Safe and Efficient Motion Planning in Robotic Manipulation through Formal Methods

by

Mingxin Yu

B.S. Physics, Peking University, 2022

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Mingxin Yu Department of Aeronautics and Astronautics May 10, 2024
Certified by:	Chuchu Fan Wilson Assistant Professor of Aeronautics and Astronautics, Thesis Supervisor
Accepted by:	Jonathan P. How R. C. Maclaurin Professor of Aeronautics and Astronautics Chair, Graduate Program Committee

Safe and Efficient Motion Planning in Robotic Manipulation through Formal Methods

by

Mingxin Yu

Submitted to the Department of Aeronautics and Astronautics
on May 10, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

ABSTRACT

Manipulating rigid body objects in crowded environments poses significant challenges due to the need for rapid, real-time planning and the assurance of safe operational paths. The challenges come from varying shapes of the manipulated objects and high-dimensional nature of manipulators.

This thesis addresses these issues by developing (1) a mixed-integer linear programming (MILP)-based approach to plan safe paths for rigid-body objects; and (2) a learned control barrier function (CBF) tailored for manipulators with multiple degrees of freedom (DoF) and an associated framework CBF-RRT to enable efficient planning for robotic manipulators. Comprehensive experimental results have shown that the proposed methods outperform baseline methods, providing tools for improving the safety and efficiency of robotic manipulators in complex environments.

Thesis supervisor: Chuchu Fan

Title: Wilson Assistant Professor of Aeronautics and Astronautics

Acknowledgments

I would like to start by expressing my gratitude to my advisor, Prof. Chuchu Fan, for her remarkable mentorship and invaluable support throughout the past two years. Chuchu provided me with the freedom to explore research directions and whenver I feel lost, she is always there with me. I would also like to thank my undergraduate mentors, especially Prof. Lin Shao, Dr. Jie Fu and Prof. Hao Dong. Thank you for all the patience, support and guidance you have given to me during my gap year and beyond. Your encouragement reignited my passion for research, and I would not be where I am today without your influence and inspiration.

I would like to thank my wonderful labmates from REALM lab. Thank you for sharing both tears and joys, and for all the inspirational research discussion and group activities that have enriched our collective journey. I would also thank my amazing collaborators. The invaluable feedback and contributions have been instrumental during the research journey.

More broadly, I would like to thanks my friends around Boston, back in China and all over the world despite our different time zones. Thanks for always being with me and offering the accompanion and support through ups and downs. Those late-night calls, walks, random discussions, delicious food and trips we enjoyed together will always be unforgettable memories of my life.

Last but definitely not least, I would like to extend my deepest gratitude to my parents. Thank you for shaping who I am and continuously respecting and supporting my decisions, granting me the freedom to explore my own journey. Your unconditional love and unwavering

support have been my anchor, even though miles may lie between us.

Contents

Title page **1**

Abstract **3**

Acknowledgments **5**

List of Figures **11**

List of Tables **15**

1 Introduction **17**

1.1 Planning Safe Paths in Robotic Manipulation 17

1.1.1 Rigid-body Objects 18

1.1.2 Multi-DoF Manipulators 19

1.2 Contributions 20

1.3 Thesis Outline 22

2 Related Work **23**

3 Planning Paths for Rigid Objects **27**

3.1 Planning Pipeline for Point Objects 27

3.1.1 Construct Coarse Graph G_c 28

3.1.2 Construct Dense Graph G_d 29

3.1.3 Online Query 30
3.1.4 Discussion 31
3.2 From Point Objects to Polytope Objects 31
3.2.1 MILP Formulation 32
3.2.2 Collision-Avoidance Constraint 34
3.2.3 Revisit Vertices $\mathcal{V}_d$ 40
3.3 Experiments 40
3.3.1 Planning Results 41
3.3.2 Comparison with Sampling-based Algorithms 43
3.3.3 Comparison with GCS 43
3.4 Conclusion 47

4 Planning for Multi-DoF Manipulators **49**
4.1 Problem Statement and Preliminaries 49
4.2 Learning Control Barrier Function for Manipulators 51
4.2.1 Learning Framework for CBF-INF 51
4.2.2 Specializing Functions for Manipulators: State-based and LiDAR-based CBF-INF 52
4.3 Planning with Learned CBF 54
4.3.1 Synthesize controller. 55
4.3.2 Safe-steering RRT. 55
4.4 Experiments 56
4.4.1 Motion Planning in Simulation 57
4.4.2 Ablation Study in Simulation 59
4.4.3 Hardware Demonstration 60
4.5 Conclusion 62

5 Conclusion and Future Work **63**

References **65**

List of Figures

3.1 An overview of our pipeline. (a) Decomposition of the free workspace into a set of convex polytopes. The polytopes serve as graph vertices and are interconnected if they overlap G_c (Section 3.1.1). (b) Construction of G_d(Section 3.1.2), where each vertex is a set of free configurations - illustrated as the set of green points enclosed by the green ring - positioned on the boundary (blue ring) of intersections of polytopes(Section 3.1.2). We verify all the candidate edges (gray) via MILPs, and the valid edges (blue) demonstrate the existence of path segments between vertices (**??**). (c) Online query stage: the start and end configurations are connected to graph G_d and the planned path is retrieved(Section 3.1.3). 28

3.2 Reachable set (yellow region) of 2D object. 34

3.3 Models of collision avoidance in environments with convex polytopes (blue and yellow) and obstacles (gray). Green lines indicate collision-free paths, and red lines indicate detected collisions. *Left:* This model only checks endpoints of each line segment. The dashed red line demonstrates that endpoint-only checking is insufficient as it cuts through an obstacle. *Right* [4]: This overly restrictive model allows only paths like the dotted line where both endpoints lie within the same free region, rejecting the simpler, viable solid line. *Middle (ours):* modeled by Proposition 1. This approach prevents the dashed line from cutting through obstacles while including more direct paths between regions, like the solid line. 35

3.4 *Left:* all the vertices are contained in the same green polytope. *Right:* one vertex (C) is not contained in the same polytope as vertices A, B. 36

3.5 *Upper left:* all the vertices are contained in the same green polytope. *Upper right:* vertex C is not contained in the same polytope as the other vertices (A, B, D) *Lower left:* two adjacent vertices (C, D) are contained in a different polytope as A, B. *Lower right:* vertices along the same diagonal (AC and BD) are contained in a same polytope. 37

3.6 Visualization of planning results in 2D (obstacles: pink) and 3D scenarios with convex objects. The start and end configurations are ploted with red, while waypoints and their corresponding reachable sets are shown in green. In 2D scenarios, we also illustrate the graph G_c with vertices ($\mathcal{V}_c$) as blue polytopes and edges ($\mathcal{E}_c$) as blue lines connecting red dots, which represent the centers of vertex polytopes. 43

3.7 Visualization of planning results in 2D (obstacles: pink) and 3D scenarios with L-shaped non-convex objects. The start and end configurations are ploted with red, while waypoints and their corresponding reachable sets are shown in green. In 2D scenarios, we also illustrate the graph G_c with vertices ($\mathcal{V}_c$) as blue polytopes and edges ($\mathcal{E}_c$) as blue lines connecting red dots, which represent the centers of vertex polytopes. 44

3.8 A 2D motion planning problem consists of a stick getting out of a trap and passing through a narrow gap, where our MILP-based method is able to achieve almost 100% coverage in workspace Fig. 3.6b. The C-space is a subset of $SE(2)$, with two translation axes and a periodic axis corresponding to the rotation. We visualize the collision-free C-space by sampling (left) and the C-IRIS cover acquired from GCS (right). 44

4.1 Illustration of observations. Left: sCBF-INF takes the signed distance to the nearest obstacle as observation. Middle: For *fully-observable* environments, oCBF-INF uses a point cloud sampled uniformly on the obstacles as observation. Right: For *partially-observable* environments, oCBF-INF observes the point cloud from a mounted LiDAR sensor. 53

4.2 The overall neural network architecture. Left: The architecture of the sCBF-INF. Right: The architecture of the oCBF-INF. 54

4.3 Motion planning experiments under LiDAR-based setting. 59

4.4 Left: Slices of oCBF-INF value for Panda, obtained by sweeping across two joints, with all other joint states and obstacle positions held constant. Right: Learning curves of constraint satisfaction rate on Panda. 59

4.5 Safety and goal-reaching performance of LiDAR-based controllers in an end-to-end manner (without motion planning module). 61

4.6 Left & middle: snapshots of solving a motion planning problem with our method. Right: the curve of minimum signed distance to all obstacles along a trajectory. Videos are included in the supplementary. 61

List of Tables

3.1 Comparison of our MILP-based method to baselines in all benchmark scenarios. 45

3.2 Comparison of our method with PRM and RRT for an L-shaped object navigating a bugtrap environment Fig. 3.6b. Online time is evaluated in both large and narrow environments. In large settings, the mid-right corridor remains constant while dimensions are doubled. In narrow settings, the corridor width is reduced to 60% of its original size. The online time limit for PRM is 40s. 45

3.3 Metrics of offline stages for our MILP-based method and GCS. 47

4.1 Experiments under state-based setting. Performance on average success rate (SR), number of explored nodes on 1000 test cases, and summed planning time on 100 testing cases, averaged over 3 random seeds. 58

Chapter 1

Introduction

Robotic manipulators are becoming increasingly prevalent in our lives, ranging from stacking packages in the warehouse, assembling products on manufacturing lines to performing everyday tasks in household environments. Despite the increasing popularity of robotic manipulators in these real-world scenarios, the need for rapid, real-time planning and guaranteeing safe execution of the operational paths still presents significant challenges, particularly in densely crowded environments. The challenges come from varying shapes of the manipulated objects and high-dimensional nature of manipulators. In this chapter, we will discuss the specific problems and challenges this thesis addresses, highlight our contributions, and provide an outline of the thesis.

1.1 Planning Safe Paths in Robotic Manipulation

Safe path planning in robotic manipulation must consider both the paths of the manipulated objects and the manipulator. However, the challenges associated with these two components are distinct, and this thesis addresses them separately.

1.1.1 Rigid-body Objects

The first challenge addressed in this thesis is planning a path for manipulated objects of various shapes, which is tightly related to *Piano Mover's Problem*. The *Piano Mover's Problem* [1] asks whether one can find a sequence of rigid body motions from a given initial position to a desired final position, subject to certain geometric constraints during the motion. The difficulty over planning for point objects comes from the dimension of the objects, which prevents the feasibility of naively tracking the motion of a points, especially when narrow corridors are presented in the environment and a followed higher-dimension configuration space of $SE(2)$ or $SE(3)$ rather than $\mathbb{R}^2$ or $\mathbb{R}^3$. This problem has inspired many motion planning works, mainly streamed as sampling-based and optimization-based methods.

Sampling-based methods like probabilistic roadmap (PRM) [2] are widely adopted because of their simplicity. The approaches employ discrete collision detection, which is straightforward and efficient but highly dependent on the chosen resolution. Setting improper parameters can lead to missed obstacles, resulting in invalid paths. In contrast, optimization-based methods, such as Mixed-Integer Linear Programming (MILP) [3], [4] or Graphs of Convex Sets (GCS) [5] offer a more robust solution by directly incorporating collision avoidance into the formulations. These methods, unlike sampling-based methods, ensure paths being collision-free by design, thus bypassing the dependency of resolution in collision detection.

Despite their advantages, current optimization formulations come with their own set of challenges. Solving MILPs, for example, is an NP-hard problem and can be computationally intensive as the environment complexity increases. The time to directly solve a MILP for a feasible path grows exponentially with the number of waypoints, which is proportional to the required maneuvers in an environment. This scalability issue makes brute-force MILP-based methods difficult to apply. On the other hand, the effectiveness of GCS depends on precise initial seeding to capture narrow passages within the configuration space, which brings feasibility challenges to GCS methods.

To address these inefficiencies, our approach simplifies the problem by decomposing a single large MILP into manageable, fixed-size smaller MILPs. Additionally, by focusing directly on the workspace rather than the configuration space, our method mitigates the challenges associated with identifying critical narrow passages, thereby enhancing efficiency and scalability in path planning.

1.1.2 Multi-DoF Manipulators

The second challenge discussed in this thesis is safe control of manipulators with multiple degress of freedom (DoFs) and efficient planning in the corresponding configuration space. Despite the wide adoption of robotic manipulators in many real-world applications, real-time planning of safe execution paths for manipulators in crowded environments can still be challenging due to high-dimensional complex dynamics.

Sampling-based motion planning methods, such as Rapidly Exploring Random Trees (RRT) [6], Probabilistic Roadmaps (PRM) [2], and their extensions [7]–[11] have demonstrated their efficacy in generating collision-free paths in complex environments. However, the high sampling complexity of those methods is prominent for manipulators. Moreover, those methods require accurate state estimation before planning, which often assumes static environments and precise knowledge of the shapes and positions of obstacles. The number of samples used in RRT can be reduced if expanding a node has a higher likelihood of success, which helps to reduce the node number for finding a solution. The expansion of a node will terminate when a collision is detected. On the contrary to recklessly heading towards the sampled state, using a safe steering function will substantially reduce the early termination of expansion.

In the context of safe control, CBF has demonstrated success in rather complex robotic systems [12], [13], including manipulators [14], [15]. However, the CBFs used in the above works are typically manually designed as functions over the state of the environment, requiring both extensive experts' experience and accurate estimation from sensory data. When applying to robotic manipulator systems, the articulated nature of robots also poses an extensive computation burden for state estimation to handle the varying geometry of robots [14], [16]. For the simplicity of construction, geometric shapes of a robot are often over-approximated [16]–[19] and CBFs are often selected in simple forms like signed distance or [14], [16] or in quadratic form [20], [21]. The cost for simplicity is that the CBFs may be over-conservative and a feasible control signal is not guaranteed to be found.

To address these challenges, we are inspired by the recent advances in learning CBFs for the efficient synthesis of safe controllers, which has shown success in walking [22], flight [23] and multi-agent setting [24], with some other attempts to extend these neural CBFs to observation-feedback systems [25], [26]. We also combine the use of control barrier functions (CBF) [27], [28] in multiple robotics applications for safe control to design a steering function for sampling-based methods. Prior works [29]–[32] have shown significant advancements in combining the safe CBF controller with sampling-based motion planning algorithms for low-dimensional systems.

1.2 Contributions

To address the challenges associated with rigid body objects and manipulators, we present two frameworks, respectively.

For rigid polytope objects, we propose a hybrid approach that utilizes information from both workspace and configuration space. Our method consists of three primary stages: i) constructing a coarse graph of convex sets in the workspace to efficiently cover obstacle-free areas and simplify the complexity typically associated with higher-dimensional configuration spaces; ii) building a pre-computed graph in the configuration space where nodes represent bottleneck configurations and edges are feasible paths validated through Mixed Integer Linear Programming (MILPs); and iii) an online query phase that connects start and goal configurations with the precomputed graph in the second stage and retrieves the solution path.

The **contributions** can be summarized as:

1. By operating directly within the workspace, our method effectively mitigates the challenges associated with covering narrow tunnels in the free space, compared to configuration space-based approaches.

2. We simplify the path planning process by breaking down a large, complex optimization problem into a series of smaller MILP problems. Each smaller problem is focused on finding a valid path segment within a region, significantly enhancing the scalability of our approach.

3. We conduct comprehensive experiments in both 2D and 3D environments, comparing against baseline methods PRM and GCS. Our results demonstrate scalability with the environment, higher feasibility, and shorter computation time.

With manipulators, we build upon the motion planning framework RRT [6] and learn a CBF-INC to steer the system towards newly sampled configuration. CBF-INC has two variants handling different inputs: state (signed distance) and point cloud input from LiDAR. Given state input, our framework CBF-INC-RRT increases the success rate by 14% and reduces the number of explored nodes by 30% on the most challenging test cases, compared with vanilla RRT and other neural-controller-enhanced RRT. CBF-INC-RRT also doubles the success rate and halves the explored nodes, compared with the hand-crafted CBF-enhanced RRT method by avoiding over-conservativeness. With point cloud input setting, where many methods (like vanilla RRT and hand-crafted CBF) are not directly applicable, CBF-INC-RRT still improve the success rate by 10% on challenging cases, compared with planning with other steering controllers.

The **contributions** can be summarized as:

1. We present a CBF-INC specialized for robotic manipulators. Our neural networks tackle high-dimensional observations and complex geometric link shapes. To the best of our knowledge, this is the first CBF-style controller taking raw sensor input in high-dimensional manipulators.

2. We present a framework - CBF-INC-RRT that incorporates the learned neural CBF into the motion planning algorithm. Such a framework preserves the completeness of traditional motion planning methods while benefiting from the safe exploration of CBF-INC.

3. Through extensive experiments on 4D and 7D manipulators, we demonstrate that planning algorithms using CBF-INC significantly outperform baselines, in terms of success rate and exploration efficiency. We also show CBF-INC generalizes to dynamic environments and evaluate CBF-INC-RRT on hardware.

1.3 Thesis Outline

The rest of this thesis is organized as follows:

- Chapter 2 provides an overview of related works in motion planning and .

- Chapters 3 and 4 introduces our MILP-based method and CBF-INC-RRT, to tackle the challenges for manipulated objects and manipulators, respectively. The chapters include the problem statements, the general algorithm structure and experimental results.

- Chapter 5 concludes by summarizing the main points of this thesis and discusses several possible directions for future works.

Chapter 2

Related Work

Graph-based methods. Graph-based methods are particularly suitable for scenarios requiring multiple queries withn the same environment, once the graph is constructed, it can be reused to find multiple paths. One category builds the graph of discrete configurations, like sampling-based methods PRM [2] and lattice planners [33]. PRM randomly samples states in C-space as nodes in the graph. Nodes in the graph are connected based on the feasibility of direct paths. Lattice planners are commonly used in nonholonomic vehicle parking scenarios and leverage high levels of parralelization [33], [34]. They use a regular grid in the configuration space to define graph nodes [35]. And the edges are computed offline [36], representing motions that adhere to specific constraints. However, one inherent limitation of lattice planners is their scaling with grid resolution. Employing regular lattices across the entire configuration space can be computationally expensive and inefficient. In contrast, our method reduces the size of the graph by only utilizing a regular grid on a set of manifolds within the configuration space, where the metric is strictly zero.

Another category of graph-based planners relies on space decomposition, where the graph explicitly represent the free space. Based on constrained Delaunay triangulation [37], [38], a complete yet non-overlapping decomposition method, previous works perform well in 2D environments for point objects [3], [39], either through search or by solving MILPs. However, applying constrained Delaunay triangulation to 3D environments or MILP to non-circular objects presents significant challenges, limiting its utility in more complex scenarios. GCS line of works [5], [40], [41] adopts a different approach by attempting to cover the free space with a set of convex polytopes, thereby retrieving paths through optimization. This method is capable to work in both workspace and configuration space, thus accommodating any shape and scaling to more than 10 dimensions.

Exact collision detection Most path-planning algorithms work with discrete collision detection, due to their easy implementation, broad applicability, and fast execution [2], [42]. This method checks a path in C-Space by creating samples along the path and when all these samples are checked to be collision-free, the entire path is assumed to be collision-free. While efficient, its accuracy depends on the sampling resolution, risking missed collisions with coarse samples or delays with overly fine-grained samples. To enhance reliability, one method is free bubble [43], recursively bisecting the samples on the path to guarantee collision-free [44] by calculating the maximum allowable movement without collision. Another method is continuous collision checking [45], [46], which performs well for rigid body motions by directly checking the collisions of the reachable set with the obstacles. But they suffer from the speed. We integrate both methods in different modules of the algorithm, aiming for accuracy and efficiency.

Local Safety for robotic arms. Safe deployment in the real world is crucial for general-purpose robotic arms. Various non-learning techniques have been proposed to tackle the collision avoidance problems on manipulators, ranging from potential field methods [47]–[49], reachability analysis [50], to CBF [15], [16], a trusted tool for ensuring safety in control systems [27], [51]. These methods, including CBF are usually hand-crafted via signed distance [14], [16], [52], quadratic form [20], [21] or minimum uniform scaling factor [15]. While effective for certain environments, they require substantial design efforts and perfect knowledge of the environment. For simplicity, CBFs designed for robotic arms with multiple degrees of freedom (DoF) often use a set of simple convex shapes to over-approximate the rigid body links [16]–[19], leading to over-conservative policies. Extending these approximations from one shape to another is also not straightforward.

In contrast, data-driven methods, exploiting the power of neural networks, have shown promise in addressing complex geometric shapes of manipulators [53], [54], even when only high-dimensional observations are available [55]. Recently, learning-based CBFs have demonstrated potential in resolving these challenges on drones in multi-agent setting [24] and visual navigation systems [25]. However, in robotic manipulator systems, the adoption of neural networks in CBFs is restricted to parameter search guidance [56] due to the convoluted collision-free configuration space.

Inductive bias in sampling-based motion planning. One stream of methods [57]–[60] has been proposed to improve sampling-based planning methods by incorporating inductive bias. The methods fall into two categories. One set of methods seeks to find heuristic functions to prioritize the samples to explore, including Fast Marching Trees [61], sampling-based A* [62], and recent GNN-related works [60], [63]. Some works also consider improving the sampling strategy [57], [64]. However, these methods still suffer from finding a control policy for the planned reference trajectory, especially for complex dynamics. The other class of methods conceives motion planning problems as sequential decision-making problems and relies on neural policies [59], [60], [65] such as imitation learning [66] or reinforcement learning [58] in an end-to-end manner, aiming to find a collision-free trajectory directly with a neural network. Some further consider adding explicit safety constraints during training to accelerate [53], [67]. Though these methods have shown impressive results, they sacrifice the completeness guarantee of many motion planning algorithms.

Some related works to the second part of this thesis are [29]–[32], which explore incorporating CBF into sampling-based motion planning. [30]–[32] proposed improvements on sampling methods and [31], [32] focused on optimal planning.

Chapter 3

Planning Paths for Rigid Objects

In this chapter, we propose a path planning algorithm for rigid body using mixed-integer linear programming (MILP).

3.1 Planning Pipeline for Point Objects

In this paper, we consider the path planning problem of a rigid body object O. The workspace $\mathcal{W}$, the physical space where the object lies, can be $\mathbb{R}^2$ or $\mathbb{R}^3$ depending on the particular application at hand. And the configuration space for O is $SE(2)$ and $SE(3)$, respectively. A configuration is represented as $q = (p, R)$ with a position vector $p \in \mathbb{R}^2$ or $\mathbb{R}^3$ and a rotation matrix $R \in SO(2)$ or $SO(3)$. The pipeline of our method is structured into three key stages, as shown in Fig. 3.1. We first construct a graph of convex set G_c covering the free workspace. This stage is performed offline and the graph is reusable for multiple queries across different objects. We exploit the lower dimensionality of the workspace compared to the configuration space to obtain a higher coverage ratio and fewer narrow tunnels, which are typically challenging to identify and cover. Then we construct a graph G_d in configuration space, which is also offline. Here, the nodes represent bottleneck configurations identified by G_c and the edges are viable paths validated through MILPs. This graph allows multiple queries for different start and goal configurations. At the online stage, we connect the start and goal configuration into G_d and retrieve the solution path. Our experiment results show that this method significantly reduces online time consumption across various objects and environments, in both 2D and 3D settings.

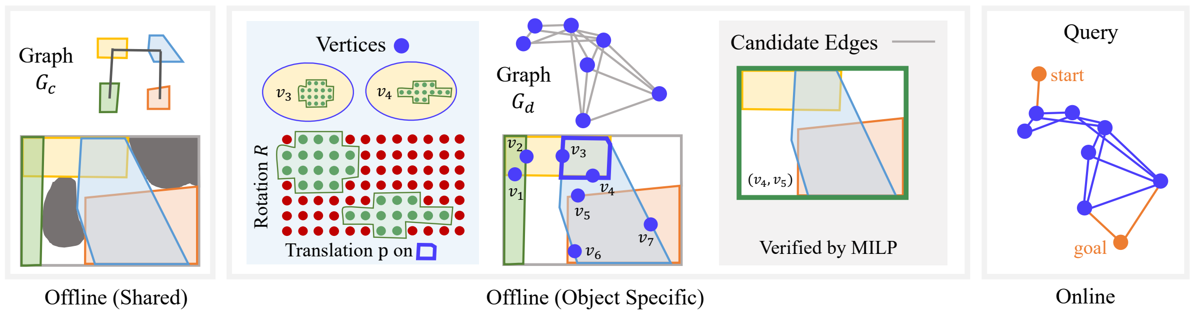


Figure 3.1: An overview of our pipeline. (a) Decomposition of the free workspace into a set of convex polytopes. The polytopes serve as graph vertices and are interconnected if they overlap G_c (Section 3.1.1). (b) Construction of G_d(Section 3.1.2), where each vertex is a set of free configurations - illustrated as the set of green points enclosed by the green ring - positioned on the boundary (blue ring) of intersections of polytopes(Section 3.1.2). We verify all the candidate edges (gray) via MILPs, and the valid edges (blue) demonstrate the existence of path segments between vertices (**??**). (c) Online query stage: the start and end configurations are connected to graph G_d and the planned path is retrieved(Section 3.1.3).

3.1.1 Construct Coarse Graph G_c

At this stage, we aim to approximately decompose the free workspace $\mathcal{W}_{\text{free}}$ into a set of convex polytopes $\mathcal{P}$ and build a graph G_c on $\mathcal{P}$ as summarized in Algorithm 1. The graph G_c is designed to be reused across various objects within the same environment.

Following [68], we first construct a visibility graph $G_v := (\mathcal{V}_v, \mathcal{E}_v)$ where $\mathcal{V}_v$ is uniformly sampled from $\mathcal{W}$ with points inside obstacles excluded, and $\mathcal{E}_v$ is added by checking for collisions along the line segments connecting each pair of points within some distance using Proposition 1. During each iteration, the subroutine SAMPLEVISIBILITYEDGE samples n_s points on $\mathcal{E}_v$ that are not yet covered by the polytopes in $\mathcal{P}$. The n_s points are then used as initial points for IRIS algorithm [40], after which the newly computed polytopes are added into $\mathcal{P}$. The iteration terminates when the coverage ratio of $\mathcal{P}$ over $\mathcal{E}_v$ exceeds threshold $\alpha > 0$. Subsequently, an undirected graph $G_c := (\mathcal{V}_c, \mathcal{E}_c)$ is constructed with vertices $\mathcal{V}_c = \mathcal{P}$, and with an edge $(P_{i_1}, P_{i_2}) \in \mathcal{E}_c$ for every pair of intersected polytopes P_{i_1} and P_{i_2}.

Algorithm 1 Construct a graph of convex polytopes G_c

```
Require: list of obstacles $\mathcal{O}_{\text{obs}}$, number of samples for visibility graph $n_v$, number of samples
    per iteration $n_s$, coverage threshold $\alpha$.
1: $\mathcal{P} \leftarrow \emptyset$
2: $G_v \leftarrow$ SampleVisibilityGraph($\mathcal{O}_{\text{obs}}, n_v$)
3: while CheckCoverage($\mathcal{E}_v, \mathcal{P}$) $< \alpha$ do
4:     $\mathcal{S} \leftarrow$ SampleVisibilityEdge($\mathcal{E}_v, \mathcal{P}, n_s$)
5:     $\mathcal{P} \leftarrow \mathcal{P} \cup$ Iris($\mathcal{S}$)
6: end while
7: $G_c \leftarrow$ ConstructGraph($\mathcal{P}$)
8: return $G_c$
```

3.1.2 Construct Dense Graph G_d

Graph G_c is a GCS, and an optimal solution of the path planning problem for a point object can be found via [41]. However, its direct applicability is limited when considering non-point objects. The limitation arises because an intersection between polytopes - a valid pathway for point objects - does not inherently guarantee feasible traversal for objects with non-negligible dimensions and orientations.

To address this challenge, we introduce another undirected graph $G_d := (\mathcal{V}_d, \mathcal{E}_d)$ that describes the connectivity between adjacent polytopes for a specific object O, detailed in Algorithm 2. Our key insight is that, for an object O to move from P_i to P_j, its center c must traverse the boundary ∂P_{ij} of the intersection $P_{ij} := P_i \cap P_j$.

Vertices $\mathcal{V}_d$. The vertices $\mathcal{V}_d$ in G_d are generated through subroutine Sample&Group. The process begins with discretizing ∂P_{ij} into regular grids.

a) Discretization in 2D environments. For 2D, the boundary ∂P_{ij} is a closed ring, which can be parameterized linearly from $\lambda = 0$ to $\lambda = 1$, with the points for $\lambda = 0$ and $\lambda = 1$ being identical. Translations along this boundary are selected with a fixed interval $\delta\lambda > 0$. Rotations are discretized into $n_R > 0$ possible values, with angles $(\theta_1, \cdots, \theta_{n_R})$ being $(1 \cdot 2\pi/n_R, 2 \cdot 2\pi/n_R, \cdots, n_R \cdot 2\pi/n_R)$. These correspond to rotation matrices $(R_1, \cdots, R_{n_R})$.

Together, the translation parameter λ and the set of rotations θ create a 2D grid of possible configurations, as shown in Fig. 3.1.

b) Discretization in 3D environments. In 3D scenarios, the boundary ∂P_{ij} consists of several facets, each treated as an independent boundary without considering the connections between them as patches. On each facet, the translations are constructed using a regular rectangle grid, and the rotations are a selected set of rotation matrices $(R_1, \cdots, R_{n_R})$. Together, the translations and rotations create a 3D grid on a facet.

After discretization, we collect the set of free configurations $\mathcal{C}_{ij}$ on all grids, where a free configuration refers to the object being completely contained inside $P_i \cup P_j$. We then try to connect adjacent free configurations on the same grid, which isn't needed for point objects as the configurations belong to the same convex polytope. This process results in the formation of several disjoint subgraphs $\{G_{p,ij,n}\}_{n=1}^{n_{ij}}$, as shown in Fig. 3.1. Within each subgraph, any two configurations can be interconnected via a path. Subsequently, the set of vertices $\mathcal{V}_{p,ij,n} \subset \mathcal{C}_{ij}$ of n-th subgraph $G_{p,ij,n}$ on ∂P_{ij} becomes a node $v_{ij,n} = \mathcal{V}_{p,ij,n} \in \mathcal{V}_d$ to serve as potential waypoints in detailed path planning.

Edges $\mathcal{E}_d$. We assess all possible connections among $\mathcal{V}_d$ to establish $\mathcal{E}_d$. For any two vertices v_{ij,n_1} and v_{ik,n_2}, an edge is considered if the proposed traversal between vertices is verified as feasible by VerifyTraversal. The edge (v_{ij,n_1}, v_{ik,n_2}) represents the feasibility of moving from polytope P_j to P_k via P_i. Specifically, for a point object, as all the points in $v_{ij,n_1} \cup v_{ik,n_2}$ are inside a same convex polytope, any points from v_{ij,n_1} can be connected with any points in v_{ik,n_2}. So an edge can be added to $\mathcal{E}_d$ as long as they share the same polytope for a point object.

3.1.3 Online Query

Graph G_d serves as an offline pre-computed roadmap, enabling rapid online path planning from a start configuration q_{start} to an end configuration q_{end}. By introducing new vertices $v_{\text{start}} = \{q_{\text{start}}\}$ and $v_{\text{end}} = \{q_{\text{end}}\}$ into G_d, we can follow the same VerifyTraversal

Algorithm 2 Construct a graph of convex polytopes G_d

```
Require: list of obstacles $\mathcal{O}_{\text{obs}}$, graph $G_c$, number of intermediate waypoints $N$.
 1: $\mathcal{V}_d \leftarrow \emptyset$, $\mathcal{E}_d \leftarrow \emptyset$
 2: for $(P_i, P_j) \in \mathcal{E}_c$ do
 3:     $v_{ij,1}, \cdots, v_{ij,n_{ij}} \leftarrow \textsc{Sample\&Group}(\partial P_{ij}, \mathcal{O}_{\text{obs}})$
 4:     $\mathcal{V}_d \leftarrow \mathcal{V}_d \cup \{v_{ij,1}, \cdots, v_{ij,n_{ij}}\}$
 5: end for
 6: CandidateEdgeSet = $\{(v_{ij,n_1}, v_{ik,n_2}) | v_{ij,n_1}, v_{ik,n_2} \in \mathcal{V}_d, v_{ij,n_1} \neq v_{ik,n_2}\}$
 7: for $(v_{ij,n_1}, v_{ik,n_2}) \in$ CandidateEdgeSet do
 8:     if $\textsc{VerifyTraversal}(v_{ij,n_1}, v_{ik,n_2}, G_c, N)$ then
 9:         $\mathcal{E}_d \leftarrow \mathcal{E}_d \cup \{(v_{ij,n_1}, v_{ik,n_2})\}$
10:     end if
11: end for
12: return $\mathcal{V}_d, \mathcal{E}_d$
```

subroutine to connect the vertices with G_d and retrieve the planned path online. The path is composed of two parts: motion between vertices and motion inside the configuration inside a vertex.

3.1.4 Discussion

As the problem is decomposed and solved optimally for each subproblem, the approach does not guarantee global optimal for the entire path. However, the algorithm is capable of finding multiple solutions or modalities, which can then serve as initial solutions for further refinements.

3.2 From Point Objects to Polytope Objects

Moving from points to non-point objects, a significant challenge arises in verifying the motion between waypoints due to the complex geometry of the objects. In the literature, this issue is commonly addressed by bloating the obstacles to account for the size of objects, thereby transforming the problem into path-planning within this bloated environment as with a point object [39]. However, this method can be overly restrictive, especially for objects whose shapes deviate significantly from circular or spherical forms.

In this section, we will tackle the challenge using MILP to handle polytope objects. We assume the rigid-body object O to be simply connected (no holes) and the line segments extending from its geometric center to each vertex are entirely contained within the boundaries of the object. We denote the vertices of O are $\{v_{O,i}\}$.

3.2.1 MILP Formulation

As discussed in Section 3.1.2, the traversal between vertices (v_{ij,n_1}, v_{ik,n_2}) in G_d is not straightforward for non-point objects. Therefore, we formulate the subroutine VERIFYTRAVERSAL as an MILP, which is to verify whether there exists a piece-wise linear path from a configuration in v_{ij,n_1} to another one in v_{ik,n_2} with N intermediate waypoints.

For the sake of simplicity, we'll denote v_{ij,n_1} as u and v_{ik,n_2} as w in the discussion of MILP, while denoting the set of polytopes $\{P_i, P_j, P_k\} \subset \mathcal{E}_c$ as $\mathcal{P}_c$. The list of free configurations in u or w are denoted as $(q_{u,1}, \cdots, q_{u,n_u})$ and $(q_{w,1}, \cdots, q_{w,n_w})$.

Let q_t be the waypoints in the path, including the start and end configurations, indexed by $t \in \{0, \cdots, N+1\}$. Each waypoint q_t consists of a translational part p_t and a rotational part R_t. The translational part in a 2D scenario is given by $p_t = (x_t, y_t)$, and in 3D, it extends to $p_t = (x_t, y_t, z_t)$.

Model. The objective is to minimize the total 1-norm of the translational displacement along the path. The entire MILP model is formulated in Eq. (3.1):

$$\min_{\{p_t\}_t, \{R_t\}_t, \mathcal{B}} \sum_{t=0}^{N} \|p_{t+1} - p_t\|_1, \tag{3.1a}$$

$$\text{s.t.} \quad Eqs. \ (3.2), (3.3), (3.4) and \ (3.5). \tag{3.1b}$$

Here $\mathcal{B}$ is the set of all binary variables we'll introduce. The objective in Eq. (3.1a) can be easily transformed to standard linear expressions with additional variables, which we will not detail here.

Basic constraints. The rotational part R_t is encoded as a one-hot vector $\beta_{R,t}$ using binary variables. This is achieved through the constraints

$$R_t = \sum_{i=1}^{n_R} \beta_{R,t,i} R_i, \qquad \forall t = 0, \cdots, N+1 \tag{3.2a}$$

$$1 = \beta_{R,t}^T \mathbf{1}, \qquad \forall t = 0, \cdots, N+1 \tag{3.2b}$$

Each element in R is a possible rotation matrix, as specified in Section 3.1.2. The index where $b_{R,t} = 1$ indicates the selection of the corresponding rotation at waypoint t.

Between two consecutive waypoints, we enforce the object to either translate or rotate, a decision encoded by binary variable $\beta_{\text{or},t}$ in Eq. (3.3). This constraint addresses the challenges associated with encoding collision-free conditions in MILPs. By restricting the motion to either translation or rotation at each step, this simplified model becomes computationally tractable. Specifically, in 3D scenarios, we further limit the object to translational motion only, equivalent to setting $\beta_{\text{or},t} = 1$.

$$\|p_{t+1} - p_t\|_1 \leq M \cdot \beta_{\text{or},t}, \qquad \forall t = 0, \cdots, N \tag{3.3a}$$

$$\|\beta_{R,t+1} - \beta_{R,t}\|_1 \leq M(1 - \beta_{\text{or},t}), \qquad \forall t = 0, \cdots, N \tag{3.3b}$$

$$\beta_{\text{or},t} = 1, \text{ (only present in 3D)} \qquad \forall t = 0, \cdots, N \tag{3.3c}$$

The constraints for stand and end configuration are that first and last waypoints are in u and w correspondingly. We introduce two additional one-hot vectors β_{start} and β_{end}, which are used to select specific configurations from the sets u and w, shown in Eq. (3.4).

$$p_0 = \sum_{i=1}^{n_u} \beta_{\text{start},i} p_{u,i}, \qquad p_{N+1} = \sum_{i=1}^{n_w} \beta_{\text{end},i} p_{w,i}, \tag{3.4a}$$

$$R_0 = \sum_{i=1}^{n_u} \beta_{\text{start},i} R_{u,i}, \qquad R_{N+1} = \sum_{i=1}^{n_w} \beta_{\text{end},i} R_{w,i}, \tag{3.4b}$$

$$\sum_{i=1}^{n_u} \beta_{\text{start},i} = 1, \qquad \sum_{i=1}^{n_w} \beta_{\text{end},i} = 1 \tag{3.4c}$$

3.2.2 Collision-Avoidance Constraint

The remaining constraint in the MILP is encoding collision avoidance for non-point objects. While the start (q_0) and end (q_{N+1}) configurations are already ensured to be collision-free, ensuring continuous collision avoidance throughout the motion path is critical. Here, the collision avoidance constraint is encoded by enforcing that the each reachable set between two consecutive waypoints (q_t, q_{t+1}) $(t = 0, \cdots, N)$ does not intersect with the boundary of the union of related polytopes. Specifically, this is achieved by requiring that the surface of reachable set be completely contained within the union of all relevant collision-free polytopes, denoted as $\bigcup \mathcal{P}_c$.

Compute surface of reachable set

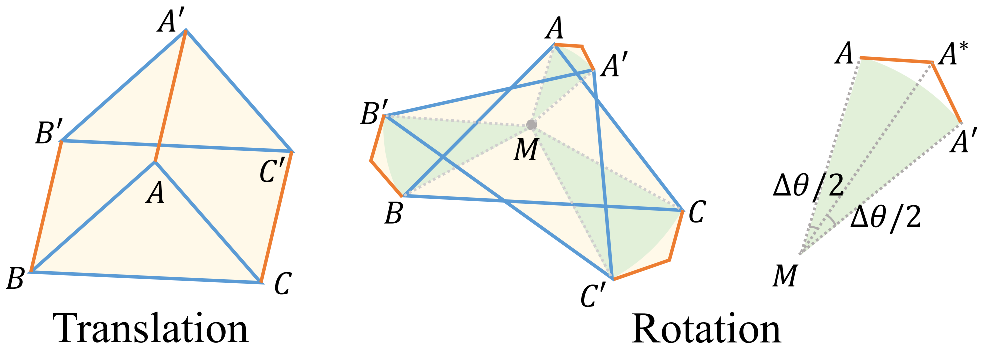


Figure 3.2: Reachable set (yellow region) of 2D object.

In 2D, the reachable set is approximated as Fig. 3.2. For translation, the boundary contains the blue edges of O at q_t and q_{t+1} and orange lines connecting a same vertex at two waypoints. For rotation, the difference is that the lines connecting a same vertex at two waypoints are arcs. While the arcs can not be expressed in linear expression, we overapproximate the arc with two line segments, with $\|MA^*\| \cos(\Delta\theta_{\max}/2) = \|MA\|$. $\Delta\theta_{\max}$ is the maximum rotation angle allowed in one step, selected as $\pi/3$.

In 3D, only translation is allowed. The reachable set is the union of the polytopes swept by each face of O, which is also a polytope. So the faces of the reachable set must be a subset of the union of the faces of O at q_t and q_{t+1} and the parallelogram swept by each edge of O.

Collision detection between surface and polytope

We observed the following, so we just need to check each line segments (no matter 2D or 3D) to satisfy Proposition 1.

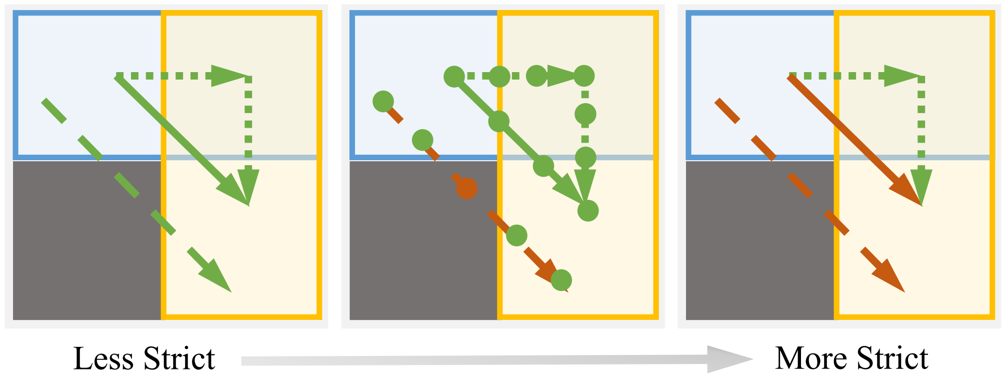


Figure 3.3: Models of collision avoidance in environments with convex polytopes (blue and yellow) and obstacles (gray). Green lines indicate collision-free paths, and red lines indicate detected collisions. *Left:* This model only checks endpoints of each line segment. The dashed red line demonstrates that endpoint-only checking is insufficient as it cuts through an obstacle. *Right* [4]: This overly restrictive model allows only paths like the dotted line where both endpoints lie within the same free region, rejecting the simpler, viable solid line. *Middle (ours):* modeled by Proposition 1. This approach prevents the dashed line from cutting through obstacles while including more direct paths between regions, like the solid line.

Proposition 1 (Line segment inside two convex polytopes). *Given two convex polytopes $P_i = \{x | A_i x \leq b_i\}$, $i = 1, 2$, and a line segment l with two endpoints (x_a, x_b), l is contained inside the union of P_1 and P_2 if one of following conditions holds:*

1. *x_a and x_b are within a same polytope P_i,*
2. *x_a and x_b are within different polytopes, but there exists a point x satisfying $x \in (P_1 \cap P_2)$.*

Proof. For condition (1), the convexity of P_i ensures that all points on l, being a line segment between x_a and x_b, are also contained within P_i. If condition (2) holds, line segments (x_a, x)

and (x, x_b) both satisfy condition (1). Thus, the entire line segment l remains within the union of P_1 and P_2. □

Corollary 1.1 (Triangle). *Given two convex polytopes $P_i = \{x|A_i x \leq b_i\}$, $i = 1, 2$, and a triangle T, T is contained inside the union of P_1 and P_2 if all edges of T satisfy at least one of the conditions in Proposition 1.*

Proof. We first consider the case when all the vertices of triangle T are contained in exactly one of P_i, there are two possible cases as demonstrated in Fig. 3.4:

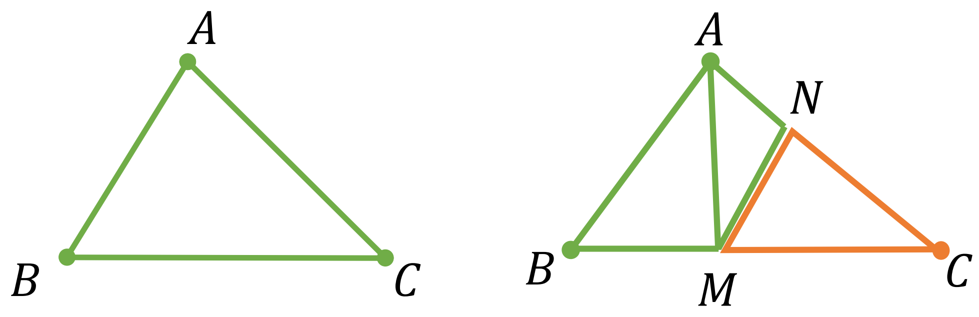


Figure 3.4: *Left:* all the vertices are contained in the same green polytope. *Right:* one vertex (C) is not contained in the same polytope as vertices A, B.

1. All vertices of triangle T are contained within the same convex polytope, here referred to as the green polytope. Since convex polytopes maintain the property that any point on the line segment between any two points within the polytope also lies within the polytope, it follows that the entire area of triangle T is contained within the green polytope.

2. One vertex C is located in a different polytope from vertices A, B (*right*). Given that the edge BC satisfies Proposition 1, there exists a point M on BC such that $M \in P_1 \cap P_2$. Similarly, for edge AC, there exists a point N on AC that also belongs to $P_1 \cap P_2$. The triangle T, i.e. $\triangle ABC$ can be divided into three triangles: $\triangle ABM, \triangle AMN, \triangle MNC$. Since each triangle's vertices are entirely contained within at least one polytope and each polytope is convex, each of these smaller triangles is

contained within a polytope. Hence, the entire triangle T is contained within the union of P_1 and P_2.

When extending the conditions to allow vertices of triangle T to be contained in more than one of P_i, it is equivalent to considering an enlargement of the polytopes P_1 and P_2. Therefore, the proposition that triangle T is contained within the union of the polytopes still holds. □

Corollary 1.2 (Convex quadrilateral)**.** *Given two convex polytopes* $P_i = \{x|A_i x \leq b_i\}$, $i = 1, 2$, *and a convex quadrilateral* P_0, P_0 *is contained inside the union of* P_1 *and* P_2 *if all edges of* P_0 *satisfy at least one of the conditions in Proposition 1.*

Proof. Similar to the proof of triangles, we first consider the case when all the vertices of convex quadrilateral P_0 are contained in exactly one of P_i, there are four possible cases as demonstrated in Fig. 3.5:

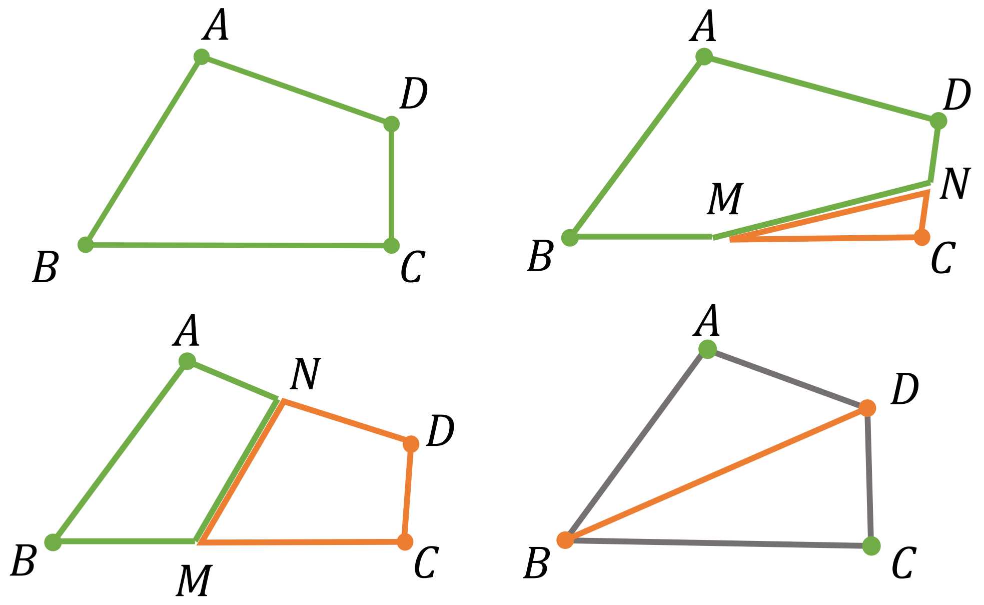


Figure 3.5: *Upper left:* all the vertices are contained in the same green polytope. *Upper right:* vertex C is not contained in the same polytope as the other vertices (A, B, D) *Lower left:* two adjacent vertices (C, D) are contained in a different polytope as A, B. *Lower right:* vertices along the same diagonal (AC and BD) are contained in a same polytope.

1. All vertices of convex quadrilateral P_0 are contained within the same convex polytope, here referred to as the green polytope. Since convex polytopes maintain the property that any point on the line segment between any two points within the polytope also lies within the polytope, it follows that the entire area of convex quadrilateral $P_0$is contained within the green polytope.

2. One vertex C is located in a different polytope from vertices A, B, D. Given that the edge BC satisfies Proposition 1, there exists a point M on BC such that $M \in P_1 \cap P_2$. Similarly, for edge CD, there exists a point N on CD that also belongs to $P_1 \cap P_2$. The convex quadrilateral P_0, i.e. $\triangle ABC$ can be divided into two polytopes: polytope $ABMND$ and $\triangle MNC$. Since each small polytope's vertices are entirely contained within at least one P_i and each P_i is convex, each of these smaller polytopes is contained within a polytope. Hence, the entire convex quadrilateral P_0 is contained within the union of P_1 and P_2.

3. Two adjacent vertices C, D are contained in a different polytope as A, B. Similarly, there exists a point M on BC and a point N on AD such that $M, N \in P_1 \cap P_2$. The convex quadrilateral P_0, i.e. $\triangle ABC$ can be divided into two polytopes: quadrilateral $ABMN$ and quadrilateral $MNDC$. Since each small polytope's vertices are entirely contained within at least one P_i and each P_i is convex, each of these smaller polytopes is contained within a polytope. Hence, the entire convex quadrilateral P_0 is contained within the union of P_1 and P_2.

4. Vertices along the same diagonal (AC and BD) are contained in a same polytope. We divide the quadrilateral $ABCD$ into two triangles $\triangle ABD$ and $\triangle BCD$. The two vertices of edge BD are contained in the same convex polytope, so both two small triangles satisfy Corollary 1.1. Hence, the entire convex quadrilateral P_0 is contained within the union of P_1 and P_2.

When extending the conditions to allow vertices of convex quadrilateral P_0 to be contained in more than one of P_i, it is equivalent to considering an enlargement of the polytopes P_1 and P_2. Therefore, the proposition that convex quadrilateral P_0 is contained within the union of the polytopes still holds. $\square$

So we just need to check the line segments within the sets $\mathcal{S}$ based on Proposition 1:

1. the edges of O at waypoints q_t and q_{t+1}
2. the line segments connecting a same vertex of O between waypoints q_t and q_{t+1}.

Accurately verifying condition 2) of Proposition 1 involves a product of two sets of variables - the positions of x_a, x_b and the proportion (x, x_a) occupies. This product cannot be encoded into a mixed integer linear programming. Instead, we divide the line segments into 10 equal parts and only check the 11 endpoints in practice. So the MILP constraints can be written as, $\forall e_s = (p_{s,\text{start}}, p_{s,\text{end}}) \in \mathcal{S}$:

$$\begin{aligned}&A_i(\eta p_{s,\text{start}} + (1-\eta)p_{s,\text{end}}) \le b_i + M(1-\beta_{s,i,\eta}),\\ &\quad \forall P_i \in \mathcal{P}_c, \forall \eta \in \{0, 0.1, \cdots, 1\}\end{aligned} \tag{3.5a}$$

$$\sum_i \beta_{s,i,\eta} \ge 1, \quad \forall \eta \in \{0, 0.1, \cdots, 1\} \tag{3.5b}$$

$$\beta_{s,\text{cond1},i} \le \beta_{s,i,0}, \quad \beta_{s,\text{cond1},i} \le \beta_{s,i,1} \tag{3.5c}$$

$$\begin{aligned}&\sum_\eta (\beta_{s,i,\eta} + \beta_{s,j,\eta} - 1) \ge 1 - M(1-\beta_{s,\text{cond2},ij}),\\ &\quad \forall P_i, P_j \in \mathcal{P}_c,\ P_i \ne P_j\end{aligned} \tag{3.5d}$$

$$\sum_i \beta_{s,\text{cond1},i} + \sum_{(i,j)} \beta_{s,\text{cond2},ij} \ge 1 \tag{3.5e}$$

where $\beta_{s,i,\eta}$, $\beta_{s,\text{cond1},i}$, $\beta_{s,\text{cond2},ij}$ are binary variables for line segment e_s. $\beta_{s,i,\eta} = 1$ indicates that interpolated points are inside polytope P_i. Similarly, $\beta_{s,\text{cond1},i}$ or $\beta_{s,\text{cond2},ij}$ being 1 means the corresponding condition is satisfied. Eqs. (3.5a) and (3.5b) encodes the preconditions, while Eqs. (3.5c) and (3.5d) encodes the two conditions in Proposition 1, respectively.

3.2.3 Revisit Vertices $\mathcal{V}_d$

In constructing the vertices $\mathcal{V}_d$, we connect adjacent free configurations on the grid to form disjoint subgraphs $\{G_{p,ij,n}\}_{n=1}^{n_ij}$. However, simply connecting adjacent free configurations may lead to collisions for non-point objects. To address this, we enlarge the polytope object O during collision checking. This ensures that movements between adjacent configurations remain collision-free.

2D Scenario. For each discrete configuration on this grid, we conduct collision checks for a bloated object O'. Each edge on object is O bloated by distance $\max(l_\lambda, l_\theta)$ where l_λ is the maximum distance between two consecutive translation points and l_θ is the maximum distance a point on O could possibly travel between consecutive rotations, calculated by $\max_i 2\|v_{O,i}\|_2 \sin(\pi/n_R)$.

3D Scenario. We check collisions within a ball of radius $\max_i |v_{O,i}|_2$, which accommodates arbitrary rotations for configurations that share the same translation. For configurations on a same facet that share the same rotation, connectivity does not require verification due to the convexity of P_i and P_j. This is justified as follows: Object O is divided into two parts by the facet, one part resides entirely within P_i and the other inside P_j. This division remains constant for configurations on the same facet under the same rotation. Given that both P_i and P_j are convex, any swept region of each part of O remains confined within its respective convex polytope.

3.3 Experiments

We empirically validate our method in this section. We initially set the number of intermediate waypoints, N, to 0. If a solution is infeasible with $N = 0$, we increase N to 1. Additionally, rotations are discretized into 12 distinct rotations for 2D environments and 24 for 3D environments, respectively. All experiments were launched on a server with 1 AMD Ryzen Threadripper 3990X 64-Core Processor. We adopt Gurobi 10.0.0 [69] as the MILP solver and handle the graphs G_c, G_d with NetworkX [70].

Benchmarks. We collect 8 benchmark environments, visualized in Fig. 3.6. We consider two type of objects: convex and non-convex. In 2D environments, the convex object is selected as a stick of length 1.2 and width 0.1, while the non-convex object is an L-shape with longer side 1.2, shorter side 0.8 and width 0.1. In 3D environments, the convex object is a pad of size $1.0 \times 0.8 \times 0.1$, and the non-convex object is L-shaped, composed of two pads of size $1.0 \times 0.8 \times 0.1$ and $1.0 \times 0.1 \times 0.4$.

Baselines. We compare our method with following multi-query motion planning algorithms, PRM [2] and GCS [71]. PRM, a sampling-based method, is evaluated across 5 trials for each problem set with an offline phase of 15 seconds allocated to develop a roadmap before each trial, implemented with OMPL [72]. GCS constructs a graph of convex set in configuration space [68], followed by optimizing for an optimal path with piece-wise linear curves [71]. To ensure a fair comparison in generating an IRIS cover for both GCS and our MILP-based method , though their application in different space, we select a same set of hyperparameters. Specifically, we set $n_v = 512$, $\alpha = 0.95$ and select $n_s = 5$ for our MILP-based method .

3.3.1 Planning Results

Metrics. In evaluating the performance among three methods, we assess two key metrics. First, we measure the **online time** required to compute a path when a new start and goal configuration pair is specified. Secondly, we evaluate the **number of waypoints** in the solution path. This metric serves as an indicator of the path's complexity and efficiency in navigating from start to goal. A lower number of waypoints generally suggests a smoother execution in practice.

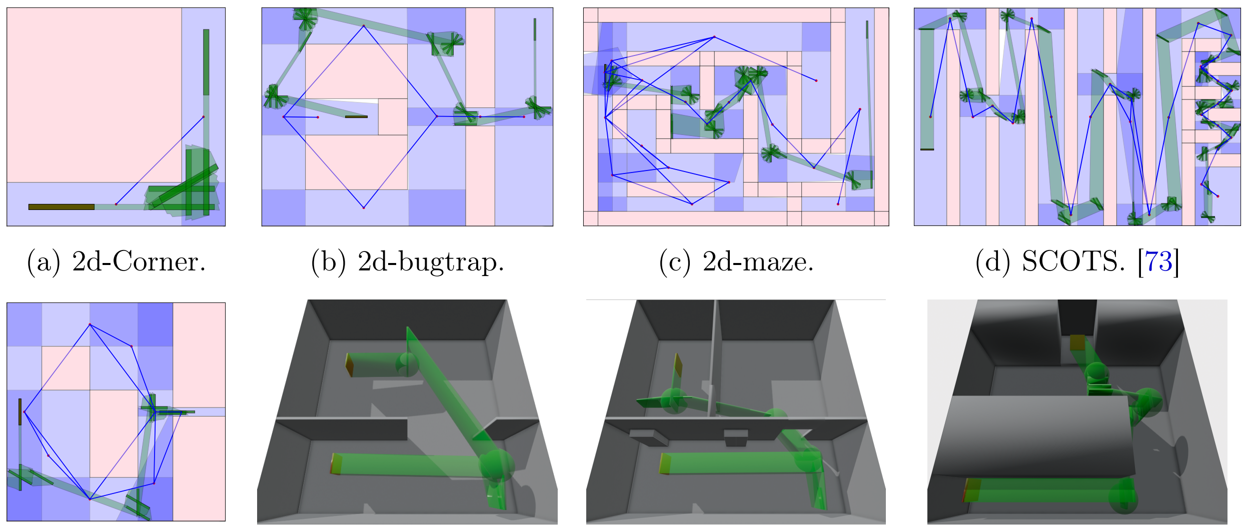
Result We compare with baselines and show the numerical results in Table 3.1. We also visualized the solution paths in Fig. 3.6.

Online Time. Our method consistently achieved the shortest online computation times, typically under 100 ms across most test scenarios. This efficiency contrasts with the GCS, which struggles to find feasible solutions in several environments. This is particularly evident in configurations involving narrow tunnels not covered by the C-IRIS strategy, which is explored further in Section 3.3.3.

The online time for our method involves retrieving the path from G_d and connecting the start and end configurations to the graph. In practice, we set the number of intermediate waypoints $N = 0$, allowing us to leverage matrix operations for quick verification instead of solving MILPs for faster online computation. In contrast, the online stage for GCS involves solving an optimization problem. And for PRM, it includes discrete collision checking to connect start and end configurations to a densely populated graph, much larger than ours.

Waypoint Comparison. GCS, which optimizes for path optimality, generally produces smoother paths when it can find a solution, leading to fewer waypoints. In 2D environments, our method typically finds paths with fewer waypoints compared to PRM, indicating a more efficient pathfinding strategy. But the situation differs in 3D. This is because we over-restrict the free configurations on the boundaries in 3D scenarios, so the solution space is much smaller for our method and our method tends to generate more complex and twisted paths.

Impact of Object Shapes. Our method allows the G_c to be reused for different object shapes. With the dimensions of the object being similar, both our method and PRM are able to maintain a consistent online time for different objects. Conversely, GCS performance is significantly impacted by changes in object shape, altering the geometry of the free configuration space and affecting feasibility and computation times across various environments.



(a) 2d-Corner. (b) 2d-bugtrap. (c) 2d-maze. (d) SCOTS. [73]

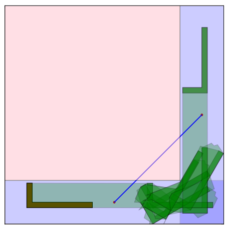
(e) 2d-peg in hole. (f) 3d piano w. (g) 3d-narrow. (h) 3d-peg-in-hole.

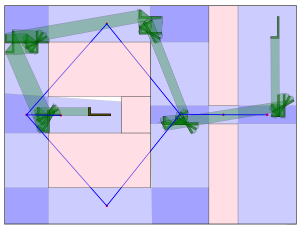
Figure 3.6: Visualization of planning results in 2D (obstacles: pink) and 3D scenarios with convex objects. The start and end configurations are ploted with red, while waypoints and their corresponding reachable sets are shown in green. In 2D scenarios, we also illustrate the graph G_c with vertices ($\mathcal{V}_c$) as blue polytopes and edges ($\mathcal{E}_c$) as blue lines connecting red dots, which represent the centers of vertex polytopes.

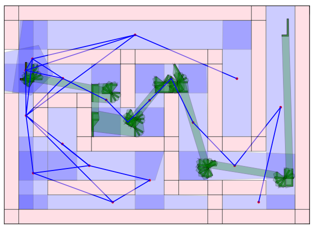
3.3.2 Comparison with Sampling-based Algorithms

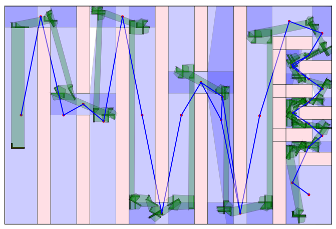
We further compare our method with PRM as detailed in Table 3.2 for scaling. The offline computation time for our method consistently remains around 15 seconds across all test cases, attributable to the minimal changes in the structure of G_c and G_d besides scaling. Similarly, the online execution time remains relatively consistent across scenarios. However, PRM struggles with environment scaling. One critical hyperparameter affecting PRM is the resolution of collision checking. We tested PRM's efficiency with varying distances between checked states—specifically at 0.25x, 0.1x, and 0.05x of the object's length—to assess its adaptability. PRM struggles to maintain the performance, unlike our method's MILP solving, which performs exact continous collision checking with infinite resolution.

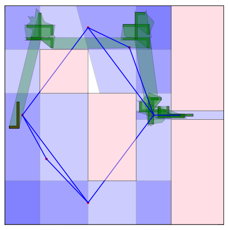
3.3.3 Comparison with GCS

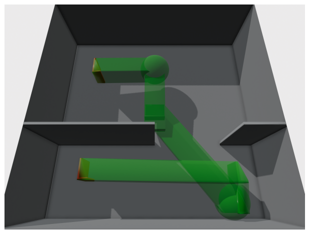

(a) 2d-Corner.

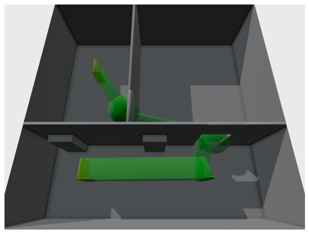

(b) 2d-bugtrap.


(c) 2d-maze.


(d) SCOTS. [73]


(e) 2d-peg in hole.


(f) 3d piano w.


(g) 3d-narrow.

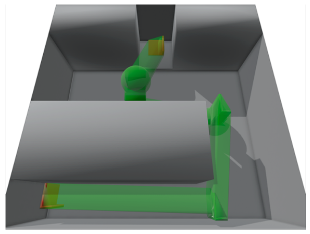

(h) 3d-peg-in-hole.

Figure 3.7: Visualization of planning results in 2D (obstacles: pink) and 3D scenarios with L-shaped non-convex objects. The start and end configurations are ploted with red, while waypoints and their corresponding reachable sets are shown in green. In 2D scenarios, we also illustrate the graph G_c with vertices ($\mathcal{V}_c$) as blue polytopes and edges ($\mathcal{E}_c$) as blue lines connecting red dots, which represent the centers of vertex polytopes.

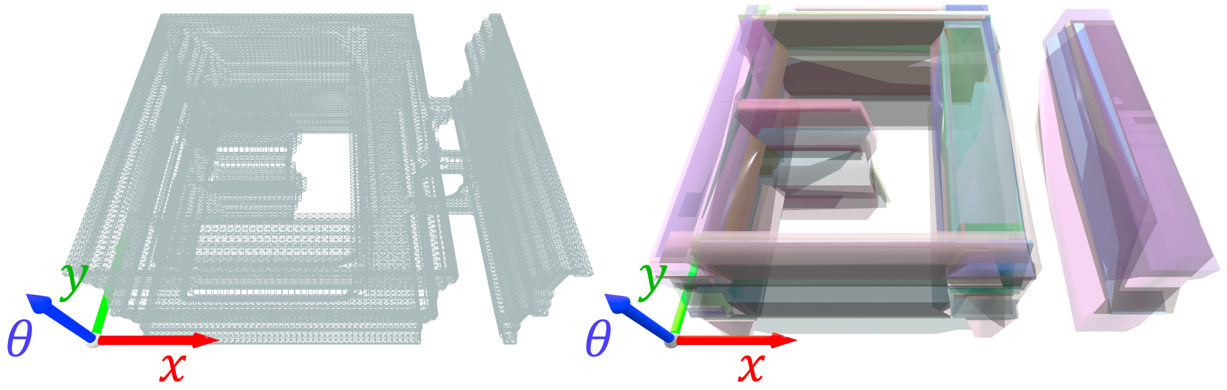


Figure 3.8: A 2D motion planning problem consists of a stick getting out of a trap and passing through a narrow gap, where our MILP-based method is able to achieve almost 100% coverage in workspace Fig. 3.6b. The C-space is a subset of $SE(2)$, with two translation axes and a periodic axis corresponding to the rotation. We visualize the collision-free C-space by sampling (left) and the C-IRIS cover acquired from GCS (right).

The major difference between our method and GCS lies in space decomposition: our method is workspace-based while GCS works in C-space. The dimensionality of C-space is usually higher than that of the workspace, accompanied by an inherent geometrical and topological differences between these two spaces. In the demonstrated scenario Fig. 3.8, where the objective is for a stick to navigate out of a trap through a narrow gap. The collision-free C-space consists of two large regions, connected by a thin tunnel. Despite

Table 3.1: Comparison of our MILP-based method to baselines in all benchmark scenarios.

(a) Results for convex objects (A stick in 2D scenarios and a box for 3D).

	Online Time ↓ (ms)			Waypoint Number ↓		
scenario	ours	GCS	PRM	ours	GCS	PRM
corner	**2.8**	607.8	105.3	8	**6**	20
bugtrap	**15.1**	INF	116.4	**21**	INF	40
maze	**57.1**	7.2e5	111.9	29	**21**	42
SCOTS	**55.0**	INF	508.3	**85**	INF	97
2d-peg	**22.1**	INF	122.6	**13**	INF	31
3d-easy	**63.8**	1.2e5	129.3	10	**9**	10
3d-narrow	**212.4**	2.0e5	1045.0	14	32	**12**
3d-peg	**48.1**	INF	287.1	20	INF	**6**

(b) Results for non-convex objects (L-shape for all scenarios).

	Online Time ↓ (ms)			Waypoint Number ↓		
scenario	ours	GCS	PRM	ours	GCS	PRM
corner	**2.8**	4.1e4	105.9	**9**	11	19
bugtrap	**11.5**	INF	113.8	**18**	INF	40
maze	**54.8**	2.6e4	110.4	29	**25**	43
SCOTS	**47.8**	INF	3822.2	**97**	INF	98
2d-peg	**12.8**	INF	115.8	**13**	INF	28
3d-easy	**66.7**	2.1e5	269.3	10	**5**	7
3d-narrow	**257.5**	INF	593.7	16	INF	**11**
3d-peg	**52.1**	INF	824.7	16	INF	**11**

Table 3.2: Comparison of our method with PRM and RRT for an L-shaped object navigating a bugtrap environment Fig. 3.6b. Online time is evaluated in both large and narrow environments. In large settings, the mid-right corridor remains constant while dimensions are doubled. In narrow settings, the corridor width is reduced to 60% of its original size. The online time limit for PRM is 40s.

Time (ms)	ours	PRM-0.25x	PRM-0.1x	PRM-0.05x
original	**11.6**	$113_{\pm 0.5}$	$109_{\pm 1.4}$	$127_{\pm 22.5}$
large	**13.4**	$521_{\pm 823}$	$4.2e3_{\pm 2.2e3}$	$1.5e4_{\pm 1.2e4}$
narrow	**13.0**	$1.4e3_{\pm 2.5e3}$	$5.8e3_{\pm 4.1e3}$	$1.2e4_{\pm 1.0e4}$
L & N	**17.4**	$2.2e4_{\pm 1.5e4}$	$3.7e4_{\pm 5.6e3}$	$3.9e4_{\pm 4.2e2}$

dense sampling reveals the tunnel, GCS, with 42 polytopes, failed to detect the narrow passage, thus incapable of finding a feasible solution for the problem. On the contrary, our method identifies the critical pathways and achieves a near-complete coverage in the workspace. This distinction indicates a challenge in C-space representation, pathways that occupy a small volume in the workspace can correspond to an even smaller fraction of the free space in C-space, considering the increased dimensionality. This observation underscores the necessity of carefully designing sampling strategy to ensure the reliability.

We also conduct a comprehensive comparison on time consumption and problem size for offline stage. The offline stage for our MILP-based method can be roughly divided into two: the construction of G_c (Section 3.1.1) and G_d (Section 3.1.2). The problem size for G_c is the number of IRIS regions ($|\mathcal{V}_c|$) and its computation only needs to be conducted once across different objects in the same environment. But the graph in GCS needs to be recomputed when either the environment or the object changes. It can be observed that the number of IRIS regions we need to grow is significantly lower than GCS, while remaining high feasibility. Moving to G_d, the problem size is determined by the number of MILP required. Despite our MILP-based method necessitates a substantial number of MILPs, the MILPs are designed to be of small sizes, batch processing is highly prefered. In the experiment, we assign 2 threads for each MILP. The offline computation time for G_d scales linearly to the number of MILP. Conversely for GCS, the number of variable is proportional to the number of IRIS region, which makes the online time extremely long for GCS when the number of IRIS region is large.

Across the majority of test cases, our MILP-based method demonstrated reduced offline computation times while maintaining feasibility. However, it's important to note the inherent advantage of GCS in its design for globally optimal solutions. While our MILP-based method focuses on achieving faster computation and higher feasibility, this comes at the sacrifice of optimality.

Table 3.3: Metrics of offline stages for our MILP-based method and GCS.

	Offline Time ↓ (s)			# IRIS regions ↓		# MILP ↓
scenario	G_c	G_d	GCS	ours	GCS	ours
corner	1.1	0.5	23.7	2	24	6
bugtrap	4.1	10.7	21.6	7	42	79
maze	6.1	130.3	251.4	19	131	1624
SCOTS	4.0	45.7	79.3	27	78	355
2d-peg	5.2	18.8	26.5	8	59	170
3d-easy	5.3	15.8	27.8	4	55	42
3d-narrow	7.7	115.5	96.9	8	95	471
3d-peg	16.4	10.1	166.9	4	164	15

3.4 Conclusion

In this study, we introduced a novel hybrid path planning approach that integrates information from both workspace and configuration space. By structuring our method into three distinct stages—constructing a coarse graph of convex sets in the workspace, building a dense graph in the configuration space, and executing an online phase for rapid path retrieval—we have demonstrated improved planning efficiency in multi-query scenarios.

Chapter 4

Planning for Multi-DoF Manipulators

In the realm of robotic manipulation, an object cannot move by itself, but has to be held by a manipulator. In this chapter, we will present a safe controller CBF-induced neural controllerand an associated motion planning framework CBF-INC-RRT.

4.1 Problem Statement and Preliminaries

We consider a robot with control-affine dynamics $\dot{q} = f(q) + g(q)u$, where $q \in \mathcal{C} \subseteq \mathbb{R}^n$ is the robot configuration and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, in a cluttered environment $\mathcal{E}$. We assume both configuration space $\mathcal{C}$ and action space $\mathcal{U}$ to be bounded. The robot perceives the environment $\mathcal{E}$ through an observation model $o = o(q, \mathcal{E}) \in \mathcal{O} \subset \mathbb{R}^k$. In this work, we consider two different observation models, namely, a signed-distance observation model and a LiDAR-based observation model, whose details are elaborated in 4.2.2. The state-observation space $\mathcal{X} := \mathcal{C} \times \mathcal{O}$ can be partitioned into three subspaces: an unsafe set $\mathcal{X}_u$ where the robot collides with or penetrates itself or environmental obstacles, a safe set $\mathcal{X}_s = \{x | \|x - x_u\| \geq r_{\text{thres}}, \forall x_u \in \mathcal{X}_u\}$ where the robot is at least r_{thres} away from the unsafe set, and a boundary set $\mathcal{X}_b = \mathcal{X} \setminus (\mathcal{X}_u \bigcup \mathcal{X}_s)$.

Given a start configuration q_0 and goal configuration q_g, satisfying $(q_0, o(q_0, \mathcal{E}))$, $(q_g, o(q_g, \mathcal{E})) \notin \mathcal{X}_u$, we seek to find a feasible control sequence $\mathbf{u} : [0, T] \to \mathcal{U}$ that steers the system from

q_0 to $q(T) \in \mathcal{X}_{\text{goal}}$, while ensuring $(q(t), o(q(t), \mathcal{E})) \notin \mathcal{X}_u, \forall t \in [0, T]$. The goal region $\mathcal{C}_{\text{goal}}$ is defined as $\{(q, o(q, \mathcal{E})) | \|q - q_g\| \leq r_{\text{goal}}\}$ for some pre-defined radius $r_{\text{goal}} \geq 0$. In this work, we build upon motion planning algorithm RRT [6] and substitute the steer function with a neural-network-based controller for control input.

Rapid-exploring random trees (RRT). RRT [6] tackles the motion planning problem by starting with sampling a set of configurations in the free space $\mathcal{X}_f = \mathcal{X}_s \bigcup \mathcal{X}_b$. The algorithm then attempts to build or expand a tree to connect these nodes with the start configuration q_0 and the goal configuration q_g. This two-step process is repeated until a collision-free path connecting the two configurations is found or until termination. Each edge on the tree has to be collision-free, thus requiring collision checking at the edge construction stage.

Control barrier function. CBF ensures safety in control systems by enforcing the states of the systems to stay in the safe set. Extended CBFs in state-observation space $\mathcal{X}$ are scalar functions $h : \mathbb{R}^n \mapsto \mathbb{R}$ such that for some $\alpha_h > 0$:

$$\begin{aligned}
&\forall (q, o) \in \mathcal{X}_s, h(q, o) \leq -\gamma \\
&\forall (q, o) \in \mathcal{X}_u, h(q, o) > \gamma \\
&\forall (q, o) \in \mathcal{X}, \ \inf_{u \in \mathcal{U}} (L_f h(q, o) + L_g h(q, o) u) + \alpha_h h(q, o) \leq -\epsilon
\end{aligned} \tag{4.1}$$

where $L_f h$ and $L_g h$ denote the Lie derivatives, which capture the rate of change of h along the system trajectories induced by f and g, respectively. Since o is also a function of q, the calculation of Lie derivatives requires the calculation of $\frac{\partial h}{\partial q}\dot{q}$ and $\frac{\partial h}{\partial o}\dot{o}$. And $\epsilon, \gamma > 0$ are small margins to encourage the strict inequality satisfaction of CBF conditions. It is proved in [27] that if the initial state $(q(0), o(0)) \in \mathcal{X}_s, h(q, o) \leq 0$ and a Lipschitz continuous policy $\pi : \mathcal{X} \mapsto \mathcal{U}$ selects actions from the set $\mathcal{K}_{\text{CBF}} = \{u \mid L_f h(q, o) + L_g h(q, o) u + \alpha_h h(q, o) \leq 0\}$, then the trajectory $q(\cdot)$ does not leave the safe set $\mathcal{X}_s$.

Here we can see a connection between motion planning and control barrier function. As long as the controller ensures the forward-invariant of the safety set, then the agent never encounters collisions. Using such a CBF controller ensures the collision-free constraint for motion planning.

4.2 Learning Control Barrier Function for Manipulators

Motivated by the connection between the motion planner's collision-free constraint and CBF's safety guarantees, we present a two-stage approach using a CBF-induced neural controller that allows a robot to avoid obstacles and a motion planner that guides the robot to jump out of stuck regions and toward its goal. We first train a neural network CBF-induced neural function (CBF-INF) for the robot. CBF-INF is trained to satisfy all the constraints in (4.1). Next, we synthesize a controller CBF-INC using the trained CBF-INF and incorporate it into the motion planning framework.

4.2.1 Learning Framework for CBF-INF

Training procedures. We utilize an offline strategy to train CBF-INF $h_\theta(q, o)$, where the training data is pre-collected. Our training dataset is a combination of two different parts, both are collected in various training environments : (i) We gather rollout trajectories using classical controllers (e.g. LQR controller). These trajectories are generated with random initial and goal states. (ii) To ensure coverage of less-explored spaces within the rollout, we uniformly sample the robot's pose in the configuration space. The presence of observation allows that the training environments not necessarily be the same as test ones and that CBF-INF can easily generalize to new environments. CBF-INF is trained to minimize an empirical loss function $\mathcal{L}$ like [25]:

$$\begin{aligned}\mathcal{L} = &\frac{\alpha_1}{N_{\text{safe}}} \sum_{(q,o)\in\mathcal{X}_s} [\gamma + h\,(q,o)]_+ \\ &+ \frac{\alpha_2}{N_{\text{unsafe}}} \sum_{(q,o)\in\mathcal{X}_u} [\gamma - h\,(q,o)]_+ + \frac{\alpha_3}{N}\cdot \\ &\sum_{(q,o)\in\mathcal{X}} [\epsilon + L_f h(q,o) + \inf_{u\in\mathcal{U}} (L_g h(q,o)\cdot u) + \alpha_h h]_+\end{aligned} \tag{4.2}$$

where $\alpha_1, \alpha_2, \alpha_3$ are positive tuning parameters, $[\cdot]_+$ is $\max(0, \cdot)$, N_{safe}, N_{unsafe} and N are the number of points in the training samples in $\mathcal{X}_s$, $\mathcal{X}_u$ and $\mathcal{X}$, respectively. As the term $\inf_{u \in \mathcal{U}}(L_g h(q, o) \cdot u)$ is linearly dependent on u, the minimum value can be easily found within a bounded action space $\mathcal{U}$ via linear programming (LP). The existence of a feasible control signal in the last condition in (4.1) can be demonstrated when the third loss term comes to 0. Note that CBF-INF is trained as a Neural Network from finite samples and therefore not a valid CBF before being verified to satisfy the CBF constraints over the entire space. The latter is a theoretically hard problem [74]. However, the learned CBF-INF can be used as a steering function and provides significant empirical improvements over vanilla RRT (shown in Sec. **??**). We will also show in Fig 4.4 that the empirical satisfaction rates of the CBF constraints over finite samples are close to 100%.

Computing Lie-derivatives. Directly computing the third condition in Eq. (4.1) requires the calculation of $\frac{\partial h}{\partial q}\dot{q}$ and $\frac{\partial h}{\partial o}\dot{o}$. We first assume the obstacle velocities are much smaller than the links of manipulators in the dynamic scenarios, so we can disregard the change of o in between two computational steps induced by the change of environment $\mathcal{E}$ when computing Lie-derivatives. Furthermore, we replace the exact calculation of $\frac{\partial h}{\partial q}$ with numerical differentiation, i.e., $[\frac{\partial h}{\partial q}]_i = \frac{h(q+e_i \cdot \epsilon, o) - h(q,o)}{\epsilon}$, where e_i is a one-hot vector with $e_i[i] = 1$. This approach bypasses the explicit expression of forward kinematics of manipulators with many degrees of freedom and only demands a "black-box" access to the numerical values of the kinematics and the Jacobian [20].

4.2.2 Specializing Functions for Manipulators: State-based and LiDAR-based CBF-INF

In contrast to hand-crafted CBFs utilized on multi-DoF robotic manipulators in [14], [16], [30], we aim to learn a neural function that encodes the safety constraint of avoiding both self-collision and collision with the exterior environment. Based on different observation models, we propose two types of CBF-INF, both sharing the same training procedure.

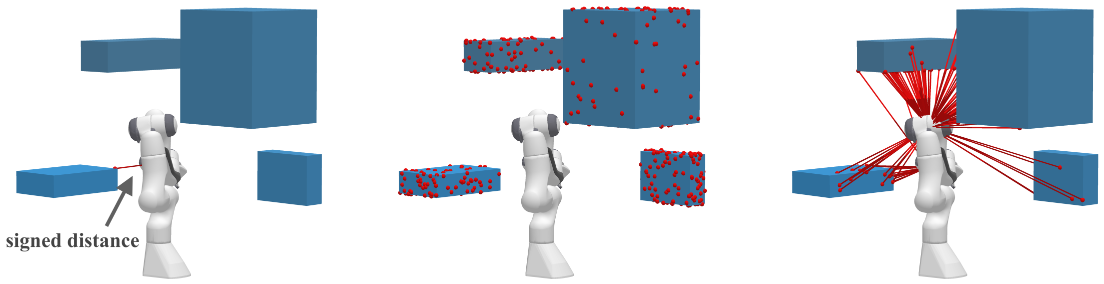


Figure 4.1: Illustration of observations. Left: sCBF-INF takes the signed distance to the nearest obstacle as observation. Middle: For *fully-observable* environments, oCBF-INF uses a point cloud sampled uniformly on the obstacles as observation. Right: For *partially-observable* environments, oCBF-INF observes the point cloud from a mounted LiDAR sensor.

State-based CBF-INF (sCBF-INF). The observation o in sCBF-INF is the minimum signed distance, d, between the obstacles and any robot links. In this setting, it is easy for the CBF-INF to determine, from the sign of d, whether the robot is in a collision-free state with respect to the environment. The only remaining challenge is to learn the self-collision pattern, which solely depends on the configuration q. To address this, we design the neural network to take the concatenated vector q and d as input, and generate a scalar as output, as in Fig. 4.2.

LiDAR-based CBF-INF (oCBF-INF). sCBF-INF relies heavily on accurate distance information from the environment, which is not available or costly to acquire in a dynamic or partially observable environment. A more flexible implementation is to define CBF-INF as functions of partial observations, e.g., LiDAR. In the LiDAR-based setting, the observation o is composed of N raw points sampled on the surface of obstacles paired with their respective normal vectors, similar to [59]. This observation is represented by a finite set $o = \{(p_i, n_i)\}_{i\in[1,\cdots,N]} \in \mathbb{R}^{N\times 6}$, with the 3 dimensional points p_i and 3 dimensional normal vector n_i in the world frame. The point cloud can be retrieved using the LiDAR sensor or a depth camera mounted on the robot.

For each link of the manipulator, we transform the point cloud into its local frame, concatenate each point with a one-hot vector of the link index, and then feed all the transformed point clouds into a PointNet [75], which encodes the point clouds while ensuring permutation

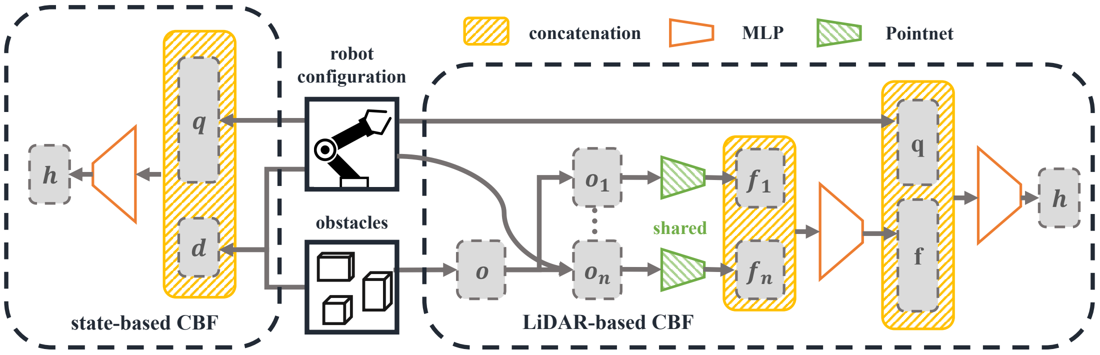


Figure 4.2: The overall neural network architecture. Left: The architecture of the sCBF-INF. Right: The architecture of the oCBF-INF.

invariance on the order of points. The feature vectors from the PointNet are further fed into an MLP, concatenated with the configuration q. The whole network architecture is shown in Fig. 4.2.

4.3 Planning with Learned CBF

Sampling-based motion planning algorithms are widely adopted to efficiently search high-dimensional spaces via building a space-filling tree. Despite their guaranteed completeness, these algorithms explore the configuration space via random shooting, and fine-grained collision-checking is required for each edge. The edge will be discarded if any part of the checking fails, even though a slight detour may save the edge, which wastes the computation in such a failed exploration. Our framework, however, encourages the planner to explore the configuration space more wisely by using safe controllers, i.e., CBF-INC. By incorporating the controller into the motion planning algorithm, the likelihood of successfully expanding a node is increased, which reduces the exploration cost given the controller's reactivity to obstacles.

4.3.1 Synthesize controller.

We construct our controller CBF-INC from CBF theory [27] that modifies any given reference controller u_{nominal} by solving the quadratic programming (QP) problem:

$$\begin{aligned} u(q, o, q_g) &= \arg\min_{u \in \mathcal{U}} \|u - u_{\text{nominal}}(q, o, q_g)\|^2, \\ \text{s.t. } & L_f h(q, o) + L_g h(q, o) \cdot u + \alpha h \leq 0. \end{aligned} \tag{4.3}$$

Here, the controller seeks to minimize the squared difference with the nominal control input within the action space $\mathcal{U}$, while satisfying the safety constraint.

4.3.2 Safe-steering RRT.

In this study, we focus on Rapidly-exploring Random Trees (RRT) [6] as a representative sampling-based motion planner. In RRT, the steer function generates a prospective edge, which extends the current search tree toward the direction of a randomly selected point. The steer function needs to conduct collision checking in the process. Different from [30], which substitutes explicit checking for the nearest neighbor and changes the original framework, we propose CBF-INC-RRT, to use CBF-INC as the steer function. This is a general approach that can be applied to any sampling-based motion planners using a steer function. Within our steer function, the robot rolls out a trajectory using CBF-INC. The q_g in (4.3) is set to be the newly sampled point. The generated trajectory and control sequence then serve as the edge added to the search tree. Details of the complete algorithm are provided in the Appendix.

There may be concerns about the completeness of this modified planning paradigm. However, we can still ensure completeness by opting to use CBF-INF to discard unsafe LQR actions instead of modifying them after a certain number of exploration steps, as suggested in [32]. This optional variation allows us to maintain the crucial aspect of completeness while improving safety and efficiency.

4.4 Experiments

Experiment setup. We evaluate our methods on a 4-DoF Dobot Magician in simulation and a 7-DoF Franka Panda both in simulation and the real world. Similar to [14], we consider direct control over the joint velocities, i.e., $\dot{q} = u$. In experiments, obstacles within the environment are depicted as cuboids. CBF-INF is trained in environments with 4 fixed-size obstacles with random poses. We test the method on more challenging environments with 8 obstacles of random sizes and poses unless specified otherwise. The nominal policy u_{nominal} for the QP controller is selected as LQR. The simulations of continuous-time robot dynamics and control frequency occur at 120 Hz and 30 Hz, respectively.

Baselines. Apart from vanilla RRT (RRT) [6], we also compare against RRT variants with the following steer controllers in both state-based and LiDAR-based settings:

- **Reinforcement Learning (sRL&oRL)[76]:** We design the reward function to encourage goal-reaching and penalize collision, then train the controller with DDPG.
- **Imitation Learning (sIL&oIL)[77]:** Use behavior cloning to mimic the planned trajectories generated from an expert motion planner BIT* [78].

Some baseline methods require complete information of the environment, thus only available in the state-based setting:

- **Hand-crafted CBF (hCBF) [15]:** The construction of this CBF adopts the minimum uniform scaling factor. Only available for 7-DoF Panda robot.
- **Safe RL method OptLayer (sOpt) [53]:** Add additional optimization layer to force the controller satisfy 4.1, where h is a signed distance function instead of CBF.

The baseline methods with steer controllers are abbreviated with a suffix '-steer'. We also conduct ablation studies evaluating the controllers only. All neural controllers are designed and trained using the same environment observations and neural network architectures. The methods are evaluated on randomly generated 1000 easy and 1000 hard testing cases, based on the time required for BIT* [78] to find a solution. All the experiments are conducted with a predefined node limit: for the 4-DoF robot, exploration is restricted to a maximum of 200 nodes, while the limit for the 7-DoF robot is 500 nodes.

4.4.1 Motion Planning in Simulation

Evaluation metrics. For motion planning problems in the section, we consider the following metrics: (1) Success rate (SR): A problem is successfully solved only if a collision-free path is found within the node limit. (2) Explored nodes: the attempts of adding a node to the search tree, regardless of success or not. (3) Total time consumption: One common concern about learning-based methods is their running speed due to the frequent calling of a large neural network model at inference time. We separate this metric into two categories: (3.1) online time, which is directly related to control frequency and observation update frequency and must be performed online during execution. This includes neural network inference time, QP solving, and perceiving observations; and (3.2) planning time, proportional to the total timesteps when expanding the search tree, is the remaining time consumption other than online time. This includes planning, running the simulations, and performing collision checking. Because online time highly depends on selected parameters, we only report planning time in the main text. Results for online time can be found in the supplementary.
Performance of state-based methods. Shown in Table 4.1, we see significant improvement in success rate and exploration efficiency (explored nodes) using sCBF-INC-RRT in the state-based setting. Remarkably, performance improvement is much more pronounced on challenging hard testing problems. Regarding planning time, sCBF-INC-RRT performs comparably with vanilla RRT and takes considerably less time compared to sRL-steer, sIL-steer methods, and even safe method sOptLayer-steer. It's worth discussing why hCBF-steer performs much worse than CBF-INC-RRT and even RRT. First, hCBF is more conservative because it over-approximates the geometry shapes of the robot. This limits its performance,

Table 4.1: Experiments under state-based setting. Performance on average success rate (SR), number of explored nodes on 1000 test cases, and summed planning time on 100 testing cases, averaged over 3 random seeds.

(a) Results on 4-DoF Magician robot.

	Easy			Hard		
method	SR↑ (%)	nodes↓	time(s)	SR↑ (%)	nodes↓	time(s)
RRT	91.2	37.4	**42.3**	68.1	81.2	**43.3**
sCBF-INC-RRT	**97.7**	**24.1**	45.8	**84.6**	**54.6**	45.3
sIL-steer	89.4	43.0	55.2	58.2	99.6	80.1
sRL-steer	90.3	40.4	51.5	60.8	94.0	73.4
sOpt-steer	84.6	51.2	65.7	64.5	93.2	70.6

(b) Results on 7-DoF Franka Panda robot.

	Easy			Hard		
method	SR↑ (%)	nodes↓	time(s)	SR↑ (%)	nodes↓	time(s)
RRT	85.7	112.3	166.0	62.8	252.5	**278.6**
sCBF-INC-RRT	**92.0**	**67.5**	**160.5**	**76.1**	162.5	345.8
hCBF-steer	39.2	134.1	472.6	26.3	**160.5**	525.3
sIL-steer	39.1	324.9	459.0	25.3	400.5	547.2
sRL-steer	83.9	124.9	235.9	60.1	266.7	395.3
sOpt-steer	26.5	155.3	902.8	16.4	174.9	942.3

especially in cluttered environments. Second, the QP controller in hCBF-steer sometimes cannot find a feasible solution. Although we've attempted to relax the optimization problem with a constraint violation penalty term, this compromises the safety guarantee of hCBF-steer.

Performance of LiDAR-based methods. We evaluate algorithms that integrate various controllers into steer functions (e.g., oCBF-INC-RRT) and those that solely leverage controllers to address the planning problems (e.g., oCBF-INC). The experiments are conducted in fully-observable environments. In Fig. 4.3, we show that motion planning methods significantly outperform end-to-end controllers regarding success rate, demonstrating that motion planning can help improve the feasibility of finding a solution under QP formulations. Among all the motion planning methods, oCBF-INC-RRT outperforms oRL-steer and oIL-steer on

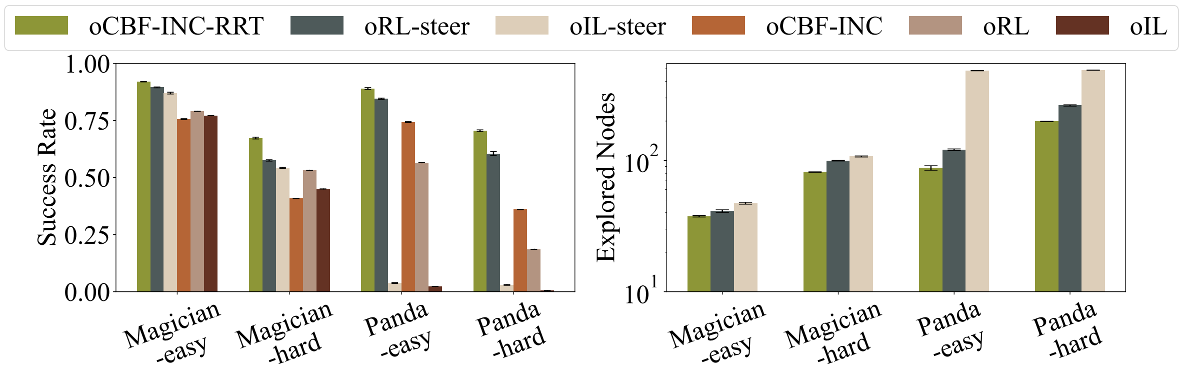


Figure 4.3: Motion planning experiments under LiDAR-based setting.

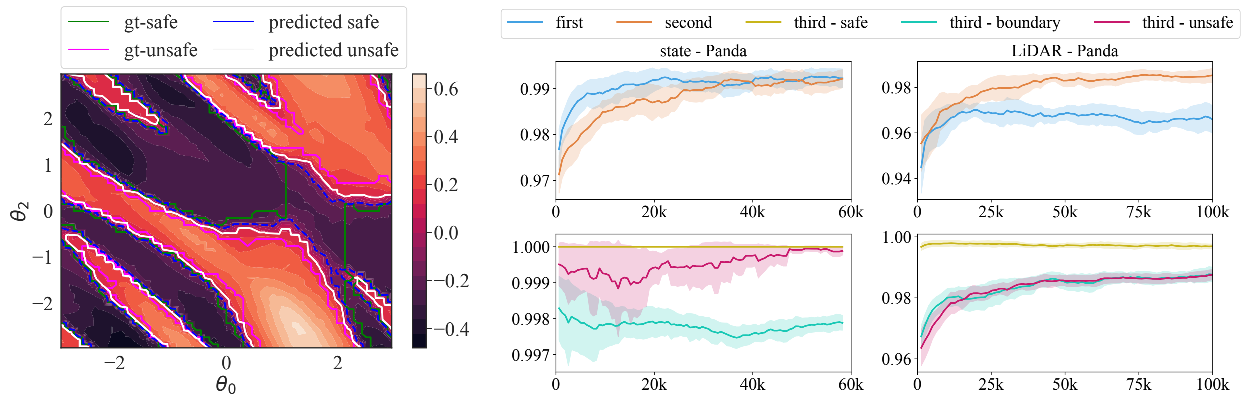


Figure 4.4: Left: Slices of oCBF-INF value for Panda, obtained by sweeping across two joints, with all other joint states and obstacle positions held constant. Right: Learning curves of constraint satisfaction rate on Panda.

success rate and number of explored nodes, especially in hard tests. Regarding the total time consumption, our method does take a slightly longer time than vanilla RRT due to frequent inference calls of neural network. However, our method takes about 0.15 s and 0.32 s on average to compute the control signals and step the simulation for a 1-second period on Dobot Magician and Franka Panda, respectively. This indicates our planning can be performed faster than real-time, further establishing applicability to the real world.

4.4.2 Ablation Study in Simulation

We first visualize CBF contour in configuration space in Fig 4.4. We also plot the learning curves of satisfaction rates of each CBF constraint on our neural CBF-induced neural controller-enhanced RRT. All the constraints are satisfied over 99% and 97% on the validation sets for state-based and LiDAR-based settings after training, respectively.

We then evaluate our controller oCBF-INC, by unrolling its control output without integrating it into the motion planning framework, over 1000 planning problems with 6 obstacles. This experiment showcases how different neural controllers balance goal-reaching and safety. We conduct experiments in two distinct environments: **(i)** the environment is static and fully observable, where the observation contains 1024 points uniformly sampled on the surfaces of obstacles; **(ii)** obstacles are dynamic and move at a constant speed and the environment is only partially observable, where the observation is acquired by two 3D LiDARs mounted on the manipulators.

Evaluation metrics. We evaluate the end-to-end controllers based on three measures averaged over the testing problems: (1) goal-reaching rate: A goal configuration is identified as reached, only if the agent does not encounter any collision during the rollout, and eventually reaches the goal within a limited time horizon. (2) safety rate: the ratio of collision-free states along the entire trajectory. (3) makespan: rollout steps for succeeded cases.

Performance. Fig 4.5 shows the performance of both Dobot Magician and Franka Panda robots in the considered environments. In the static and fully observable environment, oCBF-INC outperforms baselines by approximately 2% on the goal-reaching rate and safety rate for the Magician robot and by more than 15% for Franka Panda. Although all algorithms face a substantial performance drop in the dynamic and partial-observable environment, our controller still notably exceeds oRL and oIL baselines. The makespan performances are generally comparable across algorithms, while oCBF-INC is slightly better. This demonstrates the method achieves a great balance between efficiency and safety.

4.4.3 Hardware Demonstration

Finally, we validate our proposed method on a real Franka Emika Panda controlled at 30 Hz, the same as in the simulation. We randomly select several planning problems in 4.4.1. In

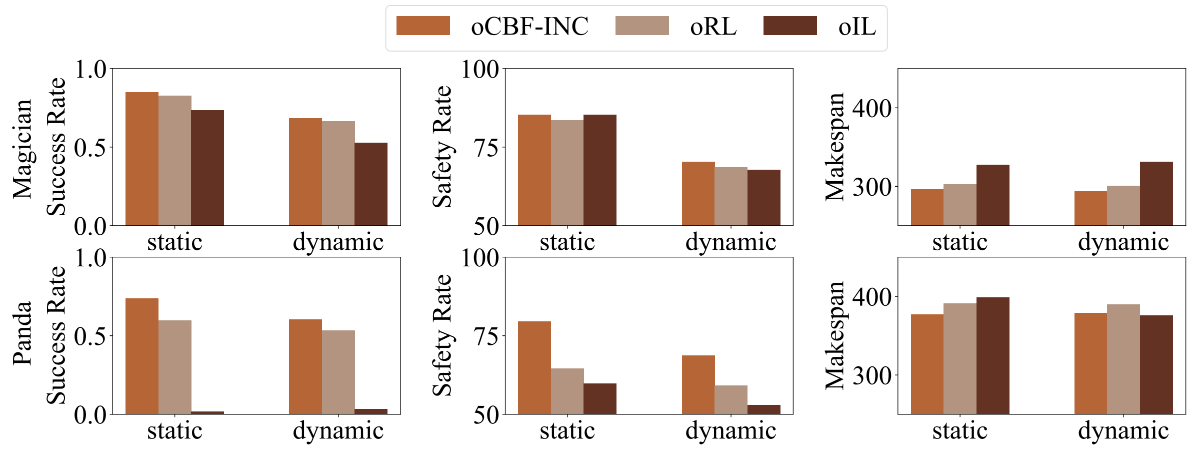


Figure 4.5: Safety and goal-reaching performance of LiDAR-based controllers in an end-to-end manner (without motion planning module).

order to avoid the floating blocks, we construct the obstacles and vision modules in the simulation, then synthesize real-world video with simulated obstacles, similar to [15]. The components are communicated via ROS.

As shown in Fig 4.6 and supplementary video, our method solves the planning problems successfully. On the right of Fig 4.6, we also show the signed distance of the robot to the environment. The experiments confirm that the planned trajectory is safe and robust to the noise in execution.

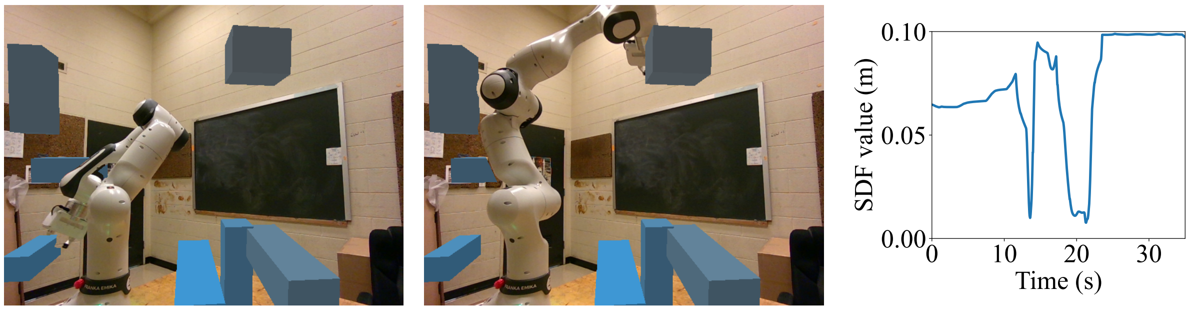


Figure 4.6: Left & middle: snapshots of solving a motion planning problem with our method. Right: the curve of minimum signed distance to all obstacles along a trajectory. Videos are included in the supplementary.

4.5 Conclusion

This paper explores a direction for robotic safety control by integrating CBF-induced neural controller - CBF-INC into motion planning. Instead of looking for a certified CBF, we train CBF-INF for robotic manipulators under different observation settings and incorporate the synthesized controller into sampling-based motion planning algorithms. We evaluate the proposed methods in various environments, including 4-DoF and 7-DoF arms, and in the real world. We demonstrate that CBF-INC generalizes well to unseen scenarios and the overall framework outperforms other methods in terms of goal-reaching rate and exploration efficiency.

However, there are several limitations of our paper: (1) Since oCBF-INF takes the raw sensor data as input, the performance is directly dependent on the sensor data quality. Quantifying the input cloud's uncertainty precisely remains an open-ended problem. (2) oCBF-INF requires transforming the input point cloud into each link frame, which poses potential scalability issues for robots with higher DoF. (3) Our computation of the Lie derivative assumes the moving speeds of obstacles are small in dynamic scenarios. We hope to relax this assumption in future work.

Chapter 5

Conclusion and Future Work

In this thesis, we study path planning problems in manipulation, focusing on rigid-body objects and manipulators, separately. For rigid-body objects, we present a MILP-based approach in Chapter 3, which decomposes the free workspace into a graph of convex polytopes and generates viable path segments between polytopes via MILP. The path segments form a graph and the graph can serve as an offline roadmap, thus allowing fast online query. For multi-dof manipulators, we propose CBF-INC-RRT in Chapter 4, which first train a neural CBF for the manipulator, which can take raw observation like point cloud as input. We then synthesize a safe controller with QP, which serve as a steering function in sampling-based motion planning algorithm, RRT, to allow for more effecient exploration in safe space. We provide extensive experimental results to demonstrate the effectiveness of the proposed algorithms over baseline methods on the path planning problems on two types of systems.

After addressing the challenges for rigid-body objects and manipulators separately, being able to efficiently planning and controlling objects with manipulators is crucial to real-world applications. In future works, we would like to explore the planning process of robotic manipulation in crowded environments especially in assembly process when the object is held by the manipulator. The configuration space is of higher dimension, and is a product of

$\mathbb{R}^n$ and $SE(3)$. We would also like to leverage the planning outcome to detect infeasibility in motion planning. Upon detection, we would like to explore the direction of incorporating high-level task planning to interact with obstacles or multi-arm collaboration, thereby resolving scenarios that are initially infeasible.

References

[1] J. T. Schwartz and M. Sharir, "On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on pure and applied mathematics*, vol. 36, no. 3, pp. 345–398, 1983.

[2] 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, no. 4, pp. 566–580, 1996. DOI: 10.1109/70.508439.

[3] M. Vitus, V. Pradeep, G. Hoffmann, S. Waslander, and C. Tomlin, "Tunnel-milp: Path planning with sequential convex polytopes," in *AIAA guidance, navigation and control conference and exhibit*, p. 7132.

[4] M. d. S. Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-Free Encoding for Chance-Constrained Nonconvex Path Planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019. DOI: 10.1109/TRO.2018.2878996.

[5] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, "Finding and optimizing certified, collision-free regions in configuration space for robot manipulators," in *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2022, pp. 328–348.

[6] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[7] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 5054–5061.

[8] R. Kala, "Rapidly exploring random graphs: Motion planning of multiple mobile robots," *Advanced Robotics*, vol. 27, no. 14, pp. 1113–1122, 2013.

[9] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 5539–5544.

[10] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural rrt*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[11] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9535–9541.

[12] G. Wu and K. Sreenath, "Safety-critical control of a planar quadrotor," in *2016 American control conference (ACC)*, IEEE, 2016, pp. 2252–2258.

[13] Q. Nguyen, A. Hereid, J. W. Grizzle, A. D. Ames, and K. Sreenath, "3d dynamic walking on stepping stones with control barrier functions," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 827–834.

[14] A. Singletary, W. Guffey, T. G. Molnar, R. Sinnet, and A. D. Ames, "Safety-critical manipulation for collision-free food preparation," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 954–10 961, 2022.

[15] B. Dai, R. Khorrambakht, P. Krishnamurthy, V. Gonçalves, A. Tzes, and F. Khorrami, "Safe navigation and obstacle avoidance using differentiable optimization based control barrier functions," *arXiv preprint arXiv:2304.08586*, 2023.

[16] C. T. Landi, F. Ferraguti, S. Costi, M. Bonfè, and C. Secchi, “Safety barrier functions for human-robot interaction with industrial manipulators,” in *2019 18th European Control Conference (ECC)*, 2019, pp. 2565–2570. DOI: 10.23919/ECC.2019.8796235.

[17] C. Chang, M. J. Chung, and B. H. Lee, “Collision avoidance of two general robot manipulators by minimum delay time,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 3, pp. 517–522, 1994. DOI: 10.1109/21.279000.

[18] E. Rimon and S. P. Boyd, “Obstacle collision detection using best ellipsoid fit,” *Journal of Intelligent and Robotic Systems*, vol. 18, pp. 105–126, 1997.

[19] H.-C. Lin, Y. Fan, T. Tang, and M. Tomizuka, “Human guidance programming on a 6-dof robot with collision avoidance,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2676–2681. DOI: 10.1109/IROS.2016.7759416.

[20] A. Singletary, P. Nilsson, T. Gurriet, and A. D. Ames, “Online active safety for robotic manipulators,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 173–178. DOI: 10.1109/IROS40897.2019.8968231.

[21] A. Singletary, S. Kolathaya, and A. D. Ames, “Safety-critical kinematic control of robotic systems,” *IEEE Control Systems Letters*, vol. 6, pp. 139–144, 2022. DOI: 10.1109/LCSYS.2021.3050609.

[22] F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, “Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics,” in *2021 American Control Conference (ACC)*, IEEE, 2021, pp. 3683–3690.

[23] D. Sun, S. Jha, and C. Fan, “Learning certified control using contraction metric,” in *Conference on Robot Learning*, PMLR, 2021, pp. 1519–1539.

[24] Z. Qin, K. Zhang, Y. Chen, J. Chen, and C. Fan, “Learning safe multi-agent control with decentralized neural barrier certificates,” *arXiv preprint arXiv:2101.05436*, 2021.

[25] C. Dawson, B. Lowenkamp, D. Goff, and C. Fan, "Learning safe, generalizable perception-based hybrid control with certificates," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1904–1911, 2022.

[26] S. Dean, N. Matni, B. Recht, and V. Ye, "Robust guarantees for perception-based control," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 350–360.

[27] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278. DOI: 10.1109/CDC.2014.7040372.

[28] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[29] A. Manjunath and Q. Nguyen, "Safe and robust motion planning for dynamic robotics via control barrier functions," in *2021 60th IEEE Conference on Decision and Control (CDC)*, IEEE, 2021, pp. 2122–2128.

[30] G. Yang, B. Vang, Z. Serlin, C. Belta, and R. Tron, "Sampling-based motion planning via control barrier functions," in *Proceedings of the 2019 3rd International Conference on Automation, Control and Robots*, 2019, pp. 22–29.

[31] A. Ahmad, C. Belta, and R. Tron, "Adaptive sampling-based motion planning with control barrier functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, IEEE, 2022, pp. 4513–4518.

[32] G. Yang, M. Cai, A. Ahmad, C. Belta, and R. Tron, "Efficient lqr-cbf-rrt*: Safe and optimal motion planning," *arXiv preprint arXiv:2304.00790*, 2023.

[33] D. Nister, J. Soundararajan, Y. Wang, and H. Sane, *Nonholonomic Motion Planning as Efficient as Piano Mover's*, 2023. arXiv: 2306.01301 `[cs.RO]`.

[34] M. McNaughton, *Parallel algorithms for real-time motion planning*. Carnegie Mellon University, 2011.

[35] A. Kelly, A. Stentz, O. Amidi, M. Bode, D. Bradley, A. Diaz-Calderon, M. Happold, H. Herman, R. Mandelbaum, T. Pilarski, *et al.*, "Toward reliable off road autonomous vehicles operating in challenging environments," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 449–483, 2006.

[36] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[37] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[38] L. P. Chew, "Constrained delaunay triangulations," in *Proceedings of the third annual symposium on Computational geometry*, 1987, pp. 215–222.

[39] D. Demyen and M. Buro, "Efficient triangulation-based pathfinding," in *AAAI*, vol. 6, 2006, pp. 942–947.

[40] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2015, pp. 109–124.

[41] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, "Shortest paths in graphs of convex sets," *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.

[42] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[43] S. Quinlan, *Real-time modification of collision-free paths*. Stanford University, 1995.

[44] F. Schwarzer, M. Saha, and J.-C. Latombe, "Adaptive dynamic collision checking for single and multiple articulated robots in complex environments," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 338–353, 2005.

[45] S. Redon, A. Kheddar, and S. Coquillart, "An algebraic solution to the problem of collision detection for rigid polyhedral objects," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 4, 2000, 3733–3738 vol.4. DOI: 10.1109/ROBOT.2000.845313.

[46] X. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using taylor models and temporal culling," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, 15–es, 2007.

[47] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[48] A. De Santis, A. Albu-Schaffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," in *2007 IEEE/ASME international conference on advanced intelligent mechatronics*, 2007, pp. 1–6. DOI: 10.1109/AIM.2007.4412606.

[49] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345. DOI: 10.1109/ICRA.2012.6225245.

[50] P. Holmes, S. Kousik, B. Zhang, D. Raz, C. Barbalata, M. Johnson-Roberson, and R. Vasudevan, "Reachable sets for safe, real-time manipulator trajectory design," *arXiv preprint arXiv:2002.01591*, 2020.

[51] P. Wieland and F. Allgöwer, "Constructive safety using control barrier functions," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 462–467, 2007.

[52] J. Haviland and P. Corke, "Neo: A novel expeditious optimisation algorithm for reactive motion control of manipulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1043–1050, 2021. DOI: 10.1109/LRA.2021.3056060.

[53] T.-H. Pham, G. De Magistris, and R. Tachibana, “Optlayer-practical constrained optimization for deep reinforcement learning in the real world,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6236–6243.

[54] M. Koptev, N. Figueroa, and A. Billard, “Neural joint space implicit signed distance functions for reactive robot manipulator control,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2023. DOI: 10.1109/LRA.2022.3227860.

[55] P. Liu, K. Zhang, D. Tateo, S. Jauhri, Z. Hu, J. Peters, and G. Chalvatzaki, “Safe reinforcement learning of dynamic high-dimensional robotic tasks: Navigation, manipulation, interaction,” *arXiv preprint arXiv:2209.13308*, 2022.

[56] S. McIlvanna, N. N. Minh, Y. Sun, M. Van, and W. Naeem, “Reinforcement learning-enhanced control barrier functions for robot manipulators,” *arXiv preprint arXiv:2211.11391*, 2022.

[57] B. Ichter, J. Harrison, and M. Pavone, “Learning sampling distributions for robot motion planning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 7087–7094.

[58] T. Jurgenson and A. Tamar, “Harnessing reinforcement learning for neural motion planning,” *arXiv preprint arXiv:1906.00214*, 2019.

[59] R. Strudel, R. G. Pinel, J. Carpentier, J.-P. Laumond, I. Laptev, and C. Schmid, “Learning obstacle representations for neural motion planning,” in *Conference on Robot Learning*, PMLR, 2021, pp. 355–364.

[60] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao, “Learning-based motion planning in dynamic environments using gnns and temporal encoding,” in *Advances in Neural Information Processing Systems*, 2022.

[61] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.

[62] S. M. Persson and I. Sharf, “Sampling-based a* algorithm for robot path-planning,” *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 2014.

[63] C. Yu and S. Gao, “Reducing collision checking for sampling-based motion planning using graph neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[64] C. Zhang, J. Huh, and D. D. Lee, “Learning implicit sampling distributions for motion planning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3654–3661.

[65] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, “From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots,” in *2017 ieee international conference on robotics and automation (icra)*, IEEE, 2017, pp. 1527–1533.

[66] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, “Motion planning networks,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 2118–2124.

[67] X. Wang, “Ensuring safety of learning-based motion planners using control barrier functions,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4773–4780, 2022.

[68] P. Werner, A. Amice, T. Marcucci, D. Rus, and R. Tedrake, “Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs,” *arXiv preprint arXiv:2310.02875*, 2023.

[69] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023.

[70] A. A. Hagberg, D. A. Schult, and P. J. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX,” in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.

[71] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, “Motion planning around obstacles with convex optimization,” *Science robotics*, vol. 8, no. 84, eadf7843, 2023.

[72] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, Dec. 2012, https://ompl.kavrakilab.org. DOI: 10.1109/MRA.2012.2205651.

[73] M. Rungger and M. Zamani, "SCOTS: A Tool for the Synthesis of Symbolic Controllers," in *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '16, Vienna, Austria: Association for Computing Machinery, 2016, pp. 99–104, ISBN: 9781450339551. DOI: 10.1145/2883817.2883834.

[74] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100 270, 2020.

[75] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[76] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[77] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[78] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Bit*: Batch informed trees for optimal sampling-based planning via dynamic programming on implicit random geometric graphs," *arXiv preprint arXiv:1405.5848*, 2014.