}}_\theta^c := \{\boldsymbol{x} \in \mathcal{X} : \theta^\top \phi_p(\boldsymbol{x}) \leq 0\}, \tag{3-13}$$

where $\theta \in \mathbb{R}^d$ and $\phi_p(x)$ is the column vector of monomials up to degree p evaluated at $\boldsymbol{x}$. For the constraints, can reuse the same rollouts as before in the regular *Scenario-Shield* algorithm, then assuming we sampled N initial states $\boldsymbol{x}_{0,1:N}$, we obtain the labels according to Equation 3-1, or in the case of conformal partial rollouts Equation 3-3; where $y = 1$ and $y = -1$ correspond to membership to $\mathcal{A}$ and $\mathcal{A}^c$ respectively.

Then, if we sample all the states in the domain $\mathcal{X}$, the complement of the region of attraction $\hat{\mathcal{A}}_\theta^c$, correct up to the accuracy of the chosen parameterization, optimization problem

$$\min_{\theta,\xi} \quad \xi \tag{3-14a}$$

$$\text{s.t.} \quad y(\boldsymbol{x}_0)\theta^\top \phi_p(\boldsymbol{x}_0) \leq \xi \qquad \forall \boldsymbol{x}_0 \in \mathcal{X} \tag{3-14b}$$

$$-\xi \leq 0, \tag{3-14c}$$

where ξ is a slack variable. Hence, both the problem's objective and constraints are linear in the decision variables θ, ξ. Allowing the use of linear programming, if it were not for the infinite number of constraints. Thus, the scenario approach is invoked to replace the infinite number of constraints 3-14b with a finite number of constraints, following from our previously obtained finite number of sampled states and labels, assuming we have N such samples, subsequently referred to as scenarios, that constraint can be written into

$$y(\boldsymbol{x}_{0,i})\theta^\top \phi_p(\boldsymbol{x}_{0,i}) \leq \xi \qquad \forall i = 1, \ldots, N. \tag{3-15a}$$

However, the scenario approach in its vanilla form assumes that the solution is unique [24], also see the preliminary introduction in subsection 2-2-2. In general, a linear program may admit multiple optimal solutions; thus, a solution is not necessarily unique. Fortunately, there exist two solutions: a strongly convex cost function can be chosen, which makes the problem, if feasible, have a unique solution. However, this will result in solving a possibly more expensive strictly convex optimization problem, rather than a linear programming one. Alternatively, a deterministic tie-breaker can be chosen [24, appendix C]; which boils down to solving multiple linear programs sequentially. Let's denote the set of optimal solutions of Equation 3-14a with Θ^*, then we can find the element(s) that optimize the tie-breaker functions $t_1(\theta), \ldots, t_d(\theta)$ until a single solution remains. [24] suggests using $t_1(\theta) = \theta_1, \ldots$, which means this can be solved by at most d linear programs, and is efficiently implemented in standard solvers such as Gurobi through lexicographic multiple objective optimization.

Hence, the scenario program is formulated as a feasibility problem that is solved using linear programming. In this context, this means that the found set $\hat{\mathcal{A}}_{\theta^*}^c$ is probabilistically valid for a new sampled initial state $\boldsymbol{x}_{0,N+1}$, provided the samples are drawn i.i.d.

and the number of scenarios N satisfies Equation 3-16. Then, $\hat{\mathcal{A}}^c_{\theta^*}$ is correct for a i.i.d. $\boldsymbol{x}_{0,N+1}$ with at least a probability of $1-\epsilon$, in which we can have a confidence of $1-\beta$, provided that the number of samples N satisfies:

$$N \geq \frac{2}{\epsilon}\left(\ln\frac{1}{\beta} + d\right), \tag{3-16}$$

where $\epsilon \in (0,1)$ is the risk, $1-\beta \in (0,1)$ is the confidence, and d is the finite number of decision variables of the optimization problem, i.e., $|\theta|$.

3-3-4 Implications for the Scenario-Shield algorithm

What does this mean for the application to the *Scenario-Shield* algorithm? Under the assumption that the number of samples satisfies Equation 3-16 and that a new sample is i.i.d., it allows us to describe the region of attraction as a semialgebraic set. Thus, we obtain a switching surface to decide whether to continue going towards a subgoal, or drop it and switch back to the original goal or another queued subgoal.

Unfortunately, however, using the set description as a switching surface directly would violate the assumption of independence, as a point defined by the intersection of a trajectory and a switching surface depends on the previously sampled data through the scenario program's solution. Instead, the independence assumption is accommodated by evaluating a previously computed solution only once, thus also avoiding the introduction of dependence along the trajectory. The assumption that a test point, i.e., the $(N+1)^{\text{th}}$ scenario, is identically distributed is more subtle. Since the *Scenario-Shield* algorithm samples uniformly within a l_∞-ball around its current configuration, we can make use of the fact that the uniform distribution is equivalent to the normalized Lebesgue measure on a bounded region [25]. Thus, assuming that the semialgebraic set description is of sufficiently high order, we can still obtain the guarantee that:

$$\mathbb{P}\left(\frac{\lambda((\hat{\mathcal{A}}^c \setminus \mathcal{A}^c) \cup (\mathcal{A}^c \setminus \hat{\mathcal{A}}^c))}{\lambda(\mathcal{A}^c \cup \hat{\mathcal{A}}^c)} \leq \epsilon \;\middle|\; \mathbb{P}(y = \hat{y}) = 1\right) \geq 1-\beta, \tag{3-17}$$

where $\lambda(.)$ is the Lebesgue measure. Note that this hinges on the correctness of the labels. As described in Section 3-2, computing the labels with certainty is computationally expensive, and its computationally cheaper variant through probabilistic relaxation can be integrated into the scenario program's guarantee as well, by adjusting the conditional probability, which is independent.

Two more aspects will be addressed. Firstly, how does the chosen semialgebraic set description scale with the number of dimensions, and secondly, how to interpret the solution of $\hat{\mathcal{A}}^c \in \mathbb{R}^n$, while the true system may evolve on a manifold $\mathcal{M}$.

Alternative set description

The advantage of expressing $\mathcal{A}^c$ as a basic semialgebraic set, specifically as the sublevel set of a sum of monomials, is that it can model disjoint sets. However, the disadvantage

is the required number of coefficients, and hence decision variables of the optimization problem, depends on the number of dimensions n and degree of monomials p as $\binom{p+n-1}{n-1}$. Thus, it grows exponentially in the number of dimensions. This demands many more samples according to Equation 3-16, because it is affine in d.

Moreover, locally, $\mathcal{A}^c$ may appear joint. Since this set approximation is done during runtime, it is anyway only possible to do this locally, and hence the scenario program can solve for a lighter set description, i.e., that may only be joint, at the benefit of requiring fewer scenario samples. Furthermore, the scenario program in Equation 3-14a is merely a feasibility program and hence does not control the size of the estimated set. Therefore, it may be advantageous to use a different parameterization that requires fewer decision variables per dimension, such as an ellipsoid. A minimum volume ellipsoid $\mathcal{E}$ covering the finite set of scenarios which are heuristically classified to belong to $\mathcal{A}^c$.

Let $\mathcal{E} := \{\boldsymbol{x} \in \mathbb{R}^n : (\boldsymbol{x} - \boldsymbol{c})^\top \Theta (\boldsymbol{x} - \boldsymbol{c}) \leq 1\}$, where $\boldsymbol{c}$ is the center and Θ is the shape matrix. Because the ellipsoid's volume is proportional to its inverse's determinant, the minimum volume enclosing ellipsoid is the solution of:

$$\min_{\Theta, \boldsymbol{c}} \quad -\log(\det \Theta) \tag{3-18a}$$

$$\text{s.t.} \quad (\boldsymbol{x}_{j,0} - \boldsymbol{c})^\top \Theta (\boldsymbol{x}_{j,0} - \boldsymbol{c}) \leq 1 \qquad \forall j \in \{i \in 1, \dots N \mid y(\boldsymbol{x}_{i,0}) \leq 0\} \tag{3-18b}$$

$$\Theta \succeq 0. \tag{3-18c}$$

However, the constraint 3-18b is not convex. Fortunately, [26, sec. 8.4] shows how to rewrite it into the following convex constraint:

$$||\Theta \boldsymbol{x}_{j,0} + \boldsymbol{c}||_2 \leq 1 \qquad \forall j \in \{i \in 1, \dots N \mid y(\boldsymbol{x}_{i,0}) \leq 0\}. \tag{3-19}$$

Then, the objective is convex, and all constraints are convex in the decision variables, and hence, the problem can be solved efficiently. Note, however, to retain convexity, we can only consider the samples classified to belong to $\mathcal{A}^c$, as imposing both enclosing and inscribing constraints on the ellipsoid may render the problem non-convex, or even infeasible.

Mapping $\hat{\mathcal{A}}^c$ from $\mathbb{R}^n$ to $\mathcal{M}$

Both provided set descriptions, the semialgebraic set, and the enclosing ellipsoid, are defined in the Euclidean space. However, this representation is not aligned with many robotic applications, e.g. satellites, quadrotors, or mobile manipulators require the configuration space to include rotations. Such configurations live on a Riemannian Manifold $\mathcal{M}$.

Hence, one must either find a new set description or project the previously introduced set descriptions onto the manifold. [27] describes sets on $\mathcal{M}$ through the interior of a Riemannian ball, however, that can easily lead to overapproximation. Instead, we will make use of the fact that any point on $\mathcal{M}$ is locally homeomorphic to the Euclidean space. Hence, by working in the tangent space of $\mathcal{M}$, the above methods apply to

obtain $\hat{\mathcal{A}}^c \subset \mathbb{R}^n$. Then, we can use the exponential map to map the estimated region of attraction $\hat{\mathcal{A}}$ back onto $\mathcal{M}$.

As an example, we may work with a manipulator that has two rotational degrees of freedom. Thus, its configuration space is defined as the torus $S^1 \times S^1$. Conveniently, this is a matrix Lie group, hence one can enter the tangent space around the configuration x by premultiplying with the matrix T_x, the left-trivialization of the tangent space.

To conclude this section, we can state that the scenario approach enables us to obtain a continuous estimate of the underlying region of attraction, with a probably approximately correct (PAC) guarantee. Therefore, it may be used in the *Scenario-Shield* extension as a scenario program, which allows to drop a subgoal once the region of attraction is entered, as outlined in Section 3-4.

3-4 Extensions of the Scenario-Shield algorithm

The *Scenario-Shield* algorithm from algorithm 1 can be extended in isolation with the partial rollouts (SS-PR), developed in Section 3-2, the scenario program (SS-SP), developed in subsection 3-3-3, or with both extensions simultaneously (SS-PR-SP). The latter is again shown in pseudo code in algorithm 2.

Specifically, the partial rollouts introduce a change in the rollout function and employ the intermediate representation of a heuristic h_i, which is a function of the corresponding sample $\boldsymbol{x}_{i,0}$, its terminal partial rollout state $\boldsymbol{x}_{i,k}$, and the current goal. Moreover, the scenario program introduced the estimation of the complement of the region of attraction at the end of each verification cycle, as well as another parallel loop which allows to drop the latest subgoal upon entering the estimated region of attraction.

Algorithm 2: *Scenario-Shield* with partial rollouts and a scenario program (SS-PR-SP)

Input: $t, \boldsymbol{x}$
Output: $\tilde{\boldsymbol{x}}_G$

Initialization (once at startup):
  $Q \leftarrow [\boldsymbol{x}_G]$; `// queue of goal configurations`
  $\hat{\mathcal{A}}^c \leftarrow \mathcal{X}$; `// Estimate of the complement of the region of attraction`
do in parallel
  for $t = 0$ *until* T, *if* t *mod* $f_{verify}^{-1} = 0$ **do**
    $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
    $S \leftarrow \{\boldsymbol{x}_i\}_{i=1}^{N}$, $\boldsymbol{x}_i \sim \mathbb{P}$, with $\text{supp}(\mathbb{P}) \subseteq \Omega(\boldsymbol{x}_0)$; `// Locally sample states`
    if $f^{(k)}(\boldsymbol{x}_0) \notin B_\varepsilon\left[Q.last()\right]$ **then**
      $S_k \leftarrow \{f^{(k)}(\boldsymbol{x}_i) : \boldsymbol{x}_i \in S\}$; `// Compute rollouts`
      $H \leftarrow \{h(\boldsymbol{x}_{i,0}, \boldsymbol{x}_{i,k}, Q.\text{last}()) \mid \boldsymbol{x}_{i,0} \in S, \boldsymbol{x}_{i,k} \in S_k\}$
      if $\exists(\boldsymbol{x}_{i,0}, h_i) \in S \bowtie H$ *such that* $h_i \geq C_{\mathcal{A}}$ **then**
        $Q.\text{append}(\boldsymbol{x}_{i,0})$
    $\hat{\mathcal{A}}^c \leftarrow \text{scenario-program}(S, S_T)$
  for $t = 0$ *until* T, *if* t *mod* $(2f_{verify})^{-1} = 0$ **do**
    $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$
    if $\boldsymbol{x}_0 \notin \hat{\mathcal{A}}^c$ **then**
      $Q.\text{pop}()$
  while *true* **do**
    `// Real-time execution loop`
    if $d(\boldsymbol{x}, Q.last()) \leq \varepsilon$ **then**
      $Q.\text{pop}()$
    $\tilde{\boldsymbol{x}}_G \leftarrow Q.\text{last}()$
    return $\tilde{\boldsymbol{x}}_G$

Chapter 4

Implementation

In this chapter, the details about implementing *Scenario-Shield* are outlined for two robotic systems, namely, a holonomic ground robot and a mobile manipulator. In both cases, the underlying feedback motion planner is a geometric fabric.

On both robotic platforms, parallel computing is utilized for fast batch rollouts, as shown in Section 4-1. The platform-specific details for the ground robot and the mobile manipulator are presented in Section 4-2 and Section 4-3, respectively.

4-1 Batch trajectory rollouts

The proposed algorithm requires forward simulation of the autonomous system dynamics from many initial conditions. This task is well-suited to the parallel computing paradigm of single instruction, multiple data (SIMD), which allows multiple rollouts to be computed simultaneously. As a result, the computational cost per rollout scales sublinearly with the batch size.

The Python library `JAX` [28] facilitates writing SIMD-style programs that execute efficiently on parallel compute resources such as graphics processing units (GPUs). In practice, this is leveraged by compiling the mathematical expression that defines a fabric rollout prior to runtime. This compilation step requires fixing the rollout horizon and the number of inputs. The inputs to the compiled function include a batch of initial states, circular obstacle locations and radii, and the goal configuration; the outputs are the corresponding terminal states.

`JAX` employs just-in-time (JIT) compilation, meaning that the function is only compiled into platform-specific machine code upon its first evaluation. This compilation is performed through accelerated linear algebra (XLA). To avoid a runtime delay due to this initial compilation, it is beneficial to perform a warm-start by evaluating the function once during system initialization. While the compiled function can be cached for reuse,

it is important to note that evaluating it in a new process will trigger retracing before a cache hit can be recognized.

The underlying geometric fabrics are implemented using the symbolic differentiation capabilities of `CasADi` [29]. These expressions are converted to `JAX`-compatible code using the `Jaxadi` library [30].

4-2 Holonomic ground robot

The system dynamics of a holonomic ground robot on a plane with acceleration-based control are simply integrating the velocities and acceleration inputs:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}_k, \tag{4-1}$$

where Δt is the discrete time step, $\boldsymbol{x}_k$ is the stack of the configuration vector $\boldsymbol{q} \in \mathbb{R}^2$ and its velocity $\Delta \boldsymbol{q}$, i.e., $\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{q}_k & \Delta \boldsymbol{q}_k \end{bmatrix}^\top$, at time k. The control input integrates the geometric fabrics' continuous-time acceleration signal, i.e., $\boldsymbol{u}_k = \int_{k \cdot \Delta t}^{(k+1) \cdot \Delta t} \ddot{\boldsymbol{q}}(\tau) d\tau$. However, since an analytic solution of the integral is not available, the forward Euler numerical integration method is used. Note that while some geometric fabrics applications use fourth-order Runge-Kutta integration schemes [31], forward Euler was successfully applied to geometric fabrics in [32], see [33] for a detailed discussion, since the rollouts' computational complexity depends linearly on the number of function evaluations necessary, forward Euler is chosen. As introduced in subsection 2-1-2, the acceleration policy is obtained through:

$$\ddot{\boldsymbol{q}} = -(\Big(M(\boldsymbol{q}, \dot{\boldsymbol{q}})\Big)^{-1} \xi(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \Big(M(\boldsymbol{q}, \dot{\boldsymbol{q}})\Big)^{-1} (\partial_{\boldsymbol{q}} \psi(\boldsymbol{q}) + B(\boldsymbol{q}, \dot{\boldsymbol{q}}) \dot{\boldsymbol{q}})), \tag{4-2}$$

which is constructed according to the recipe in subsection 2-1-2, but requires the following choices of geometries and differential maps for baseline, goal, and obstacle.

For each obstacle, first, the differential map $\phi_{\text{obst},i} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ introduces a measure of the distance between the obstacle's contour and the robot's center:

$$\phi_{\text{obst},i}(\boldsymbol{q}) = \frac{||\boldsymbol{q} - \boldsymbol{o}_i||_2}{r_i} - 1, \tag{4-3}$$

where $\boldsymbol{o}_i$ and r_i are the i^{th} obstacle's center and radius, respectively. Then, the obstacle geometry on the latent variable $z = \phi_{\text{obst},i}(\boldsymbol{q})$ is chosen to be:

$$h_{\text{obst}} = \frac{-\dot{z}^2 \cdot (\text{sign}(\dot{z}) - 1)}{z^{10} + \varepsilon}, \tag{4-4}$$

where ε is a small positive number to avoid division by zero.

The goal reaching potential, priority matrix, and damping matrix are :

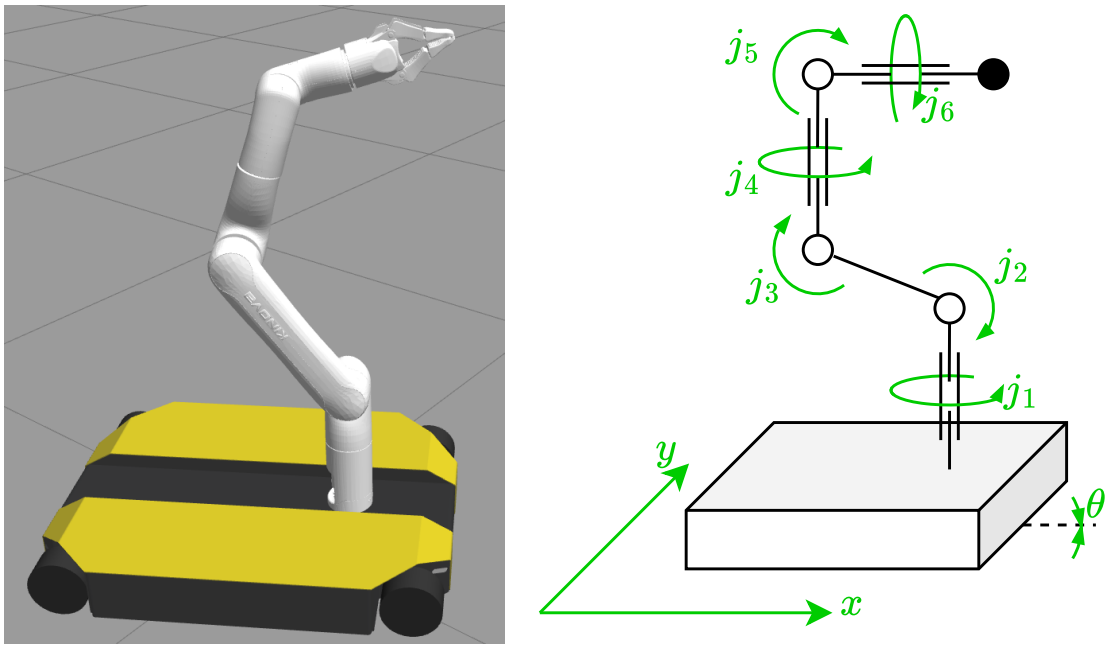
$$\psi(\boldsymbol{q}) = ||\boldsymbol{q} - \boldsymbol{q}_G||_2, \quad M(\boldsymbol{q}, \dot{\boldsymbol{q}}) = w_M \cdot \boldsymbol{I}_2, \text{ and } \quad B(\boldsymbol{q}, \dot{\boldsymbol{q}}) = w_B \cdot \boldsymbol{I}_2, \tag{4-5}$$

respectively, where $w_M \in \mathbb{R}_{>0}$ and $w_B \in \mathbb{R}_{>0}$ are weighting factors, and $\boldsymbol{I}_2$ is the identity matrix of size 2.

Then, the feedback motion planner is queried at frequency f_{planner}, while the *Scenario-Shield* algorithm runs in parallel at f_{verify}, adapting the geometric fabrics' goal $\boldsymbol{q}_G$. The verification algorithm's scenario program is solved using the optimization toolbox CVXPY [34].

4-3 Mobile manipulator

The considered mobile manipulator is the Clearpath mobile manipulator platform. This system comprises a holonomic mobile base, the Clearpath Dingo, integrated with a 6 degrees of freedom (DoF) Kinova robotic arm, as shown in Figure 4-1.



(a) in Gazebo simulation **(b)** configuration description

Figure 4-1: Dingo ground robot with a Kinova arm attached.

The robot's configuration vector $\boldsymbol{q}$ is hence defined as

$$\boldsymbol{q} = \begin{bmatrix} x & y & \theta & j_1 & j_2 & j_3 & j_4 & j_5 & j_6 \end{bmatrix}^\top, \tag{4-6}$$

where $(x, y, \theta) \in \mathbb{R}^2 \times S^1$ are the base's cartesian position and orientation, and $(j_1, j_2, j_3, j_4, j_5, j_6) \in T^6$ are the revolute joints of the manipulator. Hence, the configuration vector lives in the 9-dimensional configuration space $\mathcal{C} = \mathbb{R}^2 \times T^7$. The robot is holonomic and thus, the vanilla theory of geometric fabrics can be applied.

Due to the omnidirectional drive of the base, the robot is a holonomic system. Hence, similar to the previous ground robot, the system dynamics follow those of a double integrator, with the first integration to map the geometric fabrics' acceleration policy to a velocity command, and once in the simple dynamics

$$\boldsymbol{x}_{k+1} = \left[\begin{array}{c|c} \boldsymbol{I}_9 & \Delta t \cdot \boldsymbol{I}_9 \\ \hline \boldsymbol{0} & \boldsymbol{I}_9 \end{array}\right] \boldsymbol{x}_k + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_9 \end{bmatrix} \boldsymbol{u}_k, \tag{4-7}$$

where again $\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{q}_k & \Delta \boldsymbol{q}_k \end{bmatrix}^\top$, $\boldsymbol{u}_k = \mathit{clipped}(\tilde{\boldsymbol{u}}_k)$, and $\tilde{\boldsymbol{u}}_k = \int_{k\cdot\Delta t}^{(k+1)\cdot\Delta t} \ddot{\boldsymbol{q}}(\tau) d\tau$. The nominal control input $\tilde{\boldsymbol{u}}$ is clipped to avoid velocity commands that exceed the robot's limits, with maximum velocities of $v_{\text{max, base}}$ and $v_{\text{max, arm}}$ for the base and arm, respectively, thus,

$$\mathit{clipped}(\tilde{\boldsymbol{u}}_k) = \left[\min\left(1, \frac{v_{\text{max, base}}}{||\tilde{\boldsymbol{u}}_k^{(1:3)}||_2}\right) (\tilde{\boldsymbol{u}}_k^{(1:3)})^\top \quad \min\left(1, \frac{v_{\text{max, arm}}}{||\tilde{\boldsymbol{u}}_k^{(4:9)}||_2}\right) (\tilde{\boldsymbol{u}}_k^{(4:9)})^\top\right]^\top, \tag{4-8}$$

where $\tilde{\boldsymbol{u}}^{(i:j)}$ denotes subvector $\tilde{\boldsymbol{u}}$ containing components i through j.

Next, the goal potential and geometries for obstacle and joint limit avoidance are specified, which follow the vanilla choices of the toolbox's implementation.

The goal reaching follows from the potential [9]

$$\psi(\boldsymbol{x}) = \alpha_0 \left(\|\boldsymbol{x}\| + \frac{1}{\alpha_1} \ln\left(1 + e^{-2\alpha_1 \|\boldsymbol{x}\|}\right)\right), \tag{4-9}$$

the joint limit avoidance geometry, and associated energy Lagrangian

$$h_{\text{limit}}(x, \dot{x}) = -\frac{\dot{x}^2}{x}, \qquad \mathcal{L}_e = \frac{-\dot{x}^2(\text{sign}(\dot{x}) - 1)}{x^2}, \tag{4-10}$$

and obstacle avoidance geometry and corresponding energy

$$h_{\text{collision}}(x, \dot{x}) = -\frac{\dot{x}^2}{x}, \mathit{and} \qquad \mathcal{L}_e = \frac{0.1\dot{x}^2}{x^2}, \tag{4-11}$$

all other terms are as introduced in subsection 2-1-2.

Unfortunately, the toolbox's expression contains many elementary operations, such that even though the feedback law is closed-form, its evaluation is slow. This effect is amplified over rollouts; therefore, the next subsection demonstrates how this problem can be mitigated.

4-3-1 Simplifying the system dynamics

Control engineers are commonly concerned with order reduction and linearly approximating dynamical systems. In this case, the high order of the dynamical system is not directly a problem; instead, the number of primitive operations is a bottleneck for the forward simulation. Specifically, the geometric fabric terms related to obstacle

avoidance involve numerous symbolic gradient calculations, which, once converted to `JAX`, are evaluated numerically, necessitating many function evaluations.

The evaluation time for batched rollouts is, for long horizons (> 100 steps), roughly proportional to the number of primitive operations necessary for one timestep of the system dynamics; for shorter horizons, the overhead of launching the kernel for the GPU computations dominates. Therefore, it is of interest to inspect the geometric fabrics' expression and check whether dropping certain expressions reduces the number of primitive operations. A comparative table for the number of obstacles and different combinations of collision links is shown in Table 4-1. In the end, it was chosen to use the fabrics expression with 10 obstacles and collision spheres only at the chassis and the wrist, which reduced the number of expressions by over 78%.

Table 4-1: Number of primitive computations for different geometric fabrics expressions

No. of obstacles	**Collision Links (C=chassis, W=wrist, EE=end-effector)**			
	C	C, W	C, EE	C, W, EE
3	1,982	29,801	44,013	64,768
5	2,336	43,891	65,165	99,656
10	3,221	79,116	118,045	186,876
20	4,991	149,566	223,805	361,316

Alternatively, computing Jacobians can be made less complex by reducing the number of configuration variables to take the partial derivative against. This can be realized by locking some of the manipulator's joints.

4-3-2 ROS architecture

While *Scenario-Shield* can be applied to any underlying motion planner, the goal was to integrate it with the lab's existing implementation of geometric fabrics for a mobile manipulator. This existing implementation uses robot operating system (ROS) Noetic. The geometric fabrics planner is an action server that, upon receiving a goal configuration in the joint space, starts sending velocity commands to the robot.

The Scenario-Shield implementation integrates this through a new action server, which inherits all methods of the previous one, but adds a periodically executed verification loop. Unfortunately, ROS Noetic is compatible with Python up to version 3.8, while the parallelization code of `JAX` requires Python 3.10. This is why the parallel rollouts are moved to a separate JAX-rollout-server node, which communicates via rosbridge.

For the code please refer to this GitHub repository.

Chapter 5

Experiment

The described implementation of the *Scenario-Shield* algorithm together with a fabrics motion planner was simulated for a holonomic ground robot and a mobile manipulator. In both scenarios, we compare the proposed *Scenario-Shield* algorithm, including its extensions, as mentioned in Section 3-4, with a vanilla geometric fabrics planner.

This chapter begins with Section 5-1, which outlines the setup of the experiments, including the evaluation metrics, the computation time, and the randomization of the environment. Next, Section 5-2 and Section 5-3 display the experiment results for the holonomic ground robot and the mobile manipulator, respectively.

5-1 Setup

For each algorithm configuration, the experiment will be conducted over N randomized environments for K time steps. In each environment, we analyze the objective to reach the goal configuration $\boldsymbol{q}_G$ while avoiding a set of obstacles. The i^{th} experiment will yield the trajectory $\boldsymbol{x}_{1:K}^{(i)} = \begin{bmatrix} \boldsymbol{q}_{1:K}^{(i)} & \dot{\boldsymbol{q}}_{1:K}^{(i)} \end{bmatrix}^\top$. The following four metrics are evaluated:

1. The success rate $R_{\text{succ}} := \sum_{i=1}^{N} \mathbf{1}_{\mathcal{Q}_G}(\boldsymbol{q}_K^{(i)})/N$,
2. The average time to reach the goal among the successful runs $\bar{\tau} := \frac{\sum_{i=1}^{N} \mathbf{1}_{\mathcal{Q}_G}(\boldsymbol{q}_K^{(i)})\tau_i}{R_{\text{succ}} \cdot N}$,
3. The average of the resulting path lengths normalized by the corresponding path length found by optimal rapidly-exploring random tree (RRT)*, among the successful runs, and
4. The average total curvature among successful runs normalized to that of RRT* $\bar{\kappa} := \frac{\sum_{i=1}^{N} \mathbf{1}_{\mathcal{Q}_G}(\boldsymbol{q}_K^{(i)})\kappa_i/\kappa_i^{\text{RRT}}}{R_{\text{succ}} \cdot N}$,

where $\mathbf{1}_{\mathcal{Q}_G}(\boldsymbol{q}_K)$ is the indicator function evaluating to 1 if if the final configuration is within a small region around the goal $\mathcal{Q}_G$, i.e., $\boldsymbol{q}_K \in \mathcal{Q}_G$, and 0 otherwise. The time it took to reach the goal $\tau_i = \min\{k \in \{1, \dots, K\} \mid \boldsymbol{q}_k^{(i)} \in Q_G\}$, the total cuvature $\kappa_i = \sum_{k=1}^{K-1} \arccos \frac{(\boldsymbol{q}_k - \boldsymbol{q}_{k-1})^\top (\boldsymbol{q}_{k+1} - \boldsymbol{q}_k)}{||\boldsymbol{q}_k - \boldsymbol{q}_{k-1}|| \cdot ||\boldsymbol{q}_{k+1} - \boldsymbol{q}_k||}$, and κ_i^{RRT} is the total curvature of RRT's solution for the i^{th} environment. The reason to normalize by the path length and total curvature of RRT* is that neither the proposed method nor the baseline is an optimal method. Moreover, only the successful runs are counted towards the mean path length and mean total curvature, as they would otherwise skew the result.

Computation time

The computation time for the in-parallel running verification loop is comprised of three main components. First, offline, i.e., before runtime, the compilation time of the parallel fabrics rollout and during runtime, querying computing the rollouts, and solving the scenario program.

All computations were done on a computer running Ubuntu 22.04 with the following hardware specs: 128GB of RAM, an Intel(R) Core(TM) i9-14900 CPU, and a NVIDIA GeForce RTX 5090 GPU.

Randomizing the environment

To incentivize clustering of obstacles, which produces regions of no attraction, the spherical obstacles' centers are randomized following trajectories of finite, discrete random walks of fixed length. For N_{walks} walks, each of fixed length N_{length}, the set of obstacles $O \subset \mathcal{W}$ can be written as:

$$O := \bigcup_{i=1}^{N_{\text{walks}}} \{\boldsymbol{x} \in \mathcal{W} : ||\boldsymbol{x} - \boldsymbol{o_{i,j}}||_2^2 \leq o_r^2\} \text{ with} \tag{5-1}$$

$$\boldsymbol{o_{i,j+1}} = \boldsymbol{o_{i,j}} + s \cdot \begin{bmatrix} \cos(w_j) & \sin(w_j) \end{bmatrix}^\top \qquad \text{for } j \in \{1, \dots, N_{\text{length}}\}, \tag{5-2}$$

where s is a stepsize, $w_j \sim \mathcal{U}(S^1)$, and $\boldsymbol{o_{i,0}} \sim \mathcal{U}(\mathcal{W})$. An example environment is shown in Figure 3-1a.

After sampling the obstacles, the path planning algorithm RRT is employed to verify the feasibility of this environment. Since RRT is probabilistically complete, an environment for which no solution is found can be safely discarded without biasing the sampled environments too much.

5-2 Holonomic ground robot

The system dynamics of the holonomic ground robot follow the implementation details outlined in Section 4-2. Since the obstacles are circular, defined by their position and radius, the robot's geometry can be accounted for by expanding each obstacle by the robot's configuration space footprint. In the special case of a circular robot, this corresponds to expanding each obstacle by the robot's radius, effectively reducing the problem to that of a point robot navigating among enlarged circular obstacles. Here, it is assumed that the ground robot's physical shape projected onto the workspace resembles a circle.

All relevant parameters related to the underlying geometric fabrics planner, the *Scenario-Shield* algorithm, and the environment are collected and summarized in Table 5-1, where $\mathcal{B}_\infty(c, r) = \{q \in \mathcal{W} : \|q - c\|_\infty \leq r\}$ denotes the closed infinity norm ball.

Table 5-1: Parameters for the geometric fabric and *Scenario-Shield* for the ground robot

Description	Notation	Value
Geometric Fabric		
Discrete time step	Δt	0.02s
Integration scheme	-	forward Euler
Damping weight	w_B	1
Priority matrix weight	w_M	3
Scenario Shield		
Verification frequency	f_{verify}	0.2 Hz
Full rollout length	T	3000
Partial rollout length	t	1150
Sampling space	$B(q_0)$	$\mathcal{B}_\infty(q_0, 2)$
Mondrian coverage region $\mathcal{A}$	$C_{\mathcal{A}}$	1
Mondrian coverage region $\mathcal{A}^c$	$C_{\mathcal{A}^c}$	0.99
Monomials up to degree	d	9
Set failure probability of a rollout	δ	0.2
Risk of the scenario program	ϵ	0.2
Number of sampled scenarios	N	100

During execution, there are two computationally expensive operations: first, the rollouts, which can be computed sequentially on central processing unit (CPU) or in parallel on graphics processing unit (GPU), and second, solving the scenario program. The former's relation to the number of sampled states is depicted in Figure 5-1.

As expected, the CPU's computational complexity is linear in the batch size, while the GPU's is independent of the batch size, yet, up to batches of 120 initial conditions the overhead of launching the single instruction, multiple data (SIMD) program dominates, and for such small batch sizes computing on the CPU is faster w.r.t. the walltime.

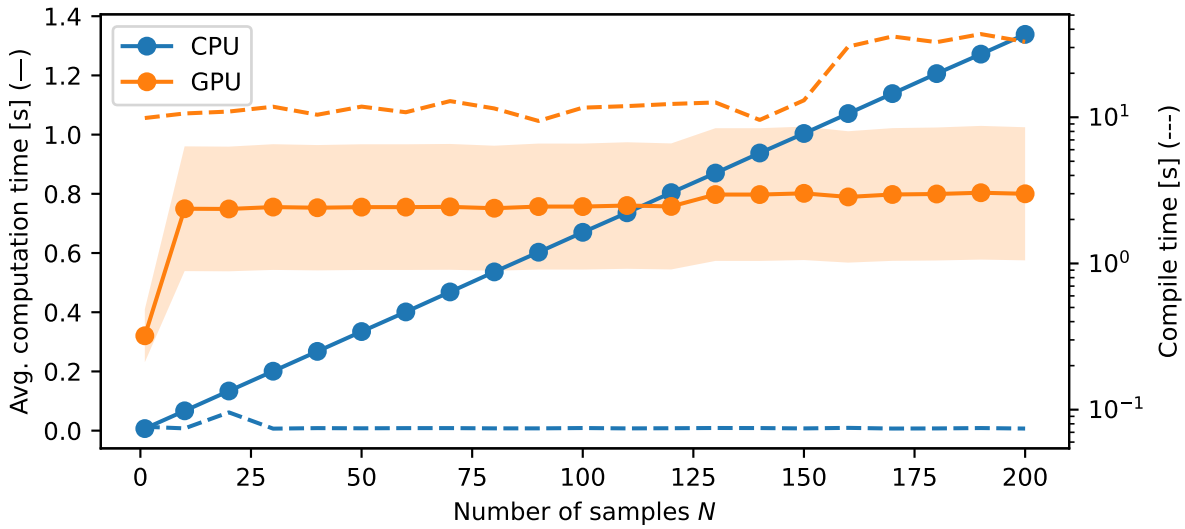


Figure 5-1: Rollout computation and compile times displayed with solid and dashed lines, respectively, for the ground robot's fabric with 100 obstacles and 1150 steps. Opaque band indicating one standard deviation.

Results

The results for the 100 experiments per algorithm configuration are shown in Table 5-2. The randomization is consistent across different algorithm configurations, i.e., the randomly generated environments have the same starting seed for each algorithm configuration. The baseline (0) is the underlying geometric fabrics planner without any runtime verification. For the following algorithm configurations, this baseline is equipped with different variants of the *Scenario-Shield* algorithm; these are, (1) the basic *Scenario-Shield* (*SS*) at 0.2Hz, (2) equipped with the scenario program, *SS-SP*, at a relatively lower frequency of 0.15Hz to accommodate solving the scenario optimization program, (3) the partial rollout extension *SS-PR* at the same frequency as *SS*, (4) *SS-PR* at a high frequency of 1Hz enabled through the shorter computation time of the partial rollouts opposed to full ones, and (5) both extensions combined, again at a slightly lower frequency to accommodate solving the optimization problem.

Table 5-2: Simulation results of each algorithm configuration of the ground robot over 100 randomized environments. Best value in bold. Standard deviations in brackets. Baseline in grey.

Algorithm Configuration	**Evaluation Metrics**			
(f_{verify} [Hz])	R_{succ}	$\bar{\tau}$ [s]	$\bar{l}$ [-]	$\bar{\kappa}$ [-]
baseline (-)	0.48	42.8 (5.02)	1.21 (0.091)	1.47 (3.39)
SS (0.2)	**0.93**	44.8 (6.92)	1.16 (0.079)	1.38 (1.56)
SS-SP (0.15)	0.89	44.0 (6.04)	1.10 (0.081)	**1.18** (1.31)
SS-PR (0.2)	0.60	**37.0** (1.98)	1.15 (0.067)	1.30 (1.32)
SS-PR (1)	0.84	38.6 (2.93)	1.21 (0.187)	1.44 (1.50)
SS-PR-SP (0.9)	0.78	39.4 (5.26)	**1.05** (0.256)	1.31 (1.32)

The *SS* increases the success rate by a factor of 1.9 compared to the baseline. However, due to the cost of full rollouts, its verification frequency is limited to a maximum of 0.2Hz. This is a significant limitation, as local samples from 5 seconds ago may no longer be informative to the present robot's state. In this case, this was addressed by holding the simulation until the verification loop had completed. However, this approach is not realistic for real-time operation. Consequently, in its unmodified form, the algorithm's verification results are only applicable to slow-moving systems.

When adding the scenario program, i.e., *SS-SP*, the allowable frequency drops even more to allow for the scenario program to be solved; moreover, the early switch back through the scenario program seems to decrease the path length, yet not in a statistically significant way. It appears that the scenario program decreases the success rate slightly, which, upon investigation, as displayed in Section 5-2, may be caused by incorrectly dropping a subgoal, despite not yet being in the region of attraction of the subgoal next in line. This could possibly be mitigated by decreasing the user-set allowable risk parameter ϵ in the scenario program; however, this would then demand a larger number of samples.

The next algorithm configuration *SS-PR* experiences a drastic drop in the success rate if run at the same low frequency of 0.2Hz, this is expected, as the partial rollouts classified through the conformal heuristic may include incorrect labels, hence, introducing misleading subgoals. However, the success rate performance can be largely recovered to 0.84% if the verification frequency is increased to 1Hz. This is an important milestone, as only in this configuration the verification becomes real-time viable for the ground robot, as the robot does not make significant progress within the sampling region within a verification period.

Lastly, combining both extensions into the *SS-PR-SP* configuration run at 0.9Hz, to again account for the scenario program, experiences a slight drop in success rate. It is not possible to attribute that to the slightly lower verification frequency or the risk of the scenario program. Furthermore, in this configuration, the relative path length is the shortest; however, its standard deviation is too large to deduce statistical significance at a reasonable significance level.

Example cases

In this section, a few examples are provided to give more intuition about the differences between the different extensions. In all cases, the ground robot must go from the start configuration $\boldsymbol{q}_0$, denoted by the green star, to the goal configuration $\boldsymbol{q}_g$, denoted by the red star, while avoiding the grey obstacles.

Figure 5-2 shows some key frames of the *SS* configuration at a verification frequency of 0.2Hz. At a simulation time of 10^s, the samples include, for the first time, those that belong to the region of attraction within the rollout horizon $\mathcal{A}$. By adapting the underlying dynamical system by selecting the configuration of one of those samples as a subgoal, the robot can be successfully steered out of the region of no attraction $\mathcal{A}^c$. However, note that the robot's state at $t = 15$s is not in the backwards reachable set

of the picked subgoal; hence, as shown at $t = 30$s, another subgoal must be placed into the queue. Upon successfully reaching the proximity of the second subgoal, the first subgoal can eventually be reached, successfully steering the robot into the region of attraction of the initial underlying dynamical system.

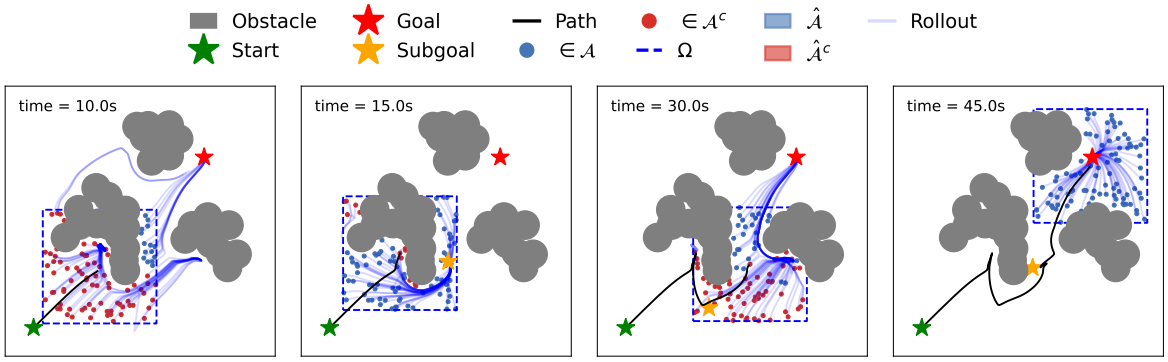


Figure 5-2: *Scenario-Shield* at 0.2Hz (selected frames)

Figure 5-3 shows how the partial rollouts affect the process. As can be seen, the rollouts appear shorter; however, due to the five times higher verification frequency, effectively, five times more samples are obtained during the same simulation interval. While this effect does not appear significant in the statistics, in this specific environment, the larger amount of samples allows to find a piece of the region of attraction earlier, hence, avoiding first getting stuck behind the first obstacle before adapting, as seen in Figure 5-2 at $t = 10$s.

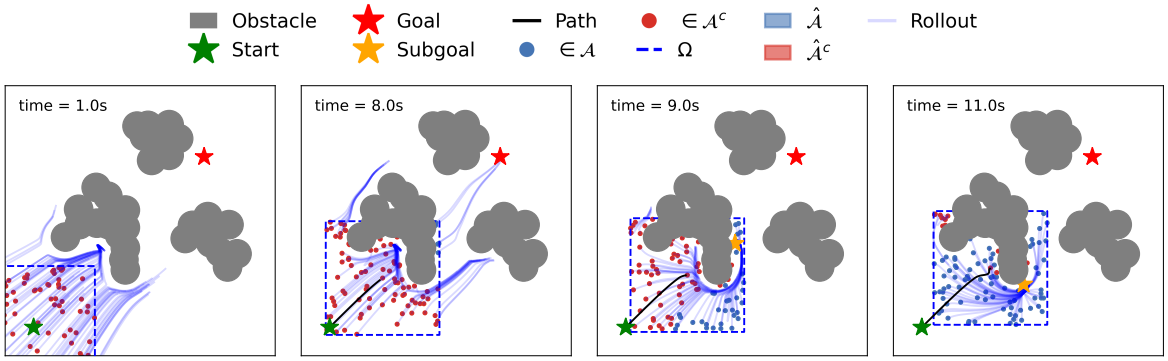


Figure 5-3: *Scenario-Shield* with partial rollouts at 1Hz (selected frames)

Next, Figure 5-4 shows a run of *SS-SP* which suffers from an incorrect estimate of the region of attraction, leading to taking a detour all the way to the right-most obstacle. In particular, in the last frame, it is apparent that the fidelity of the semi-algebraic set description leads to a spurious, incorrect region of attraction approximation. While this is unexpected, as the scenario program should allow for avoiding detours, this specific instance can be fixed by increasing the frequency, achieved by combining both extensions, as displayed in the selected frames of *SS-PR-SP* Figure 5-5.

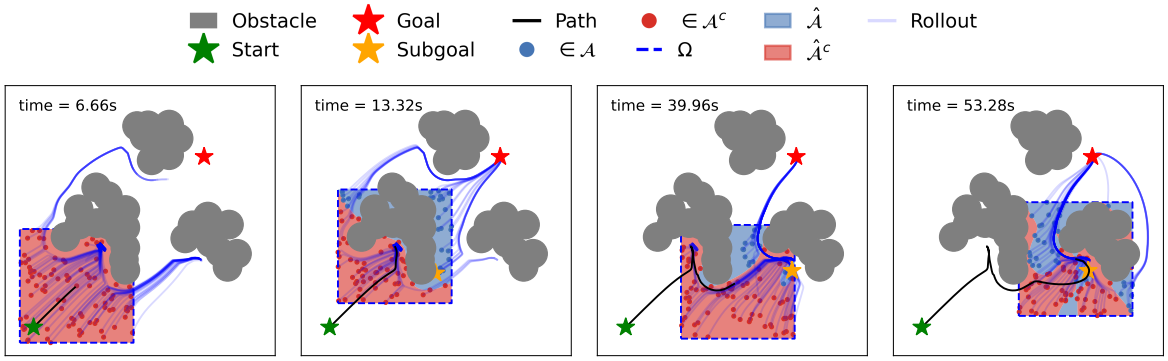


Figure 5-4: *Scenario-Shield* with the scenario program at 0.15Hz (selected frames)

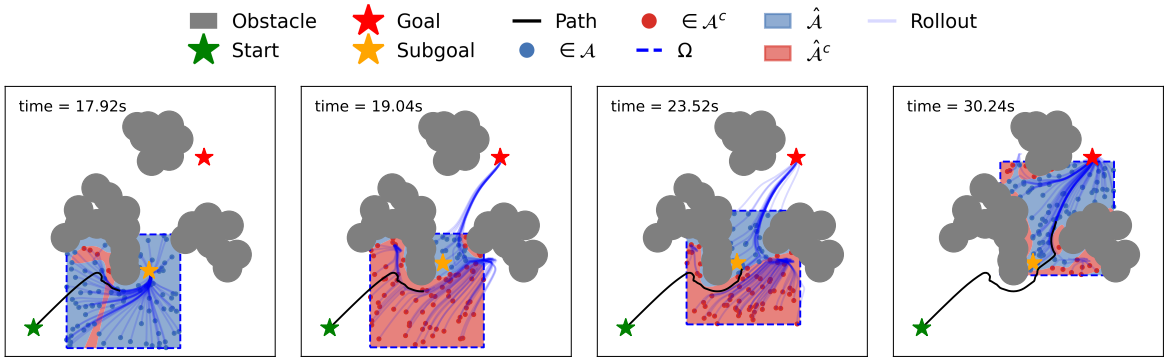


Figure 5-5: *Scenario-Shield* with partial rollouts and the scenario program at 0.9Hz (selected frames)

5-3 Mobile manipulator

For the mobile manipulator, the baseline approach using pure geometric fabrics is compared to the *Scenario-Shield* (SS) method, as well as two variants of the scenario program (SS-SP). These variants approximate the region of attraction in different ways: one using a semi-algebraic set, and the other using a minimum enclosing ellipsoid.

Conformal partial rollouts could not be applied to the manipulator. This was due to the poor predictive performance of the simple heuristic, which rendered the conformal prediction intervals uninformative. In particular, the label-conditional coverage regions produced by Mondrian conformal prediction (CP) failed to separate, making probably correct classification infeasible. A preliminary explanation is twofold: first, the reachability induced by geometric fabrics for manipulators depends on the pseudoinverse of the Jacobian, which can become singular in certain joint angle regimes; this likely has not been accounted for by the heuristic. Second, the heuristic may be overly simplistic. Incorporating additional information, such as joint velocities, or employing a more capable predictor, such as a multilayer perceptron, could potentially resolve this issue.

Therefore, we cannot leverage partial rollouts for the mobile manipulator scenario, resulting in the verification loop being run at only 0.2Hz due to the higher computational load of the full-length rollouts.

For the experiments, the parameter choices are summarized in Table 5-3, and the quantitative results are summarized in Table 5-4. The *Scenario-Shield* algorithm with the ellipsoidal approximation of the complement of the region of attraction improved the success rate by 87.5%. Interestingly, the scenario program, opposed to the previous ground robot case, further improved the success rate and decreased the mean time to goal compared to the basic *Scenario-Shield* algorithm. The reason for this is presented below, together with motion sequences of the different algorithms.

Table 5-3: Parameters for the geometric fabric and *Scenario-Shield* for the mobile manipulator robot

Description	Notation	Value
Geometric Fabric		
Discrete time step	Δt	0.05s
Integration scheme	-	forward Euler
Scenario Shield		
Verification frequency	f_{verify}	0.2 Hz
Full rollout length	K	500
Sampling space	$B(q_0)$	$\mathcal{B}_\infty(q_{0,1:3}, 1.5) \times \mathcal{B}_\infty(q_{0,4:9}, 0.3)$
Monomials up to degree	d	3
Risk of the scenario program	ϵ	0.2
Number of sampled scenarios	N	20

Table 5-4: Simulation results of each algorithm configuration of the ground robot over 20 randomized environments. Best value in bold. Standard deviations in brackets. Baseline in grey.

Algorithm Configuration	Evaluation Metrics			
	R_{succ}	$\bar{\tau}$ [s]	$\bar{l}$ [-]	$\bar{\kappa}$ [-]
baseline (-)	0.40	**33.2** (4.01)	**2.11** (0.167)	**1.87** (0.91)
SS	0.60	62.1 (18.2)	4.69 (0.888)	3.19 (2.73)
SS-SP (ellipsoid)	**0.75**	42.4 (12.9)	3.10 (0.498)	2.11 (1.30)
SS-SP (semi-algebraic)	0.70	45.7 (5.81)	4.03 (0.917)	2.05 (1.89)

The motion sequence of the vanilla geometric fabrics planner is shown in Figure 5-6, where the grey spheres represent obstacles, and the trailing green lines trace the paths of the robot's chassis and wrist joint. The planner temporarily becomes stuck behind the obstacles, which sufficiently reduces the robot's velocity for the obstacle avoidance term to vanish; note its dependency on velocity in Equation 4-11. Once the goal-reaching potential outweighs the now-negligible collision avoidance term, the robot proceeds to cut through the obstacles. This illustrates the limitation of geometric fabrics, which,

under the assumption of boundary-conforming collision geometries, may not preserve safety.

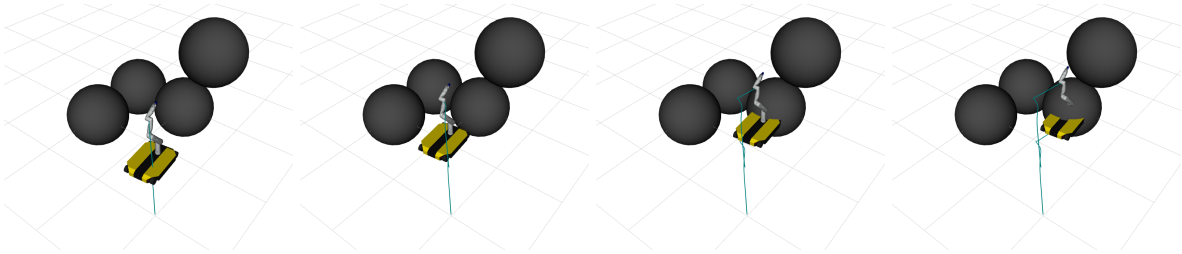


Figure 5-6: Geometric Fabrics getting first trapped in a deadlock, and then, violating a collision constraint. Green lines trail the chassis and wrist joints.

Next, Figure 5-7 shows the rollouts from locally sampled configurations; for visualization purposes, only the base positions of the rollouts are displayed. Blue paths indicate successful reach of the (sub)goal within the specified rollout horizon, while red paths indicate failure. While *Scenario-Shield* successfully navigates around the obstacles without getting stuck, the delay introduced by rollout computation leads to a key issue: after sampling the neighboring configurations, the robot continues to move while rollouts are still being computed in parallel. As a result, by the time a successful rollout is identified, the robot may have already progressed toward the goal; thus, executing the earlier sampled rollout may actually represent a regression. In the basic implementation of *Scenario-Shield*, the robot must move to a subgoal in order to remove it from the queue. This effect is visible in the second and third snapshots of Figure 5-7, where, in the second frame, the robot is actually closer to the goal than in the subsequent third frame.

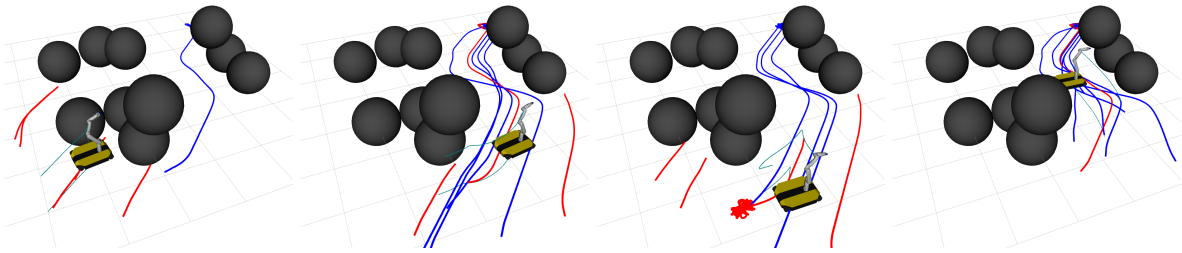


Figure 5-7: *Scenario-Shield* allowing the robot to navigate around the obstacles.

It turns out that using a continuous region of attraction estimate significantly alleviates this issue. When the robot has moved beyond the originally sampled configurations, it may no longer need to return fully to the initial state of a successful rollout, provided its current state lies within the estimated region of attraction. This would not be possible with a finite set approximation, where membership of intermediate states cannot be evaluated. This benefit is illustrated in the third snapshot of Figure 5-8: despite being spatially distant from the successful rollouts, the robot is able to proceed, as it has already entered the estimated region of attraction. The semi-algebraic set parameterization, i.e., describing the region of attraction $\mathcal{A}$ as the sublevel set of an optimized sum of monomials, exhibits similar behavior, as shown in Figure 5-9.

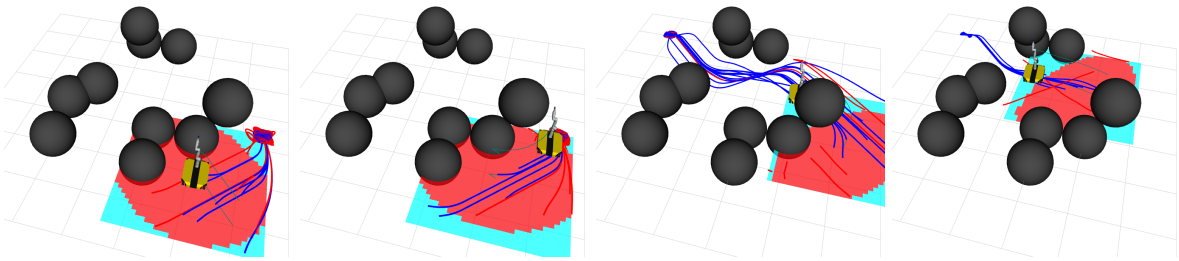


Figure 5-8: *Scenario-Shield* with the ellipsoid scenario program.

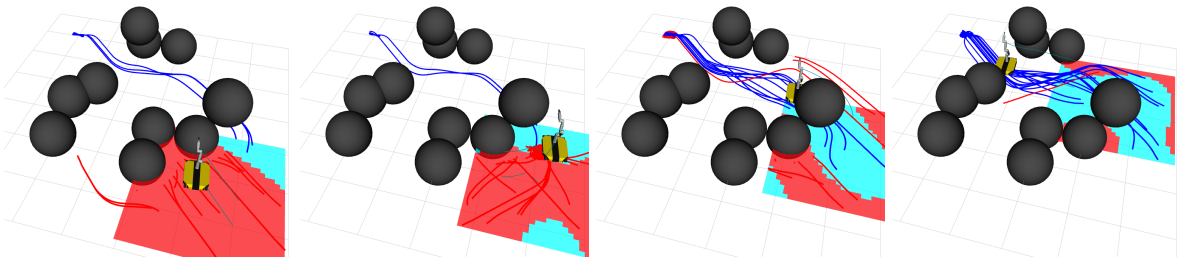


Figure 5-9: *Scenario-Shield* with the semi-algebraic scenario program.

Note that there is a trade-off between the two scenario program approaches. The semi-algebraic set description allows for a hard separation of the samples, provided that sufficiently high-order monomials are used. In contrast, the minimum enclosing ellipsoid approach does not offer such separation. Instead, it approximates the complement of the region of attraction, $\mathcal{A}^c$, by computing the minimum volume ellipsoid that encloses the negative samples. Imposing both inscribing and enclosing constraints would render the problem non-convex, and moreover, potentially infeasible, because a single ellipsoid cannot, in general, separate two classes. However, the semi-algebraic set approach suffers from a rapid growth in the number of decision variables, which increases exponentially with the dimensionality of the configuration space. Consequently, even though the semi-algebraic formulation can be solved via linear programming, while the ellipsoidal approach requires solving a convex quadratic program, Figure 5-10 shows that for four or more dimensions, the ellipsoid-based method is computationally more efficient.

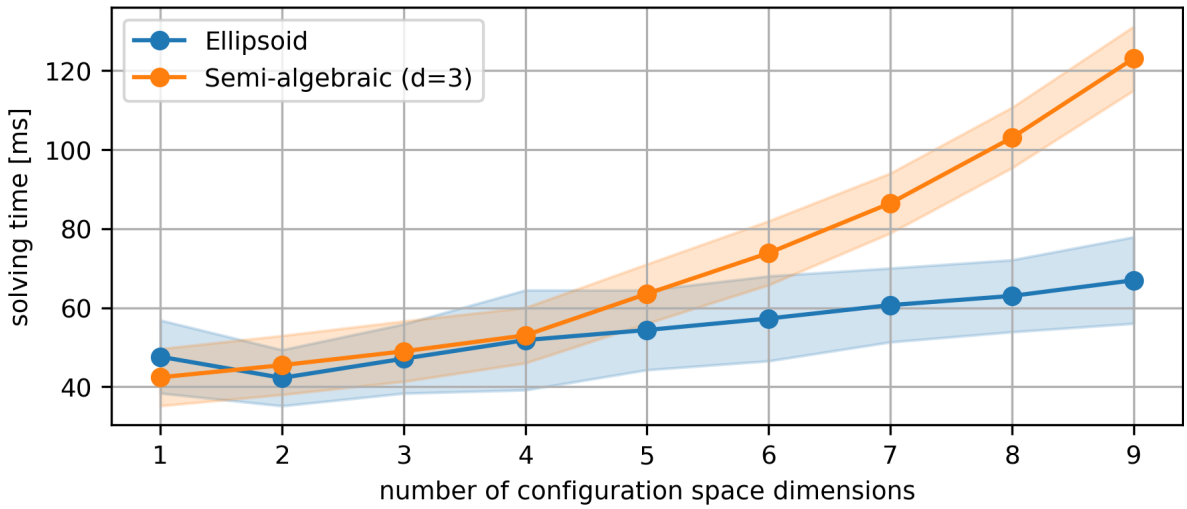


Figure 5-10: Mean solving times with standard deviations over 100 randomized scenario programs.

Chapter 6

Conclusion

This thesis addressed the global attractivity problem in geometric motion planning. While several modern feedback planners guarantee almost global attractivity, i.e., for all initial conditions except those on a set of Lebesgue measure zero, these guarantees typically rely on restrictive assumptions. Notably, most geometric planners with an underlying dynamical system require that the obstacle set is a union of disjoint, star-shaped obstacles, a condition rarely satisfied in practical scenarios encountered by robots. When such assumptions are violated, the system can remain stable; however, the complement of the region of attraction may have positive measure, thereby introducing deadlocks in robotic motion planning.

A runtime verification algorithm, called *Scenario-Shield*, is proposed that adapts the underlying dynamical system during runtime to steer it back into the region of attraction. Motivated by the availability of closed-form feedback laws and recent advances in parallel computing, this work proposes a sampling-based approach. Neighboring robot configurations are locally sampled and forward simulated under the geometric planner's dynamics to approximate the finite-time region of attraction. This enables the selection of intermediate goals that steer the robot toward these verified regions. The process is repeated iteratively, online, until the current configuration is connected queue of finite-time backward reachable sets.

The adaptation mechanism operates at a fixed verification frequency, while the underlying feedback planner can run at a higher control rate. To make the verification loop real-time viable, forward simulations are computed in parallel. Additionally, *conformal prediction* is employed to decide correctly with high probability whether a sampled configuration lies within the K-step backward reachable set using only a shorter k-step forward simulation (with $k \ll K$), thereby significantly reducing computational overhead.

A further fragment of picking subgoals from a finite subset of the region of attraction is that, due to possibly large dispersion of the samples, the path connecting all subgoals

will make unnecessary detours. This is addressed by dropping subgoals based on reaching an uncountably infinite set approximation, which is obtained through the *scenario approach*.

Finally, the adaptation of goals introduces a switched dynamical system. While individual subsystems may be stable, switching can lead to instability in general [35]. However, in this work, the use of probably correct attractivity classifications and probably approximately correct region estimates allows to ensure that the probability of infinite switches approaches zero, thus, with almost certainty not inducing an instability.

While a guarantee of global attractivity could not be achieved, the experimental validation demonstrated that the proposed *Scenario-Shield* algorithm boosts the success rate by a factor of ≈ 1.94 and ≈ 1.88 for a holonomic ground robot and a mobile manipulator, respectively. Moreover, an argument can be made that if the sampling space of *Scenario-Shield* is sufficiently large, such that it intersects the underlying region of attraction and the queue of backward reachable sets can be established, the algorithm will successfully adapt the underlying geometric planner to reach the goal.

6-1 Limitations

Despite its effectiveness, the method has several limitations related to the statistical uncertainty quantification techniques as outlined below.

First, the partial rollout technique is calibrated using conformal prediction (CP), enabling shorter, thus faster, rollouts that still allow for probably correct classification of whether initial conditions lie within the backward reachable set of a goal configuration. However, this relies on the assumption that nonconformity scores are exchangeable between calibration and test time, a condition that may not always hold.
Moreover, as seen for the mobile manipulator, the simple heuristic is such a poor predictor that the coverage region obtained through CP is uninformative. Specifically, in the case of Mondrian CP, the coverage regions overlap.

Second, the underlying region of attraction is approximated using the scenario approach, which comes with two main limitations: (1) To keep the scenario program tractable when optimizing the chosen semi-algebraic set, only monomials up to a finite degree are considered, potentially limiting the approximation accuracy. (2) Ideally, the approximated region of attraction $\hat{\mathcal{A}}$ would define as a switching surface. However, the scenario approach requires that test constraints be independent and identically distributed (i.i.d.) relative to the sampled constraints. Switching precisely at the contour of $\hat{\mathcal{A}}$ would require evaluating a point that depends on the sampled scenarios through the solution of the scenario program, hence, violating this i.i.d. assumption.

Third, the local sampling space of neighboring robot configurations used in this method is currently static, i.e., it translates with the robot but does not adapt in size or shape. Consequently, if the robot starts outside the region of attraction and never flows toward its boundary, the sampling region may never intersect with the region of attraction, leading to a persistent deadlock. This observation motivates the first direction for future work.

6-2 Future work

Based on the limitations identified in this work, three main directions for future research are proposed: (1) improving the sampling strategy, (2) addressing the exchangeability assumption in conformal prediction, and (3) integrating the method into hierarchical motion planning frameworks.

The sampling region could be made adaptive, expanding or shifting based on the observed fraction of samples that lie within the estimated region of attraction. This would help prevent deadlocks by increasing the likelihood of intersecting the region of attraction even when the initial configuration is far from it. In addition, the current sampling strategy for the ground robot follows a uniform distribution but rejects configurations within obstacles. In the case of the mobile manipulator, however, pure uniform sampling is performed, which often produces configurations in self-collision. Rolling out such invalid samples wastes computational resources. To address this, rejection sampling, or more sophisticated sampling techniques, should be implemented. However, this introduces a tradeoff between the computational cost of collision checking and the inefficiency of blindly simulating invalid samples.

Verifying the assumption of exchangeable nonconformity scores in practice is challenging. This is primarily because nonconformity scores depend on both the prediction and the ground truth, but the latter is unavailable at test time. As a result, potential distribution shifts may go undetected. Robust conformal prediction [36], which retains guarantees under shifts within an a priori assumed bound, can help mitigate this issue, but they ultimately face the same challenge: ensuring that the actual test-time distribution shift does not exceed the assumed bounds.

In the context of feedback motion planning, however, uncertainty is primarily driven by environmental variability. Interestingly, distinct distributions of obstacle placements may still induce similar distributions of nonconformity scores. Analyzing this relationship in more detail could provide a practical pathway for verifying the exchangeability assumption; namely, by checking whether the observed environment aligns with an acceptable distribution, which is known not to cause a shift within the assumed bounds.

This thesis focused on improving the feedback motion planning layer, but its contributions also have implications for hierarchical motion planning architectures. In such systems, a global planner may operate with global obstacle information and a simplified dynamical model to compute a coarse plan, while a local planner refines this plan using locally perceived obstacles and a more accurate representation of the system dynamics. Connecting these layers in an assume-guarantee framework typically requires explicit knowledge of the local planner's region of attraction. However, as shown in this thesis, local planners in the class of feedback planners, such as navigation functions or geometric fabrics, do not an analytic expression if the assumptions for almost global attractivity are violated. Instead, the proposed sampling-based method can be employed to empirically approximate the local planner's region of attraction, enabling assume-guarantee style relationships.

This thesis contributes toward closing the gap between geometric motion planning guarantees and the practical realities of robotics applications, where required assumptions for reach-avoid guarantees do not hold.

Bibliography

[1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 1, 2002, ISSN: 0005-1098. DOI: 10.1016/S0005-1098(01)00174-1. (visited on 11/26/2025).

[2] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Advances in Applied Mathematics*, vol. 11, no. 4, pp. 412–442, Dec. 1, 1990, ISSN: 0196-8858. DOI: 10.1016/0196-8858(90)90017-S. (visited on 07/30/2025).

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*, I. J. Cox and G. T. Wilfong, Eds., New York, NY: Springer, 1990, pp. 396–404, ISBN: 978-1-4613-8997-2. DOI: 10.1007/978-1-4613-8997-2_29. (visited on 10/25/2025).

[4] S. M. LaValle, *Planning Algorithms*, 1st ed. Cambridge University Press, May 29, 2006, ISBN: 978-0-521-86205-9. DOI: 10.1017/CBO9780511546877. (visited on 04/02/2025).

[5] S. Bakker, R. Perez-Dattari, C. D. Santina, W. Böhmer, and J. Alonso-Mora, "Tamed-PUMA: Safe and stable imitation learning with geometric fabrics," in *Proceedings of the 7th Annual Learning for Dynamics \& Control Conference*, PMLR, May 22, 2025, pp. 405–418. (visited on 11/26/2025).

[6] L. Huber, A. Billard, and J.-J. Slotine, "Avoidance of convex and concave obstacles with convergence ensured through contraction," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1462–1469, Apr. 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2893676. (visited on 09/02/2025).

[7] W. Yao, B. Lin, B. D. O. Anderson, and M. Cao, "Guiding vector fields for following occluded paths," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 4091–4106, Aug. 2022, ISSN: 1558-2523. DOI: 10.1109/TAC.2022.3179215. (visited on 09/02/2025).

[8] N. Ratliff and K. V. Wyk, *Fabrics: A foundationally stable medium for encoding prior experience*, Sep. 14, 2023. DOI: 10.48550/arXiv.2309.07368. (visited on 03/18/2025).

[9] T. Merva, S. Bakker, M. Spahn, D. Zhao, I. Virgala, and J. Alonso-Mora, "Globally-guided geometric fabrics for reactive mobile manipulation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 10, no. 6, pp. 5553–5560, Jun. 2025, ISSN: 2377-3766. DOI: 10.1109/LRA.2025.3562005. (visited on 12/03/2025).

[10] K. Van Wyk, M. Xie, A. Li, *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, Apr. 2022, ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3143311. (visited on 10/15/2025).

[11] J. Barraquand and J.-C. Latombe, "A monte-carlo algorithm for path planning with many degrees of freedom," in *, IEEE International Conference on Robotics and Automation Proceedings*, May 1990, 1712–1717 vol.3. DOI: 10.1109/ROBOT.1990.126256. (visited on 10/25/2025).

[12] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992, ISSN: 2374-958X. DOI: 10.1109/70.163777. (visited on 10/25/2025).

[13] D. J. Gonon, D. Paez-Granados, and A. Billard, "Reactive navigation in crowds for non-holonomic robots with convex bounding shape," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4728–4735, Jul. 2021, ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3068660. (visited on 12/05/2025).

[14] S. M. Khansari-Zadeh and A. Billard, "A dynamical system approach to realtime obstacle avoidance," *Autonomous Robots*, vol. 32, no. 4, pp. 433–454, May 1, 2012, ISSN: 1573-7527. DOI: 10.1007/s10514-012-9287-y. (visited on 12/05/2025).

[15] C. K. Verginis and D. V. Dimarogonas, "Adaptive robot navigation with collision avoidance subject to 2nd-order uncertain dynamics," *Automatica*, vol. 123, p. 109 303, Jan. 1, 2021, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2020.109303. (visited on 07/30/2025).

[16] A. Bloch, D. E. Chang, N. Leonard, and J. Marsden, "Controlled lagrangians and the stabilization of mechanical systems. II. potential shaping," *IEEE Transactions on Automatic Control*, vol. 46, no. 10, pp. 1556–1571, Oct. 2001, ISSN: 1558-2523. DOI: 10.1109/9.956051. (visited on 12/05/2025).

[17] A. Bylard, R. Bonalli, and M. Pavone, "Composable geometric motion policies using multi-task pullback bundle dynamical systems," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 7464–7470. DOI: 10.1109/ICRA48506.2021.9561320. (visited on 06/09/2025).

[18] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Cham: Springer International Publishing, 2022, ISBN: 978-3-031-06648-1. DOI: 10.1007/978-3-031-06649-8. (visited on 08/01/2025).

[19] L. Lindemann, Y. Zhao, X. Yu, G. J. Pappas, and J. V. Deshmukh, *Formal verification and control with conformal prediction*, Aug. 31, 2024. DOI: 10.48550/arXiv.2409.00536. (visited on 03/04/2025).

[20] M. C. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, Jan. 2008, ISSN: 1052-6234, 1095-7189. DOI: 10.1137/07069821X. (visited on 11/14/2025).

[21] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, Dec. 1, 2009, ISSN: 1367-5788. DOI: 10.1016/j.arcontrol.2009.07.001. (visited on 12/03/2025).

[22] K. Burns and M. Gidea, *Differential Geometry and Topology: With a View to Dynamical Systems*. CRC Press, May 27, 2005, 403 pp., ISBN: 978-1-4200-5753-9.

[23] T. Cunis, "Probabilistic safety analysis for model predictive control with a case study on aircraft upset recovery," in *Nonlinear and Constrained Control: Applications, Synergies, Challenges and Opportunities*, E. Garone, I. Kolmanovsky, and T. W. Nguyen, Eds., Cham: Springer Nature Switzerland, 2025, pp. 181–206, ISBN: 978-3-031-82681-8. DOI: 10.1007/978-3-031-82681-8_8. (visited on 11/25/2025).

[24] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, May 2006, ISSN: 0018-9286. DOI: 10.1109/TAC.2006.875041. (visited on 10/21/2025).

[25] P. Billingsley, *Probability and measure* (Wiley series in probability and mathematical statistics), 3rd ed. New York: Wiley, 1995, 593 pp., ISBN: 978-0-471-00710-4.

[26] S. P. Boyd and L. Vandenberghe, *Convex optimization*, Version 29. Cambridge New York Melbourne New Delhi Singapore: Cambridge University Press, 2023, 716 pp., ISBN: 978-0-521-83378-3.

[27] Y. Tang, J.-B. Lasserre, and H. Yang, "Uncertainty quantification of set-membership estimation in control and perception: Revisiting the minimum enclosing ellipsoid," in *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, PMLR, Jun. 11, 2024, pp. 286–298. (visited on 08/29/2025).

[28] J. Bradbury, R. Frostig, P. Hawkins, *et al.* "JAX: Composable transformations of python+NumPy programs." (2018), [Online]. Available: http://github.com/jax-ml/jax.

[29] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, 2018.

[30] I. Alentev, L. Kozlov, and S. Nedelchev, *JaxADi: Bridging CasADi and JAX for efficient numerical computing*, 2024.

[31] M. Xie, K. V. Wyk, A. Li, M. A. Rana, Q. Wan, D. Fox, B. Boots, and N. Ratliff, *Geometric fabrics for the acceleration-based design of robotic motion*, Jun. 25, 2021. DOI: 10.48550/arXiv.2010.14750. (visited on 11/30/2025).

[32] R. Pérez-Dattari, C. Della Santina, and J. Kober, "PUMA: Deep metric imitation learning for stable motion primitives," *Advanced Intelligent Systems*, vol. 6, no. 11, p. 2 400 144, 2024, ISSN: 2640-4567. DOI: 10.1002/aisy.202400144. (visited on 11/30/2025).

[33] K.-J. Simmoteit, P. Schillinger, and L. Rozo, "Diffeomorphic obstacle avoidance for contractive dynamical systems via implicit representations," presented at the Robotics: Science and Systems XXI, vol. 21, Jun. 21, 2025, ISBN: 979-8-9902848-1-4. (visited on 11/30/2025).

[34] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.

[35] D. Liberzon, *Switching in Systems and Control* (Systems & Control: Foundations & Applications), red. by T. Başar. Boston, MA: Birkhäuser, 2003, ISBN: 9781461265740 9781461200178. DOI: 10.1007/978-1-4612-0017-8. (visited on 03/06/2025).

[36] M. Cauchois, S. Gupta, A. Ali, and J. C. Duchi, "Robust validation: Confident predictions even when distributions shift," *Journal of the American Statistical Association*, vol. 119, no. 548, pp. 3033–3044, Oct. 1, 2024, ISSN: 0162-1459. DOI: 10.1080/01621459.2023.2298037. (visited on 12/05/2025).

Glossary

List of Acronyms

CP	conformal prediction
MPC	model predictive control
PAC	probably approximately correct
ROS	robot operating system
DoF	degrees of freedom
RRT	rapidly-exploring random tree
i.i.d.	independent and identically distributed
SIMD	single instruction, multiple data
GPU	graphics processing unit
XLA	accelerated linear algebra
CPU	central processing unit