Reactive Motion Generation through Probabilistic Dynamic Graphs

Reaktive Bewegungserzeugung durch probabilistische dynamische Graphen
Master thesis by Sebastian Zach
Date of submission: May 4, 2025

1. Review: Jan Peters
2. Review: Kay Pompetzki
3. Review: An Thai Le
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Intelligent Autonomous Systems

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 APB TU Darmstadt

Hiermit erkläre ich, Sebastian Zach, dass ich die vorliegende Arbeit gemäß § 22 Abs. 7 APB der TU Darmstadt selbstständig, ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt habe. Ich habe mit Ausnahme der zitierten Literatur und anderer in der Arbeit genannter Quellen keine fremden Hilfsmittel benutzt. Die von mir bei der Anfertigung dieser wissenschaftlichen Arbeit wörtlich oder inhaltlich benutzte Literatur und alle anderen Quellen habe ich im Text deutlich gekennzeichnet und gesondert aufgeführt. Dies gilt auch für Quellen oder Hilfsmittel aus dem Internet.

Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 4. Mai 2025

S. Zach

Abstract

Robust and efficient motion planning is essential for robots operating in complex, dynamic environments, but achieving both fast reactions and globally optimal paths remains a major challenge. This thesis tackles the problem of balancing computational efficiency and effective global path-finding in robot motion planning. Reactive methods like Riemannian Motion Policies (RMPs) offer real-time performance and safety guarantees but often get stuck in local minima in complex environments. In contrast, global planning methods can find optimal paths but are often too slow for dynamic settings. We propose a hierarchical approach that combines local reactive control with global planning using probabilistic dynamic graphs. Our method integrates RMPs for reactive control, Stein Variational Probabilistic Roadmaps (SVPRMs) for adaptive sampling-based planning, and Value Iteration for computing paths. In this setup, roadmap nodes act as attraction points in configuration space, and their positions are continuously adapted to environmental changes using Stein Variational Gradient Descent. The advantage function from Value Iteration adjusts the strength of these attraction points within the RMP framework, blending local and global strategies. Experiments in 2D environments and with a simulated 7-DoF Franka robot show that our approach outperforms purely reactive methods in environments with local minima and achieves competitive success rates in dynamic settings compared to established techniques. The results also suggest that the global roadmap can be updated less frequently than the reactive controller, with Stein Variational Gradient Descent helping to maintain the accuracy of the value function between updates. Overall, this work demonstrates a promising strategy for achieving fast, safe, and globally-aware robot motion planning in dynamic environments.

Contents

Abbreviations, Symbols and Operators

List of Abbreviations

Notation	Description
APF	Artificial Potential Field
DoF	Degree of Freedom
GDS	Geometric Dynamical System
GPMP	Gaussian Process Motion Planning
JIT	Just-in-Time
KL	Kullback-Leibler divergence
MDP	Markov Decision Process
MPC	Model Predictive Control
MPPI	Model Predictive Path Integral
ODE	Ordinary Differential Equation
OSC	Operational Space Control

PD	Proportional-Derivative controller
PRM	Probabilistic Roadmap
RBF	Radial Basis Function
RMP	Riemannian Motion Policy
RRT	Rapidly-Exploring Random Tree
SBMP	Sampling-based Motion Planning
SD	Standard Deviation
SDF	Signed Distance Field
SVGD	Stein Variational Gradient Descent
SVPRM	Stein Variational Probabilistic Roadmap

1. Introduction

The increasing integration of robots into human environments demands navigation systems that ensure safety while efficiently reaching goals. As robots transition from controlled industrial settings to dynamic spaces shared with humans, the capability to react to unpredictable movements becomes crucial. This thesis addresses a fundamental challenge in robot motion planning: balancing reactive collision avoidance with global path optimization in dynamic environments.

1.1. Motivation

Human-robot collaboration offers significant potential across various domains, from manufacturing to healthcare. However, realizing this potential requires robots to operate safely alongside humans, who represent dynamic obstacles with unpredictable trajectories. Unlike static environments where paths can be pre-planned, dynamic settings require continuous adaptation to changing conditions.

Safety relies on detecting potential future collisions and generating reactive movements. Detection of obstacles can be done in various ways [1, 2, 3], we will focus on generating optimal reactive motions. Robots must avoid collisions and reach their destinations without getting stuck in local minima.

Motion planning varies from reactive methods [4, 5, 6, 7] to global techniques [8, 9, 10, 11]. Reactive motion generation methods prioritize computational speed, enabling high-frequency control loops that minimize delays between obstacle detection and robot response. This responsiveness is achieved by limiting the considered environmental information and planning horizon, focusing only on immediate actions rather than complete future trajectories. While these approaches excel at local obstacle avoidance, they

frequently encounter local minima: situations where the robot becomes trapped in configurations from which purely reactive strategies cannot progress toward the goal.

Conversely, global planning methods provide completeness guarantees, ensuring that the optimal path will be found if one exists. These approaches consider the entire environment and planning horizon, generating complete trajectories from start to goal. However, this comprehensive planning typically requires significant computational resources, reducing the frequency at which plans can be updated in response to environmental changes. This limitation can compromise safety in highly dynamic environments where rapid reactions are necessary.

The challenge lies in developing approaches that combine the complementary strengths of reactive and global methods, maintaining the computational efficiency and responsiveness of reactive approaches while overcoming their susceptibility to local minima through global awareness.

1.2. Proposed Approach

This thesis presents a hierarchical approach that integrates reactive control with global planning to enable robots to navigate safely in dynamic environments while avoiding local minima. The key insight driving our method is that different components of the planning system can operate at different frequencies and in different coordinate spaces, creating a hierarchy that balances computational efficiency with plan optimality.

Our approach combines three key components:

1. Riemannian Motion Policies (RMPs) [5] provide a framework for reactive control with safety guarantees, enabling high-frequency responses to dynamic obstacles.
2. Stein Variational Probabilistic Roadmaps (SVPRMs) [12] create and continuously adapt a graph of configuration points that effectively represents the connectivity of the free space.
3. Value Iteration [13] computes optimal paths through the roadmap, producing a policy that directs the robot from any node toward the goal.

The innovation lies in how these components interact: the global planner sets sub-goals in the environment that are dynamically weighted based on their value. These sub-goals are followed by the fast reactive controller, while their positions are continuously updated

using Stein Variational Gradient Descent (SVGD) to maintain a valid representation of the free space despite moving obstacles. This approach creates a smooth transition between the different timescales of planning and control, allowing the system to adapt to dynamic environments while maintaining global awareness.

While learning-based methods such as reinforcement learning have shown remarkable success recently [14, 15, 16, 17] they present challenges for safety-critical applications. These approaches often achieve robustness through domain randomization [18], training on perturbed environment dynamics with the expectation that real-world conditions represent just another variation. However, when real systems differ significantly from training environments or variations are difficult to simulate, e.g. when human behavior is involved, performance guarantees become difficult to establish.

Safety-critical applications also benefit the ability to understand why the system made specific decisions and how failures occur. Classical approaches offer explicit domain knowledge and theoretical guarantees that can be difficult to establish with learning-based methods. Our method relies primarily on classical approaches while maintaining integration points where learning techniques could be applied in future work. This design choice prioritizes safety guarantees and system transparency while preserving the potential for data-driven improvements.

Different levels in planning hierarchies often operate most effectively in different coordinate spaces. For example, high-level planning might reason in task space (end-effector position and orientation), while low-level control operates in joint space (robot configuration). For robots with redundant degrees of freedom, such as 7-DoF manipulators, this creates the need for controllers that can efficiently map between these spaces. Our approach addresses this challenge through RMPs, which provide a mathematically rigorous framework for combining policies defined in different spaces.

The main contributions of this thesis are:

1. The composition of RMPs, SVPRMs and Value Iteration in a hierarchical motion planning framework into a cohesive system that enables robots to navigate dynamic environments while avoiding local minima.
2. A multi-timescale approach that decouples global planning from local control while maintaining coordination between them: advantage functions weight reactive policies at the local level, while SVGD continuously adapts roadmap nodes to bridge the temporal gap between slow global updates and fast reactive control.

3. Experimental validation in both 2D toy examples and physically simulated 7-DoF robot environments, demonstrating the approaches effectiveness across different dimensionalities and dynamics.

1.3. Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 reviews relevant literature on motion planning approaches, from myopic reactive methods to global planners, with particular attention to hierarchical approaches and techniques for dynamic environments.

Chapter 3 introduces the theoretical foundations underpinning our approach, including Riemannian Motion Policies, Stein Variational Probabilistic Roadmaps, and Value Iteration.

Chapter 4 details our proposed hierarchical approach, explaining how global planning and reactive control are integrated through dynamic roadmaps and advantage functions.

Chapter 5 describes the experimental setup used to evaluate our approach, including both 2D toy environments and simulated 7-DoF robot scenarios.

Chapter 6 presents and analyzes the experimental results, comparing our approach against established baselines and examining the impact of different parameter settings.

Chapter 7 summarizes the findings, discusses limitations, and suggests directions for future research.

2. Related Work

Motion planning is a well-studied problem in robotics, concerned with the generation of collision-free trajectories from initial states to goals. As robots increasingly operate in complex, dynamic environments alongside humans, planning approaches must balance computational efficiency, path optimality, and reactivity.

Methods in this field can be categorized by their environmental awareness, temporal scope, and computational strategy. Each category presents distinct trade-offs between real-time performance and solution quality.

This chapter examines five families of approaches: myopic methods that offer reactivity but risk local minima, sampling-based planners that efficiently handle high-dimensional spaces, trajectory optimization techniques that generate smooth paths, hierarchical approaches that combine planning levels, and specialized extensions for dynamic environments with moving obstacles.

2.1. Myopic Methods

Myopic (reactive or local) methods compute control commands based on the current state and immediate surroundings without planning complete trajectories. While computationally efficient, they are susceptible to local minima problems.

The foundational Artificial Potential Field (APF) approach [4] models the robot's configuration space as a potential field with attractive goals and repulsive obstacles. Khatib extended this to Operational Space Control (OSC) [19] for dynamic control of robotic manipulators, transforming dynamics from joint to task space along the kinematic chain. Both methods suffer when attractive and repulsive forces balance, creating local minima.

Although myopic methods are limited to creating movement based on the current state alone, they can still be used to generate complex behavior, e.g. with diffeomorphic

mappings. Saveriano et al. [20] model demonstrations as trajectories of a stable linear attractor on some Riemannian manifold and use a Gaussian Mixture Model to map desired space. ImitationFlow [21] similarly employs normalizing flows to find mappings, supporting not just point attractors but also oscillatory patterns.

Riemannian Geometry is well suited for describing robotics problems [22]. It can be used to combine multiple simpler motion policies, defined in different spaces, into more complex motions in a stable geometrically consistent manner as done with Riemannian Motion Policies (RMPs) [5]. A more detailed description of RMPs and how tree-structured task spaces [23] and automatic differentiation libraries [24] can be used to compute them efficiently is provided in Chapter 3.

RMPs have been applied to multi-robot systems [25], under-actuated systems [26], LiDAR-to-camera transfer learning [27], and human-robot interaction [28]. In the latter, mission-specific policies are reweighed by their alignment to the human operators intent, while preserving independent motions, identified with an orthogonality measure to the input and already processed policies.

There are several extensions to the RMP framework. The challenge of incorporating changing importance weights, additional to the distance based metrics, is addressed with RMPfusion [29]. They demonstrate that additional correction terms, based on the weight functions' gradients, are necessary to maintain Lyapunov stability. This approach requires all subtask policies to be Geometric Dynamical Systems (GDS) [23] but allows for weight-changing functions that can also be learned.

A significant theoretical extension, Geometric Fabrics [30], overcomes GDS limitations by allowing metrics to depend on both position and velocity while choosing the forcing function independently. Instead of using solely position-dependent Riemannian metrics or having to limit expressivity of forcing functions with curvature terms, it employs Finsler energies for velocity-dependent geometries. The approach modifies the pullback operations and replaces the resolve operation with an "energize" operation, decoupling priority weights from the geometry while ensuring stability, thus enabling more expressive and natural motions.

A probabilistic generalization is presented in Composable Energy Policies [6]. This framework combines multiple probabilistic policies by maximizing their joint probability, which they show is equivalent to minimizing a sum of energy functions. The authors evaluate Cross-Entropy Method and reward-weighted regression to find the maximum likelihood without explicit task map inversions. They show that RMPs are a special case where all policies are Gaussian, limiting expressivity. Whereas their method supports multimodal

or partially uniform distributions that can better represent optimal behavior, especially in complex environments.

Control Barrier Functions offer explicit safety guarantees by defining safe regions in state space. Recent work has focused on dynamic environments [31] and combining multiple safety objectives [7], where they minimize not only the Mahalanobis distance of desired accelerations in the joint space, but the sum of distances in all task spaces.

While these reactive approaches offer computational efficiency and have been extended to give control-theoretical safety guarantees, all of these can become trapped in local minima when navigating complex environments.

2.2. Sampling-Based Motion Planning

Sampling-based Motion Planning (SBMP) methods are at the opposite end of the planning spectrum. They provide more global solutions by constructing a sparse representation of the configuration space.

Unlike grid-based approaches that suffer from the curse of dimensionality, sampling-based methods scale more effectively to high-dimensional spaces. Grid-based methods such as A* [32], D* [33], and hybrid A* [34] discretize the state space and define motion primitives connecting states into a graph structure. While effective for low-dimensional problems, these methods become computationally intractable as dimensions increase due to the exponential growth in the number of grid cells.

Sampling-based methods address this limitation by constructing a sparse representation of the configuration space. The two foundational algorithms in this category are Probabilistic Roadmaps (PRMs) [8] and Rapidly-Exploring Random Trees (RRTs) [9]. PRMs build a roadmap by randomly sampling configurations in the free space and connecting them with feasible paths, creating a graph that can be queried for multiple planning problems. RRTs incrementally build a tree from the start configuration, extending branches toward randomly sampled points until reaching the goal region.

Numerous extensions to these algorithms can be found in literature. RRT-Connect [35] accelerates convergence by growing trees from both the start and goal configurations. PRM* and RRT* [36] provide asymptotic optimality guarantees, ensuring that the solution approaches the optimal path as the number of samples increases. Informed RRT* [37]

further improves efficiency by focusing sampling within a shrinking ellipsoidal region that biases sampling towards the optimal solution.

Several approaches have been developed to address the computational burden of collision checking. Lazy PRM [38] initially assumes all edges are collision-free, then iteratively finds shortest paths and removes colliding edges until a valid path is found or additional sampling is required. This significantly reduces the number of collision checks during graph construction.

The quality of sampling strategies has also received considerable attention. LEGO [39] addresses the challenge of adequately representing difficult regions such as narrow passages by learning a distribution that identifies these critical areas based on feature vectors describing the environment. HPPRM [40] creates an artificial potential field discretized into a grid, using the mean potential in each cell to guide node placement in the PRM. Chen et al. [41] propose an improved sampling strategy where nodes repel each other except at obstacle boundaries or move toward boundaries if they are inside obstacles. Their method also adds additional samples near difficult connections when the local planner success rate is low.

Recently efforts have been made to leverage parallel computing on GPUs and CPUs. VAMP [42] uses single instruction multiple data structures to accelerate sampling-based motion planning, using body spheres and parallel edge evaluation for efficient collision checking. GTMP [11] introduces a random multipartite graph for planning in configuration-time space. Each layer of the graph represents one timestep, with all nodes of one layer connected to all nodes of the next layer. Value Iteration is then applied to find optimal paths, with JAX vectorization enabling efficient computation. Kino-PAX [43] parallelizes RRTs for GPUs by extending multiple nodes simultaneously and using space decomposition to ensure uniform node distribution.

While sampling-based methods offer significant advantages for high-dimensional planning problems, it is worth noting that the resulting paths often lack smoothness and may require post-processing. Additionally, many more extensions and variations exist beyond those discussed here [44]. The development of sampling-based methods continues to be an active research area, with ongoing efforts to improve computational efficiency, solution quality, and applicability to dynamic environments, which we address separately in Section 2.5.

2.3. Trajectory Optimization

Another family of approaches formulate motion planning as optimization over multiple control inputs or trajectory parameters, seeking to minimize a cost function depending on system dynamics and collision constraints. Early optimal-control methods include Differential Dynamic Programming (DDP) [45], based on the second order approximations and iterative Linear Quadratic Regulator (iLQR) [46], which alternates between linearizing the dynamics around a nominal trajectory and solving an LQR sub-problem to update the control law.

Trajectory optimizers can also be classified by their planning horizon. In receding-horizon Model Predictive Control (MPC), a finite time window of duration T is optimized at each step, and only the first control action is executed and the remainder of the trajectory is further optimized, after shifting the window forward. The endpoint of this window is evaluated by its proximity to the goal rather than being strictly constrained to reach it, Model Predictive Path Integral (MPPI) control [47] implements such an approach, using Monte Carlo integration with importance sampling. In full-horizon methods, the entire trajectory from start to goal is optimized in one shot, fixing the final state to the goal configuration, as in CHOMP [48], which optimizes trajectories using covariant gradient descent with respect to a cost that balances smoothness and collision avoidance. From this perspective myopic controllers can be viewed as reducing the horizon to a single timestep, optimizing only the immediate action at the current state.

Gradient-oriented methods characterize one branch of optimization frameworks, with sampling-based techniques forming the alternative. Sampling-based trajectory optimization explores the solution space by evaluating multiple trajectory candidates and selecting the best solution.

Gradient-based planners other than CHOMP include Gaussian Process Motion Planning (GPMP) [10], generating a smooth trajectory initialization by sampling from a Gaussian process and subsequently optimizing the sample to both avoid obstacles and stay close to the means of the Gaussian process to ensure smoothness. Its extensions GPMP2 [49] and GPMP-Graph [50] formulate the whole problem as probabilistic inference on factor graphs, and search for the maximum a-posteriori solution.

Examples for sampling-based optimization are the Cross-Entropy Method (CEM) [51] and its sample-efficient variant iCEM [52], which iteratively update a sampling distribution to fit the top-performing samples from previous iterations. Howell et al. [53] evaluate random perturbations to a trajectory in the MuJoCo physics simulator and replace it by

the current best sample. STORM [54] employs Halton sequences for low-discrepancy sampling of control inputs, then fits B-splines and parallelizes trajectory evaluation on GPUs. Its cost functions incorporate learned approximations for collision avoidance, joint limits, and singularity avoidance. VP-STO [55] performs MPC by iteratively updating the mean and covariance of trajectories represented by via-points that are interpolated with cubic splines. PRIEST [56] enhances CEM by projecting sampled trajectories onto a feasible set through an iterative optimization process using Lagrange multipliers for collision and joint constraints.

Klein et al. [57] optimize cubic splines to become geodesics by minimizing the kinetic energy in a Riemannian manifold. They investigate different Riemannian metrics for region avoidance, finding that only inverse barrier function metrics enforce hard constraints on geodesics to avoid undesirable regions.

Despite their successes, trajectory optimizers face challenges: gradient-based methods can become trapped in non-convex obstacle spaces or suboptimal homotopy classes, and both sampling and gradient approaches are highly sensitive to initialization.

2.4. Hierarchical Motion Planning

To mitigate limitations of individual approaches, hierarchical motion planning combines planning at different levels of abstraction, temporal scales, and computational complexities. Purely reactive methods can become trapped in local minima, while global planners may be computationally prohibitive for real-time applications. By decomposing the motion planning problem into layers, hierarchical approaches can balance computational efficiency with plan optimality while enabling appropriate abstraction for different tasks.

The elastic bands method [58] represents one of the earliest hierarchical approaches. This method models a trajectory as overlapping, collision-free bubbles. Bubbles move via artificial forces to avoid obstacles and stay connected, creating tension. New bubbles are added when gaps form, and redundant ones are removed. While effective in smoothing paths in a planar environment, it doesn't address high-dimensional challenges in $\mathrm{SE}(3)$ for articulated manipulators.

Structured control hierarchies offer an alternative approach. Sentis and Khatib [59] proposed projecting lower-priority tasks into the null space of higher-priority tasks, ensuring critical constraints are never violated by subordinate objectives. More flexible prioritization appears in recent work like RAMP [60], which employs a two-level MPC architecture with

a high-level MPPI planner generating global paths and a low-level vector field controller providing immediate reactivity. Similarly, Zhu et al. [61] developed a dual-MPC hierarchy with different planning horizons for global guidance and local obstacle avoidance.

Several approaches integrate sampling-based planning with trajectory optimization. IOS-MP [62] alternates between the two, using PRM* or BIT* to find a path, then refines it using augmented Lagrangians and adds the optimized path back into the planner's graph, possibly enhancing it with new samples. BITKOMO [63] uses a similar concept using KOMO [64] to refine, while optimization costs inform subsequent sampling. PINSAT [65] employs a hierarchical planning framework where a graph search uses "dummy actions" that overlook collision constraints, while B-spline optimization at the lower level determines actual edges and collision, which after evaluation adjust the successor states and priority queue.

Some recent hierarchical approaches leverage RMPs. Chen et al. [66] develop a pipeline that uses Informed RRT* as a global planner with Riemannian Motion Policies for local control, allowing the robot to follow global paths while reactively avoiding obstacles detected from point cloud data.

Hansel et al. [67] introduce HiPBI, which reformulates RMPs within a probabilistic framework to enable policy blending. HiPBI [67] uses the probabilistic perspective on RMPs as Gaussian distributions, shifting from metric-weighted averaging to a Boltzmann Product of Experts optimized via iCEM. For multi-agent systems, HiPBOT [68] frames RMP policy blending as an Optimal Transport problem, jointly optimizing motion policy weights across all agents while maintaining individual reactivity. The approach defines a cost function for transferring weights from prior expert distributions to agent-specific expert distributions. Both HiPBI and HiPBOT evaluate different policy weights through forward simulations over fixed time windows, effectively functioning as MPC.

Our method aligns with this hierarchical paradigm by establishing explicit subgoals through a PRM and following them with low-level attractor policies whose weights are dynamically adapted using Value Iteration and SVGD. This approach integrates global planning with reactive control while maintaining computational efficiency and overcoming local minima.

2.5. Motion Planning in Dynamic Environments

Dynamic environments present additional challenges for motion planning, as both the validity and optimality of paths can change over time. While some methods in the

previous sections can be accommodate for moving obstacles, they often struggle with rapidly changing environments. This section surveys specialized techniques for preserving feasible paths in dynamic environments without the need for full replanning.

Some strategies directly incorporate temporal dimensions into the planning process. In SIPP [69] "safe intervals" are introduced, time periods during which specific regions remain collision-free. These allow a modified A* algorithm to efficiently find optimal paths without checking safety at every time step. Building on this concept, Hüppi et al. [70] and Gentner et al. [71] identified collision-free time intervals for roadmap nodes when object trajectories are known. Both approaches implement time-sensitive A* variants accounting for estimated arrival times at nodes, with Gentner additionally introducing "exit nodes", a heuristic from which moving goals become reachable.

Several RRT extensions address dynamic environments specifically. RRT^X [72] builds on the rewiring of RRT# [73] while improving efficiency through the concept of consistency, measuring differences between stored graph costs and local minimum costs-to-come. When consistency is violated due to environmental changes, nodes are rewired to better-path parents, until a sufficient consistency is achieved. Liao et al.'s Bi-HS-RRT^X [74] enhances RRT^X with bidirectional search and a hybrid sampling strategy combining global bias with local planning. Qi et al.'s Mod-RRT [75] prunes nodes if they become colliding and the resulting orphans of the tree search for the lowest cost parents in their neighbors, and dynamically chooses alternatives when the robot cannot reach its next waypoint based on Pareto dominance of path length and steering angle. Yuan et al. [76] also focuses on reconnection strategies for goal-biased RRTs. They implement a hierarchy of reconnection methods when obstacles invalidate paths and estimate "safe points" until when the robot can still safely follow its original path before a re-routing has to be found.

Optimization-based methods have also been adapted for dynamic scenarios. Kolur et al.'s POSH [77] extends GPMP-Graph by optimizing the factor-graph, representing multiple lines from start to goal, via the Levenberg-Marquardt algorithm, then pruning unreachable states and re-optimizing after each step. Similarly, JIST [78] uses an RRT-like approach to grow the factor graph with a fixed node budget, pruning invalid portions after each step and selecting the lowest-cost leaf for execution.

We adopt concepts from the Elastic Roadmap approach by Yang and Brock [79]. Unlike traditional roadmaps with fixed configurations, Elastic Roadmaps feature nodes that actively respond to changes. These nodes are sampled near obstacle boundaries and continuously repositioned using a task-level controller to remain near obstacle features while avoiding collisions. Node connectivity is determined by straight-line visibility in task space, with a graph search algorithm continuously computing optimal node sequences.

Lehner et al. [80] extend this concept with a workspace connectivity graph of spheres, using Exploring/Exploiting Trees to find optimal paths within spheres that are then integrated into the Elastic Roadmap.

Building on the Stein Variational Probabilistic Roadmap (SVPRM) approach [12], which we discuss in more detail in Chapter 3, we replace the Elastic Roadmap's null-space control for node updates and edge traversal with Stein Variational Gradient Descent (SVGD) to adapt the nodes in dynamic environments, and represent edges with Riemannian Motion Policies. This substitution exploits RMP's multi-objective flexibility and theoretical control guarantees, while SVGD's probabilistic foundation concentrates nodes in regions of high posterior feasibility, yielding more sample-efficient, globally distributed coverage of the free configuration space than null-space control's local constraint–focused approach.

The approaches discussed demonstrate diverse strategies for addressing dynamic environment challenges, from adaptive roadmaps to time-aware planners and rapidly reconnecting trees to iterative optimization methods. Our proposed method, detailed in Chapter 4, combines SVPRM's adaptability with RMPs' reactive capabilities in a hierarchical framework that efficiently handles dynamic obstacles while avoiding local minima.

3. Foundations

This chapter introduces the fundamentals that form the theoretical foundation of the proposed method. The approach presented in this thesis combines reactive control with global planning to enable robots to navigate safely in dynamic environments.

Three key components underpin this method: Riemannian Motion Policies provide reactive control with safety guarantees, Stein Variational Probabilistic Roadmaps implement adaptive sampling-based planning, and Value Iteration to determine optimal paths through the roadmap.

3.1. Riemannian Motion Policies

Riemannian Motion Policies (RMPs) provide a framework for generating reactive robot motions while maintaining safety guarantees in dynamic environments. Originally proposed by Ratliff et al., RMPs [5] leverage differential geometry principles to develop controllers that operate across different coordinate systems.

In this section, we first examine the general problem of motion generation. We then investigate how quadratic costs motivate the development of the RMP framework and demonstrate how task maps connect different spaces. Following this, we discuss example policies for common robotics tasks and finally describe two algorithms, which efficiently compute RMPs through structured composition of task spaces.

3.1.1. Reactive Motion Generation

Motion generation can be understood as finding the optimal control function [81] $\mathbf{u} : \mathbb{R} \to \mathcal{U}$ for an agent in an environment described by a dynamical system $\mathbf{f} : \mathcal{X} \times \mathcal{U} \to T\mathcal{X}$, such that the resulting trajectory of states $\mathbf{x} : \mathbb{R} \to \mathcal{X}$ achieves a specified goal. One way this

can be found, is via the constrained minimization of a cost function. In this formulation, $\mathcal{U} \subseteq \mathbb{R}^d$ denotes the d-dimensional space of possible controls, and $\mathcal{X} \subseteq \mathbb{R}^n$ represents the space of all possible system states, with $T\mathcal{X}$ being its tangent space.

The general problem can be expressed as:

$$\begin{aligned} \min_{\mathbf{u}(\cdot),T} \quad & \int_0^T L(\mathbf{x}(t), \mathbf{u}(t))\, dt + \Phi(\mathbf{x}(T)) \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \\ & \mathbf{u}(t) \in \mathcal{U}, \quad \mathbf{x}(t) \in \mathcal{X}, \\ & \mathbf{x}(0) = \mathbf{x}_0, \ \mathbf{x}(T) \in \mathcal{X}_T, \\ & \mathbf{0} \geq \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)), \\ & \mathbf{0} = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)), \qquad \forall t \in [0, T], \end{aligned} \tag{3.1}$$

where L is the running cost function and Φ is the final state cost function. The boundary conditions constrain the system to start at the current state $\mathbf{x}_0$ and end in the space of desired goal states $\mathcal{X}_T$. State and action inequality constraints $\mathbf{g}$ may represent environmental conditions to avoid, such as collisions with obstacles, while equality constraints $\mathbf{h}$ might enforce specific relationships between state variables e.g. caused by the kinematic chain of a multi-body agent.

This optimal control problem does not have a general closed-form solution for arbitrary cost functions, constraints and dynamics. Moreover, our model of the system dynamics $\mathbf{f}(\mathbf{x}, \mathbf{u})$ may change during execution, for instance, when obstacles move unexpectedly. Therefore, we require computational efficiency to react quickly to environmental changes.

This leads us to high-frequency reactive motion generation, where the agent continuously responds to the current observed state. The objective is to define a motion policy that rapidly computes control commands $\mathbf{u}(t)$ based on the current state $\mathbf{x}_0$, while guaranteeing that the resulting trajectory $\mathbf{x}(t)$ maintains safe distances from obstacles and satisfies the remaining constraints.

3.1.2. Quadratic Costs and Metric-Weighted Optimization

A class of such reactive motion policies are RMPs [5]. They provide control-theoretical guarantees while maintaining computational efficiency, as they only consider the current state (positions and velocities). These policies provide control commands in the form of

accelerations $\mathcal{U} \subseteq T^2\mathcal{X}$, functioning as second-order dynamical systems that can describe a rich variety of behaviors.

With a slight relaxation of notation, we reuse $\mathbf{f}$ to describe the motion policy, rather than the whole system,

$$\mathbf{u}(t) = \ddot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t)), \tag{3.2}$$

and explicitly represent the position as $\mathbf{x}$, velocity as $\dot{\mathbf{x}}$ and acceleration as $\ddot{\mathbf{x}}$, instead of using a reduced form as above.

Not all objectives are intuitive to describe in the same space as the control inputs. Consider a robot arm needing to avoid a specific region occupied by an obstacle. Defining this region in $\mathrm{SE}(3)$ space (position and orientation), and deriving avoidance movements there, is more intuitive than working directly in the robot's joint space. We refer to this space as the task space, or operational space, denoted $\mathcal{X}$. The region where the robot arm would be in collision is the collision space $\mathcal{X}_{\text{coll}} \subseteq \mathcal{X}$, and the set of states that are not in collision is the free space $\mathcal{X}_{\text{free}} = \mathcal{X} / \mathcal{X}_{\text{coll}}$ [82].

The space of joint positions of a robot arm is referred to as configuration space $\mathcal{C}$ or joint space. Defining the free space in configuration space $\mathcal{C}_{\text{free}} \subseteq \mathcal{C}$ is typically non-trivial. This is, among other reasons, because the task space is often represented as a 6-dimensional vector $\mathcal{X} \subseteq \mathbb{R}^6$, using the logarithmic map to represent the $\mathrm{SE}(3)$ space, while the configuration space for many robots is a 7-dimensional vector $\mathcal{C} \subseteq \mathbb{R}^7$, as most robot arms have 7 degrees of freedom. Also the connection between those spaces is often highly non-linear, due to the complex kinematic chains.

Task maps ψ capture the relationship between different spaces:

$$\begin{aligned}
\mathbf{x} &= \psi(\mathbf{q}), \quad \mathbf{x} \in \mathcal{X}, \ \mathbf{q} \in \mathcal{C} \\
\dot{\mathbf{x}} &= \frac{d}{dt}\psi(\mathbf{q}) \quad = \mathbf{J}_\psi(\mathbf{q})\dot{\mathbf{q}}, \\
\ddot{\mathbf{x}} &= \frac{d^2}{dt^2}\psi(\mathbf{q}) = \mathbf{J}_\psi(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\psi(\mathbf{q})\dot{\mathbf{q}},
\end{aligned} \tag{3.3}$$

using the Jacobian matrix $\mathbf{J}_\psi$ and it's time derivative $\dot{\mathbf{J}}_\psi$. In the case of a robot arm, these equations include the forward kinematics. Describing the agent's behaviour as a dynamical system in the task space (or operational space) and using these Jacobians enables a more intuitive way to control, for example, a robot's end-effector [19].

Suppose we have multiple functions $\mathbf{f}_i(\mathbf{x}, \dot{\mathbf{x}})$ providing desired accelerations $\ddot{\mathbf{x}}_i^d$ in potentially different task spaces. Finding the acceleration in joint space that best satisfies these desired

accelerations can be formulated as the solution to the following least squares problem:

$$\mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) = \arg\min_{\ddot{\mathbf{q}}} \frac{1}{2} \sum_i \| \ddot{\mathbf{x}}_i^d - \underbrace{\mathbf{J}_{\psi_i}\ddot{\mathbf{q}} - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}}}_{\ddot{\mathbf{x}}_i} \|_{\mathbf{M}_i}^2, \tag{3.4}$$

$$\|\mathbf{x}\|_{\mathbf{M}}^2 = \mathbf{x}^\top \mathbf{M} \mathbf{x},$$

where the Riemannian Matrix $\mathbf{M}$ is a symmetric positive semidefinite matrix. We drop the notation showing $\mathbf{J}_\psi$'s dependency on $\mathbf{q}$. To find the optimal solution, we look for the roots of the gradient with respect to $\ddot{\mathbf{q}}$:

$$\nabla_{\ddot{\mathbf{q}}} \left[\frac{1}{2} \sum_i \| \ddot{\mathbf{x}}_i^d - \mathbf{J}_{\psi_i}\ddot{\mathbf{q}} - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}} \|_{\mathbf{M}_i}^2 \right] \stackrel{!}{=} \mathbf{0}$$

$$\frac{1}{2} \sum_i \mathbf{J}_{\psi_i}^\top \, 2\mathbf{M}_i (\ddot{\mathbf{x}}_i^d - \mathbf{J}_{\psi_i}\ddot{\mathbf{q}} - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}}) = \mathbf{0}$$

$$\sum_i \mathbf{J}_{\psi_i}^\top \mathbf{M}_i \mathbf{J}_{\psi_i} \ddot{\mathbf{q}} = \sum_i \mathbf{J}_{\psi_i}^\top \mathbf{M}_i (\ddot{\mathbf{x}}_i^d - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}})$$

$$\ddot{\mathbf{q}} = (\sum_i \mathbf{J}_{\psi_i}^\top \mathbf{M}_i \mathbf{J}_{\psi_i})^\dagger (\sum_i \mathbf{J}_{\psi_i}^\top \mathbf{M}_i (\ddot{\mathbf{x}}_i^d - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}})), \tag{3.5}$$

where † denotes the Moore–Penrose pseudo-inverse.

The Riemannian metric allows us to weight different desired accelerations $\ddot{\mathbf{x}}_i^d$ according to their importance. It's worth noting that if only a single desired acceleration is considered, the metric matrix would be canceled out by its inverse. Additionally, if joint and task space are identical (i.e., $\mathbf{J}_\psi = \mathbf{I}$), then the result simplifies to a metric-weighted average. If we then consider $\mathbf{J}_\psi^\top \mathbf{M} \mathbf{J}_\psi$ to be the pull-back of a metric $\mathbf{M}$ to the domain of ψ, and $\mathbf{J}_\psi^\top \mathbf{M} (\ddot{\mathbf{x}}_i^d - \dot{\mathbf{J}}_{\psi_i}\dot{\mathbf{q}})$ the transformation of a vector field $\ddot{\mathbf{x}}$ to the same domain, we can view the solution again as a metric-weighted average.

An RMP is defined as a tuple in a space $(\mathbf{f}, \mathbf{M})_{\mathcal{X}}$. The pull-back of this RMP through a task map is defined as:

$$\text{pull}_\psi(\mathbf{f}, \mathbf{M})_{\mathcal{X}} = \left(\left(\mathbf{J}^T \mathbf{M} \mathbf{J}\right)^\dagger \mathbf{J}^T \mathbf{M} \mathbf{f}, \ \mathbf{J}^T \mathbf{M} \mathbf{J} \right)_{\mathcal{C}} ,$$

and similarly, a push-forward operation can be defined as:

$$\text{push}^\psi(\mathbf{f}, \mathbf{M})_{\mathcal{C}} = \left(\mathbf{J}\mathbf{f} + \dot{\mathbf{J}}\dot{\mathbf{q}}, \ \mathbf{J}^\dagger \mathbf{M} (\mathbf{J}^\dagger)^T \right)_{\mathcal{X}} ,$$

mapping a configuration space policy to a task space policy.

After pushing from joint space to task space, evaluating the policy there, and then pulling the result back into joint space, addition of RMPs is allowed like:

$$\mathcal{R}_c = \sum_i \mathcal{R}_i = {}^{\Omega}\left(\left(\sum_i \mathbf{M}_i\right)^{+} \sum_i \mathbf{M}_i \mathbf{f}_i,\ \sum_i \mathbf{M}_i\right),$$

resulting in the same metric-weighted average as in 3.5, finalizing the definition of an algebra on RMPs.

3.1.3. Policy Formulations for Goal Attraction and Obstacle Avoidance

To generate safe and stable motions with the above operations, Cheng et al. [23] analyze a class of systems called Geometric Dynamical Systems (GDS), which can be shown to stay coordinate-free and Lyapunov stable under the RMP-Algebra. In this framework, the desired acceleration $\ddot{\mathbf{x}}_d$ should follow the gradient of an artificial potential field $\nabla_{\mathbf{x}}\Phi$, and the Riemannian metric $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$ is set to depend on the position and velocity.

With velocity dependency in the metric, we can modify behaviors contextually. For example, a collision avoidance policy can be deactivated if the obstacle is moving away from the agent. This smooth transition of importance allows for more nuanced behavior than a strict hierarchy of importance, as seen in methods like Null Space Projection [59].

We use non-linear PD-like controllers of the following form:

$$\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = -\Big(\nabla_{\mathbf{x}}\Phi(\mathbf{x}) + \mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \boldsymbol{\xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}})\Big), \tag{3.6}$$

where $\Phi(\mathbf{x})$ is the potential to be minimized, $\mathbf{B}(\mathbf{x}, \dot{\mathbf{x}})$ represents dampening terms, and $\boldsymbol{\xi}_{\mathbf{M}}(\mathbf{x}, \dot{\mathbf{x}})$ are terms induced by the curvature of the space.

As an example of such a system realizing a goal attraction policy from [25], we can define the space to be the vector from the end-effector position $\mathbf{x}_{\text{EE}}$ to a goal position $\mathbf{x}_{\text{goal}}$, denoting a state in this space generally as $\mathbf{x}$:

$$\mathbf{x} = \psi_{\text{attraction}}(\mathbf{q}) = \mathbf{x}_{goal} - \psi_{\text{EE}}(\mathbf{q}),$$

where ψ_{EE} maps the joint positions to the position of the end-effector in task space.

The forcing term in the form of Equation 3.6 is:

$$
\begin{aligned}
\gamma(\mathbf{x}) &= \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right), \\
w(\mathbf{x}) &= \gamma(\mathbf{x})\ w_u + (1-\gamma(\mathbf{x}))\ w_l, \quad 0 \le w_l \le w_u < \infty, \\
\nabla_{\mathbf{x}}\Phi(\mathbf{x}) &= \tanh\left(\alpha\ \|\mathbf{x}\|\right)\ w(\mathbf{x})\ \frac{\mathbf{x}}{\|\mathbf{x}\|}, \\
\mathbf{B}(\mathbf{x},\dot{\mathbf{x}}) &= \eta\ w(\mathbf{x})\ \mathbf{I}, \\
\boldsymbol{\xi}_{\mathbf{M}} &= \frac{dw}{dt}\dot{\mathbf{x}} - \frac{1}{2}\nabla_{\mathbf{x}} w(\mathbf{x})\|\dot{\mathbf{x}}\|^2, \\
&= -\frac{1}{2}\|\dot{\mathbf{x}}\|^2\left(\mathbf{I} - 2\ \hat{\dot{\mathbf{x}}}\hat{\dot{\mathbf{x}}}^T\right)\nabla_{\mathbf{x}} w(\mathbf{x}), \\
\hat{\dot{\mathbf{x}}} &= \frac{\dot{\mathbf{x}}}{\|\dot{\mathbf{x}}\|},
\end{aligned}
$$

with σ setting the spatial scale of $\gamma(\mathbf{x})$, affecting how fast the influence diminishes with distance. Weights w_u and w_l define metric scaling limits. Parameter α adjusts potential gradient steepness near the goal, where control should weaken. η is a damping coefficient for the velocity-dependent term $\mathbf{B}$. Terms $\boldsymbol{\xi}$ maintain stability in curved spaces. We use a scaled identity matrix as metric $\mathbf{M}(\mathbf{x},\dot{\mathbf{x}}) = w(\mathbf{x})\ \mathbf{I}$. See the appendix of [23] for a thorough stability analysis.

As another example, we will use the obstacle avoidance policy from [83], where the space is the distance between an agent's link and an obstacle, denoted as the scalar x:

$$
\begin{aligned}
\nabla_x \Phi(x) &= k_p \exp(-x/\ell_p), \\
\xi(x,\dot{x}) &= -k_d\left[1 - \frac{1}{1+\exp(-\dot{x}/v_d)}\right]\frac{\dot{x}}{x/\ell_d + \epsilon_d}, \\
m(x,\dot{x}) &= \left[1 - \frac{1}{1+\exp(-\dot{x}/v_d)}\right] g(x)\frac{\mu}{x/\ell_m + \epsilon_m}, \\
g(x) &= \begin{cases} x^2/r^2 - 2x/r + 1, & x \le r, \\ 0, & x > r, \end{cases}
\end{aligned}
$$

In this formulation, the gradient repels the agent from the obstacle, with the curvature term acting as a barrier. This differs from the collision controller derived in [23]. An analysis of this policy's geometric dynamical properties and Lyapunov stability is beyond the scope of this work.

3.1.4. Composition of Task Spaces

An efficient, recursive calculation of the pull-back operation can be achieved when task maps can be arranged in a tree structure. The RMP-*flow* algorithm [23] defines the nodes of the RMP-Tree, to contain the functions $\mathbf{f}_i(\mathbf{x}, \dot{\mathbf{x}}_i)$ and $\mathbf{M}(\mathbf{x}, \dot{\mathbf{x}})$, along with the states $\mathbf{x}$ and $\dot{\mathbf{x}}$, while each edge represents a task map ψ.

In alignment with the algebra mentioned in Section 3.1.2 we execute a push-forward throughout the tree, propagating states from parent to child nodes:

$$(\mathbf{x}_c, \dot{\mathbf{x}}_c) = (\psi(\mathbf{x}_p), \mathbf{J}(\mathbf{x}_p)\dot{\mathbf{x}}_p),$$

followed by a pull-back operation:

$$\mathbf{f} = \sum_{i=1}^{K} \mathbf{J}_i^\top (\mathbf{f}_i - \mathbf{M}_i \dot{\mathbf{J}}_i \dot{\mathbf{x}}), \quad \mathbf{M} = \sum_{i=1}^{K} \mathbf{J}_i^\top \mathbf{M}_i \mathbf{J}_i \ ,$$

aggregating the forces and metrics from child to parent nodes. Finally a resolve operation is added:

$$\ddot{\mathbf{x}} = \mathbf{M}^\dagger \mathbf{f}, \tag{3.7}$$

computing the acceleration used as control command.

Li et al. introduce RMP2 [24] where the task map relationships are constructed automatically using automatic differentiation libraries, which relaxes the structure from a tree to a directed acyclic graph. This approach simplifies implementation and improves space complexity when the Jacobian vector product is correctly utilized. However, if the Jacobian matrix is calculated explicitly, the time complexity can be worse than that of the original RMP-*flow* algorithm.

3.2. Stein Variational Probabilistic Roadmaps

In this section, we examine Stein Variational Probabilistic Roadmaps (SVPRMs) [12], which is a global sampling based approach. This method effectively covers the free space and can incorporate prior information, emphasizing critical regions. We begin with a brief overview of Probabilistic Roadmaps (PRMs) in general, then explore how SVPRMs utilize Stein Variational Gradient Descent to systematically optimize the distribution of roadmap nodes. Finally, we analyze how different node connection strategies impact the roadmap's effectiveness in dynamic environments with moving obstacles.

3.2.1. Sample-Based Planning with Probabilistic Roadmaps

Probabilistic Roadmaps (PRMs) are sampling-based planners that can efficiently answer multiple queries about how to navigate from one point to another. A key advantage of this approach is that it retains most of its information between queries, making subsequent path computations relatively fast.

The basic structure of the PRM algorithm involves

1. sampling configurations,
2. deciding which samples should be included in the roadmap, and
3. determining how to connect them.

The resulting graph can then be used in various ways to generate motion plans, e.g. with graph-search algorithms as discussed in Chapter 2.

This approach is computationally more efficient than using a grid-like structure to discretize the space, which would suffer severely from the curse of dimensionality. Higher-dimensional spaces remain to be more challenging than lower-dimensional ones for probabilistic roadmaps, but the scaling properties are significantly better.

An important consideration is how to sample the configurations effectively. Algorithms like PRM* [36] can guarantee asymptotic completeness, ensuring that a solution will be found if one exists and if enough samples are taken. A good roadmap should represent the connectivity of the search space adequately in all important regions, particularly in narrow passages and around the goal.

3.2.2. Particle-Based Distribution Approximation with Stein Variational Gradient Descent

In our context, the search space is the free configuration space $\mathcal{C}_{\text{free}}$. For generality, we formulate this section for a general case $\mathcal{X}_{\text{free}}$, which does not necessarily have to be part of the $\text{SE}(3)$ space where we defined our obstacle positions. Lambert et al. formulate the problem probabilistically [12], treating the state as a random variable X. To ensure sampled points are collision-free, they introduce a binary random variable Z. The parametrized conditional distribution $p(Z = 1|X = \mathbf{x}; \Theta)$ represents the probability that a configuration is collision-free. And we assume we have a method to evaluate this efficiently for specific samples.

Using Bayes' rule, this can be expressed as:

$$p(\mathbf{x}|z;\Theta) = \frac{p(z|\mathbf{x};\Theta)p(\mathbf{x})}{\int_{\mathcal{X}} p(z|\mathbf{x})} \ ,$$

providing us with a distribution of states. For brevity, we omit explicit mention of random variables in the notation. Sampling directly from this distribution is generally intractable due to the need to integrate over the entire space to compute the marginal distribution.

Instead, the SVPRM approximates $p(\mathbf{x}|z;\Theta)$ with a set of N particles representing a distribution $\tilde{p}(\mathbf{x})$:

$$\{\mathbf{x}_i\}_{i=1}^{N} \sim \tilde{p}(\mathbf{x}), \quad \mathbf{x}_i \in \mathbb{R}^d.$$

These particles are iteratively updated via:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon \boldsymbol{\phi}^*(\mathbf{x}_i),$$

to minimize the Kullback-Leibler divergence (KL) $D_{KL}(\tilde{p}(\mathbf{x}) \parallel p(\mathbf{x}|z))$, using Stein Variational Gradient Descent (SVGD):

$$\boldsymbol{\phi}^*(\mathbf{x}_i) = \frac{1}{N}\sum_{j=1}^{N}\Big[k(\mathbf{x}_j,\mathbf{x}_i)\nabla_{\mathbf{x}_j}\log p(\mathbf{x}_j|z) + \nabla_{\mathbf{x}_j}k(\mathbf{x}_j,\mathbf{x}_i)\Big], \tag{3.8}$$

$$\nabla_{\mathbf{x}}\log p(\mathbf{x}_j|z;\Theta) = \nabla_{\mathbf{x}}\log p(z|\mathbf{x}_j;\Theta) + \nabla_{\mathbf{x}}\log p(\mathbf{x}_j),$$

$$k(\mathbf{x}_j,\mathbf{x}_i) = \exp\left(-\frac{1}{2h}(\mathbf{x}_j-\mathbf{x}_i)^\top \mathbf{M}(\mathbf{x}_j-\mathbf{x}_i)\right),$$

using the score of the collision probability scaled with an Radial Basis Function (RBF) kernel k.

The kernel incorporates a Hessian:

$$\mathbf{M} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{H}(\mathbf{x}_j),$$

$$\mathbf{H}(\mathbf{x}) = -\nabla_{\mathbf{x}}^2 \log p(\mathbf{x}|x),$$

computed using the $\mathbf{x}$ values from previous iterations to repel nodes from obstacles. The gradient of the RBF kernel ensures the nodes are spread out in the search space.

To approximate the collision probability SVPRM uses a cost function $\mathbf{h}(\mathbf{x})$, based on the distance of the agent to the nearest obstacle, similar to approaches in [10, 48]:

$$
\begin{aligned}
p(z = 1|\mathbf{x}; \Theta) &\propto \exp(-\alpha \left\|\mathbf{h}(\mathbf{x})\right\|^2), \\
\mathbf{h}(\mathbf{x}) &= \left[c(\mathbf{x}, s_j)\right] \Big|_{j=1:K}, \\
c(\mathbf{x}, s_j) &= \begin{cases} -d(\mathbf{x}, s_j) + \epsilon & d(\mathbf{x}, s_j) \leq \epsilon, \\ 0 & \text{otherwise}, \end{cases}
\end{aligned}
\tag{3.9}
$$

representing the robot using body-spheres s_j, a set of spheres encapsulating the robot's geometry, to simplify the computation [84].

Priors, such as mixtures of Gaussians, can be used to move the nodes towards the agent's current position and the goal position. This increases the density of nodes to improve the range of possible directions in these regions of interest.

3.2.3. Graph Connectivity for Dynamic Scenarios

Determining how to connect nodes in the roadmap is another critical aspect of the algorithm. Some approaches use steering functions [85], some just assume straight lines [8]. While having more edges increases the computational intensity of subsequent motion planning steps due to the increased need of collision checking, it is important to have cycles in the graph in dynamic environments. This redundancy allows for alternative paths if certain edges become blocked during execution [86, 87]. Since we compute pairwise distances for the kernel in SVGD and in Value Iteration as well, a distance-based condition to determine connectivity does not introduce a lot of additional computation steps. Moreover, this approach helps strike a balance between edge density and graph redundancy. We will use straight lines, as this approach can be easily vectorized, whereas methods like following a steering function might not be as well-suited for GPU implementation [42].

3.3. Value Iteration

As discussed previously, we need a method to find the optimal path along the graph resulting from the PRM. Since the graph is only an abstraction, we might need to recompute this path frequently, e.g. if the agent moves to a different part of the graph, after evading an obstacle.

Therefore, it would be beneficial if most of the planning information could be reused when it becomes necessary to recompute the optimal sequence of nodes. Value Iteration, a component of dynamic programming [13, 88], produces a policy that indicates the direction to the goal from any node in the graph simultaneously. We first outline the Bellman equations defining the value function and describe how optimal actions are derived from it.

3.3.1. Bellman Equations

To find the best path to the goal, we must frame the problem as a finite Markov Decision Process (MDP) [89]. An Markov Decision Process (MDP) consists of a finite set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, a set of rewards $\mathcal{R}$, a discount factor $\gamma \in [0, 1]$, and a dynamics function $p_{\text{MDP}} : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, which represents the joint probability of transitioning to a particular state and receiving a specific reward based on the previous state and the last action taken.

In our context, the states are the nodes of the PRM, so $\mathcal{S} \subset \mathcal{X}$. The possible actions at each node are described by attraction policies $\mathcal{A} = \mathcal{F} = \{\mathbf{f}_i \mid \mathbf{f}_i : \mathcal{X}_i \times T\mathcal{X}_i \to \mathcal{U}_i\}$. If we select one motion primitive for each edge and assume they deterministically guide us toward the intended node, the method approaches an all pairs shortest path algorithm. However, the Bellman formulation allows for more general definitions.

Value Iteration provides us with the value function $v_{\pi^*} : \mathcal{S} \to \mathbb{R}$ and the action-value function $q_{\pi^*} : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ for the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$. The value function represents the expected return G, which is the sum of discounted rewards R received when starting in state $\mathbf{x}$ and following policy π. The action-value function is similar, except that the first action is substituted by the action $\mathbf{f}_i$.

$$v_\pi(\mathbf{x}) \doteq \mathbb{E}_\pi\left[G_t \mid X_t = \mathbf{x}\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, X_t = \mathbf{x}\right], \qquad \mathbf{x} \in \mathcal{S},$$

$$q_\pi(\mathbf{f}_i, \mathbf{x}) \doteq \mathbb{E}_\pi\left[G_t \mid A_t = \mathbf{f}_i, X_t = \mathbf{x}\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, A_t = \mathbf{f}_i, X_t = \mathbf{x}\right], \mathbf{f}_i \in \mathcal{A},$$

where X is the random variable representing the state, and A is the random variable representing the action. Optimal policies maximize this return, so all optimal policies share the same optimal value and action-value functions.

These functions can be formulated recursively through the Bellman Optimality Equations:

$$\begin{aligned} v_*(\mathbf{x}) &= \max_{\mathbf{f_i} \in \mathcal{F}} q_{\pi_*}(\mathbf{f}_i, \mathbf{x}), \\ &= \max_{\mathbf{f}_i} \mathbb{E}_{\pi_*} \left[G_t \mid A_t = \mathbf{f}_i, X_t = \mathbf{x} \right], \\ &= \max_{\mathbf{f}_i} \mathbb{E}_{\pi_*} \left[R_{t+1} + \gamma G_{t+1} \mid A_t = \mathbf{f}_i, X_t = \mathbf{x} \right], \\ &= \max_{\mathbf{f}_i} \mathbb{E} \left[R_{t+1} + \gamma \mathrm{v}_*(S_{t+1}) \mid A_t = \mathbf{f}_i, X_t = \mathbf{x} \right], \end{aligned} \tag{3.10}$$

$$q_*(\mathbf{f}_i, \mathbf{x}) = \mathbb{E}\Big[R_{t+1} + \gamma \max_{\mathbf{f}_j \in \mathcal{F}} q_*(\mathbf{f}_j, X_{t+1}) \;\Big|\; A_t = \mathbf{f}_i, X_t = \mathbf{x} \Big]. \tag{3.11}$$

Using this recursive property, Value Iteration iteratively updates an approximation of the optimal value function:

$$v_{k+1}(\mathbf{x}) \;\doteq\; \max_{\mathbf{f}_i} \mathbb{E}[R_{t+1} + \gamma\, v_k(X_{t+1}) \mid A_t = \mathbf{f}_i, X_t = \mathbf{x}], \tag{3.12}$$

which can be shown to converge for arbitrary initializations.

Finally, the Advantage Function [90] is defined as:

$$a_* \left(\mathbf{f}_i, \mathbf{x} \right) = q_* \left(\mathbf{f}_i, \mathbf{x} \right) - \max_{\mathbf{f}_j} q_* \left(\mathbf{f}_j, \mathbf{x} \right) = q_*(\mathbf{f}_i, \mathbf{x}) - v_*(\mathbf{x}), \tag{3.13}$$

providing a measure on how much moving to a certain state improves the value.

3.3.2. Policy Extraction and Exploration

Once Value Iteration converges to the optimal value function $v_*(\mathbf{x})$, the optimal policy $\pi_*(\mathbf{x})$ can be extracted by selecting the action that maximizes the expected value:

$$\pi_*(\mathbf{x}) = \arg\max_{\mathbf{f}_i \in \mathcal{F}} \mathbb{E}[R_{t+1} + \gamma v_*(X_{t+1}) | A_t = \mathbf{f}_i, X_t = \mathbf{x}].$$

However, sometimes it is beneficial to incorporate exploration strategies, e.g. if there are areas where the approximation of the optimal action is inaccurate due to the dynamic nature of the environment. Two common approaches for balancing exploration and exploitation are the ϵ-greedy strategy and the Boltzmann or Gibbs distribution.

The ϵ-greedy approach introduces stochasticity into the policy by selecting a random action with probability ϵ instead of following the greedy policy. The Boltzmann distribution

assigns selection probabilities proportional to the exponential of the preference of an action, controlled by a temperature parameter τ using the softmax function:

$$\text{softmax}(\mathbf{x}, \tau) = \frac{1}{\sum_i \exp(\frac{x_i}{\tau})} \exp\left(\frac{\mathbf{x}}{\tau}\right) , \tag{3.14}$$

where x_i are the elements of the vector $\mathbf{x}$ and exp is the exponential function, applied element wise.

4. Integration of Global Planning and Reactive Motion Generation

This chapter details the methodology of our approach, focusing on overcoming local minima of reactive motion generation in dynamic settings. Our main contribution lies in the integration of fast local reactive control with computationally more intensive global planning using a Probabilistic Roadmap (PRM) and Riemannian Motion Policies (RMPs), while Stein Variational Gradient Descent (SVGD) bridges between these different operational frequencies.

The architecture includes the following processes, also shown in Figure 4.1. We overcome the local minima that RMPs can get stuck in by following subgoals that are part of a PRM. SVGD continuously optimizes subgoal positions to adapt to the changing environment. The decision, which subgoals to follow, is based on a policy resulting from Value Iteration on the PRM graph. This approach enables efficient navigation while maintaining safety guarantees and computational feasibility.

In the following sections we will discuss the details on the approximation of collisions and the choice between configuration space and task space subgoals. We then elaborate how the optimal policy is followed by reweighing RMPs with the advantage function and finish by hypothesizing on the effects of using SVGD to attain an approximation of the value in-between collision checks.

4.1. Collision Probability Estimation for Motion Planning

To enable efficient computation of collision probabilities and safe motion generation, we implement a distance measure that balances accuracy with computational efficiency. This measure informs both the SVGD for updating roadmap nodes and the reactive control of the RMPs.

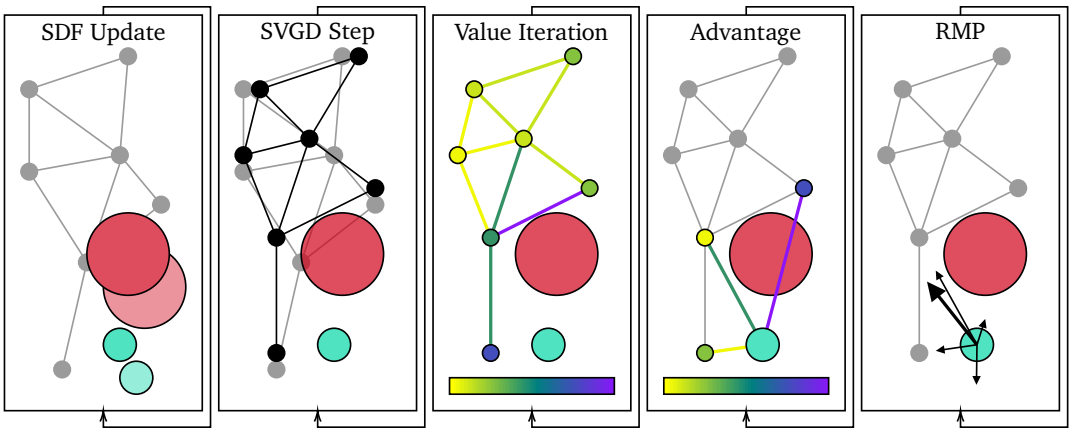


Figure 4.1.: Decoupled modules. The continuously updated Signed Distance Fields (SDFs) informs Stein Variational Gradient Descents (SVGDs) for node repositioning, away from the obstacle (red). Value Iteration evaluates edge traversability via SDFs as well, producing advantage functions that connect the agent (turquoise) to the roadmap and weight attraction policies in the RMP Framework.

We employ several conservative simplifications to reduce implementation complexity and computational burden. The agent is represented by a set of body spheres, as described in Section 3.2.2. Similarly, obstacles are approximated using sphere or box primitives. These approximations significantly simplify the calculation of distances between the agent and obstacles, making the implementation straightforward and computationally efficient.

The metric consists of multiple parts. To prevent self-collisions, we compute pairwise distances between the links of the agent. For efficiency, we exclude links that cannot collide with each other due to the kinematics of the agent, such as adjacent links. While more complex collision detection methods exist, including learned approximators [3, 91], they are not the focus of this work. We also account for workspace boundaries by evaluating the distances of individual link centers to the boundaries after computing the kinematics. In addition, we consider the distance of the joint configurations to joint limits. These various distance measures are scaled to ensure that distances measured in different spaces (e.g., joint limits in radians versus obstacle distances in meters) can be meaningfully compared.

The minimum of these scaled distances is used for both collision avoidance and estimating collision probability. Since the distances toward joint limits that we consider harmful are relatively small, a linear scaling approach is sufficient for our purposes. Our method

therefore differs from approaches such as [12, 10, 48], where a sum of squares is employed. Here, the gradient points away from just the closest obstacle, rather than multiple ones. Since both SVGD and RMP can be evaluated at relatively high frequencies, allowing for small step sizes, considering only the closest constraint is sufficient for effective collision avoidance, while being independent on the number of body spheres.

4.2. Probabilistic Roadmap Nodes as Attractors

In our approach, the nodes of the PRM represent the goals of attraction policies defined as Riemannian Motion Policies. As discussed previously, RMPs can be formulated in different spaces, which provides flexibility but also requires careful design choices. A key consideration, then, is how to select the space for the class of policies, which we will use to guide the agent along the subgoals towards the end-goal.

One potential approach could be to construct the PRM in task space, which would simplify certain aspects of the problem since task space typically has fewer dimensions and is more intuitive in most scenarios. However, determining configuration space connectivity from task space connectivity presents significant challenges. For example, consider a SCARA robot with two Degrees of Freedom (DoFs) that can reach the same point in both an "elbow down" and an "elbow up" configuration. In this case, the connectivity of two workspace points might depend on the specific elbow position, making it difficult to accurately represent the connectivity in task space alone. This problem becomes intractable as the agent's degrees of freedom exceed the dimensions of the workspace.

Therefore, we have chosen to construct our PRM in joint space. Additionally, we incorporate another RMP defined in task space to track the goal once the agent is sufficiently close. For the current implementation, this task-space policy is not part of the roadmap. In Section 7.2 we discuss how we can achieve a tighter integration in future work.

4.3. Dynamic Metric Weighting Through Advantage Functions

As mentioned in Section 3.3 the states of our MDP are the nodes of the PRM $\mathcal{S} \subset \mathcal{X}$ and the actions are the edges of the PRM, representing the choice to follow a specific attraction policy $\mathcal{A} = \mathcal{F}$. To effectively guide the agent's motion through the environment,

we establish a reward function $r : \mathcal{C} \times \mathcal{C} \to \mathbb{R}$ designed to find the shortest path in the free space $\mathcal{C}_{\text{free}}$:

$$\mathbb{E}[R_{t+1}|\mathbf{q}, \mathbf{f}_i] = r(\mathbf{q}, \mathbf{q}_T) = \frac{g(\mathbf{q}, \mathbf{q}_T) - \|\mathbf{q} - \mathbf{q}_T\|_2}{p(z|\mathbf{q}, \mathbf{q}_T)} \,, \tag{4.1}$$

it is important to note that our reward function is defined solely on states. We assume that executing an action means following the policy $\mathbf{f}_i$ from the current state $\mathbf{q}$ to the attraction center of the policy, denoted as $\mathbf{q}_T$. This assumption results in deterministic dynamics and enables us to uniquely identify actions with their goal state.

We use the configuration-space Euclidean distance $\|\mathbf{q} - \mathbf{q}_T\|_2$, applying collision penalties inversely proportional to $p(z|\mathbf{q}, \mathbf{q}_T)$. This probability is estimated by sampling n_d discrete points along the path and assuming independence:

$$\begin{aligned} p(z|\mathbf{q}_i, \mathbf{q}_j) &= \prod_{k=0}^{n_d} p\left(z \;\middle|\; \mathbf{q}_i + \frac{k}{n_d}(\mathbf{q}_j - \mathbf{q}_i)\right) \\ &= \exp\left(-\alpha \sum_{k=0}^{n_d} \mathbf{h}\left(\mathbf{q}_i + \frac{k}{n_d}(\mathbf{q}_j - \mathbf{q}_i)\right)\right). \end{aligned}$$

This discretization is a pragmatic approximation that yields effective results despite theoretical limitations [42], and possible deviations from the actual trajectory, resulting from $\mathbf{f}_i$. Here, $\mathbf{h}$ represents the cost function defined earlier as Equation 3.9.

Following the terminology of [71], we define exit nodes $\mathcal{S}_{\text{exit}}$ near goals. When solving inverse kinematics is inexpensive, exit nodes correspond to exact goal positions. Otherwise, they're selected based on potential proximity to the closest goal if following a target tracking RMP. For distances below a threshold Δ_g, we assign the reward:

$$g(\mathbf{q}, \mathbf{q}_T) = \begin{cases} \left(1 - \frac{\|\psi_{\text{ee}}(\mathbf{q}) - \mathbf{x}_{\text{goal}}\|_2}{\Delta_g}\right)^2 r_g, & \text{if } \mathbf{q} = \mathbf{q}_T \wedge \mathbf{q} \in S_{\text{exit}} \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

where $\mathbf{x}_{\text{goal}}$ is the position of the closest goal in task space and $r_g > 0$ is the reward given, if matching the goal position exactly. ψ_{EE} maps the configuration $\mathbf{q}$ to the task space. Note the reward only applies to edges connecting nodes to themselves.

After establishing the reward function, we perform Value Iteration to obtain the value function $v(\mathbf{q})$. It is important to note that the agent's current position is not explicitly

included as a node in the PRM. Instead, we compute the action-value function $q : \mathcal{F} \times \mathcal{C} \to \mathbb{R}$ by evaluating the paths from the agent's current position $\mathbf{q}_{\text{agent}}$ to the nearest nodes:

$$q(\mathbf{f}_i, \mathbf{q}_{\text{agent}}) = r(\mathbf{q}_{\text{agent}}, \mathbf{q}_T) + v(\mathbf{q}_T).$$

Since we modeled the action to deterministically end in state $\mathbf{q}_T$, the expectation can be reduced to this simple sum of one local edge reward and the value of the corresponding PRM node.

We then want to use the softmax (see Equation 3.14) of the action-value function to weight the attraction policies that guide the agent towards the nodes of the PRM. To enhance numerical stability, we substitute the action-value function with the Advantage Function as described in [92]:

$$\begin{aligned}\text{softmax}\left(q\left(\mathbf{f}_i, \mathbf{q}_{\text{agent}}\right)\right) &= \text{softmax}\left(q(\mathbf{f}_i, \mathbf{q}_{\text{agent}}) - \max_{\mathbf{f}_i} q(\mathbf{f}_i, \mathbf{q}_{\text{agent}})\right) \\ &= \text{softmax}(a(\mathbf{f}_i, \mathbf{q}_{\text{agent}})).\end{aligned}$$

Consequently, the metric function of the roadmap following policy becomes:

$$\mathbf{M}(\mathbf{q}_{\text{agent}}) = w(\mathbf{q}_{\text{agent}})\ \text{softmax}(a(\mathbf{f}_i, \mathbf{q}_{\text{agent}}))\ \mathbf{I}, \tag{4.3}$$

where we omit the temperature parameter of the softmax function for simplicity of notation, $w(\mathbf{q})$ represents the distance-based weighting factor defined in Section 3.1.3, and $\mathbf{I}$ is the identity matrix. The forcing term of the attraction policy remains identical.

4.4. Alternative Approach to Action-Value Function Integration

It is important to distinguish our approach from the integration of RMPs and action-value functions discussed in the appendix of [5]. They demonstrate how extracting the optimal policy π^* of a second-order Taylor approximation of an action-value function $\widetilde{q}$, results in a least squares problem as in Equation 3.4:

$$\pi^*(\mathbf{x}) = \arg\min_{\mathbf{a} \in \mathcal{A}} \widetilde{q}(\mathbf{a}|\mathbf{s}) = \arg\min_{\ddot{\mathbf{x}}} \frac{1}{2} \|\pi^*(\mathbf{x}, \dot{\mathbf{x}}) - \ddot{\mathbf{x}}\|^2_{\nabla^2_{\mathbf{a}} \widetilde{q}}\,. \tag{4.4}$$

Here, the action space $\mathcal{A}$ consists of control inputs $\mathcal{U}$, which are accelerations in the RMP framework $\mathcal{A} \subseteq \mathcal{U} \subseteq T^2\mathcal{X}$. The state of the MDP consists of both the agents position and

velocity $\mathcal{S} = \mathcal{X} \times T\mathcal{X}$, $\pi^* : \mathcal{S} \to \mathcal{A}$ is the optimal policy, and $\nabla^2_{\mathbf{a}} \widetilde{q}$ is the Hessian of the action-value function with respect to actions.

This formulation allows them to define the RMP $\left(\pi^*(\mathbf{x}, \dot{\mathbf{x}}), \nabla^2_{\mathbf{a}} \widetilde{q}\right)_{\mathcal{X}}$. Our approach differs in that we discretize the state-action space by decomposing it into multiple weighted point attraction policies $\mathcal{S} = \mathcal{X}, \mathcal{A} = \mathcal{F}$.

This discretization enables us to decouple the state-action function into a local part (r) and a global part (v), which can be updated at different rates. The global part can be estimated over a sparse discretized state-action space using Value Iteration.

While the actions in the continuous case (accelerations) and our discrete case (graph edges) differ conceptually, the edges in our approach represent following a controller that computes accelerations, similar to elastic roadmaps [79]. When we sum all attraction policies, we obtain:

$$
\begin{aligned}
\mathbf{f} &= \left(\sum_i \mathbf{M}_i \right)^{\dagger} \left(\sum_i \mathbf{M}_i \, \mathbf{f}_i \right) , \\
&= \left(\sum_i \text{softmax}_i(q(\mathbf{f}_i, \mathbf{q}_{\text{agent}})) \; w_i \, \mathbf{I} \right)^{\dagger} \left(\sum_i \text{softmax}_i(q(\mathbf{f}_i, \mathbf{q}_{\text{agent}})) \; w_i \, \mathbf{I} \, \mathbf{f}_i \right) , \\
&= \sum_i \frac{\text{softmax}_i(q(\mathbf{f}_i, \mathbf{q}_{\text{agent}})) \; w_i}{\sum_j \text{softmax}_j(q(\mathbf{f}_j, \mathbf{q}_{\text{agent}})) \; w_j} \mathbf{f}_i \, , \\
&= \sum_i \text{softmax}_i(q(\mathbf{f}_i, \mathbf{q}_{\text{agent}}) + \ln w_i) \; \mathbf{f}_i \, ,
\end{aligned}
$$

leaving out the Jacobian of the task map, as it is the identity matrix $\mathbf{I}$, since all attraction points are in the same space in our case. The notation softmax_i refers to the softmax function applied to the i-th component, and ln is the natural logarithm.

This sum of RMPs results in the expected value of a Boltzmann action distribution π^*_{EBM}, which is commonly employed to select actions based on a preference function H_t while still enabling exploration, which we brought up in Section 3.3.2:

$$
\begin{aligned}
\mathbb{E}[\pi^*_{\text{EBM}}(\mathbf{f})] &= \sum_i \text{softmax}_i(H_t(\mathbf{f}_i)) \; \mathbf{f}_i \, , \\
H_t(\mathbf{f}_i) &= q(\mathbf{f}_i, \mathbf{q}_{\text{agent}}) + \ln w_i \, .
\end{aligned}
$$

As noted in Chapter 13.1 of [89], using an estimated action-value function as a preference function has certain limitations. For instance, to ensure that the softmax approximates

the true optimal policy well, the temperature parameter τ would need to be appropriately initialized, which presents practical challenges. However, since our work does not yet incorporate learning-based approximation methods for the action-value function, where scheduling of the temperature parameter would also be an issue, this does not significantly impact our approach at this stage.

The term $\ln(w)$, resulting from the weighting function defined in Section 3.1.3, can be interpreted as a bias that enhances the importance of an attraction point when it is closer to the agent. In such cases, we assume the discretized approximation will more accurately represent the actual action-value function. This also results in a smoother transitions when subgoals get into the local reach of the agent.

4.5. Stein Variational Gradient Descent as Prior for Replanning with an Iterative Motion Planner

An essential feature of our approach is the integration of SVGD with iterative motion planning. As the nodes of our graph adjust their positions to avoid obstacles, the values associated with large portions of the graph maintain proximity to their actual values. This property allows us to update the value function at a lower frequency than the edges needed for the evaluation of the advantage function.

When the advantage function can be evaluated quickly with reasonable accuracy, the weights of the RMP remain relatively correct. This characteristic ensures safety through the quick reactive execution of the motion policies while also adaptively avoiding local minima. We expect SVGD to accelerate the convergence of the Value Iteration, by preserving the accuracy despite environmental changes, thereby further improving planning efficiency.

5. Experiments

This chapter presents the experimental evaluation of the hierarchical reactive motion generation method proposed in this thesis. The experiments are designed to assess the viability of combining local reactive control with global planning through probabilistic dynamic graphs.

We begin by detailing the implementation specifics, followed by descriptions of the testing environments used for evaluation. Next, we compare our approach against baseline methods. Finally, the chapter examines strategies in near-goal regions and analyzes the impact of different update frequencies on planning and reactive control performance.

5.1. Implementation Details

We implemented our system using JAX [93], a high-performance numerical computing library that provides GPU acceleration and automatic differentiation capabilities. JAX's Just-in-Time (JIT) compilation utilizes the XLA compiler [94] to convert Python code to executables optimized for hardware accelerators. In our implementation, we used JAX version 0.5.2 on NVIDIA RTX 2080 and RTX 3080 GPUs.

JAX executes code with Tracer objects rather than concrete arrays, tracking operations to extract a computation graph that is subsequently optimized by XLA. The library's parallelization performance is primarily achieved through vectorization, which can be distributed across multiple devices executing the same program, however we did not utilize multi-device parallelization in our experiments.

To compute the Jacobians of the task maps as described in Section 3.1 and the gradients necessary for the Stein Variational Gradient Descent detailed in Section 3.2, we leveraged JAX's automatic differentiation capabilities. We omitted the $\dot{\mathbf{J}}\dot{\mathbf{q}}$ term in our implementation, to simplify computation and in turn increase control frequency, compensating for errors

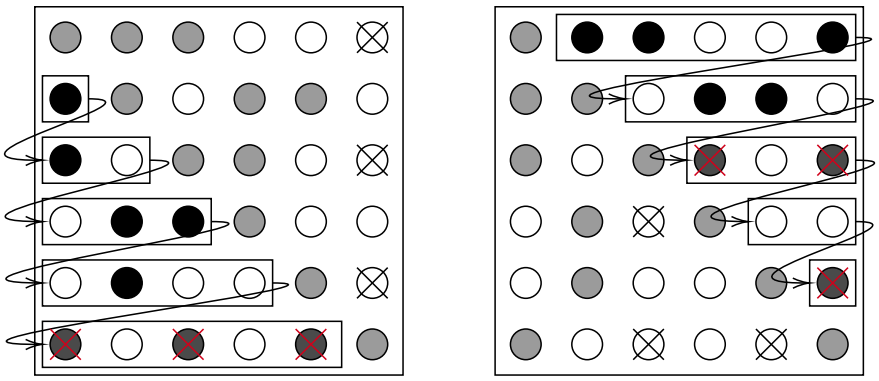


Figure 5.1.: A sparse 6-node adjacency matrix. Only the first 5 non-zero entries (filled circles) are stored. Implicit elements (grey) include self-connections (identity) and one triangular half (due to symmetry). Choosing upper vs lower triangle affects truncation: red crosses denote entries removed due to storage limits, black crosses denote their symmetric counterparts. Storing the lower triangle truncates the last node's edges entirely, while the upper triangle preserves at least one edge for all nodes.

introduced by this [6]. This simplification also reduces the space complexity of the algorithm [24]. Similarly, we chose not to incorporate curvature information into the RBF kernel during the SVGD step, as convergence was already sufficiently fast, and the additional computation would not significantly improve runtime.

For efficient storage of the Probabilistic Roadmap, we employed Batched-coordinate (BCOO) sparse matrices in JAX. The adjacency matrix of the PRM is relatively sparse, because edges are only created between nearby nodes. BCOO storage explicitly saves only non-zero entries, but to use this format on the GPU, we have to fix a maximum number of non-zero entries. This constraint risks truncating the adjacency matrix when the number of edges exceeds the limit.

To mitigate possible negative effects we leverage that we use an undirected graph as our PRM and the adjacency matrix is symmetrical. Therefore only one triangle needs to be stored. We chose to save the upper triangle, excluding the diagonal, which we assume is always connected. When the actual number of edges exceeds the set maximum, nodes with edges that get truncated have a higher probability of retaining other connections. In contrast, truncating the lower triangle would increase the risk of disconnecting nodes entirely, as illustrated in in Figure 5.1.

5.2. Simulation Environments and Scenarios

To comprehensively evaluate our method, we conducted experiments in two distinct environments: a simplified two dimensional environment and a simulated 7-DoF Franka robot environment. In this section we will review their dynamics and initializations.

For both environments we assume full and exact knowledge of the robot's and obstacles' positions is available, as sensor processing and state estimation are not the focus of this thesis. Additionally, we require that differentiable Signed Distance Fields (SDFs) of the obstacles are available.

5.2.1. Planar Navigation

This 2D toy environment provides a simplified setting to evaluate the fundamental capabilities of our method. In this environment, the agent is modeled as a circular robot with radius $r = 0.75\,\mathrm{m}$, capable of omnidirectional movement with instantaneous acceleration changes. Consequently, the configuration space of the robot is identical to the task space.

We use multiple scenes featuring diverse obstacles and target placements to highlight the strengths and limitations of our method. Varying in complexity, they include mazes with local minima, narrow passages, and scenarios requiring to traverse long, winding paths to reach the goal.

We evaluated our method in both static scenarios and dynamic scenarios where obstacles move according to Brownian noise acceleration inputs. In dynamic scenarios, obstacles have a maximum speed of $4\,\mathrm{m\,s^{-1}}$. The initial positions and velocities of the obstacles are uniformly randomly chosen within the bounds of the scene, a $60\,\mathrm{m} \times 30\,\mathrm{m}$ rectangle. Unless otherwise specified we used 100 different initializations of each scene in both the static and dynamic setting.

Motion is simulated using Euler integration, with the environment updating at $10\,\mathrm{Hz}$. A simplified collision dynamics model is implemented to simulate obstacles bouncing off of each other. Figure 5.2 presents the various scenes used in our 2D environment experiments, with detailed descriptions provided in the caption.

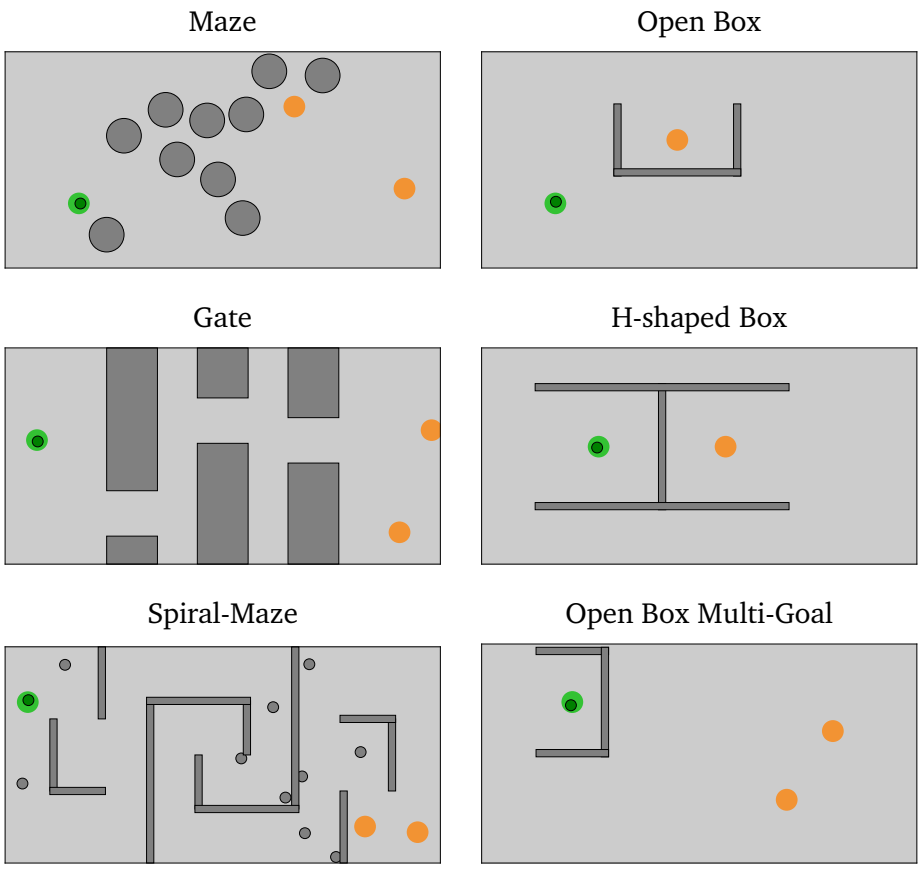


Figure 5.2.: The scenes in the planar environment. The agent (dark green) is in it's starting area (light green), and the goal areas are marked as orange. "Maze" has circular obstacles moving independently changing the optimality of different homotopies over time. The "Box" scenes feature moving local minima around the start or goal. The "Gate" environment has multiple narrow passages that move up and down. The path to the goal in the "Spiral-Maze" scene is relatively long and deviates substantially from a straight line, and is further complicated by moving circular obstacles.

5.2.2. Franka Robot Environment in MuJoCo XLA

To evaluate our method in a more complex, higher-dimensional setting, we also conducted experiments with a 7-DoF Franka robot in simulation, using MuJoCo XLA (MJX) with the BRAX framework [95]. This framework is built on JAX, making it easier to access and transfer the state of the simulator to our method.

Similar to the 2D environment, we tested scenarios with both static and moving obstacles, using 30 different initializations each. In this environment, when obstacles collide with each other, the ground, or an invisible wall forming a $2\,\mathrm{m} \times 2\,\mathrm{m}$ box around the robot, they simply reverse their velocity.

The simulator accepts positions as inputs, which an internal controller uses to calculate exact joint forces. To generate these position commands from the accelerations produced by the RMPs, we integrated the RMPs' Ordinary Differential Equation (ODE) numerically using Tsitouras' 5/4 method [96] implemented in diffrax [97]. This integration is performed over a short future time horizon $T_{\text{ODE}} = 0.1\,\mathrm{s}$, and the resulting position is passed to the lower-level controllers. The environment updates at $100\,\mathrm{Hz}$ for the RMP, while the internal controllers operate at a higher frequency of $500\,\mathrm{Hz}$.

We test in two different scenes, illustrated in Figure 5.3. In the "Maze" scene, five obstacles, randomly composed of spheres and boxes, have their positions $\mathbf{x}_o = (x_o, y_o, z_o)^T$, velocities $\dot{\mathbf{x}}_o = (\dot{x}_o, \dot{y}_o, \dot{z}_o)^T$, and sizes $\mathbf{s}_o$ uniformly randomly chosen within the following intervals:

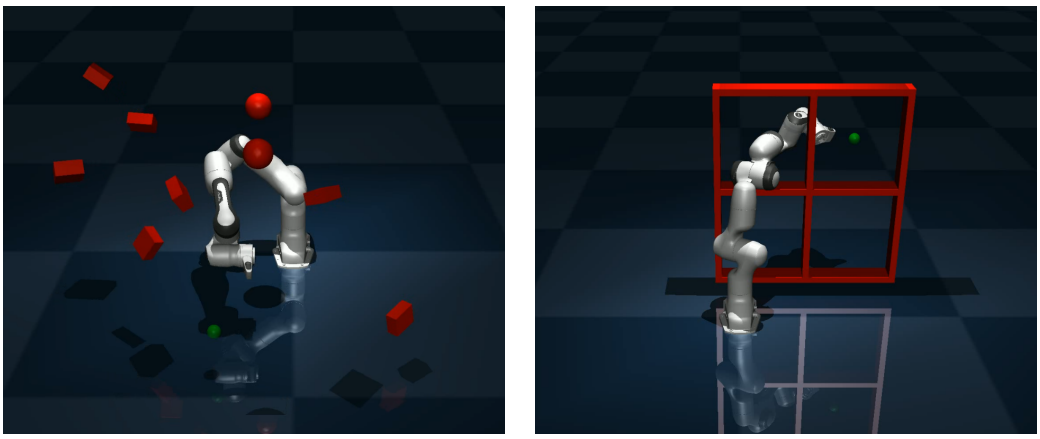
$$
\begin{aligned}
\text{Position:} \quad & x_o, y_o \in [-1\,\mathrm{m}, 1\,\mathrm{m}], \\
& z_o \in [0\,\mathrm{m}, 1\,\mathrm{m}], \\
\text{Velocity:} \quad & \dot{x}_o, \dot{y}_o, \dot{z}_o \in [0\,\mathrm{m\,s^{-1}}, 2\,\mathrm{m\,s^{-1}}], \\
\text{Size:} \quad & s_{o,i} \in [0.02\,\mathrm{m}, 0.1\,\mathrm{m}], \quad \forall\, i \in \{1, \ldots, d\},
\end{aligned}
$$

where s represents either the radius of the sphere or the half-width, -height or -depth of the box respectively.

The goal location $\mathbf{x}_g$ is uniformly sampled within the following bounds:

$$x_g \in [-0.6\,\mathrm{m}, -0.1\,\mathrm{m}], \tag{5.1}$$
$$y_g \in [-0.6\,\mathrm{m}, 0.6\,\mathrm{m}], \tag{5.2}$$
$$z_g \in [0.1\,\mathrm{m}, 0.6\,\mathrm{m}], \tag{5.3}$$

ensuring the goal is within reach of the arm.



Maze Window

Figure 5.3.: The Franka robot environments. Left: The Maze environment with randomly positioned spherical and box obstacles. Right: The Window environment where the robot must navigate through a moving opening to reach the goal.

For the "Window" scene, the window position $\mathbf{x}_w = (x_w, y_w, z_w)$ and size $\mathbf{s}_w = (w_w, h_w, d_w)$ are restricted to the following bounds:

$$
\begin{aligned}
\text{Position:} \quad & x_w && \in [0.2\,\text{m}, 0.45\,\text{m}], \\
& y_w && \in [-0.75\,\text{m}, 0.5\,\text{m}], \\
& z_w && \in [0.0\,\text{m}, 1.1\,\text{m}], \\
\text{Velocity:} \quad & \dot{x}_w, \dot{y}_w, \dot{z}_w && \in [0\,\text{m}\,\text{s}^{-1}, 0.5\,\text{m}\,\text{s}^{-1}], \\
\text{Size:} \quad & w_w, h_w && \in [0.385\,\text{m}, 0.525\,\text{m}], \\
& d_w && \in [0.035\,\text{m}, 0.125\,\text{m}],
\end{aligned}
$$

where the starting configuration always has the robot arm reaching through the top left quadrant of the window, trying to reach a goal in one of the other quadrants.

5.3. Benchmark Evaluation

To evaluate the effectiveness of our approach, we compared it against several baseline methods. This section details the metrics used for comparison, the parameter settings for our experiments, and finally discusses each baseline.

5.3.1. Metrics

We evaluated performance using four key metrics:

Success rate measures the proportion of trajectories in which the agent reached the goal within a specified distance threshold ϵ and time bound T without collisions. In the 2D environments, we set $\epsilon = 0.5\,\text{m}$ and $T = 100\,\text{s}$. For the Franka robot environment, we evaluated the distance between the end-effector position and the goal, with $\epsilon = 0.02\,\text{m}$ considered a success and a time limit of $T = 10\,\text{s}$.

Collision rate indicates the proportion of trajectories that failed due to collisions.

Path length is calculated as the piecewise linear approximation of the arc length of the path from the start position to the first instance when the goal is reached in configuration space.

Time to goal is measured by counting the evaluated timesteps until the first time the goal is reached.

As our primary aim is to demonstrate the feasibility of the method, we do not directly compare computation times against the baselines but instead provide an analysis of computation time in Section 6.4.

5.3.2. Configuration Details

For our experiments, we configured our system with carefully selected parameters to ensure meaningful comparisons. Because the two environment models feature different space definitions and result in different computational complexities, we opted to vary certain parameters between them.

We employed $n = 120$ nodes in the PRM in the 2D environment. to accommodate the higher-dimensional space and stricter tolerances, we increased this to $n = 500$ nodes in the Franka robot environment.

In the planar setting nodes connect if within a distance $\Delta_{\text{max}} = 15\,\text{m}$, with up to 10000 edges stored and during collision checks, each edge is tested at 30 equidistant points for path safety. For the Franka environment, the upper triangular adjacency matrix stored a maximum of 5000 non-zero entries, as the collision checking there is more computationally intense. Configurations were connected if their Euclidean distance was under $1.4\,\text{rad}$, with the fifth and sixth joint dimensions scaled by 0.1 and the seventh joint excluded from distance calculations. This balanced end-effector position and configuration space distances efficiently. An increased sampling density of 100 configurations checked per edge was necessary, due to the edge lengths being longer in relation to the agent's potential occupancy of the space.

We tuned the RBF kernel bandwidth to spread nodes evenly throughout the space: $h = 5$ for the 2D environment and $h = 0.5$ for the robot environment. For rewards, we used horizons of $T_r = 1\,\text{s}$ (2D) and $T_r = 0.1\,\text{s}$ (robot), with a reward of 100 for exact goal achievement and goal region parameters $\Delta_g = 0.2\,\text{m}$ (2D) and $\Delta_g = 0.05\,\text{rad}$ (Franka), see Section 4.3.

All modules ran sequentially, with Value Iteration running to convergence. This allows us to focus on establishing method feasibility while deferring investigation of module decoupling to future work.

5.3.3. Overcoming Local Optima in Reactive Control

To demonstrate that a myopic policy can get trapped in local minima, we compared the success rate of RMPs [5, 24] with our method. This comparison was conducted in both the 2D toy environments and the Franka robot environment, with particular interest in how the methods handle local minima in the "Open Box" and "Window" scenes.

For a fair comparison, we maintained the same parameters, i.e. gains and dampening, across both methods, as these significantly influence the agent's speed. The RMP baseline incorporates collision avoidance policies, a target reaching policy, and a dampening policy. In 2D environments, collision policies keep a safe distance between the spherical agent, task space limits, and obstacles. In the Franka environment, obstacle avoidance policies maintain safe distances from obstacles, task space limits, potential self-collisions, and joint limits. The dampening and target reaching policies are defined in task space.

The key distinction between our method and the baseline RMP lies in the target-reaching. The RMP has one single attraction policy, defined in task space. In our approach this policy remains inactive (i.e., its metric is zero) until the agent approaches the goal region, at which point its influence gradually increases, smoothly superseding a PRM-following policy, not present in the baseline, to ensure a seamless transition to goal tracking.

5.3.4. Impact of Planning Horizon Length

Some MPC based methods attempt to overcome local minima by incorporating exploration mechanisms that diversify the set of possible future trajectories for evaluation. However, if the time horizon is too short, these methods can also get trapped because no exploration trajectory can reach a better state outside the local minimum. To demonstrate this limitation, we compared the success rate of HiPBI [67] with our method.

As discussed in the related works section, HiPBI functions by blending different RMPs based on evaluated rollouts. To reduce bias, we maintained the same parameters for the target reaching and collision avoidance RMP as in previous experiments. Compared to the RMP baseline, HiPBI includes policies that introduce clockwise and counterclockwise curling motions to do exploration. It then uses 200 evaluations looking ahead for $10\,\mathrm{s}$, to find the best blend.

5.3.5. Global Optimization and Initial Conditions

Other optimization-based methods offer a more global perspective on trajectories by setting the end of the time horizon to coincide with goal attainment. However, their ability to find the global optimum in trajectory space often depends heavily on initialization.

We compared our method against a Gaussian Process Motion Planning (GPMP) [10] baseline, which initializes with a Gaussian process prior, assuming constant velocity for a smooth trajectory connecting the start and goal positions. The trajectory is represented as 40 particles, and 24 different initializations are optimized for 10000 steps. The computed solution is tracked using the same RMP as the baseline, with continuous updates to the goal position to follow the computed trajectory at the maximum possible speed.

To highlight GPMP's sensitivity to the initial prior, we focus on the "H-shaped Box", and "Spiral Maze" environments. Since the baseline GPMP method lacks specific adaptations for handling moving obstacles as in [77, 78], we focus this comparison on static scenes.

5.4. Goal Region Motion Generation

The transition from following the Probabilistic Roadmap to reaching the goal presents different challenges depending on the environment. In the 2D environment, adding the goal position to the PRM is straightforward, because the configuration and task spaces are simple translations of the same space. However, the inverse kinematic problem becomes more complex in the Franka robot environment due to the redundancy in its degrees of freedom, among other reasons.

To address this challenge in the robot environment, we compared several approaches:

Lookahead The reward of the edge from each node to itself is dependent on the future distance between the end-effector position and the goal, after following the target tracking policy, as discussed in Section 4.3. For this approach, we used the same ODE integration method as in the position control of the Franka robot with the Horizon $T_r = 0.1\,\text{s}$.

Direct We also evaluated using the straight-line distance from the node to the goal without ODE integration, referred to as $T_r = 0\,\text{s}$ in our experiments.

Inverse Kinematics We generated multiple goal nodes and added them to the graph by solving the inverse kinematics problem at every time-step. This was accomplished by optimizing the sum of the log likelihood of collision and the distance between the end-effector and the goal, using a stochastic optimizer. We then selected the top $k = 15$ results. While not computationally efficient, this approach is straightforward to implement and serves our experimental purposes.

In Section 7.2, we briefly discuss how these approaches could be extended to still view the goal as part of the graph and better scale the task tracking policy's metric.

Additionally, we evaluate a sampling prior in the Panda environment, where the scenes only have a single goal. It is based on a Boltzmann/Gibbs distribution $p(\mathbf{x}_j) \propto \exp \|\mathbf{x}_j - \mathbf{x}_{\text{goal}}\|$ to increase node density around the goal region. This results in a linear attraction towards the goal to all nodes of the PRM during the Stein Variational Gradient Descent.

5.5. Hierarchical Planning Frequency Analysis

To evaluate the computational efficiency of our method and the relation of different update frequencies, we measured execution times for the various components of our system using the IO host callback of JAX. It is important to note that these measurements exclude delays associated with receiving sensor data and sending control commands to the actuator. And due to XLA optimization potentially reordering and intertwining different code sections, the times may only provide a hint to the real times. To mitigate this, we enforced blocking between each step during measurement, which slows execution by preventing some compute graph optimizations. Our current implementation is also not fully optimized, containing duplicate executions and data structures. Consequently, an optimized implementation would likely achieve higher frequencies. All measurements were conducted on an NVIDIA RTX 3080 GPU.

To evaluate the impact of planning frequency on system performance, we simulated different update rates for the collision checking of the PRM edges by only updating every $\{1, 5, 10, 25, 50, 100, 200, 300\}$ steps. During these experiments, the other modules, Value Iteration, and local advantage function calculation, continued to run at every environment step. We also vary the update frequency for the SVGD to demonstrate its ability to preserve the accuracy of the nodes' value function between edge collision updates.

These experiments were conducted in the two dimensional "Maze" scene with 25 obstacles moving at a reduced maximum speed of $1\,\mathrm{m\,s^{-1}}$ on 50 different random seeds. This environment was chosen because the paths to the goal are narrow, frequently blocked and warped and therefore require continuous consideration of different homotopies.

6. Results

This chapter presents a comprehensive analysis of the experimental results obtained from testing our hierarchical reactive motion generation method. We assess the performance of our approach in comparison to established baselines in both static and dynamic settings.

We examine success rates, path length and time-to-goal metrics, evaluate strategies for near-goal behavior in the Franka robot environment. Additionally we investigate collision-checking frequencies, highlighting SVGD's effectiveness in mitigating the negative effects of reduced rates.

6.1. Performance Comparison with Established Approaches

To assess the effectiveness of our approach, we compared it against three baseline methods: local reactive motion generation with RMP [5, 24], HiPBI [67], and GPMP [10]. Descriptions of these approaches are provided in Chapter 2.

6.1.1. Static Planar Environments

The results in Figure 6.1 demonstrate that the RMP baseline, which relies solely on local reactive control, exhibits limited success in navigating complex environments. Even in static scenarios, the local reactive properties of RMPs frequently cause the agent to become trapped in local minima. While the RMP achieves some success in the "Maze" scene, it consistently fails in scenarios with deliberate local minima. For instance, in the "Open Box" and "Open Box Multi-Goal" scenes, where the agent begins surrounded by obstacles, the method cannot escape the local minimum created by these barriers.

The MPC method, HiPBI, demonstrates an improved ability to avoid local minima and reach goals compared to the purely reactive approach. This improvement stems from

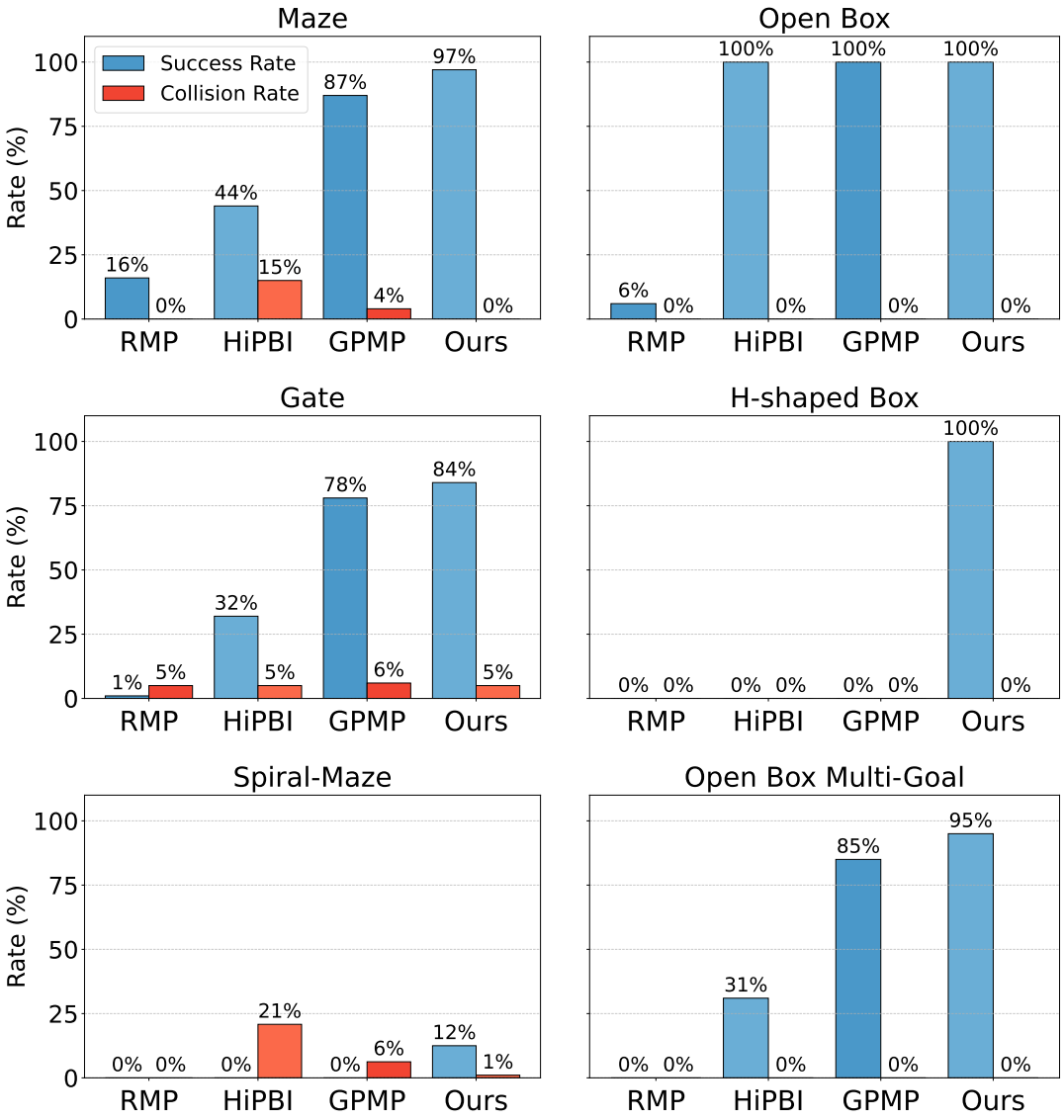


Figure 6.1.: Performance comparison in static planar environments. Success rates (blue) and collision rates (red) for our method versus three baselines: reactive RMP, model-predictive HiPBI, and optimization-based GPMP. Our approach consistently achieves higher success rates across diverse environments with local minima and narrow passages.

its capacity to evaluate multiple potential future trajectories. However, HiPBI still does not match the performance of our method or GPMP in most scenarios. Notably, in the "Gate" environment, GPMP more effectively navigates through narrow passages due to the gradients leading the path towards the gap, while HiPBI must rely on exploration to discover these passages.

It is worth noting that the higher collision rate HiPBI exhibits, compared to other methods, could be attributed to the reduced importance assigned to obstacle avoidance policies during the blending process. It is possible to constraint the blending to not impact the influence of these policies. A comparison with these constraints would be an interesting direction for future work.

GPMP is effective in most static environments but struggles in the "H-shaped Box" scene. The initializations, shooting through the middle section of the shape, fails to escape local minimums caused by a balance between collision avoidance and smoothness costs. The non-convex obstacle shape leads to an increased smoothness costs without reducing collision costs in the direction from the current trajectory candidate to a global minimum. In the "Open Box" scenes, GPMP generates successful navigation trajectories, while the "Spiral-Maze" environment challenges all methods with its long, narrow passages.

Our method consistently outperforms the baselines across most environments. This success rate demonstrates the effectiveness of combining local reactive control with sampling-based motion planning. In contrast to the other methods our method has a broader awareness of the free space, and can utilize the connectivity information more effectively in these purposefully difficult scenes.

6.1.2. Adaptation to Dynamic Obstacle Movements

In dynamic environments, see Figure 6.2, the RMP baseline achieves higher success rates in the "Maze" scene compared to static environments. This improvement occurs because moving obstacles may temporarily clear previously blocked paths. However, in the "Open Box" scenarios, the local minimum continuously traps the RMP agent.

The GPMP baseline performs worse in dynamic environments, as the implementation is not optimized for dynamic obstacle handling. As standard GPMP relies solely on full replanning for recovery of path deviations caused by unexpected obstacle movements, we reinitialize trajectories during execution. Since we tuned the initialization parameters based on a few distinct goal distances, their adaptability to other distances is limited and might lower success rates further. By employing a roadmap that spans the entire free

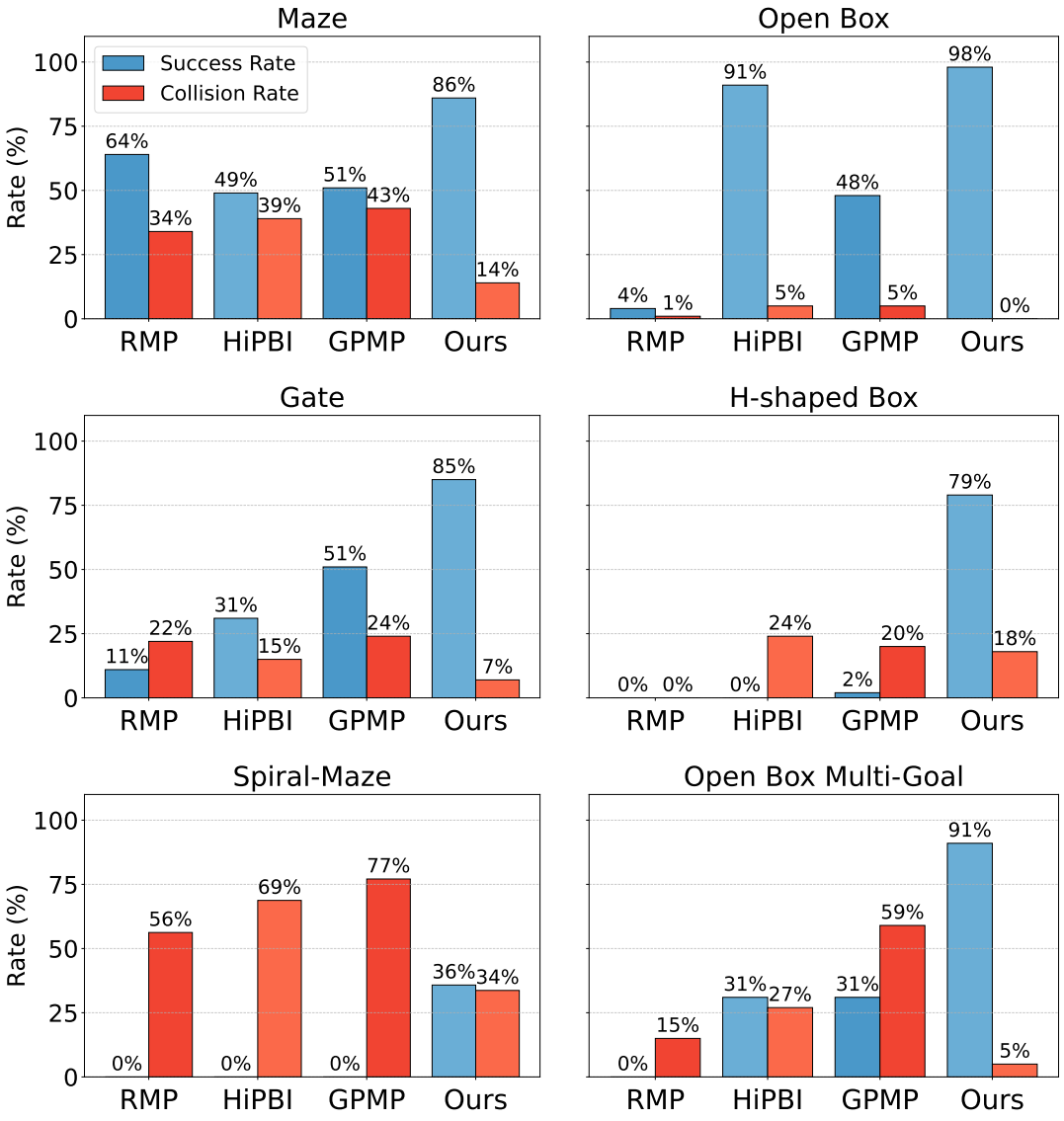


Figure 6.2.: Performance comparison in planar environments with moving obstacles. Success rates (blue) and collision rates (red) for our method versus three baselines: reactive RMP, model-predictive HiPBI, and optimization-based GPMP. Results highlight our method's adaptability to changing environments while demonstrating the increased collision risk faced by all approaches under dynamic conditions.

space and using Value Iteration to compute optimal paths from each node, our method inherently includes a built-in recovery mechanism.

Our method maintains the highest success rate across dynamic environments, however, we can not infer from these results how competitive our method is against specialized approaches for dynamic environments. The comparatively low success rate of HiPBI opposed to RMP in the "Maze" scene is unexpected and suggests a suboptimal parameter choice for the MPC method. Nonetheless, we have demonstrated that our method possesses a robust capability to handle changing conditions while effectively avoiding local minima.

All methods show increased collision rates in dynamic environments, partially attributed to parameter settings and situations where obstacles completely surround the agent. Since we only select subgoals that can be reached without collision, attraction and obstacle avoidance policies are less frequently in conflict. This contributes to the increased safety of our approach. The "H-shaped Box" scene presents an interesting case where the RMP baseline exhibits the lowest collision rate because it never leaves the starting area. To successfully reach the goal, agents must navigate through constrained spaces where they risk being trapped between the moving obstacle and the boundaries.

6.2. Trajectory Quality Assessment

Figure 6.3 presents path length and time-to-goal metrics aggregated across all static and dynamic scenes. More detailed results for individual scenes can be found in the appendix in Chapter A.

HiPBI generally produces longer paths and trajectory durations, due to its curling policies. This exploration tendency, while beneficial for avoiding local minima, reduces path efficiency. With better parameter tuning, HiPBI's performance in these metrics could potentially improve.

Similarly, GPMP's longer time-to-goal in many scenarios can be attributed to the trajectory following implementation rather than the algorithm itself. The trajectory generation may be efficient, but the execution through the RMP introduces additional detours and delays.

Our method demonstrates shorter paths and faster goal achievement in most scenarios. However, this advantage should be interpreted cautiously, as more extensive hyperparameter optimization could significantly alter the relative performance. For instance, extending

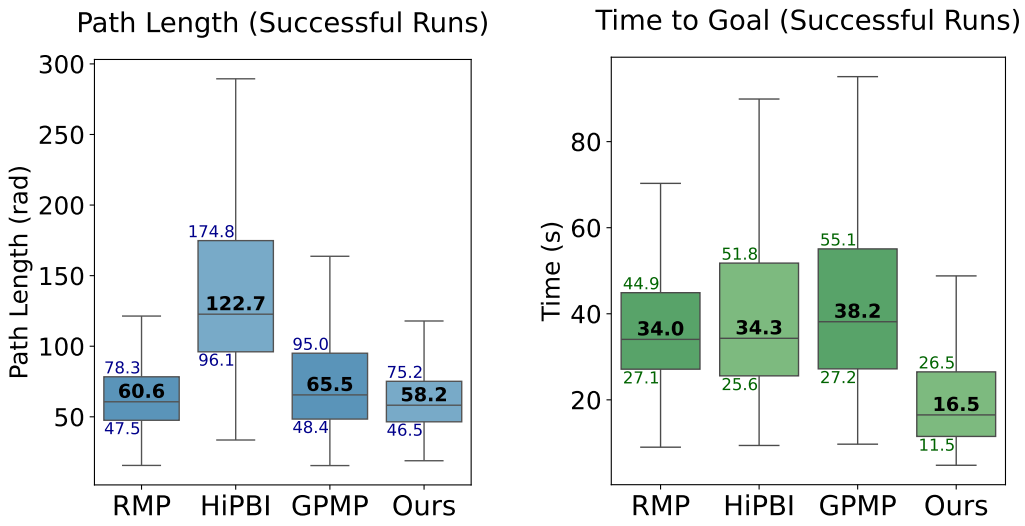


Figure 6.3.: Path length (left) and time-to-goal (right) metrics for different motion planning approaches, averaged across all planar scenes. Our method demonstrates efficient navigation while maintaining high success rates shown in previous figures.

the time horizon for HiPBI or implementing more fine-grained GPMP initializations might yield different comparative results.

Further investigations into the smoothness, which can be an issue for sampling-based approaches, is deferred until after tuning the advantage calculation parameters. Preventing oscillations due to sub-optimal low-level policy gains is also essential. But for our efforts we can conclude that the path lengths and trajectory durations are within the same scales as established methods, suggesting our method achieves both reliable goal achievement and efficient paths.

6.3. Exit Nodes in Higher-Dimensional Configuration Spaces

In the Franka robot environment, the RMP baseline achieves reasonable success in the "Maze" scene, see Figure 6.4, but occasionally encounters local minima where target-reaching and obstacle-avoidance policies counteract each other. This results in oscillatory behavior with the robot moving in small circles without making progress. The "Window" scene presents a more challenging scenario where the end-effector begins close to the goal but is separated by the window structure, causing the purely reactive approach to fail.

We compare different strategies for selecting exit nodes within our method. Incorporating goal nodes found through inverse kinematics substantially improves performance. When inverse kinematics solutions are not added to the PRM, using a time horizon to check if the goal is reachable from a current configuration proves more effective than simply evaluating the Euclidean distance between nodes and the goal. This is negatively impacted by two primary factors: starting configurations might be too distant for the goal-tracking policy to be switched on, and the composition of parameters across all policies leads to numerical instability during Ordinary Differential Equation (ODE) integration. This causes failures even when viable paths exist.

The collision rate was consistently 0% across all experiments in the Franka environment, suggesting that unsuccessful runs result from inadequate space connectivity representation. The impact of the goal prior further supports this hypothesis. Currently the parameters of the prior cause a higher impact in the "Window" scene, where without it the PRM's ability to connect to the goal are obstructed more by the present narrow passage.

The goal prior based on the Gibbs distribution effectively implements a single step of the optimization problem used for the inverse kinematics solutions, so the PRM should eventually converge to a very similar state as with inserting the nodes after optimization. This indicates the success rate of our method, using the time horizon approach, could potentially surpass the RMP baseline even in the "Maze" setting.

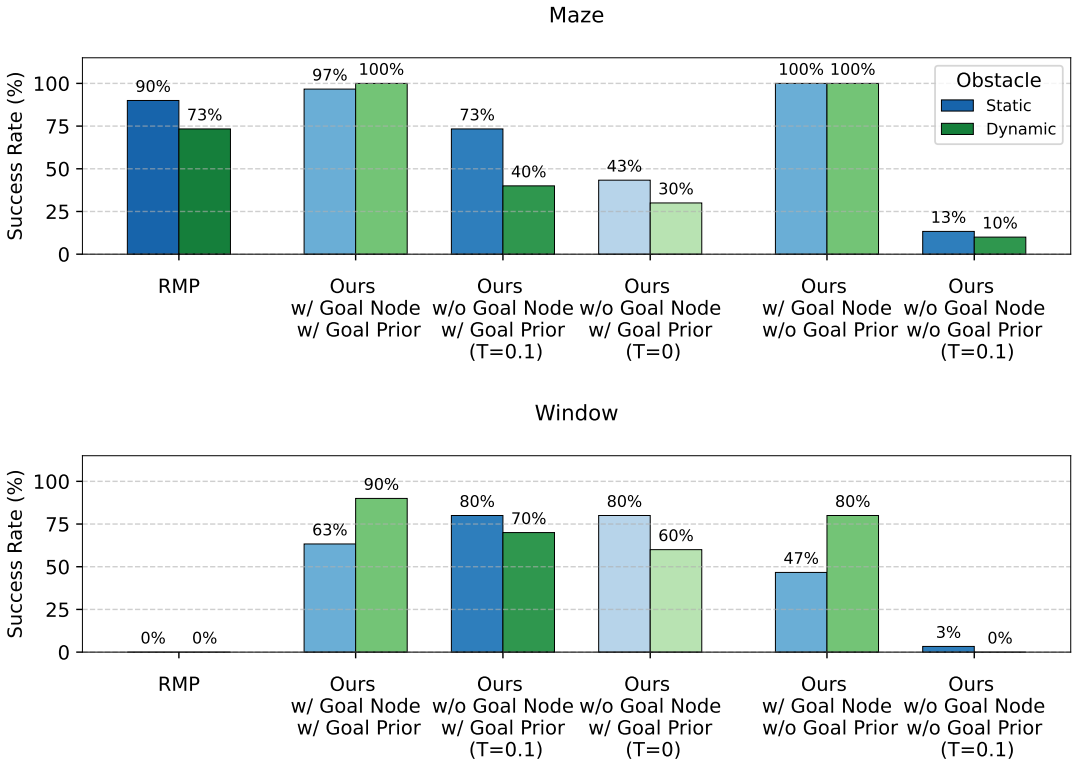


Figure 6.4.: Success rates of the RMP baseline and different exit node selection strategies in the Franka robot environments. Our method with inverse kinematics solutions (w/ Goal Node) consistently outperforms other variants, especially in the challenging “Window” scenario. Time-horizon evaluation $(T = 0.1\,\mathrm{s})$ proves more effective than direct distance measurement $(T = 0\,\mathrm{s})$ when goal nodes are not added, but fails to find solutions when the SVGD does not bias the nodes towards the goal (w/o Goal Prior). The RMP baseline demonstrates decent performance in “Maze” scenarios but struggles with the “Window” environment’s local minimum.

Task	2D Wall Time (ms)			Panda Wall Time (ms)		
SDF Update	0.575	±	0.008	4.690	±	0.031
SVGD Step	2.043	±	0.164	2.994	±	0.024
PRM Update	6.152	±	0.077	45.098	±	85.489
Value Iteration	0.490	±	0.009	2.263	±	38.138
Advantage	0.432	±	0.006	1.025	±	0.037
RMP	1.801	±	0.029	1.669	±	0.040

Table 6.1.: Computational performance of modules in milliseconds $(\text{mean} \pm \text{std})$. Results highlight the significant difference between edge collision checking and other operations. Measurements are from an NVIDIA RTX 3080 GPU execution with potential for further optimization through reduced redundant calculations and improved numerical conditioning of the RMPs.

6.4. Hierarchical Planning Frequency Assessment

Table 6.1 presents the computational times for system components. As expected, edge collision checking requires the most resources, while other modules operate at 0.5-2 kHz in the 2D environment and slightly slower in the Panda environment. Whether these measurements reflect achievable frequencies in real-world robotic systems remains an open question, due to the inaccuracy of the measurement method, though the results align with our intuition on relative complexities.

The parameters of the RMP policies lead to numerical stiffness in certain regions, particularly when the end-effector approaches the goal. This stiffness necessitates smaller timesteps during ODE integration, which could explain the high variance observed in the PRM Step timing for the robot environment.

Value Iteration runs to convergence in our implementation, so higher variance in its execution time is expected. Interestingly, this variance is not present in the 2D environment, possibly due to the higher node density and increased edge redundancy, which could make the values more stable over time.

To test the effectiveness of SVGD as a smoothing operation between collision checks, Figure 6.5 presents success rates at different update rates of both modules. In this

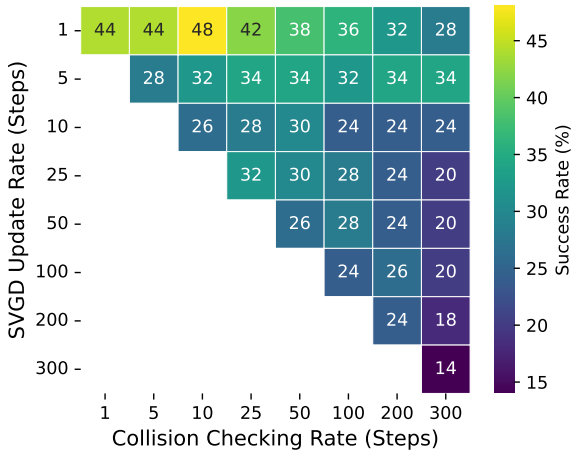


Figure 6.5.: Success rates of PRM edge collision check frequency in relation to SVGD in dynamic, crowded environments. The x-axis and y-axis represent RMP steps between edge updates and SVGD steps, respectively. Results show global planning requires fewer updates than reactive control, with SVGD maintaining accuracy between checks.

particularly crowded scene the overall success rate is relatively low, as sometimes no feasible path exists, and the agent may become surrounded without an escape route.

The results demonstrate that the PRM edges do not require updating at the same frequency as the RMP. Success rates only decline when edges are updated significantly less frequently. This effect would likely be even less pronounced in less crowded scenes.

We can infer from the data that SVGD helps preserve value accuracy between collision checks, as the success rate drops with decreasing SVGD frequency. Figure 6.6 provides a visual illustration of this effect, showing how the value function (color gradient) maintains accuracy for a period between edge updates (at $t = 20\,\mathrm{s}$ and $t = 30\,\mathrm{s}$) before gradually losing precision in certain regions.

This analysis confirms that using SVGD as a smoothing operation between collision checks effectively maintains planning accuracy while reducing computational requirements, enabling effective planning in dynamic environments.

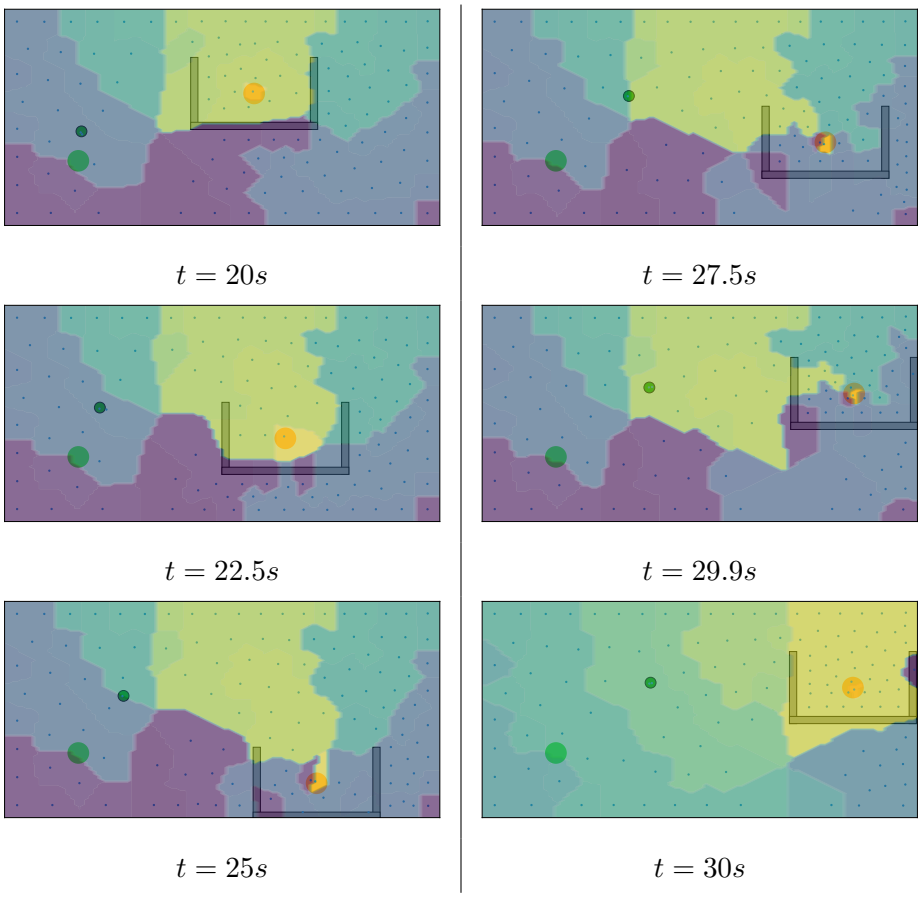


Figure 6.6.: SVGD as smoothing in-between collision checking. The rewards for the edges of the PRM are updated at $t = \{20, 30\}s$. During these times the position of the nodes is updated using SVGD. The colour gradient shows the value function, which keeps it's accuracy for a while, but over time parts of the graph loose their accuracy.

7. Conclusions

This chapter concludes the thesis by examining the limitations of the proposed approach, suggesting directions for future research, and summarizing the key contributions. The integration of global planning with reactive motion generation with Stein Variational Gradient Descent bridging between the two demonstrates significant potential for robot navigation in dynamic environments, as our experimental results have shown. However, several constraints and challenges remain to be addressed.

7.1. Limitations

Despite the promising results, our approach exhibits several limitations that warrant consideration. These limitations span from theoretical assumptions to practical implementation challenges and provide important context for interpreting our findings.

A fundamental limitation of our method is the assumption of full observability. In real-world robotic applications, sensors provide incomplete and noisy information about the environment, creating discrepancies between the robot's internal model and the actual state of the world. We do not explicitly model uncertainty in either the local reactive control or the global planning components, which limits the robustness of our solution in uncertain environments.

Our approach also presents theoretical limitations in representation. Since we do not consider the configuration-time space explicitly, certain solution trajectories cannot be found by the roadmap. For example, in scenarios with periodically opening and closing gaps (such as doors on opposite sides of a corridor), the method may fail to find viable paths or get stuck alternating between following two opposing solutions. Moreover, the robot might navigate to states that appear collision-free in the current time step but will

result in collisions in the future without possibility of escape, a problem known in the literature as inevitable collision states [98].

As discussed in Section 5.2, our method assumes the availability of differentiable cost maps. While learned approximations such as those presented in [3, 91] could serve as alternatives, using such approximations impact the applicability of theoretical safety guarantees that depend on the accuracy of obstacle representations.

The proposed approach also has limitations in handling kino-dynamic constraints. While these constraints are enforced through the parameters of the low-level policies, they are not fully considered during the planning phase. Consequently, the Value Iteration on the PRM may produce trajectories that are not optimal for the robot's actual dynamics. This discrepancy between planning and execution could lead to suboptimal performance in scenarios where dynamic constraints significantly impact feasible motions.

From a practical perspective, the tuning of parameters presents a significant challenge. The different components, SVPRM, RMP, and Value Iteration, each have their own set of parameters, and these parameters interact in complex ways that affect the overall performance of the combined method. This interdependence makes parameter optimization a tedious process that currently requires extensive trial and error, but could potentially be automated to a certain degree.

Additional practical limitations concern the theoretical stability guarantees of our approach. The current implementation omits correction terms that depend on the time derivative of the Jacobian $\dot{\mathbf{J}}$ and does not explicitly consider the change of weights as mentioned in [29]. The absence of these terms means that the stability guarantees of the original RMP framework might not fully apply to our implementation. Furthermore, since we operate in environments with moving obstacles, the stability of our system could be compromised if the obstacle movements themselves result from unstable systems, effectively transferring their instability to our reactive control.

The proposed method also has limitations in handling specific types of dynamic environments. In scenarios where obstacles move in an adversarial manner, more sophisticated motion policies and planners would be required. Likewise, if all obstacles move in a similar direction, they could push all roadmap nodes toward the same region, resulting in high node density in some areas and insufficient coverage in others. This uneven distribution would compromise the roadmap's ability to represent the connectivity of the free space.

Narrow regions of free space present another challenge, as they might not be traversable due to suboptimal node positions. The SVGD component of our method can adapt to continuously moving obstacles, but it may struggle with abruptly appearing obstacles.

This limitation is particularly relevant for mobile manipulators operating with incomplete map information, where obstacle data only becomes available once detected by sensors. In such cases, the disturbance to the roadmap might be too severe for continuous adaptation to preserve the accuracy of the value function.

Finally, our current approach is not suitable for planning with contact forces. Simply replacing attraction and collision avoidance controllers with impedance controllers would be insufficient, as the edge collision measure would need substantial revision to accommodate controlled contact with obstacles rather than enforcing minimum distances.

7.2. Outlook

Building upon the identified limitations, this section outlines promising directions for future research that could enhance the capabilities and applicability of our approach.

A natural extension of our method would be to refine the representation of the probabilistic roadmap. Currently, the roadmap consists only of configuration space poses, with an attraction policy defined in the same space to guide the agent. An interesting direction would be to conceptualize the goal tracking RMP as a unidirectional edge to a goal node defined in task space. This representation could replace the current self-connecting edges that are weighted based on the success of a rollout of the RMP. Such a modification would allow changing the metric of the target tracking policy according to the local advantage function, providing greater accuracy than the current distance-based metric.

Further enhancements could involve incorporating more complex motion policies as nodes in the graph. For example, gait control and locomotion tasks or pouring motions could be represented as specialized nodes. However, defining edges between these complex motion policies and simple attraction policies presents challenges, particularly in developing recovery policies for transitioning away from the influential region of a complex node.

Transitioning the approach to real-life experiments represents an important future direction. This practical implementation would require careful parameter selection to avoid numerical stiffness, robust handling of sensor noise and latencies, and thoughtful balancing of communication overhead when decoupling methods across different frequency domains. The practical benefits of executing certain components at higher frequencies must be weighed against the computational and communication costs. Measuring the effect of Stein Variational Gradient Descent on Value Iteration convergence in terms of

iteration count could provide insights into the computational efficiency improvements gained by warm-starting the process.

Several technical improvements could enhance the method's performance and theoretical foundations. Implementing RMPfusion as described in [29] could mitigate the effects of changing weights, while using geometric fabrics [30] as underlying reactive motion policies might provide better adaptation to robot and obstacle dynamics.

Adding arrival time estimation and safe intervals to the global planner, similar to the approach in [71], could improve safety in dynamic environments. To further improve safety, future work could incorporate certainty metrics in the reward function to help avoid inevitable collision states. The current edge weight computation could also be enhanced by utilizing Riemannian distances of the low-level root RMP, which should encode collision information in a way similar to [20].

Another valuable comparison would be between SVGD and using RMP controllers to move the PRM nodes. While RMPs perform extensive calculations to ensure stability, which may not be strictly necessary for the roadmap, and require more parameter tuning than SVGD, understanding the relative advantages of each approach could lead to beneficial hybrid solutions. Additionally, analyzing whether XLA automatically optimizes the current RMP formulation could reveal opportunities to avoid explicit computation and storage of the Jacobian, which is computationally and spatially expensive as noted in [24].

From a theoretical perspective, studying our approach through the lens of composable energy policies [6] could be fruitful. Particularly since these policies also use the notion of an advantage function to formulate probabilistic policies, albeit typically with only a one-step lookahead.

Our experimental results suggest that the impact of the goal prior is stronger in environments like the "Window" scene, where the goal is positioned inside a narrow passage. Developing a prior that increases node density around both the agent and the goal, e.g. based on a mixture of Gaussians, would likely improve success rates, particularly where the agent begins in narrow passages. Increasing the density around critical regions, identified for example by approaches like LEGO [39], could potentially increase the success rate in a broader set of scenarios.

Several algorithmic improvements could enhance the method's efficiency and robustness. Developing a better, possibly learned estimator for determining edge collision-free status could reduce computational costs. Reusing collision information and its gradients, available through automatic differentiation, could enable updating edge feasibility without explicit checking, or serve as a prioritization measure similar to lazy PRM [38] approaches.

From a theoretical standpoint, establishing proofs that the method is guaranteed to converge to the goal in a stable manner would strengthen its theoretical foundations. Studying the relationship between success rate, computational complexity, PRM node density, and obstacle density would provide valuable insights for parameter selection in different environments.

Finally, an intriguing direction would be to inform the higher-level planner, the Value Iteration, based on the current trajectory, either by replacing the Value Iteration completely or by changing the edge weights. This approach would lead to a form of Generalized Policy Iteration [89], effectively incorporating reinforcement learning principles.

7.3. Summary

This thesis has presented an approach to robot motion planning that integrates local reactive control with global planning to address challenges in dynamic environments. By combining Riemannian Motion Policies (RMPs) for reactive control with Stein Variational Probabilistic Roadmaps (SVPRMs) for adaptive global planning, we have demonstrated a method that maintains safety guarantees while efficiently navigating complex, changing environments. Value Iteration on this roadmap generates policies that guide the robot toward the goal while avoiding local minima, and the advantage function weights different attraction policies to effectively blend local control with global planning.

Results in both 2D environments and simulated 7-DOF robot scenarios demonstrate that our method outperforms purely reactive approaches in environments with local minima. It also shows higher success rates and fewer collisions in dynamic environments than comparison methods that do not employ concepts specialized for these scenarios.

A key concept of our approach is how Stein Variational Gradient Descent bridges different operational timescales, a critical challenge in dynamic environments. Robot motion control requires high-frequency reactivity for safety, while global planning is computationally intensive and traditionally operates at much lower frequencies. This adaptation preserves the accuracy of the value function, allowing the system to operate effectively even when edge collision checking occurs at substantially reduced rates.

Our experimental results validated this multi-timescale approach. We demonstrated that edge collision checking can be performed up to 25 times less frequently than reactive control without significant performance degradation when SVGD continuously updates node positions. This represents a substantial computational efficiency gain while maintaining

high success rates, as the advantage function derived from the Value Iteration continues to provide effective guidance despite the reduced update frequency of the global roadmap. The visual evidence of the value function preservation between updates, shown in our results, offers intuition for the effectiveness of SVGD as a bridging mechanism between the different timescales of global planning and local motion generation.

The limitations identified in this chapter highlight important areas for future work, particularly in closing the sim-to-real gap and providing formal guarantees. The outlined research directions suggest avenues for extending the method's capabilities and addressing its current limitations.

In conclusion, the integration of reactive motion generation with probabilistic dynamic graphs presents an approach to robot motion planning in dynamic environments. The ability to handle different operational timescales through SVGD contributes to efficient planning while maintaining safety considerations. While challenges remain, the method's performance in complex scenarios suggests potential for applications where navigation amidst moving obstacles is required, e.g. in human-robot interaction tasks.

Bibliography

[1] A. Discant, A. Rogozan, C. Rusu, and A. Bensrhair, "Sensors for Obstacle Detection - A Survey," in *2007 30th International Spring Seminar on Electronics Technology (ISSE)*, pp. 100–105, May 2007.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, pp. 1231–1237, Sept. 2013.

[3] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized Deep Signed Distance Fields for Reactive Motion Generation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6673–6680, Oct. 2022.

[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, vol. 2, pp. 500–505, Mar. 1985.

[5] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian Motion Policies," *arXiv preprint arXiv:1801.02854*, July 2018.

[6] J. Urain, A. Li, P. Liu, C. D'Eramo, and J. Peters, "Composable energy policies for reactive motion generation and reinforcement learning," *The International Journal of Robotics Research*, vol. 42, pp. 827–858, Sept. 2023.

[7] D. Morton and M. Pavone, "Safe, Task-Consistent Manipulation with Operational Space Control Barrier Functions," *arXiv preprint arXiv:2503.06736*, Mar. 2025.

[8] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, Aug. 1996.

[9] S. LaValle, "Rapidly-exploring random trees : A new tool for path planning," *Research Report 9811*, 1998.

[10] M. Mukadam, X. Yan, and B. Boots, "Gaussian Process Motion planning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9–15, May 2016.

[11] A. T. Le, K. Hansel, J. Carvalho, J. Watson, J. Urain, A. Biess, G. Chalvatzaki, and J. Peters, "Global Tensor Motion Planning," *arXiv preprint arXiv:2411.19393*, Dec. 2024.

[12] A. Lambert, B. Hou, R. Scalise, S. S. Srinivasa, and B. Boots, "Stein Variational Probabilistic Roadmaps," in *2022 International Conference on Robotics and Automation (ICRA)*, (Philadelphia, PA, USA), pp. 11094–11101, IEEE, May 2022.

[13] R. E. Bellman, *Dynamic programming*. Princeton, NJ: Princeton University Press, Oct. 1957.

[14] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving Rubik's Cube with a Robot Hand," *arXiv preprint arXiv:1910.07113*, Oct. 2019. arXiv:1910.07113 [cs].

[15] M. Weissenbacher, S. Sinha, A. Garg, and Y. Kawahara, "Koopman Q-learning: Offline Reinforcement Learning via Symmetries of Dynamics," *arXiv preprint 2111.01365*, June 2022.

[16] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, May 2022.

[17] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering Diverse Domains through World Models," *arXiv preprint arXiv:2301.04104*, Apr. 2024.

[18] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, Sept. 2017.

[19] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, pp. 43–53, Feb. 1987.

[20] M. Saveriano, F. J. Abu-Dakka, and V. Kyrki, "Learning stable robotic skills on Riemannian manifolds," *Robotics and Autonomous Systems*, vol. 169, p. 104510, Nov. 2023.

[21] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "ImitationFlow: Learning Deep Stable Stochastic Dynamic Systems by Normalizing Flows," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5231–5237, IEEE, Oct. 2020.

[22] N. Jaquier and T. Asfour, "Riemannian geometry as a unifying theory for robot motion learning and control," in *The International Symposium of Robotics Research*, pp. 395–403, arXiv, Sept. 2022.

[23] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff, "RMPflow: A Computational Graph for Automatic Motion Policy Generation," in *Proceedings of the 13th International Workshop on the Algorithmic Foundations of Robotics*, 2018.

[24] A. Li, C.-A. Cheng, M. A. Rana, M. Xie, K. V. Wyk, N. Ratliff, and B. Boots, "RMP2: A Structured Composable Policy Class for Robot Learning," *arXiv preprint arXiv:2103.05922*, Mar. 2021.

[25] A. Li, M. Mukadam, M. Egerstedt, and B. Boots, "Multi-Objective Policy Generation for Multi-Robot Systems Using Riemannian Motion Policies," in *Proceedings of the 19th International Symposium on Robotics Research*, pp. 258–274, Aug. 2019.

[26] B. Wingo, C.-A. Cheng, M. Murtaza, M. Zafar, and S. Hutchinson, "Extending Riemmanian Motion Policies to a Class of Underactuated Wheeled-Inverted-Pendulum Robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3967–3973, May 2020.

[27] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Neural Autonomous Navigation with Riemannian Motion Policy," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8860–8866, IEEE, Apr. 2019.

[28] M. Allenspach, M. Pantic, R. Girod, L. Ott, and R. Siegwart, "Task Adaptation in Industrial Human-Robot Interaction: Leveraging Riemannian Motion Policies," *Robotics: Science and System XX*, p. 026, June 2024.

[29] M. Mukadam, C.-A. Cheng, D. Fox, B. Boots, and N. D. Ratliff, "Riemannian Motion Policy Fusion through Learnable Lyapunov Function Reshaping," in *Conference on robot learning*, pp. 204–219, PMLR, 2020.

[30] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, B. Boots, and N. D. Ratliff, "Geometric Fabrics: Generalizing Classical Mechanics to Capture the Physics of Behavior," *IEEE Robotics and Automation Letters*, vol. 7, pp. 3202–3209, Apr. 2022.

[31] L. Xu, X. Xiong, and Y. Lou, "DCBF-based Trajectory Planning for Mobile Manipulators in Complex and Dynamic Work Environments," in *2024 18th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 556–561, Dec. 2024.

[32] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100–107, July 1968.

[33] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 3310–3317 vol.4, May 1994.

[34] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *The International Journal of Robotics Research*, vol. 29, pp. 485–501, Apr. 2010.

[35] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, pp. 995–1001, Apr. 2000.

[36] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, May 2011.

[37] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, Sept. 2014.

[38] R. Bohlin and L. Kavraki, "Path planning using lazy PRM," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 1, pp. 521–528, Apr. 2000.

[39] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, "LEGO: Leveraging Experience in Roadmap Generation for Sampling-Based Planning," in *2019 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pp. 1488–1495, Nov. 2019.

[40] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, "HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots," *IEEE Access*, vol. 8, pp. 221743–221766, 2020.

[41] G. Chen, N. Luo, D. Liu, Z. Zhao, and C. Liang, "Path planning for manipulators based on an improved probabilistic roadmap method," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102196, Dec. 2021.

[42] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8749–8756, 2024.

[43] N. Perrault, Q. H. Ho, and M. Lahijanian, "Kino-PAX: Highly Parallel Kinodynamic Sampling-Based Planner," *IEEE Robotics and Automation Letters*, vol. 10, pp. 2430–2437, Mar. 2025.

[44] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-Based Motion Planning: A Comparative Review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, pp. 285–310, July 2024.

[45] D. Mayne, "A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems," *International Journal of Control*, vol. 3, pp. 85–95, Jan. 1966.

[46] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *First International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp. 222–229, SciTePress, 2004.

[47] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, pp. 344–357, Feb. 2017.

[48] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *2009 IEEE International Conference on Robotics and Automation*, pp. 489–494, May 2009.

[49] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs," in *Robotics: Science and Systems XII*, Robotics: Science and Systems Foundation, 2016.

[50] E. Huang, M. Mukadam, Z. Liu, and B. Boots, "Motion planning with graph-based trajectories and Gaussian process inference," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5591–5598, May 2017.

[51] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method*. Information Science and Statistics, New York, NY: Springer, 2004.

[52] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stückler, M. Rolínek, and G. Martius, "Sample-efficient Cross-Entropy Method for Real-time Planning," in *Proceedings of the 2020 Conference on Robot Learning*, pp. 1049–1065, PMLR, Oct. 2021.

[53] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, "Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo," *arXiv preprint arXiv:2212.00541*, Dec. 2022.

[54] M. Bhardwaj, B. Sundaralingam, A. Mousavian, N. Ratliff, D. Fox, F. Ramos, and B. Boots, "STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation," in *Conference on Robot Learning*, pp. 750–759, PMLR, Sept. 2021.

[55] J. Jankowski, L. Brudermüller, N. Hawes, and S. Calinon, "VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, (London, United Kingdom), pp. 10125–10131, IEEE, May 2023.

[56] F. Rastgar, H. Masnavi, B. Sharma, A. Aabloo, J. Swevers, and A. K. Singh, "PRIEST: Projection Guided Sampling-Based Optimization for Autonomous Navigation," *IEEE Robotics and Automation Letters*, vol. 9, pp. 2630–2637, Mar. 2024.

[57] H. Klein, N. Jaquier, A. Meixner, and T. Asfour, "On the Design of Region-Avoiding Metrics for Collision-Safe Motion Generation on Riemannian Manifolds," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2346–2353, Oct. 2023.

[58] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pp. 802–807 vol.2, May 1993.

[59] L. Sentis and O. Khatib, "A whole-body control framework for humanoids operating in human environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, (Orlando, FL, USA), pp. 2641–2648, IEEE, 2006.

[60] V. Vasilopoulos, S. Garg, P. Piacenza, J. Huh, and V. Isler, "RAMP: Hierarchical Reactive Motion Planning for Manipulation Tasks Using Implicit Signed Distance Functions," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10551–10558, IEEE, July 2023.

[61] T. Zhu, J. Mao, L. Han, C. Zhang, and J. Yang, "Real-Time Dynamic Obstacle Avoidance for Robot Manipulators Based on Cascaded Nonlinear MPC With Artificial Potential Field," *IEEE Transactions on Industrial Electronics*, vol. 71, pp. 7424–7434, July 2024.

[62] A. Kuntz, C. Bowen, and R. Alterovitz, "Interleaving Optimization with Sampling-Based Motion Planning (IOS-MP): Combining Local Optimization with Global Exploration," *arXiv preprint arXiv:1607.06374*, Sept. 2016.

[63] J. Kamat, J. Ortiz-Haro, M. Toussaint, F. T. Pokorny, and A. Orthey, "BITKOMO: Combining Sampling and Optimization for Fast Convergence in Optimal Motion Planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4492–4497, Oct. 2022.

[64] M. Toussaint, "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning," in *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1930–1934, 2015.

[65] R. Natarajan, S. Mukherjee, H. Choset, and M. Likhachev, "PINSAT: Parallelized Interleaving of Graph Search and Trajectory Optimization for Kinodynamic Motion Planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13907–13914, IEEE, Mar. 2024.

[66] Y.-H. Chen, Y.-Y. Lai, and T.-H. S. Li, "Collision-free Motion Policy for Manipulator with Multi-constraint Coordination in Dynamic Environment," in *2024 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*, pp. 1–8, Aug. 2024.

[67] K. Hansel, J. Urain, J. Peters, and G. Chalvatzaki, "Hierarchical Policy Blending as Inference for Reactive Robot Control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10181–10188, IEEE, 2023.

[68] A. T. Le, K. Hansel, J. Peters, and G. Chalvatzaki, "Hierarchical Policy Blending As Optimal Transport," in *Learning for Dynamics and Control Conference*, pp. 797–812, PMLR, 2023.

[69] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5628–5635, May 2011.

[70] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-PRM: Temporal Probabilistic Roadmap for Path Planning in Dynamic Environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Kyoto, Japan), pp. 10320–10327, IEEE, Oct. 2022.

[71] M. Gentner, F. Zillenbiller, A. Kraft, and E. Steinbach, "Dynamic Multi-Query Motion Planning with Differential Constraints and Moving Goals," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 205–212, Oct. 2023.

[72] M. Otte and E. Frazzoli, "RRT^{X}: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, pp. 797–822, June 2016.

[73] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2421–2428, May 2013.

[74] L. Liao, Q. Xu, X. Zhou, X. Li, and X. Liu, "Bi-HS-RRT^{X}: an efficient sampling-based motion planning algorithm for unknown dynamic environments," *Complex & Intelligent Systems*, vol. 10, pp. 7497–7512, Dec. 2024.

[75] J. Qi, H. Yang, and H. Sun, "MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment," *IEEE Transactions on Industrial Electronics*, vol. 68, pp. 7244–7251, Aug. 2021.

[76] C. Yuan, C. Shuai, and W. Zhang, "A Dynamic Multiple-Query RRT Planning Algorithm for Manipulator Obstacle Avoidance," *Applied Sciences*, vol. 13, p. 3394, Jan. 2023.

[77] K. Kolur, S. Chintalapudi, B. Boots, and M. Mukadam, "Online Motion Planning Over Multiple Homotopy Classes with Gaussian Process Inference," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2358–2364, IEEE, arXiv, Aug. 2019.

[78] K. V. Alwala and M. Mukadam, "Joint Sampling and Trajectory Optimization over Graphs for Online Motion Planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4700–4707, Sept. 2021.

[79] Y. Yang and O. Brock, "Elastic roadmaps—motion generation for autonomous mobile manipulation," *Autonomous Robots*, vol. 28, pp. 113–130, Jan. 2010.

[80] P. Lehner, A. Sieverling, and O. Brock, "Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, (Seattle, WA, USA), pp. 4761–4767, IEEE, May 2015.

[81] H. Schättler and U. Ledzewicz, "The Pontryagin Maximum Principle: From Necessary Conditions to the Construction of an Optimal Solution," in *Geometric Optimal Control: Theory, Methods and Examples* (H. Schättler and U. Ledzewicz, eds.), pp. 83–194, New York, NY: Springer, 2012.

[82] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, pp. 560–570, Oct. 1979.

[83] Nvidia, "Rmpflow — isaac sim documentation." `https://docs.isaacsim.omniverse.nvidia.com/latest/manipulators/concepts/rmpflow.html#target-rmp-target-rmp`. Accessed: April 7, 2025.

[84] P. M. Hubbard, "Approximating polyhedra with spheres for time-critical collision detection," *ACM Trans. Graph.*, vol. 15, pp. 179–210, July 1996.

[85] C. Liu, S. Xie, X. Sui, Y. Huang, X. Ma, N. Guo, and F. Yang, "PRM-D* Method for Mobile Robot Path Planning," *Sensors*, vol. 23, p. 3512, Jan. 2023.

[86] D. Nieuwenhuisen and M. Overmars, "Useful cycles in probabilistic roadmap graphs," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 446–452, Apr. 2004.

[87] M. T. Rantanen and M. Juhola, "Using probabilistic roadmaps in changing environments," *Computer Animation and Virtual Worlds*, vol. 25, no. 1, pp. 17–31, 2014.

[88] D. P. Bertsekas, *Dynamic programming: deterministic and stochastic models*. USA: Prentice-Hall, Inc., 1987.

[89] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. Adaptive computation and machine learning series, Cambridge, Massachusetts: The MIT Press, second ed., 2018.

[90] L. Baird, "Reinforcement learning in continuous time: advantage updating," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 4, pp. 2448–2453, June 1994.

[91] F. Herrmann, S. Zach, J. Banfi, J. Peters, G. Chalvatzaki, and D. Tateo, "Safe and Efficient Path Planning Under Uncertainty via Deep Collision Probability Fields," *IEEE Robotics and Automation Letters*, vol. 9, pp. 9327–9334, Nov. 2024.

[92] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ch. 4. MIT Press, 2016. `http://www.deeplearningbook.org`.

[93] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: High performance array computing — JAX documentation." `https://docs.jax.dev/en/latest/`. Accessed: April 12, 2025.

[94] OpenXLA, "OpenXLA Project." `https://openxla.org/xla`. Accessed: March 7, 2025.

[95] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem, "Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, arXiv, June 2021.

[96] C. Tsitouras, "Runge-Kutta pairs of order 5(4) satisfying only the first column simplifying assumption," *Comput. Math. Appl.*, vol. 62, pp. 770–775, July 2011.

[97] P. Kidger, *On Neural Differential Equations*. PhD thesis, University of Oxford, Feb. 2021.

[98] T. Fraichard and H. Asama, "Inevitable collision states. A step towards safer robots?," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, pp. 388–393, Oct. 2003.

A. Comprehensive Results

To provide more details about the results that were visualized in Chapter 6, we present more comprehensive experimental data in the following tables. We report success rates and collision rates as percentages of trials that succeeded or ended in collision respectively, as well as path lengths and time to goal with their respective Standard Deviations (SDs).

Table A.1 and Table A.3 show the results for the experiments with static obstacles, for the toy example and the robot environment. Table A.2 and Table A.4 equivalently show the results in the dynamic environments.

The complementary data for the comparison of update rates is shown in Table A.5. Higher rates indicate more environment steps being taken in between each update.

Scene	Method	Success Rate (%)	Collision Rate (%)	Path Length	First Goal (steps)
HShapedBox2D	RMP Baseline	0.00	**0.00**	N/A	N/A
	HIPBI Baseline	0.00	**0.00**	N/A	N/A
	GPMP Baseline	0.00	**0.00**	N/A	N/A
	Ours (w/ Goal Node w/o Goal Prior)	**100.00**	**0.00**	**73.47 ± 1.49**	**191.04 ± 17.87**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	**100.00**	**0.00**	77.12 ± 3.87	230.74 ± 67.30
Maze2D	RMP Baseline	26.00	**0.00**	52.62 ± 11.34	352.96 ± 90.34
	HIPBI Baseline	74.00	14.00	116.45 ± 44.74	310.12 ± 115.17
	GPMP Baseline	82.00	3.00	56.77 ± 18.95	325.20 ± 148.17
	Ours (w/ Goal Node w/o Goal Prior)	**97.00**	**0.00**	**47.12 ± 5.45**	**96.47 ± 23.73**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	**97.00**	**0.00**	50.29 ± 6.87	139.97 ± 54.44
MultiGoal2D	RMP Baseline	16.00	**0.00**	**38.07 ± 6.35**	302.62 ± 125.95
	HIPBI Baseline	44.00	15.00	143.15 ± 78.11	463.30 ± 267.27
	GPMP Baseline	87.00	4.00	51.12 ± 16.81	301.05 ± 123.11
	Ours (w/ Goal Node w/o Goal Prior)	96.00	1.00	38.16 ± 7.56	**82.77 ± 35.05**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	**97.00**	**0.00**	43.45 ± 13.66	106.51 ± 66.97
OpenBox2D	RMP Baseline	6.00	**0.00**	**21.14 ± 4.02**	135.17 ± 30.76
	HIPBI Baseline	**100.00**	**0.00**	98.88 ± 40.81	257.46 ± 106.60
	GPMP Baseline	**100.00**	**0.00**	41.61 ± 13.19	239.39 ± 74.35
	Ours (w/ Goal Node w/o Goal Prior)	**100.00**	**0.00**	32.12 ± 8.39	**71.66 ± 17.90**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	**100.00**	**0.00**	39.05 ± 12.21	210.35 ± 168.43
MultiGate2D	RMP Baseline	1.00	**5.00**	76.96 ± 0.00	367.00 ± 0.00
	HIPBI Baseline	32.00	**5.00**	190.15 ± 54.58	663.09 ± 187.85
	GPMP Baseline	78.00	6.00	72.38 ± 13.54	452.27 ± 118.04
	Ours (w/ Goal Node w/o Goal Prior)	**92.00**	**5.00**	**62.00 ± 7.23**	**164.43 ± 41.91**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	84.00	**5.00**	67.51 ± 11.15	264.83 ± 152.54
OpenBox2DMultigoal	RMP Baseline	0.00	**0.00**	N/A	N/A
	HIPBI Baseline	31.00	**0.00**	138.12 ± 41.79	438.87 ± 148.79
	GPMP Baseline	85.00	**0.00**	69.48 ± 18.93	409.80 ± 127.62
	Ours (w/ Goal Node w/o Goal Prior)	**99.00**	**0.00**	**50.62 ± 8.32**	**115.92 ± 44.66**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	95.00	**0.00**	53.90 ± 8.52	163.34 ± 127.12
SpiralMaze2D	RMP Baseline	0.00	0.00	N/A	N/A
	HIPBI Baseline	0.00	20.83	N/A	N/A
	GPMP Baseline	0.00	6.25	N/A	N/A
	Ours (w/ Goal Node w/o Goal Prior)	**21.00**	2.00	**143.60 ± 16.93**	**560.43 ± 142.93**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1$ s)	12.00	**1.00**	145.80 ± 14.31	610.50 ± 167.13

Table A.1.: Performance comparison of motion planning methods in 2D static obstacle environments, showing success rate (%), collision rate (%), path length (mean ± SD meters), and time-to-goal (mean ± SD steps). Our method is evaluated with and without explicit goal nodes. Bold indicates best performance per metric within each environment ($N = 100$ trials per Scene).

Scene	Method	Success Rate (%)	Collision Rate (%)	Path Length	First Goal (steps)
Maze	RMP Baseline	90.00	**0.00**	**1.14 ± 0.15**	598.63 ± 161.25
	Ours (w/ Goal Node w/ Goal Prior	96.67	**0.00**	1.39 ± 0.24	**141.76 ± 64.92**
	Ours (w/ Goal Node w/o Goal Prior	**100.00**	**0.00**	1.41 ± 0.24	159.14 ± 87.40
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0\,\mathrm{s}$)	80.00	**0.00**	1.25 ± 0.20	704.58 ± 126.45
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0.1\,\mathrm{s}$)	73.33	**0.00**	1.91 ± 0.47	538.45 ± 202.68
	Ours (w/o Goal Node w/o Goal Prior $T_r = 0.1\,\mathrm{s}$)	13.33	**0.00**	2.23 ± 0.53	429.00 ± 65.64
Window	RMP Baseline	0.00	**0.00**	N/A	N/A
	Ours (w/ Goal Node w/ Goal Prior	63.33	**0.00**	**1.19 ± 0.11**	513.95 ± 154.93
	Ours (w/ Goal Node w/o Goal Prior	46.67	**0.00**	1.24 ± 0.16	**410.14 ± 138.61**
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0\,\mathrm{s}$)	43.33	**0.00**	1.70 ± 0.31	581.92 ± 205.76
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0.1\,\mathrm{s}$)	**80.00**	**0.00**	1.43 ± 0.35	736.46 ± 175.17
	Ours (w/o Goal Node w/o Goal Prior $T_r = 0.1\,\mathrm{s}$)	3.33	**0.00**	1.72 ± 0.00	842.00 ± 0.00

Table A.2.: Ablation study of our motion planning method and comparison with RMP baseline for the Franka robot in static obstacle environments. Results compare variations of our approach with different configurations: with and without goal nodes added to the roadmap, with and without goal-biased sampling prior, and with different time horizons (T_r) for evaluating node-to-goal connectivity. Performance metrics include success rates (%), collision rates (%), path lengths (radians, mean ± SD), and time to goal (steps, mean ± SD) across Maze and Window environments ($N = 30$ trials per configuration).

Scene	Method	Success Rate (%)	Collision Rate (%)	Path Length	First Goal (steps)
HShapedBox2D	RMP Baseline	0.00	**0.00**	N/A	N/A
	HIPBI Baseline	0.00	24.00	N/A	N/A
	GPMP Baseline	2.00	20.00	182.66 ± 3.48	856.00 ± 53.00
	Ours (w/ Goal Node)	**75.00**	20.00	122.78 ± 56.75	**306.81 ± 167.29**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	79.00	18.00	**122.40 ± 36.73**	389.91 ± 174.30
Maze2D	RMP Baseline	68.00	32.00	70.34 ± 18.81	362.96 ± 98.88
	HIPBI Baseline	61.00	39.00	130.97 ± 34.82	336.46 ± 84.29
	GPMP Baseline	39.00	53.00	127.97 ± 37.34	637.95 ± 179.27
	Ours (w/ Goal Node)	**89.00**	**11.00**	**54.96 ± 10.28**	**119.27 ± 35.80**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	**89.00**	**11.00**	60.38 ± 14.31	166.96 ± 83.46
MultiGoal2D	RMP Baseline	64.00	34.00	69.43 ± 28.19	406.02 ± 188.14
	HIPBI Baseline	49.00	39.00	148.22 ± 57.39	467.22 ± 203.05
	GPMP Baseline	51.00	43.00	122.84 ± 36.37	619.59 ± 177.78
	Ours (w/ Goal Node)	**86.00**	**14.00**	**44.05 ± 10.53**	**95.27 ± 31.28**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	**86.00**	**14.00**	49.00 ± 13.70	129.31 ± 71.09
OpenBox2D	RMP Baseline	4.00	1.00	**33.91 ± 13.12**	212.00 ± 56.44
	HIPBI Baseline	91.00	5.00	152.96 ± 76.99	392.67 ± 191.13
	GPMP Baseline	48.00	5.00	88.39 ± 37.82	500.02 ± 208.40
	Ours (w/ Goal Node)	**100.00**	**0.00**	48.07 ± 15.33	**137.48 ± 66.36**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	98.00	**0.00**	56.81 ± 21.29	231.11 ± 133.37
MultiGate2D	RMP Baseline	11.00	22.00	111.20 ± 22.57	638.18 ± 175.01
	HIPBI Baseline	31.00	15.00	174.10 ± 44.62	659.26 ± 189.10
	GPMP Baseline	51.00	24.00	102.73 ± 22.03	592.16 ± 140.05
	Ours (w/ Goal Node)	**93.00**	**5.00**	63.57 ± 12.70	**154.92 ± 93.51**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	85.00	7.00	**70.38 ± 14.94**	251.91 ± 172.33
OpenBox2DMultigoal	RMP Baseline	0.00	15.00	N/A	N/A
	HIPBI Baseline	31.00	27.00	185.65 ± 41.80	587.16 ± 169.14
	GPMP Baseline	31.00	59.00	111.51 ± 32.66	600.94 ± 168.51
	Ours (w/ Goal Node)	**94.00**	**2.00**	**70.99 ± 26.99**	**168.53 ± 83.10**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	91.00	5.00	71.72 ± 24.33	209.05 ± 151.51
SpiralMaze2D	RMP Baseline	0.00	56.25	N/A	N/A
	HIPBI Baseline	0.00	68.75	N/A	N/A
	GPMP Baseline	0.00	77.08	N/A	N/A
	Ours (w/ Goal Node)	**53.00**	39.00	**163.00 ± 25.19**	**650.18 ± 139.67**
	Ours (w/o Goal Node w/o Goal Prior $T_r = 1s$)	35.00	**33.00**	169.93 ± 20.43	754.91 ± 150.50

Table A.3.: Performance comparison of motion planning methods in 2D moving obstacle environments, showing success rate (%), collision rate (%), path length (mean ± SD meters), and time-to-goal (mean ± SD steps). Our method is evaluated with and without explicit goal nodes. Bold indicates best performance per metric within each environment ($N = 100$ trials per Scene).

Scene	Method	Success Rate (%)	Collision Rate (%)	Path Length	First Goal (steps)
Maze	RMP Baseline	73.33	16.67	**1.15 ± 0.18**	573.50 ± 150.83
	Ours (w/ Goal Node w/ Goal Prior)	**100.00**	**0.00**	1.40 ± 0.24	**132.57 ± 48.32**
	Ours (w/ Goal Node w/o Goal Prior)	**100.00**	**0.00**	1.41 ± 0.26	154.40 ± 111.60
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0s$)	30.00	36.67	1.81 ± 0.60	516.67 ± 220.18
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0.1s$)	40.00	23.33	1.82 ± 0.48	473.42 ± 228.43
	Ours (w/o Goal Node w/o Goal Prior $T_r = 0.1s$)	10.00	**0.00**	2.30 ± 0.49	486.33 ± 113.54
Window	RMP Baseline	0.00	**0.00**	N/A	N/A
	Ours (w/ Goal Node w/ Goal Prior)	**90.00**	**0.00**	**1.85 ± 0.46**	**502.33 ± 154.95**
	Ours (w/ Goal Node w/o Goal Prior)	80.00	6.67	1.94 ± 0.55	511.67 ± 171.47
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0s$)	60.00	**0.00**	2.12 ± 0.46	793.11 ± 158.35
	Ours (w/o Goal Node w/ Goal Prior $T_r = 0.1s$)	70.00	**0.00**	2.15 ± 0.31	774.57 ± 100.66
	Ours (w/o Goal Node w/o Goal Prior $T_r = 0.1s$)	0.00	3.33	N/A	N/A

Table A.4.: Ablation study of our motion planning method and comparison with RMP baseline for the Franka robot in moving obstacle environments. Results compare variations of our approach with different configurations: with and without goal nodes added to the roadmap, with and without goal-biased sampling prior, and with different time horizons (T_r) for evaluating node-to-goal connectivity. Performance metrics include success rates (%), collision rates (%), path lengths (radians, mean ± SD), and time to goal (steps, mean ± SD) across Maze and Window environments ($N = 30$ trials per configuration).

PRM Update Rate	Stein Update Rate	Success Rate (%)	Collision Rate (%)	Path Length	First Goal (steps)
1	1	44.00	4.00	97.87 ± 33.06	505.05 ± 246.04
5	1	44.00	8.00	98.16 ± 35.59	478.91 ± 234.87
	5	28.00	6.00	86.72 ± 30.70	379.57 ± 217.17
10	1	48.00	6.00	105.75 ± 36.03	534.25 ± 236.93
	5	32.00	2.00	95.05 ± 36.01	415.44 ± 214.87
	10	26.00	4.00	91.36 ± 31.23	426.85 ± 208.68
25	1	42.00	6.00	100.40 ± 37.77	520.67 ± 235.59
	5	34.00	2.00	100.82 ± 31.79	518.06 ± 249.47
	10	28.00	2.00	91.43 ± 29.81	467.00 ± 222.45
	25	32.00	2.00	100.31 ± 36.19	521.69 ± 232.31
50	1	38.00	12.00	99.81 ± 39.84	545.11 ± 271.02
	5	34.00	4.00	88.99 ± 32.02	484.06 ± 280.64
	10	30.00	4.00	91.65 ± 29.38	505.33 ± 239.96
	25	30.00	8.00	93.82 ± 28.55	546.13 ± 249.93
	50	26.00	2.00	85.99 ± 30.29	492.38 ± 234.50
100	1	36.00	4.00	93.09 ± 27.46	539.50 ± 224.63
	5	32.00	10.00	86.99 ± 32.23	534.75 ± 293.04
	10	24.00	6.00	87.20 ± 25.52	470.42 ± 167.76
	25	28.00	6.00	82.66 ± 23.41	484.64 ± 237.71
	50	28.00	4.00	89.94 ± 34.16	528.64 ± 279.24
	100	24.00	4.00	92.41 ± 28.35	535.92 ± 241.11
200	1	32.00	4.00	94.12 ± 27.50	567.56 ± 253.15
	5	34.00	4.00	90.01 ± 25.91	580.76 ± 272.56
	10	24.00	8.00	83.88 ± 27.39	446.67 ± 235.97
	25	24.00	14.00	87.41 ± 23.17	570.92 ± 252.82
	50	24.00	2.00	88.86 ± 21.09	556.75 ± 235.31
	100	26.00	2.00	96.17 ± 28.22	577.85 ± 198.70
	200	24.00	4.00	97.53 ± 32.84	560.58 ± 188.76
300	1	28.00	6.00	93.14 ± 37.14	556.79 ± 261.28
	5	34.00	2.00	88.32 ± 27.95	592.65 ± 266.85
	10	24.00	2.00	86.63 ± 33.80	498.42 ± 253.28
	25	20.00	6.00	90.88 ± 27.22	551.70 ± 242.17
	50	20.00	2.00	83.59 ± 18.77	543.70 ± 253.71
	100	20.00	4.00	91.42 ± 20.83	585.40 ± 224.03
	200	18.00	0.00	96.16 ± 31.65	565.22 ± 246.18
	300	14.00	2.00	96.25 ± 33.18	495.86 ± 254.79

Table A.5.: Comparison of update rates. Dynamic Obstacles ($N = 50$)

B. Policies

Table B.1 presents the key parameters governing the Riemannian Motion Policies. We supplemented the policies described in Section 3.1 with a configuration space dampening policy defined as $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = -\eta\, \dot{\mathbf{x}},\ \mathbf{M} = \mathbf{I}$.

While dampening is essential to mitigate overshooting, implementing it solely within the target attraction policy induced substantial curling behaviour. Redistributing some dampening to the multi-dimensional configuration space policy effectively reduced this curling effect. Furthermore, we implemented the attraction policies in multi-dimensional spaces rather than using scalar distance measures, as the latter approach generated undesirable oscillations perpendicular to the goal-directed vector.

Policy Type	Parameter	Panda Robot		2D Environment	
		Reaching Target	PRM Policy	Reaching Target	PRM Policy
Attraction Policies	k	25.0	25.0	3.0	1.0
	η	50.0	0.63	2.0	0.63
	σ	0.05	0.2	4.0	10.0
	α	64.0	64.0	1.0	1.0
	w_u	5.0	3.0	30.0	30.0
	w_l	0.0	2.0	0.0	15.0
		Panda Robot		2D Environment	
		Obstacle Avoidance	Self-Collision and Joint-Limits	Obstacle Avoidance	
Repulsion Policies	safety margin	0.01	0.03	1.2	
	k_p	1500	500	25	
	ℓ_p	0.05	0.05	20	
	v_d	0.25	0.25	1.0	
	k_d	1.0	1.0	25	
	ϵ_d	5.0	5.0	1.0	
	ℓ_d	0.01	0.01	0.25	
	μ	7000	5000	100	
	ϵ_m	100	100	0.001	
	ℓ_m	0.005	0.005	0.1	
	r	0.02	0.05	2.5	
Dampening	η	1.0		5.0	

Table B.1.: Comparison of key policy parameters between the Panda robot and 2D environments. See Section 3.1 for their role in the specific policies. The gain parameter k is an additional scaling of the gradient for the attraction policies and safety margin refers to a constant shift in the space of the repulsion policies, so the distance x is always a certain amount smaller. The dampening policy is simply $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) = -\eta\, \dot{\mathbf{x}},\ \mathbf{M} = \mathbf{I}$ in the configuration space $\mathcal{C}$.