Towards Generalizable Motion Planning: Efficient and Safe Learning-Based Frameworks

by

Mehran Ghafarian Tamizi

M.Sc., University of Tehran, 2020

B.Sc., Ferdowsi University of Mashhad, 2017

A Dissertation Submitted in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/ Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Towards Generalizable Motion Planning: Efficient and Safe Learning-Based Frameworks

by

Mehran Ghafarian Tamizi

M.Sc., University of Tehran, 2020

B.Sc., Ferdowsi University of Mashhad, 2017

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor

(Department of Mechanical Engineering)

Dr. Elizabeth Croft, Departmental Member

(Department of Electrical and Computer Engineering)

Dr. Hong Chuan Yang, Departmental Member

(Department of Electrical and Computer Engineering)

Dr. Teseo Schneider, Outside Member

(Department of Computer Science)

ABSTRACT

Robotic motion planning remains a fundamental challenge in industrial automation, with manipulators offering a clear example of the need for real-time, collision-free, and safe trajectory generation. Traditional planners often face trade-offs among optimality, adaptability, and computational efficiency, limiting their applicability in cluttered and high-dimensional industrial environments. Furthermore, most learning-based planners suffer from poor generalization, requiring retraining when deployed in new scenes or on different robot platforms. This thesis presents two learning-based frameworks designed to address these challenges. First, the Path Planning and Collision Checking Network (PPCNet) is introduced, an end-to-end neural architecture that combines a waypoint generator with a learned collision checker to enable fast, safe, and reliable planning in structured environments. PPCNet is validated in both simulated and real-world bin-picking tasks, demonstrating substantial speedups over classical planners while maintaining path quality. To overcome the generalization limitations of PPCNet, Generalizable and Adaptive Diffusion-Guided Environment-aware Trajectory generation (GADGET) is proposed, a conditional diffusion-based motion planner guided by control barrier functions. GADGET leverages voxel-based scene encoding and goal conditioning to generate safe trajectories across previously unseen environments and robotic arms without retraining. The integration of barrier-function-based guidance enables robust collision avoidance during trajectory generation. Extensive experiments demonstrate that both frameworks achieve real-time planning performance and high success rates, with GADGET offering strong generalization to novel settings. This work highlights the potential of combining deep generative models with adaptable design to create scalable and broadly generalizable motion planners, capable of transferring across diverse environments and robot platforms with minimal modification.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

This thesis, the result of four unforgettable years, could not have been completed without the support of many wonderful people. As I look back, I am deeply grateful to those who shaped my journey, both personally and academically. I hope these words, though brief, convey my sincere appreciation for all they have done.

First and foremost, I want to express my heartfelt gratitude to my supervisor, Dr. Homayoun Najjaran. His trust, encouragement, and constant support have been the foundation of my PhD journey. He not only guided me with wisdom and patience but also inspired me to remain curious, creative, and resilient. Beyond being an excellent mentor, his kindness and genuine care made this path much brighter, and for that I will always be deeply thankful.

I am also truly grateful to my colleagues and friends, Amir Soufi and Homayoun Honari, whose generous help and encouragement were a source of strength throughout this process. Without them, preparing this thesis would have been far more difficult. To all my friends at the ACIS Lab, thank you for creating an environment filled with energy, collaboration, and friendship. The lab became more than a workplace, it was a community where I grew not only as a researcher but also as a person. Finally, I would like to extend my sincere thanks to Aleksey Nozdryn-Plotnicki from Apera AI, whose guidance and support greatly enriched my invaluable internship experience at Apera AI.

I owe my deepest gratitude to my parents and my brother, whose unwavering love and emotional support sustained me throughout the challenges of pursuing a PhD abroad. Their constant encouragement, patience, and belief in me gave me the strength to overcome difficulties and stay focused on my goals. I am also profoundly thankful to my partner, Tina, whose companionship, kindness, and understanding have been a source of comfort and strength throughout this journey. This achievement would not have been possible without their sacrifices and endless support.

Above all, I feel fortunate to have walked this path surrounded by such generous, kind, and inspiring people. This thesis is a reflection not only of my own efforts but also of the many hearts and hands that carried me through. As Isaac Newton once said, “If I have seen further, it is by standing on the shoulders of giants.”

DEDICATION

To my homeland, which, despite its struggles, continues to inspire me with its strength, beauty, and unbreakable spirit.

PREFACE

The contents of this thesis were developed in the Advanced Control and Intelligent Systems (ACIS) Laboratory at the University of Victoria, with support from Apera AI and the Mathematics of Information Technology and Complex Systems (MITACS) under IT16412 Mitacs Accelerate. Chapter 2 includes material published in the International Journal of Intelligent Robotics and Applications. Chapter 3 is based on work published in Robotica journal, and finally Chapter 4 contains material submitted to Robotic and Autonomous Systems journal.

- **Mehran Ghafarian Tamizi**, Marjan Yaghoubi, and Homayoun Najjaran. "A review of recent trends in motion planning of industrial robots." International Journal of Intelligent Robotics and Applications 7.2 (2023): 253-274.
- **Mehran Ghafarian Tamizi**, Homayoun Honari, Aleksey Nozdryn-Plotnicki, and Homayoun Najjaran. "End-to-end deep learning-based framework for path planning and collision checking: bin-picking application." Robotica 42, no. 4 (2024): 1094-1112.
- **Mehran Ghafarian Tamizi**, Homayoun Honari, Amir Mehdi Soufi Enayati, Aleksey Nozdryn-Plotnicki, and Homayoun Najjaran. "A Cross-Environment and Cross-Embodiment Path Planning Framework via a Conditional Diffusion Model." arXiv preprint arXiv:2510.19128 (2025).

In these works, I, Mehran Ghafarian Tamizi, was primarily responsible for theoretical development, programming implementation, experimental design, and writing the majority of the manuscript text. Homayoun Honari contributed to research development, manuscript drafting, and programming support for Chapters 3 and 4. Amir

Mehdi Soufi Enayati contributed to research development in Chapter 4 and the real-world task implementation. Dr. Homayoun Najjaran provided supervision at every stage of the research and manuscript revision.

In addition, during my PhD I had the opportunity to contribute to other projects published in leading robotics conferences, including ICRA and IROS:

- Homayoun Honari, **Mehran Ghafarian Tamizi**, and Homayoun Najjaran. "Safety optimized reinforcement learning via multi-objective policy optimization." In 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 2873-2879. IEEE, 2024.

- Homayoun Honari, Amir M. Soufi Enayati, **Mehran Ghafarian Tamizi**, and Homayoun Najjaran. "Meta SAC-Lag: Towards Deployable Safe Reinforcement Learning via MetaGradient-based Hyperparameter Tuning." In 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 619-626. IEEE, 2024.

Chapter 1

Introduction: Motivation and Objectives

1.1 Importance of Motion Planning

Motion planning is a fundamental problem in robotics and artificial intelligence, concerned with generating collision-free and dynamically feasible trajectories for robotic systems subject to various task constraints. An ideal motion planner guarantees completeness (finding a solution whenever one exists) and optimality (finding the best possible path), while operating efficiently in real time and remaining robust against complex environments [4].

In industrial robotics these capabilities are crucial. Manipulator arms must safely and efficiently navigate cluttered workspaces under tight cycle time requirements and varying task layouts. As robots are increasingly deployed in flexible and complex production settings, the demand for planners that are both safe and adaptive continues to grow.

1.2 Challenges and Research Gaps

Despite extensive progress, existing motion planning approaches still face several fundamental challenges.

Computational Efficiency

Classical planners such as sampling based algorithms (for example RRT variants) can find feasible paths but are often slow and can get trapped in local minima when operating in high dimensional or cluttered environments. Optimization based methods can produce smooth trajectories but require long computation times or careful parameter tuning for each scenario.

Limited Generalization of Learning Based Methods

Recent learning based planners can generate motions faster than classical methods, yet they typically overfit to their training environments. A model trained for one workspace or robotic arm often fails when confronted with even moderate changes, such as new obstacle layouts or different kinematic structures. This necessitates costly retraining or fine-tuning for every new setup.

These limitations highlight an important gap: the absence of a motion planning framework that is simultaneously efficient, safe, and generalizable across diverse robots and workspaces without retraining.

1.3 Contributions

This dissertation introduces two complementary frameworks, **PPCNet** and **GADGET**, that together address these challenges by combining deep learning efficiency, generative modeling flexibility, and safety aware design.

Path Planning and Collision Checking Network (PPCNet)

PPCNet is an end-to-end deep learning framework designed for real time and safe motion planning in structured industrial environments.

- It introduces a dual network architecture that integrates a waypoint generator with a learned collision checker.
- The collision checker replaces traditional geometric collision detection with a fast data driven alternative, greatly accelerating planning while maintaining safety.
- A post processing refinement step further smooths and shortens planned paths, improving path quality and efficiency.

PPCNet addresses the computational inefficiency of classical planners by learning to predict both path feasibility and safety jointly, achieving millisecond-level planning with accuracy comparable to state of the art methods.

Generalizable and Adaptive Diffusion Guided Environment Aware Trajectory Generation (GADGET)

GADGET extends beyond efficiency to address the problems of generalization and safety in unseen environments.

- It employs a voxel based scene representation reconstructed from multi view depth sensing, enabling the planner to reason about workspace geometry.
- Through dual phase conditioning, GADGET integrates two complementary forms of guidance during trajectory generation. The first phase performs task and environment conditioning by embedding both the voxel encoded scene and

the start and goal joint configurations into a shared latent space that guides the diffusion process toward feasible goal-directed paths. The second phase applies a Control Barrier Function (CBF) inspired safety shaping term that softly biases the denoising dynamics away from unsafe configurations during inference.

- By combining these conditioning strategies within the denoising process, GADGET generates paths that are simultaneously geometry aware, goal consistent, and safety aware, achieving real time generalization across diverse environments and robotic arms without retraining.

GADGET overcomes the limited generalization and performance constraints of prior learning based planners by integrating geometry aware scene understanding and dual phase conditioning directly into the generative inference process. This formulation enables zero shot generalization to unseen environments and new robotic manipulators, while consistently achieving higher success rates, lower collision intensity, and comparable planning times compared with existing classical and learning based methods.

In summary, this thesis contributes to three major aspects:

1. **Efficiency:** PPCNet achieves fast end to end path planning with learned collision checking which is an ideal framework for repetitive industrial tasks like bin-picking.

2. **Generalization:** The dual phase conditioning in GADGET enables zero shot adaptation to unseen environments and different robotic manipulators.

3. **Safety:** GADGET integrates differentiable CBF guidance for safe trajectory generation.

Together, these innovations bridge the gap between efficiency, safety, and adaptability, establishing a foundation for generalizable and real time motion planning

applicable to industrial robotic systems.

1.4 Thesis Outline

Chapter 2: Background- This chapter reviews the foundational concepts and related work in robotic motion planning. It covers classical approaches (e.g., sampling-based, optimization-based, and heuristic methods) as well as recent advances in learning-based techniques, highlighting their limitations and motivating the need for more generalizable and efficient solutions.

Chapter 3: PPCNet – Learning-Based Motion Planning- This chapter introduces a deep learning-based framework, PPCNet, designed for efficient path planning in industrial environments. It outlines the model architecture, training strategy, and evaluation results, demonstrating improved planning speed and safety in both simulated and real-world scenarios.

Chapter 4: GADGET – Diffusion-Based Generalizable Planning- This chapter presents GADGET, a diffusion model-based motion planner aimed at generalization to new environments and robotic platforms. It describes the core components of the framework and evaluates its ability to produce safe, real-time trajectories in previously unseen settings without retraining.

Chapter 5: Concluding Remarks- The final chapter summarizes the main contributions, discusses the practical implications of the proposed frameworks, and outlines directions for future research, including enhanced scalability, constraint integration, and online adaptation.

Chapter 2

Background

Motion planning typically involves two stages: path planning and trajectory planning. Path planning focuses on generating a feasible path that satisfies geometric constraints, independent of timing. Trajectory planning, in contrast, applies a time law to the path to produce a time-parameterized trajectory. In practice, these two steps are often interdependent and executed jointly.

Motion planning approaches can be broadly categorized into classical and learning-based methods (see Figure 2.1). Classical techniques include combinatorial algorithms, artificial potential fields (APF), sampling-based planners, and bio-inspired heuristics. Learning-based methods leverage machine learning, with three major paradigms being deep learning-based planning, reinforcement learning (RL), and learning by demonstration (LbD).

Combinatorial approaches aim to determine a path by directly analyzing the continuous configuration space [5]. One such method is cell decomposition, which partitions the configuration space into smaller, manageable cells to facilitate path finding [6, 7]. However, these techniques often become impractical in complex or high-dimensional environments due to computational limitations. In contrast, artifi-

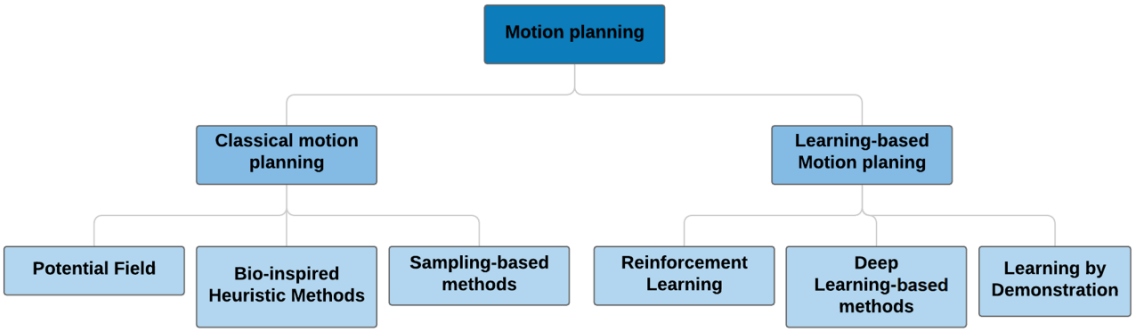

Figure 2.1: Different types of Motion planning methods.

cial potential field (APF) methods are more suitable for practical applications. APF constructs attractive and repulsive potential fields around target and obstacle configurations, respectively, to guide the robot toward the goal [8]. While this approach can handle dynamic environments with moving obstacles [9], it is prone to getting trapped in local minima, which can prevent the robot from reaching its destination.

Bio-inspired heuristic methods are reliable tools for finding optimal trajectories due to their effectiveness in solving optimization problems. However, they may struggle in complex environments and high-dimensional configuration spaces. Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) are two widely used bio-inspired optimization techniques that have demonstrated strong performance in trajectory planning tasks [10]. Despite their effectiveness, these methods are unsuitable for dynamic environments and typically require high computational time to generate near-optimal paths.

Sampling-based path planning methods are among the most widely used approaches in motion planning. These methods construct paths by randomly sampling the configuration space. While they are effective for complex and high-dimensional problems, they cannot guarantee a solution within a finite amount of time [11]. Probabilistic Roadmap Method (PRM) and Rapidly-exploring Random Trees (RRT) are the most common examples of sampling-based methods. However, they also tend

to suffer from long computation times. To address this, heuristic enhancements are often applied to improve their efficiency [12, 13].

In recent years, machine learning-based algorithms have gained significant attention for motion planning due to their generalization capability and effectiveness in handling complex problems [14, 15, 16, 17, 18]. These techniques can solve inverse kinematics problems more quickly and efficiently, which has a direct impact on the motion planning process [19, 20]. Additionally, learning-based methods can mitigate the high runtime issue of sampling-based approaches by providing biased sampling nodes, allowing these planners to operate more efficiently [21, 22, 23, 24].

The main focus of this chapter is on motion planners commonly used in robotic manipulators. The objective is to provide a comprehensive review of both classical and learning-based algorithms. This chapter highlights recent advancements in AI-based motion planners that serve as powerful alternatives to classical methods. Emphasis is placed on learning-based approaches, which have shown promising performance in addressing the limitations of traditional techniques. The chapter aims to guide the reader through the rapidly evolving literature in this field, which has gained significant momentum in recent years.

2.1 Fundamentals and Properties of Motion Planning

In this section, some general fundamentals and concepts of motion planning are discussed. One of the first and most important concepts in motion planning is the configuration space (C-space). Every position and orientation of a robot can be represented as a point in the C-space. For example, Eq. 3.1 shows a specific configuration of an n-joint arm manipulator:

$$q = \begin{bmatrix} \theta_1 & \dots & \theta_n \end{bmatrix} \tag{2.1}$$

where θ_i represents the position of the i-th joint of the arm. If this configuration lies in an obstacle-free region and satisfies all joint constraints, it is considered part of the **free configuration space** (C_{free}). On the other hand, if the configuration leads to a collision with an obstacle or violates joint constraints, it belongs to the **obstacle space** (C_{obs}). In motion planning problems, both the start configuration $q_{initial}$ and the target configuration q_{target} must lie within C_{free}. The goal of the planner is to find a continuous, collision-free path between these two points, entirely within the free configuration space.

2.1.1 Path Planning vs. Trajectory Planning

Although path planning and trajectory planning are often solved together as a single problem, it is important to understand the distinction between them. Path planning refers to the process of generating a collision-free path based solely on geometric considerations, without accounting for the robot's dynamics or mobility limitations. In other words, the path planner provides a sequence of configurations from $q_{initial}$ to q_{target} without considering time.

In contrast, trajectory planning involves the time parameterization of the path. This means that the trajectory specifies the robot's configuration at each point in time. Therefore, the output of the path planner must be time-scaled to produce a feasible trajectory (see Figure 2.2).

Path and trajectory planning are constrained by two main types of constraints: geometric and kinodynamic constraints. Geometric constraints—such as joint limits, obstacle avoidance, and self-collision avoidance—are typically handled during the path planning stage. On the other hand, kinodynamic constraints involve factors like

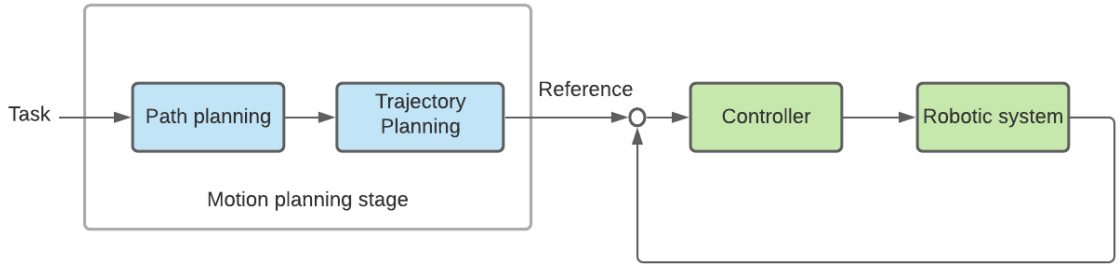

Figure 2.2: Block diagram of motion planning in a robotic system.

joint velocities, accelerations, torques, and higher-order derivatives of joint angles [25].

Trajectory planning for robotic arms and manipulators requires optimization under these constraints [26], typically aiming to satisfy three main objectives: minimum execution time [27, 28, 29], minimum energy consumption [30, 31], and minimum jerk [32, 33].

2.1.2 Motion Planning Properties

A motion planner is called **complete** when it always returns a solution if one exists. A complete planner is able to find a solution in finite time and also determine whether or not a solution exists. In this context, two related terms are also important: **resolution complete** and **probabilistically complete**. A planner is considered resolution complete if it searches over a discretized representation of the C-space and guarantees a solution when the resolution is sufficiently fine. On the other hand, a planner is **probabilistically complete** if the probability of finding a solution approaches one as the computation time approaches infinity [34].

Another important distinction in motion planning is between single-query and multiple-query planners. A multiple-query planner first attempts to construct a complete or partial representation of the C-space. As a result, it can generate paths quickly for different start and goal pairs. In contrast, in dynamic environments where

the C-space may change over time, single-query planners often perform better. This is because they solve each motion planning problem from scratch and are better suited for adapting to changes in the environment [34].

2.1.3 Collision detection

C-space consists of two subspaces: C_{free} and $C_{obstacles}$. Each configuration of the robot in C-space belongs to one of these two subspaces [35]. To detect collisions between the robot and obstacles, a distance measurement algorithm is typically used. This algorithm calculates the distance between the closest points on the robot and the obstacle [34].

The Gilbert-Johnson-Keerthi (GJK) algorithm [36] is a widely used and efficient method for computing the distance between two convex bodies. It is commonly employed in robotics by modeling both the robot and obstacles as triangular meshes. Another simpler method for collision detection is approximating objects using overlapping spheres [37]. In this approach, the number of spheres representing each object is a critical hyperparameter. Although less precise, this method is significantly faster than other analytical collision detection techniques.

In motion planning problems, regardless of the planning method, a large number of collision checks are required. This process is computationally expensive and can account for up to 90% of the total planning time [38]. In recent years, researchers have turned to machine learning techniques to approximate or bypass analytical collision checking.

Support Vector Machines (SVM) have been used to learn accurate collision boundaries in C-space [39]. Similarly, Gaussian Mixture Models (GMM) were employed in [40] to classify regions of C-space as collision-free or in-collision. Their results showed that the approach could generate paths up to five times faster than bi-

directional RRT. Another method, K-Nearest Neighbors (KNN), has also been applied to model C-space for collision detection [41]. However, KNN is generally used in static environments and faces challenges when adapting to dynamic settings.

Das et al. [42] proposed a machine learning model called Fastron to model C-space and serve as a proxy collision detector. In this method, the C-space is decomposed into subspaces, and a separate collision checker is trained for each region. Most prior works in the literature rely on such C-space decomposition techniques, which often require manual engineering and simplifications.

2.1.4 Nonlinear Optimization Solutions

In robotic systems, motion planning is commonly formulated as a constrained nonlinear optimization problem, in which a trajectory, defined in either joint space or task space, is optimized with respect to a cost function while satisfying a set of constraints. Typical objectives include generating smooth trajectories, maintaining safe distances from obstacles, and optimizing for efficiency. For instance, a smoothness term (e.g., the sum of squared velocity changes along the trajectory) may be incorporated to reduce mechanical stress on actuators. Simultaneously, an obstacle avoidance term penalizes configurations that approach or violate predefined safety margins near obstacles.

Beyond these, additional objectives can be integrated into the formulation, such as minimizing path length or travel time to shorten execution duration, or minimizing energy and torque to improve efficiency and reduce wear. Furthermore, task-specific constraints, such as maintaining a desired end-effector pose or orientation, can also be imposed. While collision avoidance has traditionally been the primary objective in motion planning, contemporary formulations often consider multiple competing criteria including smoothness, energy efficiency, and constraint satisfaction.

These objectives and constraints are typically combined into a nonlinear cost function. The resulting optimization problem is non-convex due to the inherent nonlinearities of robot kinematics, dynamics, and obstacle geometries. The planner's goal is to compute a trajectory that minimizes the total cost (usually expressed as a weighted sum of terms like trajectory smoothness, path length, and time) while satisfying kinematic and dynamic feasibility conditions, such as joint limits, collision-free constraints, and specified initial and target configurations. A general mathematical formulation of this problem is provided in Equation 2.2.

$$
\begin{aligned}
&\text{find} \quad q(t) \\
&\text{minimize} \quad J(x(t), u(t), d(t)) \\
&\text{subject to:} \\
&\quad \dot{x}(t) = f(x(t), u(t)), \\
&\quad q(t) \in C_{free}, \\
&\quad x(0) = q_{initial}, \\
&\quad x(T) = q_{target}
\end{aligned}
\tag{2.2}
$$

To address the nonlinear motion planning problem, researchers employ both gradient-based and derivative-free optimization techniques. Gradient-based methods utilize the gradients of the objective function to iteratively refine the trajectory. A widely used approach is Sequential Quadratic Programming (SQP) [43], which solves a sequence of quadratic approximations to converge toward the optimum of the original nonlinear problem. In the context of motion planning, SQP treats the entire trajectory as a decision variable and incrementally optimizes it while enforcing collision-avoidance constraints [44].

These methods—including classic gradient descent and Newton-based variants—are particularly effective when the gradients of the cost and constraint functions are avail-

able. In practical implementations, trajectory optimizers such as CHOMP (Covariant Hamiltonian Optimization for Motion Planning) [45] exploit functional gradients, which can be computed analytically and efficiently from robot kinematics. These techniques are well-suited for rapidly descending to a local minimum; however, they depend heavily on the quality of the initial trajectory and may struggle in highly non-convex optimization landscapes. Schulman et al. [46] introduced a trajectory optimization framework based on sequential convex optimization that incorporates continuous-time collision checking via convex-convex distance formulations. Their method efficiently solves high-DOF motion planning problems by penalizing collisions with a hinge loss and incrementally tightening constraints, achieving higher success rates and better path quality compared to sampling-based and CHOMP methods.

In contrast to gradient-based methods, derivative-free optimization techniques do not rely on gradient information and are particularly advantageous when the cost function is discontinuous, non-differentiable, or noisy. One prominent example is Simulated Annealing (SA) [47], a stochastic optimization algorithm inspired by the annealing process in metallurgy. SA is well-suited for highly nonlinear or discontinuous cost landscapes [48]. In motion planning, SA operates by introducing random perturbations to the trajectory and occasionally accepting worse solutions in early iterations. This acceptance criterion promotes exploration of the solution space and helps escape local minima. Over time, the algorithm "cools," progressively reducing the acceptance of suboptimal solutions and guiding the trajectory toward improvement [49].

Another widely used derivative-free method is the Nelder–Mead simplex algorithm [50], which optimizes the trajectory by iteratively modifying a simplex—a geometric structure composed of multiple candidate solutions—through reflection, expansion, and contraction steps. Nelder–Mead has been applied to trajectory smoothing

and time-parameterization, such as refining trajectories generated by sampling-based methods like RRT. For example, it has been used to optimize timing profiles to reduce jerk and travel duration in manipulator motion[51].

While casting motion planning as an optimization problem offers a structured approach, it also presents several challenges. The cost landscape is typically non-convex due to obstacle avoidance and complex kinematics, which causes gradient-based solvers to get stuck in local minima [45]. As a result, these methods are often locally optimal and incomplete and they may fail to find a feasible path even when one exists [44]. A good initial trajectory is essential: poor initialization can lead to failure, while a well-chosen seed (e.g., straight-line interpolation or sampling-based planner output) increases success rates [45]. Moreover, convergence speed and quality depend on careful tuning of hyperparameters like step sizes and constraint penalties. There's also a tradeoff between hard constraints (e.g., strict collision avoidance) and soft penalties, where the latter simplifies optimization but may temporarily violate feasibility.

Finally, computational cost is a limiting factor, especially in high-dimensional spaces. Evaluating gradients for obstacle avoidance or smoothness terms can be expensive, although recent work has introduced efficient approximations and trajectory sparsification to reduce complexity.

2.2 Classical Motion Planning methods

This section will discuss classical methods in three main categories: artificial potential fields, bio-inspired heuristic methods, and sampling-based methods. These three approaches were selected due to their strong performance in practical applications. Following subsections will elaborate on the key characteristics of each technique.

2.2.1 Artificial Potential Field

The core idea behind the Artificial Potential Field (APF) method is to guide the robot around obstacles and toward the goal configuration using an artificial potential energy field. This method is particularly effective in non-static environments with dynamic obstacles. APF generates a collision-free path by assigning high potential values around obstacles and low potential near the goal. As a result, the robot is attracted to the goal while being repelled from obstacles in the configuration space, according to the structure of the potential field.

The APF method was first introduced in [8] and later extended in [52]. Due to its simplicity and strong practical performance, the method has been widely studied in the literature. The potential field formulation is as follows: let $\mathtt{x}_d$ denote the desired target position. The manipulator's end-effector can be controlled with respect to obstacle O by subjecting it to an artificial potential field.

$$U_{\text{art}}(\mathtt{x}) = U_{\mathtt{x}d}(\mathtt{x}) + U_O(\mathtt{x}), \tag{2.3}$$

which leads to:

$$U(\mathtt{x}) = U_{\text{art}}(\mathtt{x}) + U_g(\mathtt{x}), \tag{2.4}$$

where $U_g(\mathtt{x})$ represents the gravitational potential energy. Using the Lagrangian formulation and dynamic decoupling of the end-effector, the control command vector F^*_{art} for the decoupled end-effector is given by:

$$F_{\text{art}} = F_{\mathtt{x}_d} + F^*_O, \tag{2.5}$$

where

$$F_{\mathtt{x}d} = -\nabla U_{\mathtt{x}_d}(\mathtt{x}),\ F_O\ = -\nabla U_O(\mathtt{x}), \tag{2.6}$$

and

$$U_{\mathtt{x}_d}(\mathtt{x}) = \frac{1}{2} k_p (\mathtt{x} - \mathtt{x}_d)^2. \tag{2.7}$$

Here, F_O is a repulsive force induced by the artificial potential field $U_O(\mathtt{x})$ near the surface of obstacles, while $F_{\mathtt{x}_d}$ is an attractive force that drives the end-effector position $\mathtt{x}$ toward the desired target $\mathtt{x}d$. The function $U_O(\mathtt{x})$ is defined such that the total artificial potential $U_{\text{art}}(\mathtt{x})$ remains a positive, continuous, and differentiable function with a global minimum at $\mathtt{x} = \mathtt{x}_d$.

In [53], a basic APF approach is employed using the distances between the end-effector, static obstacles, and the goal configuration. An optimal trajectory is generated by applying a pattern search algorithm, taking advantage of APF's effectiveness in minimizing a predefined cost function.

To address dynamic obstacles, it is necessary to modify both the repulsive (around obstacles) and attractive (around the goal) potential functions by incorporating the relative positions and velocities of the robot and the obstacles [54]. Although APF can handle path planning in dynamic environments, it suffers from a major drawback: the presence of local minima, where the robot may become trapped [7]. To mitigate this, an "extreme point jump-out" function is introduced in [55]. When the manipulator is stuck at a local extremum, this function initiates a small movement at the end-effector to adjust its direction and escape the local minimum. The approach combines two components: the Bloch Quantum Genetic Algorithm for generating small displacements, and the RRT method for adjusting direction. A further improvement to APF is proposed in [56], where the manipulator escapes local minima by moving

Table 2.1: Potential field methods proposed in the literature.

Study	Description	Advantage	Drawback	DOF
Liu et al. (2014) [57]	Combining configuration-oriented APF with inverse kinematics	Using damped least square method to avoid singularity	High computation time and low adaptability	6
Badawy (2016) [58]	Creating dual attraction between links by constructing a potential field with two minima	Avoiding local minimum	Not capable for high DoF manipulators	2
Long (2020) [59]	Introducing obstacle avoidance method based on APF for a manipulator in cluttered environment	Avoiding local minimum by considering virtual target point based on RRT method	Choosing virtual target point randomly; thus, deviating from the goal and collision with obstacles are probable	3
Gai et al. (2019) [60]	Introducing path planning algorithm based on potential field	Implementing practically on MA1440 industrial robot	Static environment	6
Zhang et al. (2017) [61]	Proposing a new potential function for path planning of 6 DoF RBT-6T/S03S manipulator	Using a virtual obstacle to avoid local minimum issue	Static environment and high computation time	6
Li et al. (2019) [62]	Setting the attractive and repulsive APF function for goal and obstacles in task space instead of C-space	Fast and efficient approach for deterministic environment with static obstacles	Not optimal and need high computation in uncertain environment	3
Lin, Hsieh (2018) [63]	Proposing a repulsive force field in 3-D space which makes the manipulator justify its trajectory to avoid the obstacles	Using in unknown environment with dynamic obstacles	Finding non-optimal path and trapping in local minimum is probable	-

along the direction of the repulsive potential field. This strategy enables the robot to avoid obstacles while still following a near-optimal path.

Table 2.1 summarizes additional studies in the context of APF. As mentioned earlier, APF is applicable to a wide range of path planning problems under various conditions. However, the local minimum issue remains a significant limitation. Although improved APF methods aim to resolve this problem, they often increase algorithmic complexity, which can hinder the practical implementation of APF in real-world robotic systems.

2.2.2 Bio-inspired Heuristic Methods

This section discusses the advantages and limitations of two major heuristic methods used in path and trajectory planning: Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).

Genetic Algorithm (GA) — GA is a powerful and widely used tool for optimization in robotic path planning research [64, 65, 66, 67, 68, 69]. Due to its ability to handle discrete objective functions, GA can be effectively combined with singularity avoidance techniques to find time-optimal paths [70].

Numerous studies have applied GA to motion planning for robotic manipulators. In [71], a GA-based approach is presented using bi-level optimization to generate

an optimal trajectory. Similarly, in [72], the cost function is defined based on time, energy consumption, and joint rotation, and GA is employed to obtain the optimal solution. To address the local minimum issue, an improved GA method is introduced in [73], focusing on optimal trajectory generation for a 6-DOF manipulator.

Particle Swarm Optimization (PSO) — PSO is inspired by the synchronized movement and social behavior of birds and fish [74]. It is simple to implement and highly effective for optimizing a variety of objective functions. As a result, PSO has been widely used in robotic arm motion planning [75, 76, 77]. For instance, Kucuk [78] proposed a PSO-based trajectory planner using cubic spline interpolation, which was implemented on a PUMA robot. Additionally, Cao et al. [79] presented a hybrid optimization approach that combines PSO with geometric techniques to minimize motion time for an RRR redundant robotic arm.

Other Heuristic Methods — Ant Colony Optimization (ACO)[80, 81, 82] is another bio-inspired method used in motion planning, known for its reliability in solving optimization problems. In addition, classical graph search algorithms such as Dijkstra's algorithm and the A* algorithm are commonly used in robotic operating systems (ROS) to compute shortest paths. Although these methods use heuristic estimates to reduce the number of explored nodes, they tend to have lower planning efficiency in complex environments[83].

The core idea behind Dijkstra's algorithm is to expand nodes in increasing order of their distances from the source, progressively constructing a shortest-path tree by adding one edge at each step [84]. The A* algorithm enhances this by using a heuristic function to evaluate nodes, inspecting many nearby nodes to guarantee an optimal, collision-free solution—albeit with high computational cost [85]. Theta* is a variation that relaxes the constraint of staying on grid edges, allowing more direct paths, but it does not always guarantee the true shortest path [86]. The D*

algorithm, often referred to as dynamic A*, is suitable for environments that change over time. It replans paths when new obstacles are detected, using cost-based updates to incrementally adjust the solution in real-time [87, 88].

Although heuristic methods are powerful optimization tools and perform well in many scenarios, they cannot guarantee complete solutions for trajectory planning. These approaches often require significant computation time to identify optimal paths in complex, high-dimensional environments.

2.2.3 Sampling-based Methods

Sampling-based motion planners are among the most effective algorithms in robot motion planning due to their ability to solve path planning problems in complex and unknown environments. In this context, the Probabilistic Roadmap Method (PRM) and Rapidly-exploring Random Trees (RRT) are two of the most well-known techniques.

PRM, introduced in [89], is a graph-based search algorithm. It discretizes the configuration space (C-space) and builds a roadmap composed of collision-free paths to find asymptotically optimal solutions. PRM and its variants are categorized as multiple-query algorithms. Additionally, PRM is probabilistically complete, meaning that as the number of samples increases, the probability of failing to find a solution approaches zero [90].

Algorithm 1 presents the structure of PRM [91]. At each iteration, a point $x_{\text{rand}} \in X_{\text{free}}$ is sampled and added to the vertex set V. Then, connections are attempted between x_{rand} and nearby vertices within a ball of radius r using a basic local planner. If the connection is collision-free, an edge is added to the edge set E. To avoid unnecessary computation, connections between nodes already in the same component are skipped.

Algorithm 1 PRM

Require: $V \leftarrow \emptyset$, $E \leftarrow \emptyset$
Ensure: $G = (V, E)$
  for $i = 0, \cdots, n$ **do**
    $x_{rand} \leftarrow SampleFree_i$
    $U \leftarrow Near(G = (V, E), x_{rand}, r)$
    $V \leftarrow V \cup \{x_{rand}\}$
    for each $u \in U$ **do**
      if x_{rand} and u are not in the same connected component of $G = (V, E)$ **then**
        if CollisionFree (x_{rand}, u) **then**
          $E \leftarrow E \cup \{(x_{rand}, u), (u, x_{rand})\}$
        end if
      end if
    end for
  end for

Due to these advantages, PRM has been applied to various manipulator motion planning tasks. However, the original PRM suffers from drawbacks such as high execution time. To mitigate this, Lazy PRM was proposed in [92], which reduces the number of collision checks by assuming all nodes lie in C_{free} initially, and later removing those that collide. In [93], an improved PRM was introduced for dual-arm motion planning by classifying sampled nodes based on the obstacles they collide with.

Although PRM can perform well with a small number of random nodes in simple environments, its efficiency drops in complex settings. The distribution of random samples also significantly affects the quality of the generated path.

RRT is one of the most popular alternatives to PRM. Algorithm 2 outlines its structure [94]. This method incrementally builds a random tree in C-space starting from an initial point x_{init}, extending it toward randomly sampled nodes to explore the configuration space efficiently.

RRT is simple to implement and performs well in non-convex, high-dimensional search spaces. It has been widely used for motion planning in robotic arms [95, 96,

Algorithm 2 RRT

Require: x_{init}, K
  τ.init(x_{init})
  for $k = 1$ to K **do**
    $x_{rand} \leftarrow RandomState()$
    $x_{near} \leftarrow NearestNeighbor(x_{rand}, \tau)$
    $u \leftarrow SelectInput(x_{rand}, x_{near})$
    $x_{new} \leftarrow NewState(x_{near}, u, \Delta t)$
    τ.AddVertex(x_{new})
    τ.Addedge(x_{near},x_{new},u)
  end for
  Return τ

97, 98, 99, 100, 101].

To enhance RRT's performance, several variants have been proposed. In [102], the bi-directional RRT was introduced for 7-DOF manipulators. This version grows two trees simultaneously (one from the start and one from the goal) and attempts to connect them at each step.

RRT* is an asymptotically optimal version of RRT [91]. It is probabilistically complete and computationally efficient. However, it suffers from slow convergence and becomes less efficient in high-dimensional spaces compared to grid-based methods [103]. Algorithm 3 shows the structure of RRT*.

Biased sampling is a promising strategy to improve sampling efficiency in such methods [104, 105]. Rather than uniformly sampling, biased sampling directs exploration toward more relevant regions. In [104], a combination of potential fields and RRT* is proposed to accelerate convergence.

Gammell et al.[106] introduced Informed-RRT*, which improves convergence and path quality, though it requires an initial feasible path. Later, they proposed BIT*[107], a batch-processing algorithm based on random geometric graphs (RGGs), which reuses previous information to accelerate planning. BIT* was tested on a 14-DOF dual-arm robot and showed superior convergence compared to RRT, RRT*, and

Algorithm 3 RRT*

Require: $V \leftarrow \{x_{init}\}$, $E \leftarrow \emptyset$
Ensure: $G = (V, E)$
  for $i = 1, \cdots, n$ **do**
    $x_{rand} \leftarrow SampleFree_i$
    $x_{nearest} \leftarrow Nearest(G = (V, E), x_{rand})$
    $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$
    if ObtacleFree $(x_{nearest}, x_{new})$ **then**
      $X_{near} \leftarrow Near(G = (V, E), x_{new}, r(card(V)))$;
      $V \leftarrow V \cup \{x_{new}\}$;
      $x_{min} \leftarrow x_{nearest}$;
      $c_{min} \leftarrow Cost(x_{nearest}) + c(Line(x_{nearest}, x_{new}))$
      for each $x_{near} \in X_{near}$ **do**
        if CollisionFree (x_{near}, x_{new}) *or* $Cost(x_{near}) + c(Line(x_{near}, x_{new})) <$ c_{min} **then**
          $x_{min} \leftarrow x_{near}; c_{min} \leftarrow Cost(x_{near}) + c(Line(x_{near}, x_{new}))$
        end if
        $E \leftarrow E \cup \{(x_{min}, x_{new})\}$
      end for
      for each $x_{near} \in X_{near}$ **do**
        if CollisionFree(x_{new}, x_{near}) *or* $Cost(x_{new}) + c(Line(x_{new}, x_{near})) <$ $Cost(x_{near})$ **then**
          $x_{parent} \leftarrow Parent(x_{near}); E \leftarrow (E \backslash \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$
        end if
      end for
    end if
  end for

Informed-RRT*, although it still suffers from the curse of dimensionality.

In summary, while classical sampling-based methods are widely used, they often suffer from high computation time and low convergence rates in complex or high-dimensional C-spaces. Nonetheless, their continued development remains essential due to the demand for real-time motion planning in industrial applications.

2.3 Learning-based Motion Planning Methods

In recent years, learning-based methods have gained significant traction due to advances in artificial intelligence. These methods have been applied to complex path and trajectory planning problems, driven by the increasing importance of motion planning in industrial applications [108, 17]. However, compared to classical approaches, these techniques are relatively new. This section discusses three main categories of learning-based motion planning methods.

2.3.1 Deep Learning-based Methods

Deep learning is often combined with sampling-based methods to improve their performance in motion planning. As discussed in Section 2.2.3, one of the main drawbacks of sampling-based methods is their low convergence rate in high-dimensional spaces due to random sampling. Prior experience can significantly improve the efficiency of sample node generation [109, 110, 111, 112]. Deep learning addresses this issue by leveraging prior data to generate more efficient sample nodes, ultimately reducing planning steps and improving performance (see Figure 2.3).

Qureshi et al.[2] proposed a deep sampling-based motion planner (DeepSMP), which integrates deep neural networks with sampling-based methods. This framework uses a Contractive Autoencoder (CAE) to encode the environment from raw point clouds. The output, along with initial and goal configurations, is passed to a Dropout-based stochastic deep feedforward network that generates sample nodes for planners such as RRT. This significantly improves the convergence rate and ensures the generation of collision-free paths (see Figure2.4).

Ying et al. [1] proposed a deep learning-based planner that combines a long short-term memory (LSTM) network with Bi-directional RRT (LSTM-BiRRT) for dual-

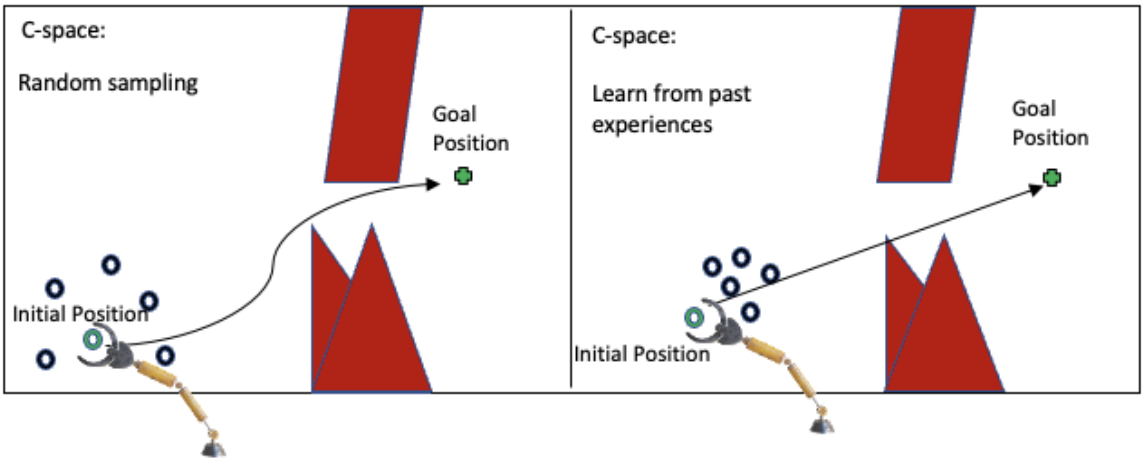

Figure 2.3: The performance of deep learning methods in generating sample nodes compared to random sampling [1].

arm motion planning. This approach differs in its sample generation process, using an LSTM Sampler trained via RRT* to produce samples. The environment, including obstacles, is encoded with an Autoencoder.

Motion Planning Network (MPNet), introduced in [113], has also shown promising results in learning feasible paths directly in unseen environments. MPNet consists of two networks: an encoder (Enet) that captures the environment and a planning network (Pnet) that generates the path. In [114], three training strategies are discussed:

- Offline batch learning: requires the full dataset in advance,
- Continual learning: enables learning from streaming data during execution,
- Active continual learning: similar to continual learning but incorporates expert demonstrations.

Constrained motion planning is another area where deep learning improves the performance of sampling-based methods. In this setting, the planner must find a collision-free path that lies on a constraint manifold—an inherently difficult problem

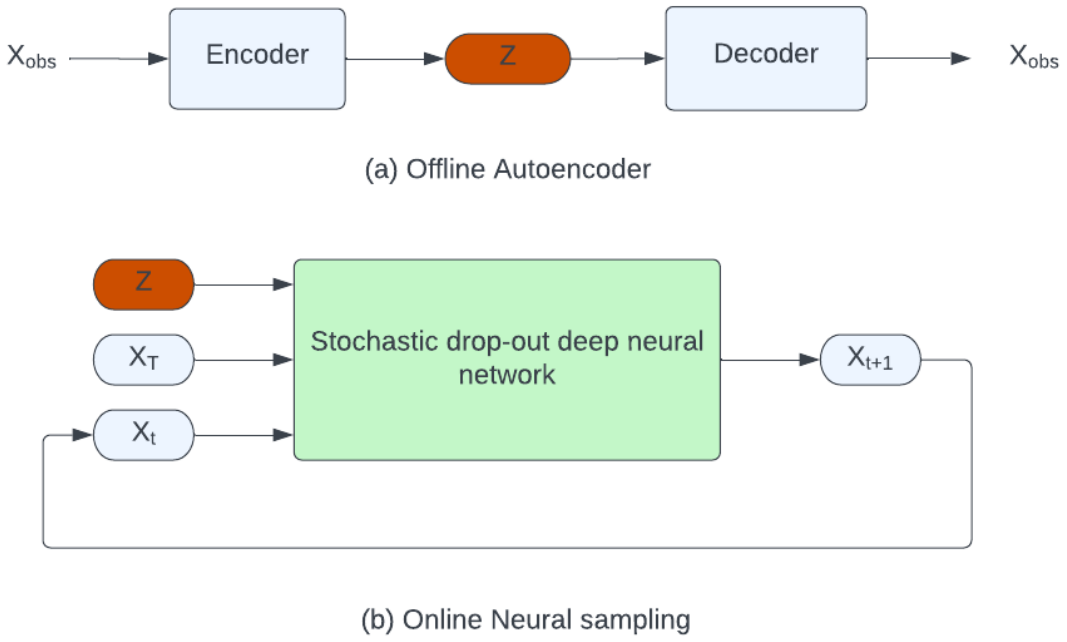

Figure 2.4: The structure of DeepSMP: CAE encodes the environment, while a deep neural network generates samples during online planning [2].

requiring efficient computation [115]. Classical sampling-based methods are typically inefficient and slow for such tasks.

To address this, Qureshi et al .[116, 117] introduced the Constrained Motion Planning Network (CoMPNet), a deep learning-based planner that takes environment data, constraint definitions, and start/goal configurations as inputs and outputs a valid path on the constraint manifold. CoMPNet was extended to CoMPNetX in [117], which introduces a task encoder, environment encoder, conditional neural generator/discriminator, and a neural sampler to enhance sample efficiency. This approach was evaluated on a robotic arm in various cluttered environments with hard kinematic constraints and demonstrated significantly improved performance over classical methods in terms of both time and path quality. However, CoMPNetX focuses solely on kinematic constraints derived from configuration space.

Generative models offer a powerful alternative for motion planning by learning

a distribution over feasible trajectories from data, rather than explicitly computing paths through optimization or sampling. This enables robots to generate diverse, task-relevant motions, even in high-dimensional and dynamic environments. Key generative approaches include Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and the more recent diffusion models, each with unique strengths for trajectory generation.

Generative Adversarial Networks (GANs) [118] leverage a generator-discriminator architecture trained in a minimax game to learn complex data distributions, enabling the generation of realistic robot trajectories from random noise. In motion planning for robotic manipulators, GANs learn from successful demonstrations to synthesize novel, physically plausible action sequences for tasks like grasping and collision-free movement. The discriminator effectively acts as a constraint checker, ensuring trajectory validity. While training can be unstable due to issues like mode collapse, GAN-based planners have shown promise in generating diverse and adaptive motions, enhancing generalization to new tasks and environments [119].

Variational Autoencoders (VAEs) [120] have been applied to robotic manipulator motion planning by learning low-dimensional latent spaces that capture the distribution of feasible motions. Such latent representations enable efficient generation of new motion plans and even smooth interpolation between known trajectories. For example, Hung et al. [121] train a VAE on random collision-free joint configurations and perform gradient-based optimization directly in its latent space to synthesize joint sequences that reach a goal while avoiding obstacles. Similarly, Ichter and Pavone [24] construct a plannable latent embedding via an autoencoder and train auxiliary networks (for local dynamics and collision checking) so that a sampling-based planner (RRT) can operate in this latent space to produce high-dimensional plans. VAEs also enable continuous variability in motions, Osa [122] shows that a VAE-based genera-

tive model can represent an effectively infinite continuum of homotopically distinct collision-free trajectories between a start and goal, allowing one to smoothly interpolate between different paths.

Recent works have employed denoising diffusion probabilistic models (DDPMs) [123] to generate motion plans for robotic arms by treating entire trajectories as data samples refined through iterative denoising. These models are trained on datasets of collision-free joint-space trajectories and conditioned on task-relevant inputs (such as start and goal configurations, and sometimes scene encodings) to produce smooth, obstacle-avoiding paths [124]. Janner et al. [125] introduced Diffuser, where planning is achieved by sampling full trajectories from a learned diffusion model, replacing traditional optimization or sampling-based methods. Chi et al. [126] condition the diffusion process on visual observations, enabling policy learning through action-by-action generation. Ze et al. [127] propose DP3, combining compact 3D point cloud inputs with diffusion models to produce dexterous manipulation actions. Huang et al. [128] present SceneDiffuser, which unifies generation, optimization, and planning in 3D scenes by incorporating physical constraints directly into the diffusion sampling. In all these approaches, diffusion models generate statistically consistent yet adaptable trajectories that generalize to new goals and environments.

Dataset Generation and Optimization Challenges

Generating an efficient and optimal dataset is a major challenge for learning-based motion planners. Since the primary goal of a learning-based planner is to replicate the behavior of an expert (oracle) planner, the quality of the dataset has a direct impact on its performance. An optimal dataset, especially one minimizing path length is able to enhance the planner performance. Most studies generate datasets using sampling-based planners, with RRT-based methods being the most common. However, as noted in Section 2.2.3, these methods are often inefficient in cluttered or complex

environments.

Post-processing can improve dataset quality. One common method is Lazy State Contraction (LSC) [114], which removes redundant waypoints to shorten paths. Sparse waypoints reduce computational overhead (e.g., in collision checking or communication with controllers), but small steps between waypoints also improve the likelihood of being collision-free and offer better learning targets for neural networks. A balanced trade-off is necessary. Resampling, as discussed in [129], helps by inserting collision-free waypoints without increasing the path length, reducing the variance in target step sizes. Additionally, the Bug2 algorithm [130] provides another efficient method for generating training datasets. In [131], a global planner based on Bug2 was proposed. This method is well-suited for dynamic environments and outperforms conventional motion planning algorithms in speed.

2.3.2 Reinforcement Learning Methods

Reinforcement learning (RL) is a powerful technique for motion planning of manipulators operating in unknown environments. In RL, an agent interacts with the environment, encounters various situations, and receives rewards based on its actions. By leveraging these experiences, the agent learns to choose optimal actions in each state [132] (see Figure 2.5). With advancements in deep learning, RL techniques can now be applied to high-dimensional continuous state spaces, enabling agents to extract relevant features and learn complex tasks. As a result, deep reinforcement learning (DRL) is well-suited for scenarios involving continuous action and state spaces.

Owing to these capabilities, DRL has been widely applied in robotic motion planning in recent years [133, 134, 135, 136, 137]. For example, Duguleana et al. [138] proposed a neural-network-based RL path planning approach for obstacle avoidance. In [139], a Soft Actor-Critic (SAC) method is used for path planning in a multi-arm

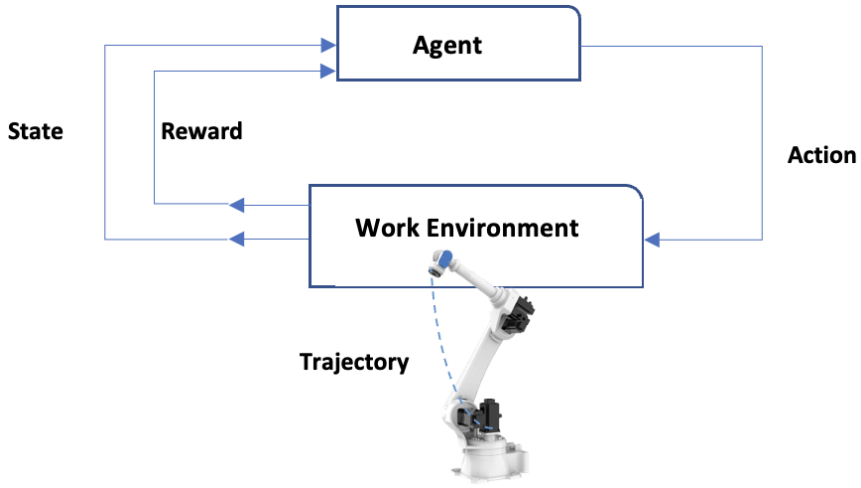

Figure 2.5: General reinforcement learning structure.

manipulator. Given the high-dimensional space and exploration requirements, traditional DRL planners were insufficient, leading to the adoption of SAC. To further enhance sample efficiency, Hindsight Experience Replay (HER) was also employed.

Katyal et al .[140] proposed a DRL-based collision-free motion planner that uses raw images of the workspace as state inputs. While this approach increases perceptual richness, it also introduces challenges due to the high-dimensional state space. Similarly, in [141], an improved Deep Deterministic Policy Gradient (DDPG) algorithm is introduced for path planning of a 6-DOF manipulator. Kamali et al. [142] developed a dynamic goal DRL planner that maps human hand movements to a robot arm, tested on the ABB IRB 120 platform.

Many of the above studies utilize sparse reward functions, which assign zero rewards to most states and only provide feedback in a few situations. The design of the reward function plays a crucial role in the performance of DRL agents. Sparse rewards often lead to inefficient exploration and slow learning [143].

To overcome this issue, Xie et al. [144] proposed two dense reward functions. The first is the Azimuth reward, which includes both position and orientation components.

It encourages maintaining a safe distance from obstacles while minimizing the gap to the goal. The second is a subtask-level reward, which decomposes the motion into simpler intermediate objectives. These reward functions were implemented on a UR3 manipulator and showed significantly better performance compared to sparse rewards.

Similarly, Peng et al. [145] introduced three new reward functions to improve DRL performance in motion planning:

- Posture reward function: models distance and directional constraints, accelerating learning.
- Stride reward function: enhances learning stability by considering step efficiency.
- Stage incentive reward function: provides structured guidance across task stages.

The results demonstrated that DRL agents trained with these dense reward functions outperformed those using sparse rewards in terms of both convergence and path quality.

To address constrained motion planning, Li et al. [146] proposed a DDPG-based planner for a free-floating dual-arm space manipulator. This approach incorporates the Generalized Jacobian Matrix (GJM) to model kinematics and enforces kinematic constraints to prevent self-collisions. A continuous reward function is defined based on the end-effector's velocity, distance to the target, and self-collision avoidance. The proposed method showed promising results in handling complex, constraint-rich scenarios.

2.3.3 Learning by Demonstration

RL requires extensive interaction with the environment to collect experience and learn tasks effectively. However, in many real-world applications, such interaction is impractical, and trial-and-error learning may be infeasible or unsafe. Learning by Demonstration (LbD) offers a powerful alternative to overcome such limitations. LbD is particularly useful for motion planning in high-dimensional spaces. In this approach, a training dataset is collected from an expert (i.e., a teacher) during task execution [147]. For robotic manipulators, human hand motions during task execution can serve as effective demonstrations [148]. In [149], various human hand movements were encoded using a set of nonlinear differential equations to capture different motion types.

Gaussian Mixture Models (GMM) [150] and Task-Parametrized Gaussian Mixture Models (TP-GMM) [151, 152] are widely used tools to encode human-demonstrated trajectories. These models leverage probability theory to manage uncertainty at all levels of motion modeling [153]. For instance, Duque et al. [3] proposed an LbD-based motion planner using TP-GMM for robotic assembly operations. Their trajectory generation framework includes three stages (see Figure2.6): demonstration acquisition, TP-GMM training for each subtask, and generation of new trajectories based on task-specific object configurations.

In addition, Kim et al.[154] introduced an LbD-based framework for constrained motion planning. This work modifies a demonstrated trajectory using Sequential Quadratic Programming (SQP) to satisfy constraints and successfully implements the method on a 3-DOF robotic arm. However, the method's scalability to high-dimensional configuration spaces remains uncertain. Similarly, in[155], imitation learning is applied for obstacle-avoidance trajectory generation for a 7-DOF space manipulator. The method prioritizes energy efficiency, which is especially critical in

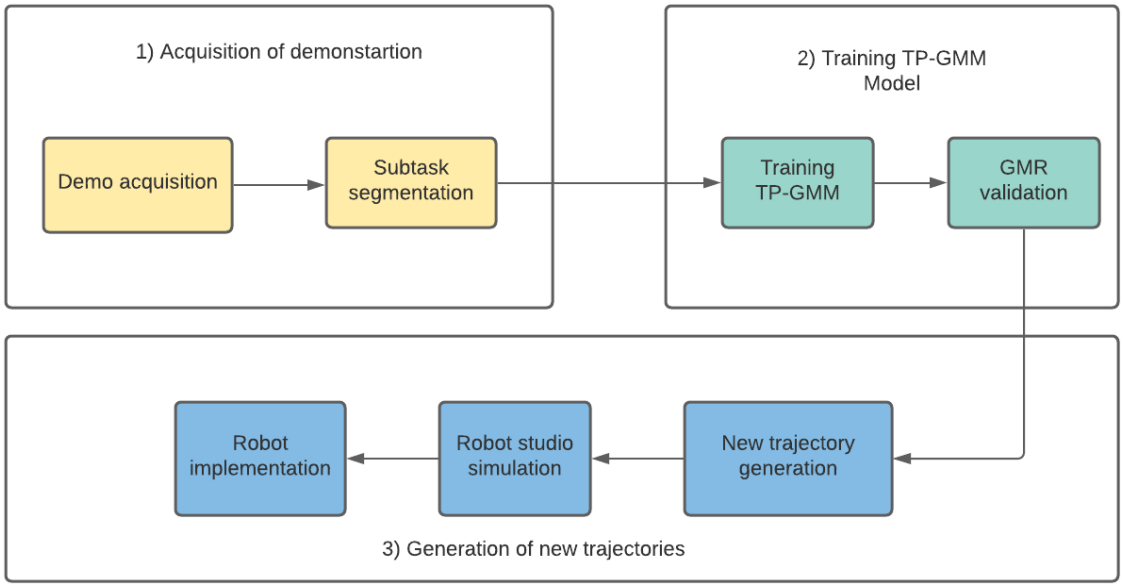

Figure 2.6: Block diagram of trajectory generation from demonstration using TP-GMM [3].

space robotics. Demonstration-based trajectories are used to learn a distribution over feasible paths.

Table 2.2 provides a comprehensive summary of recent learning-based motion planning techniques and highlighting their structures, key contributions, and implementation platforms. These works collectively demonstrate the effectiveness of learning-based approaches for solving constrained and unconstrained motion planning problems in complex environments.

Table 2.2: Summary of Learning-based motion planning methods

Reference	Learning-based method	Proposed Structure	Description	Implementation (DOF)
Bency et al. (2019) [17]	Deep learning	OracleNet: Recurrent Neural Networks to determine end-to-end trajectory	Generates High-speed and Near optimal path	7-DOF Dual-Arm Baxter Robot
Qureshi, Yip (2018) [2]	Deep learning	Deep sampling-based motion planner (DeepSMP)	Generates sample nodes for sampling-based motion planners to reduce computation time	6-DOF Universal robot (UR6)
Qureshi et al. (2020) [114]	Deep learning	Motion planning Networks (MPNet)	MPNet is able to calculate collision-free path directly or generate sample nodes for sampling-based methods	7-DOF Dual-Arm Baxter Robot
Lembono et al. (2021) [156]	Deep learning	Constrained Motion planning by using Generative Adversarial Network (GAN)	Reducing projection step and computation time by learning the distribution of robot configurations	7-DOF Panda manipulator and 28-DOF humanoid robot
Qureshi (2020) [117]	Deep learning	Constrained Motion planning Networks x (CoMPNetX)	Solves constrained motion planning problem with kinematics constrained	7-DOF Dual-Arm Baxter Robot
Guo et al. (2022) [157]	Reinforcement learning	Twin delayed deep deterministic policy gradient (TD3)	Proposed modified reward function based on the end-effector orientation and the distance between target and robot	6-DOF Arm (simulation)
Gupta and Najjaran (2022) [158]	Reinforcement learning	Exploitation of Abstract Symmetry of Environments (EASE)	Improved the sample efficiency of traditional reinforcement algorithms	5-DOF Arm (simulation)
Incremona (2021) [159]	Reinforcement learning	Hybrid control Scheme based on Deep RL	Presented a switch function that changed the motion planner with respect to the distance between the arm and obstacles	EPSON VT6
Wang et al. (2021) [160]	Reinforcement learning	Soft Actor Critic (SAC)	Proposed deep RL based optimization with maximum entropy	5-DOF Universal robot (UR5)
Yang, Peng (2021) [161]	Reinforcement learning	DDPG with Meta-Learning-Based Experience Replay Separation	Divided the original replay buffer into failure replay buffer and successful replay buffer	Simulation in V-rep
Jurgenson, Tamar (2019) [162]	Reinforcement learning	Deep deterministic policy gradient for motion planning (DDPG-MP)	Proposed a RL technique for training a neural motion planner	4-DOF WidowX robotic arm
Chen et.al. [163]	Reinforcement learning	SAC based algorithm for obstacle avoidance in dynamic environment	Employed prioritized experience replay (PER) to improve the sampling efficiency	7-DOF manipulator
Perez, Shah (2017) [164]	Learning by demonstration	Constraints learning from demonstration (C-Learn)	Learning geometric that can be used as a set of constraints in optimized-based motion planning problem	16-DOF Dual-arm Optimus robot (VI-A) 28-DOF humanoid robot
Chumkamon et al. (2021) [165]	Learning by demonstration	Learning from human hand movements	Using single shot detection to detect hand position for grasping application	7-DOF robotic arm
Li et al. (2021) [166]	Learning by demonstration	Model predictive motion planning network	Solves constrained motion planning problem with kinodynamic constrained	Acrobat, Cartpole, 12-DOF Quadrotor (simulation)
Jin et al. (2022) [167]	Learning by demonstration	Continuous Pontryagin differentiable programming (Continuous PDP)	Used sparse demonstrations such that robot follows desired trajectories	two-link robotic arm 6-DoF quadrotor

2.4 Conclusions

The field of motion planning for robotic systems is both vast and essential. Among these systems, industrial robotic manipulators play a crucial role in automating complex tasks. This chapter focused on various motion planning strategies applicable to such manipulators, reviewing both classical and learning-based approaches. For each category, the strengths and limitations are analyzed, tracking how these methods have evolved over time and identifying conditions under which each is most suitable. This chapter also provided a concise summary of the advantages and disadvantages of state-of-the-art motion planning techniques.

Table 2.3 presents a comprehensive summary of the motion planning methods discussed. Classical methods such as artificial potential fields (APF), bio-inspired heuristics, and sampling-based algorithms, such as RRT, PRM, are capable of generating feasible paths. However, they often struggle in high-dimensional and dynamically changing environments. For example, bio-inspired methods require high computation time to find optimal paths, and small changes in the environment (e.g., initial or goal configuration) can significantly impact performance. Sampling-based methods like RRT and PRM are probabilistically complete but not asymptotically optimal. Extensions such as RRT* aim to address optimality, but these still face challenges in solving both constrained and unconstrained planning problems efficiently in complex environments.

Learning-based approaches have emerged to overcome these limitations. Deep learning methods improve sampling efficiency by generating informative sample nodes, leading to reduced planning times. Reinforcement learning (RL) has shown strong potential for training robotic arms to find optimal paths in unknown environments. However, RL methods often require extensive interaction with the environment, which may not be practical in real-world scenarios. Learning by demonstration (LbD),

or imitation learning, provides a safer alternative, using human demonstrations to generate trajectories. This is particularly suitable for manipulator arms, as human hand motions serve as excellent demonstrations.

Overall, learning-based techniques present a promising alternative to classical motion planning methods, especially in scenarios requiring adaptability and speed. The ability to generate near-optimal paths in real-time enables robotic manipulators to operate efficiently in dynamic settings. However, several challenges remain, most notably the generation of efficient training datasets and robust generalization across different environments.

In industrial applications, smooth and optimal trajectories with low computation time are essential. Despite significant progress, real-time motion planning in high-dimensional spaces remains an open problem, especially when coupled with tasks like grasping [168, 169, 170, 171]. Therefore, further research is needed to develop more practical, scalable, and generalizable planning algorithms.

In conclusion, learning-based motion planners have demonstrated considerable potential and are likely to shape the future of motion planning in industrial robotics. This area offers valuable opportunities for collaboration between academia and industry.

Table 2.3: Summary of motion planning methods

Motion planning method		Example	Key Information	Selected studies
Classical methods	Potential Field	–	- Good performance in environment with dynamic obstacles - Trapping in local minimum is a significant issue in this method - This method is a good option for combining with other methods	[172],[62]
	Bio-inspired heuristic	GA, PSO	- Powerful optimization tool - This method is not complete - This method is not efficient in dynamic environment	[173], [78]
	Sampling-based	RRT, PRM	- Simple structure and easy to implement - This method is probabilistically complete - RRT and PRM are not optimal - Variants of these methods like RRT* and PRM* are asymptotic optimal - Long run time in complex environments and high-dimensional configuration space	[92],[102], [106]
Learning-based methods	Deep learning	MPNet Deep Sampling-based CoMPNet	- Generating efficient sample nodes for sampling-based methods - Able to learn optimal path with efficient dataset - Capable for solving constrained motion planning problem and generate near optimal and real time trajectory - Providing efficient dataset for training process is challenging	[2],[114]
	Reinforcement learning	DDPG, Actor-critic	- Powerful method in unknown environments - Require many experiments which is not feasible in some practical problems - Reward function is key concept in RL methods. efficient reward function can reduce number of inefficient experiments	[174], [175], [176]
	Learning by demonstration	–	- Able to generate a trajectory for complex tasks like assembling - Human hand is one of the best teachers in LbD method to generat a trajectory for robotic arms	[165],[3]

Chapter 3

PPCNet: Path Planning and Collision Checking Network

Bin-picking, which involves object detection, motion planning, and grasping, is a fundamental component of industrial automation. It has gained considerable attention in recent years due to the challenges it presents in computer vision and motion planning [177, 178]. A typical industrial bin-picking setup consists of a 2D/3D camera system for capturing environmental information (including object and obstacle detection), a conveyor system, and bins containing objects. The vision module identifies object positions and obstacle geometries, while the motion planner computes a feasible path to the grasp pose based on this information [179]. Efficient and real-time path planning for robotic manipulators can significantly improve the throughput and reliability of mass production and assembly.

In a standard bin-picking cycle, a combination of machine vision, inverse kinematics, and motion planning algorithms is used to determine an optimal grasp configuration. Once identified, the motion planner generates a collision-free path from the robot's initial (home) configuration to the grasp pose, and then to a target place

location. Finally, the robot executes the motion to complete the task. Due to the sequential and critical nature of planning and execution, the total cycle time is heavily influenced by both the planning and movement durations. It is therefore crucial to ensure that computed paths are not only efficient but also collision-free, considering both environmental and self-collisions.

While classical planners are applicable to bin-picking tasks, they often overlook the repetitive structure inherent in such operations. In static environments where the robot operates over long periods with consistent setups, the planning problem is typically constrained to a fixed set of initial configurations and a bounded workspace of potential end-effector poses. In this context, a more efficient solution tailored to bin-picking applications is proposed: the *Path Planning and Collision Checking Network (PPCNet).* PPCNet is a deep learning-based framework specifically designed to exploit the repetitive nature of bin-picking tasks by rapidly generating waypoints and evaluating their collision status.

The PPCNet architecture consists of two coordinated neural networks: one predicts the next waypoint along the path, and the other determines whether the segment connecting the current state to the next waypoint is collision-free. This structure enables fast and reliable path generation, offering competitive success rates and path quality while significantly reducing planning time compared to classical approaches. The proposed framework is especially well-suited for repetitive, real-time applications such as industrial bin-picking.

The main contributions of this chapter are summarized as follows:

- This chapter presents **PPCNet**, an end-to-end deep learning framework for efficient and reliable path planning in industrial settings.
- This work introduces a post-processing dataset refinement strategy that improves the quality and smoothness of the generated paths.

- This research addresses the bottleneck of collision detection in motion planning by integrating a learned collision checking module that significantly reduces planning time.
- This work compares two distinct training methodologies for the collision checking network, evaluating their performance in terms of safety and computational efficiency.

3.1 Problem Definition

This section formally define the path planning problem in the context of a bin-picking application. Additionally, the notations used throughout this chapter is introduced.

The central concept in motion planning is the *configuration space* (C_{space}), which represents the set of all possible configurations of the robot. A configuration $q \in C_{\text{space}}$ denotes a specific pose of an n-DOF robotic manipulator and can be defined as:

$$q = \begin{bmatrix} \theta_1 & \cdots & \theta_n \end{bmatrix}^{\mathrm{T}} \tag{3.1}$$

where θ_i is the angular position of the i-th joint of the robot arm.

The subset of C_{space} in which the robot remains collision-free is denoted as $C_{\text{free}} \subseteq C_{\text{space}}$. Let $\tau = \{q_1, q_2, \ldots, q_N\}$ be a trajectory from an initial configuration q_{initial} to a target configuration q_{target}. The goal of path planning is to find such a trajectory τ where all intermediate segments lie entirely within C_{free}. A segment between two adjacent configurations q_i and q_{i+1} is defined as the linear interpolation of the two configurations. For a valid trajectory, each segment must satisfy:

$$\big\{\alpha q_i + (1-\alpha) q_{i+1} \mid \alpha \in [0,1]\big\} \subseteq C_{\text{free}}, \quad \forall i \in \{1, \ldots, N-1\} \tag{3.2}$$

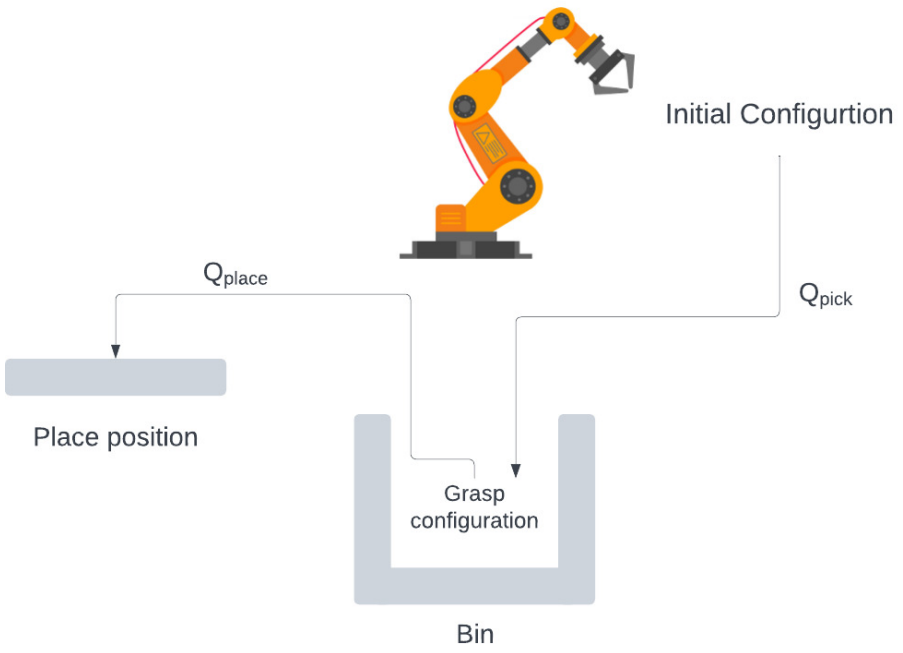

Figure 3.1: Path planning for pick and place operation

Bin-picking problem defines two path planning queries:

- **Pick path:** The trajectory from the robot's initial configuration to a grasp configuration inside the bin, denoted as Q_{pick}.
- **Place path:** The trajectory from the grasp configuration to a designated placement configuration, denoted as Q_{place}.

Figure 3.1 provides an overview of the pick-and-place operation. The objective is to design a planner that can fulfill the following criteria:

- Generate a safe, collision-free path.
- Respond to planning queries in real time.
- Achieve high path quality and a high success rate.

3.2 Path Planning and Collision Checking Framework

This section presents the proposed framework for path planning in bin-picking tasks. As illustrated in Figure 3.2, the framework consists of two sequential neural networks: the *planning network* and the *collision checking network*.

The two-network architecture is motivated by several key considerations. Foremost among them is the importance of safety in robotic motion—ensuring collision-free trajectories is a fundamental requirement. To accelerate the planning process, a collision checking network that performs significantly faster collision assessments than traditional analytical methods is introduced. Moreover, due to the inherent approximation error in the planning network, its outputs may occasionally result in collisions. To address this limitation, a subsequent verification step is necessary to assess the feasibility of each waypoint. The collision checking network fulfills this role, acting as a safeguard to ensure safety. By leveraging both networks, this approach achieves a balance between efficiency and safety, enabling rapid and reliable robot motion planning.

The planning network is a modified version of MPNet [114]. The original MPNet architecture includes two components: a planning network that sequentially generates waypoints by imitating an expert planner, and an encoder network that compresses environmental information into a fixed-length vector to inform planning.

In this setting, the primary obstacles include the bin, fixed structures, and the robot itself. These elements are fully known and incorporated into the planning environment. Unlike MPNet, PPCNet does not rely on depth maps or other sensory inputs to encode the environment, thereby reducing computational overhead and enabling faster planning—a key objective of this work. In a complete bin-picking

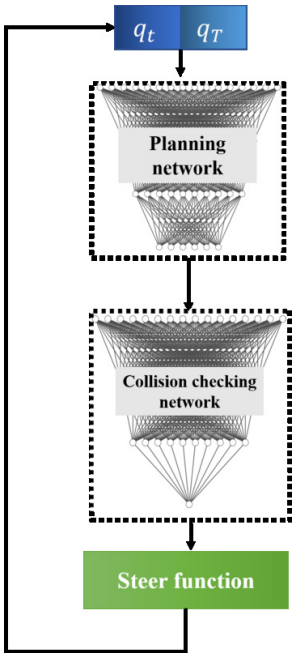

Figure 3.2: Path planning and collision checking network (PPCNet).

pipeline, the grasp selection process ensures that grasp poses are chosen to minimize the risk of collision during both grasp execution and path following.

In the proposed framework, the planning network receives the current and goal joint configurations and outputs the next joint configuration to move toward. Each configuration is represented as an n-dimensional column vector; hence, the network input is a $2n$-dimensional vector formed by concatenating the current and goal configurations. At each decision step, the proposed configuration is validated by the second network—the collision checking network—which evaluates the likelihood that the segment between the current and proposed configurations lies within the collision-free space, C_{free}. This structure enables efficient decision-making in configuration space by alternating between planning and collision checking.

3.2.1 Training and Dataset Generation

The primary objective of the path planning network is to imitate the behavior of an expert planner. When expert demonstrations are feasible, imitation learning provides an effective approach. Although RRT-based planners are not inherently sequential decision-makers, their behavior can be modeled as:

$$q_{t+1} = f(q_t, q_{\text{target}}) \tag{3.3}$$

The goal of the neural planner is to learn the function f. A foundational approach within imitation learning is *behavior cloning*, which uses supervised learning to replicate expert behavior. In this paradigm, expert demonstrations are decomposed into state-action pairs and treated as independent and identically distributed (i.i.d.) training data.

A significant challenge in behavior cloning is the risk of encountering out-of-distribution data during inference. Since the planner may encounter states not represented in the training data, even minor errors can compound and result in trajectory failures. To mitigate this issue, the Dataset Aggregation (DAgger) is algorithm [180], which incorporates online correction during training by including states visited by the learned policy.

The training procedure for both the neural planner and the collision checker is outlined in Algorithm 4. Initially, a set of expert demonstrations is generated. The function $RandomGoalConf(T)$ samples T goal configurations within the collision-free space of the bin. These sampled configurations are passed to the expert planner via the $GeneratePath$ function, which produces two datasets: the planner dataset D and the collision checker dataset C. The dataset D contains only the final, optimized paths generated by the expert planner, while C includes all intermediate segments

evaluated by the planner's geometric collision checker. In the case of RRT-based planning, D stores the final path, whereas C records the entire RRT tree and all instances of its steering function.

To ensure high-quality demonstrations, post-processing is applied to the expert-generated paths. As illustrated in Figure 3.4, the expert planner first generates a collision-free path for a given query. This path is then refined using *Binary State Contraction* (BSC), which removes redundant waypoints to produce a shorter, more efficient path. Waypoint sparsity is desirable, as it simplifies subsequent tasks such as collision checking, robot communication, and trajectory planning. BSC is computationally more efficient than the Lazy State Contraction method in [113], reducing overall runtime.

After BSC, the *resampling* procedure is applied to ensure smoothness and regularity. This step discretizes the contracted path into evenly spaced waypoints that lie on the original collision-free trajectory. These additional points do not increase path length but help reduce the mean and variance of step magnitudes $||q_{t+1} - q_t||$. While larger steps reduce the number of inference calls and collision checks, smaller steps have several advantages: (i) reduced deviation from the reference path, (ii) higher likelihood of being collision-free, and (iii) more normalized training targets. Smaller steps effectively transform the learning problem into one of direction prediction. Notably, BSC must precede resampling, as resampling relies on subdividing longer steps, which is not feasible with already dense paths. Figure 3.3 demonstrates this process in a 2D setting, showcasing its impact on path smoothness and quality. The *PostProcess* function in Algorithm 4 summarizes the combined application of BSC and resampling.

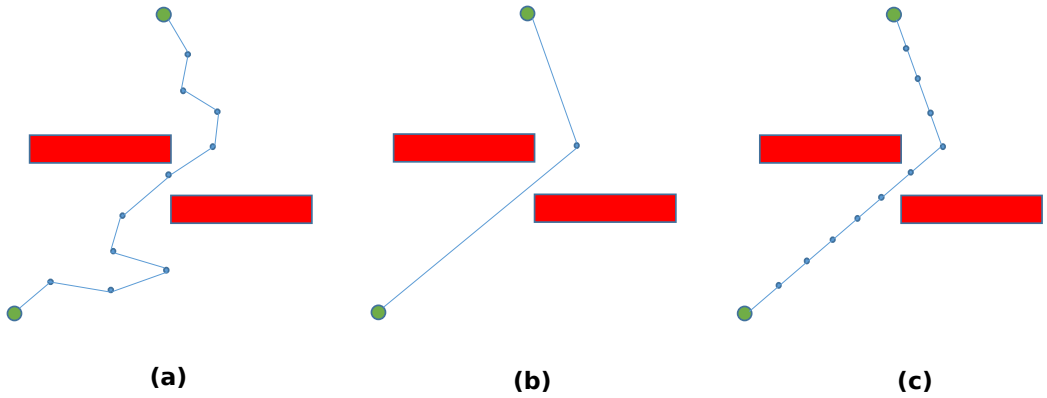

Figure 3.3: Post-processing procedure: (a) Initial path generated by the planner, (b) Path after Binary State Contraction (BSC), (c) Path after resampling.

3.2.2 Path Planner Training Formulation

Once the dataset has been generated, the network is trained to predict the next configuration $\hat{q}_{t+1}$ as accurately as possible, minimizing the discrepancy from the ground truth q_{t+1}. The network learns the underlying mapping from input to output through supervised learning by minimizing a loss function that quantifies the error between predicted and true configurations.

This optimization is performed using backpropagation, typically with stochastic gradient descent (SGD) or its variants. The objective function for training the planner network is defined as:

$$L_{\text{planner}} = \frac{1}{N_t} \sum_{j=1}^{N_d} \sum_{i=1}^{T} \|\hat{q}_{j,i} - q_{j,i}\|^2 \tag{3.4}$$

where N_t denotes the number of trajectories, N_d is the number of individual waypoints in the training dataset, and T represents the number of steps per trajectory. The loss captures the mean squared error across all predicted configurations.

3.2.3 Collision Checker Training Formulation

Two different training strategies for the collision checking network were investigated and evaluated to ensure safe motion planning. As discussed earlier, this network takes a pair of configurations—the current and the next—as input and estimates the likelihood that the connecting segment is collision-free. Thus, its training data consists of labeled segments, where each label indicates whether the segment is in collision or not.

The loss function used to train the collision checker in both approaches is the binary cross-entropy loss:

$$L(\hat{p}, p_{\text{true}}) = \hat{p} \log p_{\text{true}} + (1 - \hat{p}) \log (1 - p_{\text{true}}) \tag{3.5}$$

Here, $\hat{p}$ is the probability predicted by the network, and p_{true} is the ground-truth label. Since the task is essentially a binary classification problem, binary cross-entropy is the standard and effective choice for training such networks, as it promotes a well-calibrated probabilistic output.

Binary Labels. The most straightforward approach is to use binary labels provided directly by the analytical collision checker. In this case, p_{true} in Eq. 3.5 is either 0 (collision-free) or 1 (in-collision), and the corresponding loss is referred to as L_{binary} in Figure 3.4. This method treats each segment independently as a binary classification problem.

Population-Based Labels. Binary labels are often sparse and imbalanced, leading to biased learning toward the dominant class. A method is proposed to convert binary labels into continuous estimates, thereby better capturing the spatial context of the segment. Specifically, a reference probability $\hat{p}_{\text{ref}}$ is computed to indicate how likely a segment is to be collision-free, based on nearby segments.

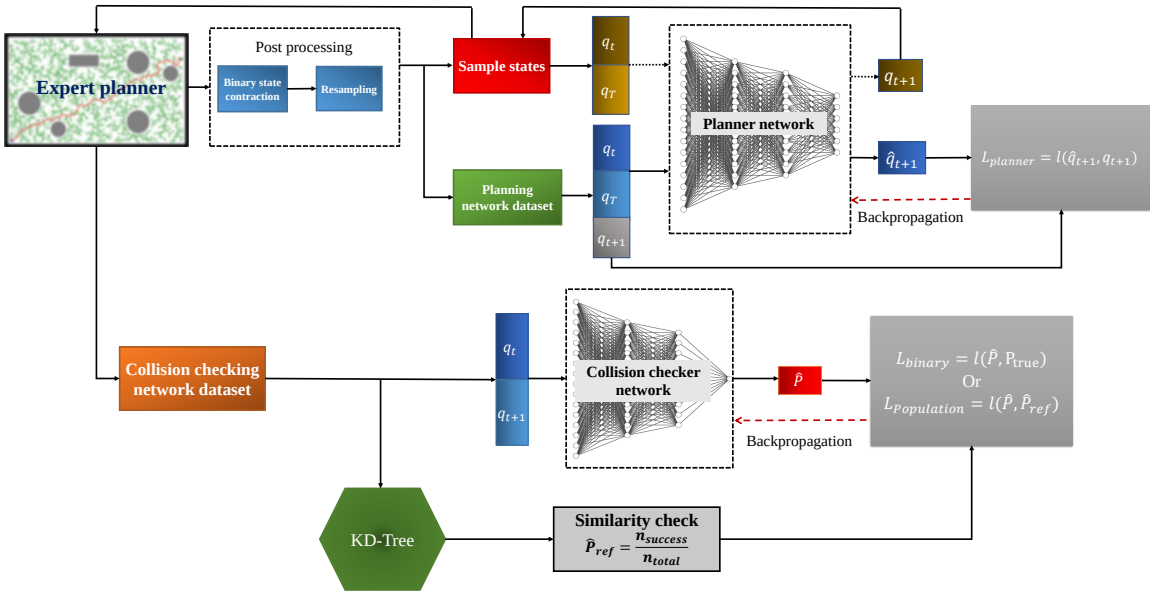

Figure 3.4: End-to-end training process of PPCNet. **Top row:** The imitation learning and data aggregation process for training the planner network. Based on random queries and the DAgger algorithm, the planner dataset is constructed with tuples of the form (q_t, q_T, q_{t+1}). The loss function L_{planner} is used for backpropagation. **Bottom row:** The population-based probability estimation and collision checker training. KD-Tree is employed to compute reference probabilities, and the collision checker is trained using either L_{binary} or $L_{\text{population}}$.

For each segment, K neighboring segments are identified within a hypersphere of fixed radius centered at its midpoint, using a KD-Tree. The population-based probability is then estimated as:

$$\hat{p}_{\text{ref}} = \frac{\sum_{k=1}^{K} \mathbb{1}[p_{\text{true}}^{(k)} = 0]}{K} \tag{3.6}$$

where $\mathbb{1}[\cdot]$ is the indicator function. This results in a continuous label that reflects the local density of collision-free segments. Near obstacles, $\hat{p}_{\text{ref}}$ approaches zero, and it increases as the distance from obstacles grows. This smooth labeling enables better gradient flow during training and improves generalization, particularly in edge cases.

All segments evaluated by the analytical collision checker during expert planning are stored to ensure comprehensive coverage. For RRT-based planners, this includes

Algorithm 4 Training process of PPCNet

Require expert planner π^*, number of initial demonstrations T, number of rollouts in each iteration T',number of state samples S, required policy success rate ζ
Initialize Planner and Collision Dataset: $D, C \leftarrow \emptyset$
Generate initial demonstrations:
  $GoalConf \leftarrow RandomGoalConf(T)$
  $D, C \leftarrow GeneratePath(\pi^*, HomeConf, GoalConf)$
Apply post-processing: $PostProcess(D)$
$SuccessRate \leftarrow 0$
for $i = 1, ...$ **do**
  $\pi_i \leftarrow TrainPolicy(D)$ ▷ Start of DAGGER
  $GoalConf \leftarrow RandomGoalConf(T')$
  $D_{\pi_i} \leftarrow GeneratePath(\pi_i, HomeConf, GoalConf)$
  $S_i \leftarrow SampleStates(D_{\pi_i}, S)$
  $D_i, C_i \leftarrow GeneratePath(\pi^*, S_i, GoalConf)$
  $D_i \leftarrow PostProcess(D_i)$
  $D \leftarrow D \cup D_i$. ▷ End of DAGGER
  $C \leftarrow C \cup C_i$.
  $\eta_i \leftarrow TrainCollisionChecker(C)$
  $SuccessRate \leftarrow TestPolicy(\pi_i, \eta_i)$
  if $SuccessRate > \zeta$ **then**
    return π_i, η_i
  end if
end for

the entire search tree and all sampled segments, both in-collision and collision-free. While this method is more computationally expensive, our experiments show that it leads to significantly better prediction performance. The corresponding loss function, $L_{\text{population}}$, is computed by substituting $\hat{p}_{\text{ref}}$ for p_{true} in Eq. 3.5.

3.2.4 End-to-End Training Process

This section provides an overview of the end-to-end training procedure for the integrated model, which comprises both the planner network and the collision checker network. The training process is summarized in Algorithm 4.

The process begins with generating an initial set of expert trajectories. These are used to populate the planner and collision checker datasets. In each training iteration,

the planner network is trained using the expert demonstrations. Following this, the DAgger process (as described in Section 3.2.1) is executed to iteratively augment both datasets with additional data generated from the current policy and corrected by the expert.

Once the datasets are updated, the collision checker network is trained using the expanded collision dataset. To evaluate overall model performance, the function *TestPolicy* is invoked. This function samples random goal configurations and attempts to generate valid trajectories between them, following the procedure outlined in Algorithm 5. If the success rate of the model exceeds a predefined threshold, the final weights of both the planner and collision checker networks are returned as the output of the training process.

3.2.5 Path Planning Process

This section describes the end-to-end path planning procedure of PPCNet. As illustrated in Figure 3.2, the two core components—the planning network and the collision checking network—operate sequentially to make decisions. In the context of bin-picking, given the start, pick, and place configurations, PPCNet generates a trajectory that enables the robot to pick an object from the bin and place it at the designated location.

The full planning process is outlined in Algorithm 5. At each iteration, the planning network predicts the next configuration for the robot arm to transition into. This configuration, along with the current one, is then evaluated by the collision checking network. For this purpose, the *Steer* function is employed.

As detailed in Algorithm 6, the *Steer* function discretizes the segment between two waypoints into smaller sub-segments of approximately the same resolution as that used in expert planner demonstrations. This ensures that the segments lie within the

Algorithm 5 Path generation process using PPCNet

Require Trained neural planner π and collision checker η, and initial and goal configurations q_{start}, q_{target}
$path \leftarrow \{q_{start}\}$
$q_{current} \leftarrow q_{start}$
for $i = 1, ..., s_{max}$ **do**
  if $Steer(q_{current}, q_{target}, \eta) == q_{target}$ **then**
    $path \leftarrow path \cup q_{target}$
    if $IsFeasible(path)$ **then**
      return $path$
    else ▷ Apply Patching
      $segments \leftarrow InCollision(path)$
      for (q_s, q_e) *in segments* **do**
        $path_{alt} \leftarrow \pi^*(q_s, q_e)$
        if $Failed(path_{alt})$ **then**
          return failure
        end if
        $path \leftarrow Patch(path, path_{alt})$
      end for
      return $path$
    end if
  end if
  $q_{next} \leftarrow \pi(q_{current}, q_{target})$
  if $Steer(q_{current}, q_{next}, \eta)$ **then**
    $q_{next} \leftarrow Steer(q_{current}, q_{next}, \eta)$
    $path \leftarrow path \cup q_{next}$
    $q_{current} \leftarrow q_{next}$
  end if
end for

distribution learned during the collision checker's training. Each segment is evaluated for collision-free probability. A segment is considered safe if its predicted probability exceeds a predefined safety threshold. If the predicted probability falls below the threshold, the segment is classified as in-collision.

If the first segment is in collision, the planner is deemed to have failed at that step. However, if the in-collision segment is not the first, the *Steer* function returns the last collision-free configuration as a valid intermediate waypoint. If the *Steer* function succeeds, the validated waypoint is added to the trajectory. If it fails, the

Algorithm 6 Steer Function $(q_{current}, q_{next})$

Require collision checking network η, safety threshold $\bar{\eta}$
$segments \leftarrow Discretize(q_{current}, q_{next})$
$q_{final} \leftarrow q_{current}$
for (q_i, q_e) *in* $segments$ **do**
  if $\eta(q_i, q_e) > \bar{\eta}$ **then**
    $q_{final} \leftarrow q_e$
  else
    if $q_{final} == q_{current}$ **then**
      return failure
    else
      return success, q_{final}
    end if
  end if
end for

planner attempts to generate a new waypoint. To support recovery from such failures, stochasticity is introduced during inference by enabling dropout layers, following the approach in [113]. Additionally, at each iteration, the algorithm checks whether the current configuration can be directly connected to the target configuration. If this direct connection is deemed safe, the target is added to the trajectory, and the planning process concludes.

Once a complete path is generated, its feasibility is verified using the analytical geometric collision checker via the $IsFeasible$ function. If the path is collision-free, it is returned as the final output. Otherwise, the algorithm identifies and attempts to repair the in-collision segments using the expert planner. The $InCollision$ function identifies such segments. When multiple consecutive segments are in collision, they are grouped into a single larger segment defined by the head and tail configurations to reduce the number of expert planner calls and improve computational efficiency. For each identified in-collision segment, the expert planner is invoked to generate an alternative collision-free path. If all such segments are successfully patched, the final path is returned. Otherwise, the planner is considered to have failed in that trial.

3.3 Experimental Results

This section presents the performance evaluation of the proposed method, implemented using PyTorch [181] for both the planner and collision checker networks. The simulation environment is built on PyBullet Planning [182, 183]. The proposed PPCNet is compared against several state-of-the-art planning algorithms: Bi-RRT [184], Informed RRT* [106], BIT* [107], and MPNet [114]. The evaluation is conducted across three distinct environments (see Figure 3.5): (i) a UR5 robot with a bin, (ii) a UR5 robot with a wall obstacle, and (iii) a Kinova Gen3 robotic arm in both simulated and real-world scenarios.

All experiments were conducted on a system equipped with a 3.7 GHz AMD Ryzen 9 processor, 96 GB of RAM, and an NVIDIA TITAN RTX GPU. The hyperparameters used for each planner are detailed in Table 3.1.

3.3.1 Comparative Study

Table 3.2 presents a comparative analysis of PPCNet against baseline planners. The results show that PPCNet significantly outperforms existing methods in terms of planning time across all environments, achieving more than a 70% improvement. Figure 3.6 provides a visual comparison of planning times. The RRT-based baselines were carefully tuned to ensure similar path lengths and completion within 10 seconds or 1000 iterations. However, BIT* and Informed RRT* still exhibited longer planning times and lower success rates compared to PPCNet.

PPCNet also generates paths with comparable lengths and success rates, which are key metrics for evaluating planning performance. The post-processing step used during dataset generation allows PPCNet to outperform even its expert planner (Bi-RRT) in path length. Specifically, the Binary State Contraction (BSC) component

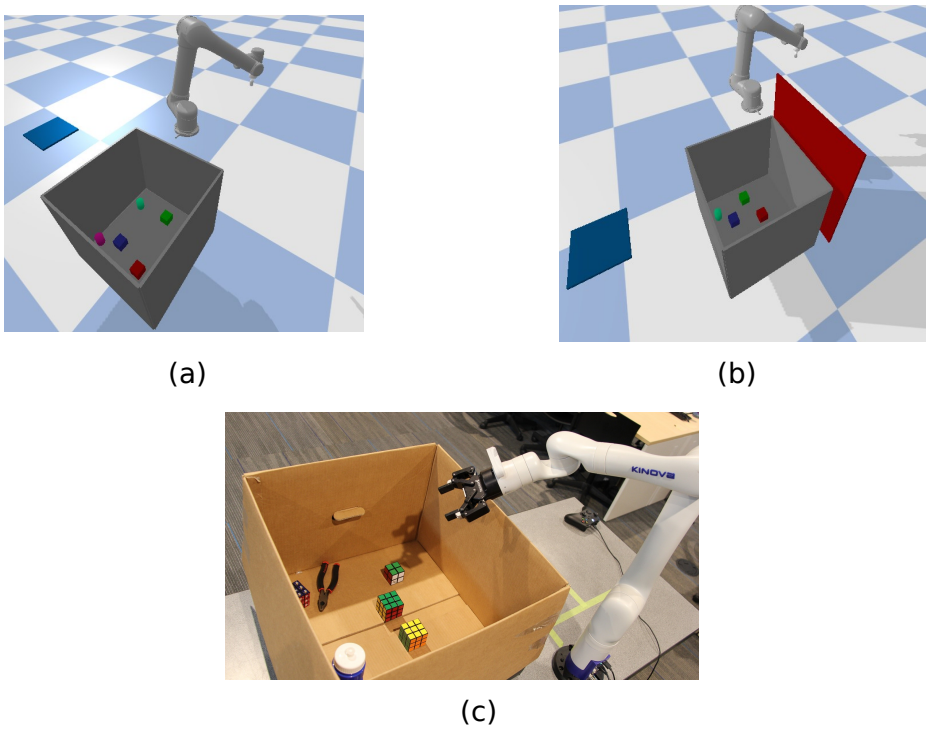

Figure 3.5: Experimental environments: a) UR5 scene, b) UR5 scene with a wall as an obstacle, c) Real-world implementation on Kinova Gen3

effectively removes redundant waypoints, reducing path length and improving quality. Combined with resampling, this refinement enhances the smoothness and consistency of generated paths.

Furthermore, while integrating collision checking into PPCNet improves robustness compared to MPNet, it introduces a slight reduction in success rate. This trade-off is a result of PPCNet prioritizing speed and path quality within strict thresholds.

The success rate not reaching 100% in known environments can be attributed to two primary factors. First, due to function approximation limitations, the planner may fail to precisely replicate expert behavior, especially near obstacles. The expert planner occasionally produces paths that closely graze obstacle boundaries, which the learned planner may not accurately replicate, resulting in failure. Second, strict thresholds are imposed to prioritize planning efficiency: a maximum planning time

Table 3.1: Hyperparameter selection for planners

Methods	Hyperparameters	Values
Bi-RRT	Max iterations	1000
	Max planning time	5 s
	Resolution	0.1 rad
Informed RRT*	Number of batches	500
	Resolution	0.05 rad
	γ	500
	Goal probability	0.1
	Max iterations	1000
	Max planning time	10 s
BIT*	Max iterations	1000
	Max planning time	10 s
	Number of samples	500
	η	20
	Goal probability	0.1
MPNet	Max iterations	100
	Learning rate	0.0001
	Expert planner	Bi-RRT
	Network architecture	Fully connected 6 layers, 300 neurons per layer
PPCNet	Post-processing resample step size	0.1745 rad
	Max iterations	100
	Optimizer	Adam
	Similarity distance	0.4 rad
	DAgger iterations	30
	Expert planner	Bi-RRT
	Safety threshold ($\bar{\eta}$)	0.8
	Network architecture (planning and collision checking networks)	Fully connected 6 layers, 300 neurons per layer

of 300 ms and a limit of 100 waypoints per trajectory. Any plan exceeding these thresholds is considered a failure. While relaxing these thresholds could increase the success rate, it would come at the cost of efficiency.

Lastly, our experiments indicate that training the collision checker using population-based labels yields superior performance compared to binary classification. This approach leads to better prediction accuracy, reducing the frequency of patching in the final trajectories and thereby lowering total planning time. Although more computationally intensive during training, the population-based method demonstrates a favorable trade-off between training cost and inference performance.

Table 3.2: Planning time and path length comparison of the proposed method and BI-RRT for 500 random pick-and-place queries

Environments	**Methods**	**Planning time (s)**	**Path length (rad)**	**Success rate** (%)
UR5	Bi-RRT	0.701 ± 0.246	7.516 ± 2.147	100
	Informed RRT*	10.186 ± 3.192	5.304 ± 1.983	79.8
	BIT*	8.621 ± 3.594	5.278±2.315	77.6
	MPNet	0.384 ± 0.079	7.305 ± 2.298	94.4
	PPCNet Collision: Binary	0.101 ± 0.031	5.887 ± 1.038	90.2
	PPCNet Collision: Population	0.093 ± 0.033	5.679 ± 1.503	93.4
UR5 with a wall	Bi-RRT	0.749 ± 0.182	7.925 ± 3.307	100
	Informed RRT*	10.856 ± 3.425	5.813 ± 2.056	73.6
	BIT*	9.299 ± 3.919	5.514 ± 2.747	70.4
	MPNet	0.392 ± 0.106	7.874 ± 3.011	93
	PPCNet Collision: Binary	0.106 ± 0.027	6.142 ± 1.979	89.8
	PPCNet Collision: Population	0.099 ± 0.034	6.016 ± 1.749	92.2
Kinova Gen3	Bi-RRT	0.645 ± 0.281	6.812 ± 2.749	100
	Informed RRT*	9.535 ± 4.122	5.124 ± 2.205	79.2
	BIT*	8.496 ± 3.082	5.099 ± 2.464	79.6
	MPNet	0.227 ± 0.151	6.714 ± 2.812	90.6
	PPCNet Collision: Binary	0.090 ± 0.029	5.315 ± 1.482	88.2
	PPCNet Collision: Population	0.085 ± 0.032	5.188 ± 1.335	89.8

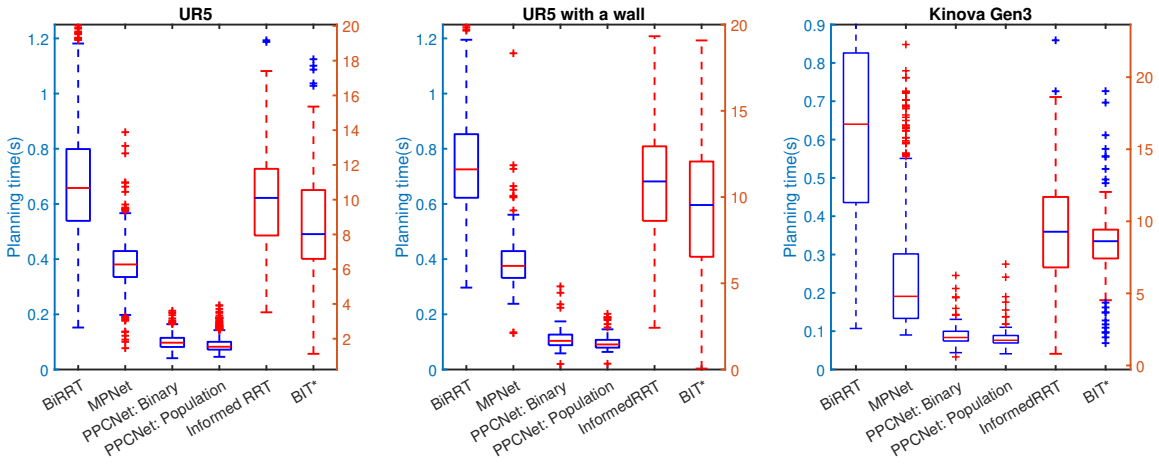

Figure 3.6: Planning time comparison between different methods

3.3.2 Real-World Implementation

This section presents the real-world deployment of PPCNet on a Kinova Gen3 robotic arm for a bin-picking task. Due to the architecture of our planning framework, the final trajectory execution depends on the specific software stack, parameter settings, and robot hardware being used. The practical applicability of PPCNet was demonstrated through bin-picking operations on the Kinova Gen3 arm, executed via the Kinova Kortex API [185]. The API allowed us to convert the planned waypoints into executable robot trajectories.

The maximum velocity of all six joints was limited to 36°/s during hardware deployment to ensure safety. This velocity cap was also used to compute the trajectory duration for each path segment. Figures 3.7 and 3.8 show motion profiles recorded during a representative bin-picking scenario. The results demonstrate that the robot followed the PPCNet-generated path with both smoothness and safety. In particular, the Cartesian trajectory in Figure 3.7 maintains a safe clearance from the bin, confirming the collision-free nature of the path. Moreover, the joint velocity and acceleration profiles in Figure 3.8 show no abrupt fluctuations, indicating stable and

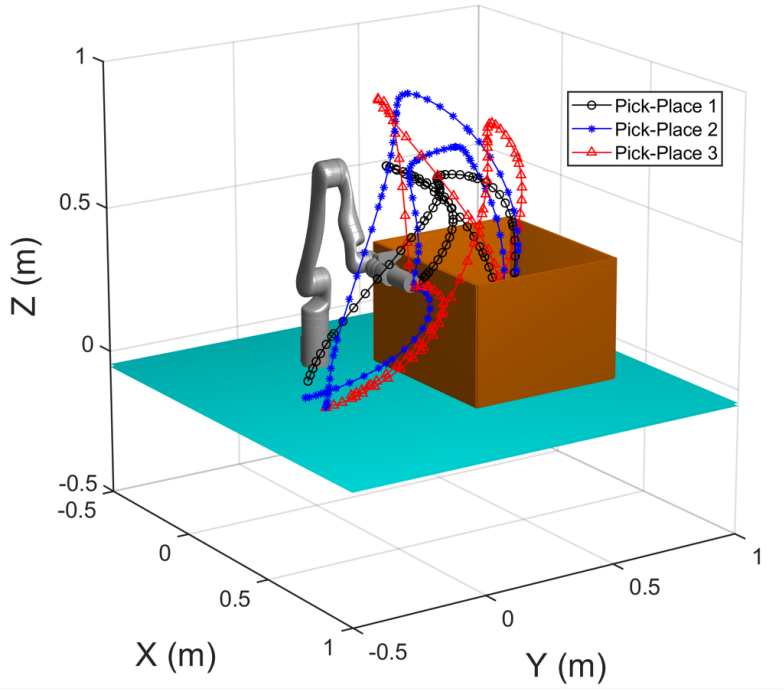

Figure 3.7: Cartesian space trajectory for a randomly selected bin-picking scenario using the Kinova Gen3 robotic arm. The robot is tasked with picking and placing three different objects onto a table.

smooth robot motion across multiple real-world trials.

Although PPCNet does not explicitly perform trajectory optimization, the results indicate that the generated paths are readily convertible into smooth executable trajectories.[1] Trajectory planning and refinement techniques in the literature—such as energy-efficient or time-optimal trajectory optimization [186, 187, 188]—can be integrated with PPCNet to further improve performance. Combining such techniques with PPCNet's efficient planning capability could enable high-quality, real-time motion generation for practical robotic applications.

[1] A demonstration video of a bin-picking task using our proposed method is available at: https://youtu.be/lYxLWPiOXuA

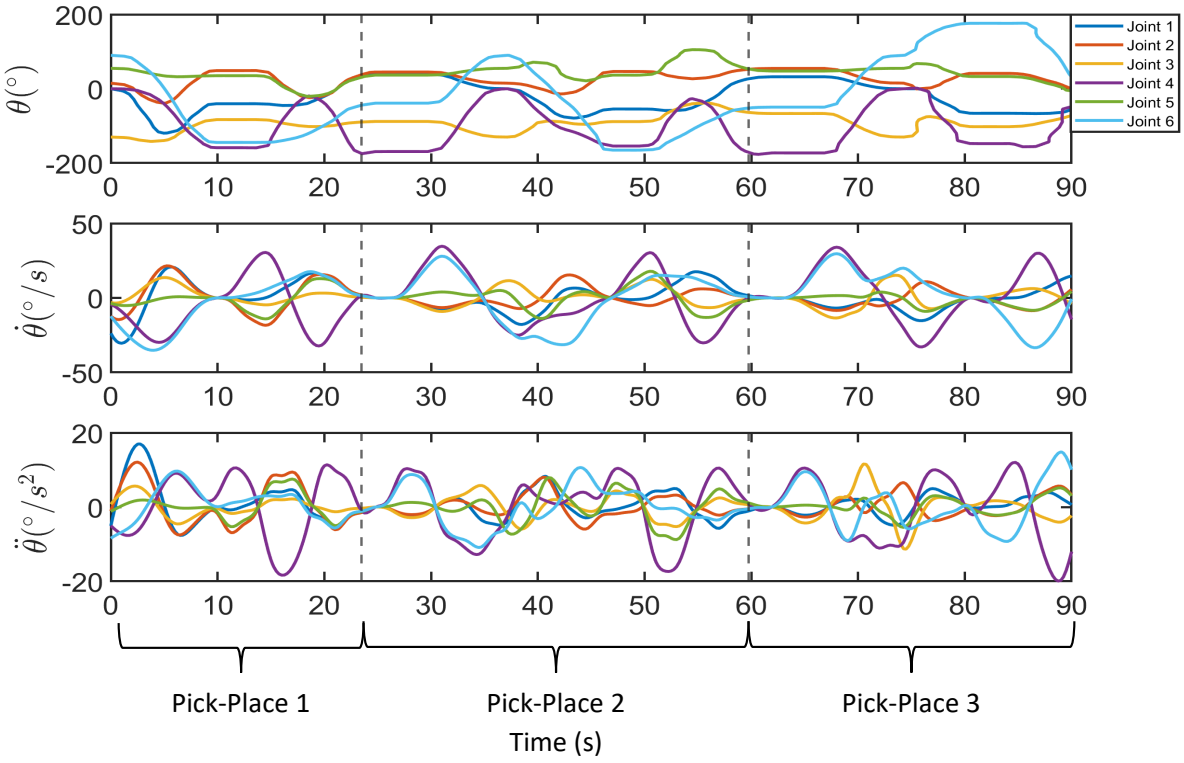

Figure 3.8: Joint-level motion profile for a randomly selected bin-picking scenario with the Kinova Gen3 arm. **Top Row**: Joint angles; **Middle Row**: Joint velocities; **Bottom Row**: Joint accelerations.

3.4 Conclusions

This paper presented a novel learning-based path planning framework, PPCNet, and demonstrated its effectiveness in bin-picking applications. Traditional motion planning algorithms often require recomputation of entire paths for each new goal configuration, making them computationally intensive and inefficient for real-time applications. Additionally, many existing methods do not leverage past experience or learned knowledge to improve future planning.

PPCNet addresses these limitations by employing deep neural networks to solve both the path planning and collision checking problems in real-time. An end-to-end training algorithm is proposed, incorporating the DAgger method to generate i.i.d.

training data for supervised learning. The use of an expert planner during training enables the model to generalize to previously unseen states.

A key contribution of PPCNet lies in its dual-network design, which includes two versions of a collision checking network that approximate the behavior of an exact geometric collision checker. These networks significantly accelerate the validation of motion segments, contributing to faster and safer planning.

PPCNet is evaluated in simulation using two robotic platforms (UR5 and Kinova Gen3) within bin-picking scenarios. Additionally, the framework is implemented on a physical Kinova Gen3 robotic arm to assess its real-world applicability. The experimental results confirm that PPCNet significantly reduces planning time while achieving path lengths and success rates on par with state-of-the-art methods.

Chapter 4

GADGET: A Diffusion-Based Framework for Generalizable Motion Planning

Motion planning is a critical component of autonomous robotic systems, particularly for manipulation tasks where a robotic arm must navigate complex configuration spaces while avoiding obstacles. In the previous chapter, PPCNet [189] was introduced, a learning-based framework capable of generating collision-free trajectories in real time. PPCNet demonstrated strong performance in static environments. However, its core limitation lies in its lack of generalization: any significant change in the workspace, obstacle configuration, or robot kinematics requires a complete retraining of the model.

Modern robotic applications often demand planners that are not only fast but also adaptable, capable of operating in previously unseen environments, handling a wide range of object configurations, and transferring across different robotic platforms without manual intervention or retraining. The high cost of data collection and

annotation, along with the need for operational flexibility, calls for planners that can generalize from limited demonstrations and reason over a diverse set of motion strategies.

To address these challenges, this chapter presents a novel path planning framework based on diffusion models, a class of generative models that have recently shown remarkable success in high-dimensional data generation tasks. Unlike conventional planners that rely on deterministic optimization or fixed policy inference, diffusion-based planners learn a data-driven distribution over feasible trajectories and generate motion plans through an iterative denoising process. This enables them to model complex, multi-modal trajectory distributions and to produce diverse, smooth, and goal-directed motions conditioned on arbitrary scene inputs—including novel obstacle configurations and robot morphologies. Crucially, this framework eliminates the need for retraining when deployed in new environments or on different robotic arms.

The following sections details the proposed framework, including the formulation of the motion planning problem as a conditional generative process, the architecture and training procedure of the diffusion model, and how conditioning is achieved through spatial and goal-based embeddings. It also describes how guidance mechanisms can be incorporated during inference to improve constraint satisfaction and goal achievement. Extensive evaluations on both simulated and real-world robotic manipulators demonstrate that our approach achieves real-time inference, strong generalization, and robustness to environmental variability, making it a powerful tool for path planning in any robotic environments.

4.1 Preliminaries

Before presenting the proposed diffusion-based planning framework, the necessary background and notation used throughout this chapter will be established. This includes a formal definition of the motion planning problem for robotic manipulators, an overview of trajectory representations, and key concepts related to diffusion models. These preliminaries form the foundation for understanding how generative modeling is integrated into path planning. In addition, Control Barrier Functions (CBFs) will be discussed, which is later used to guide the diffusion process during path generation.

4.1.1 Diffusion Models

Diffusion models have recently emerged as a powerful class of generative methods, offering notable advantages for modeling trajectories and control policies in robotics. Denoising Diffusion Probabilistic Models (DDPMs) [123] are latent variable models that synthesize data through a learned, iterative denoising process. The central idea involves a forward diffusion process in which structured data, such as a path, is progressively corrupted by Gaussian noise over multiple time steps until it becomes indistinguishable from pure noise. A neural network is then trained to reverse this process by denoising the data step by step, effectively learning to reconstruct samples from the original data distribution. During inference, new trajectories are generated by starting from pure noise and applying the learned denoising procedure, producing smooth and feasible motion plans (see Figure 4.1).

The forward diffusion process is defined by a sequence of conditional Gaussian distributions:

$$q(\tau^i|\tau^{i-1}) = \mathcal{N}\left(\tau^i; \sqrt{1-\beta_i}\,\tau^{i-1},\, \beta_i\mathbf{I}\right) \tag{4.1}$$

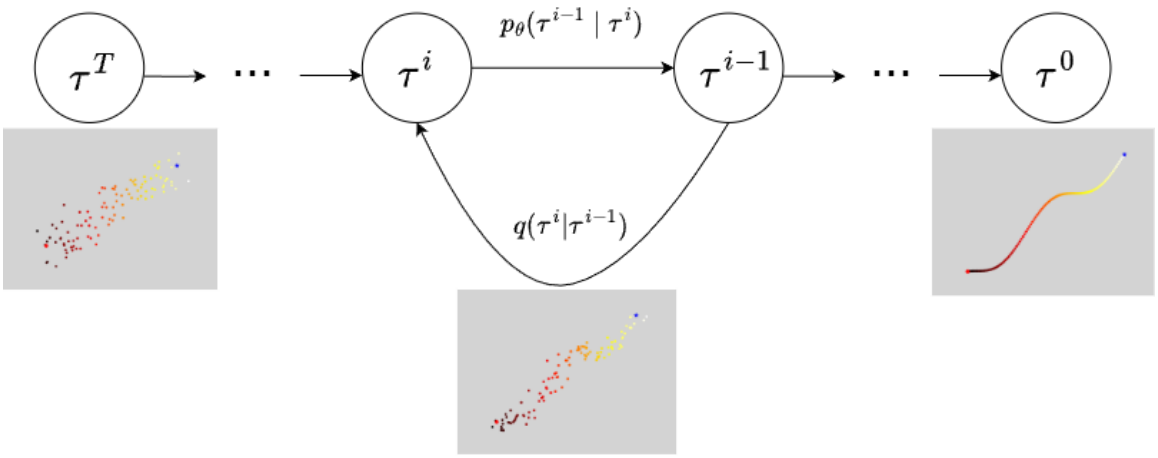

Figure 4.1: Forward and reverse (denoising) process of Diffusion model.

In the above equation:

- $q(\tau^i|\tau^{i-1})$ represents the forward diffusion process, i.e., the conditional probability of obtaining the noisy path τ^i at step i, given the path τ^{i-1} from the previous step.
- τ^i is the path at diffusion timestep i, after adding noise.
- τ^{i-1} is the path at the previous diffusion timestep $i-1$, before further noise is added.
- $\mathcal{N}(.; \mu, \Sigma)$ denotes a Gaussian distribution with mean μ and covariance Σ.
- $\sqrt{1-\beta_i}$ is a scaling factor applied to τ^{i-1} which ensures that the variance of the process increases gradually over time.
- β_i is a hyperparameter that controls the amount of noise added at each timestep i.
- $\mathbf{I}$ is the identity matrix, used here to define an isotropic Gaussian noise with variance β_i in all dimensions.

The following equation defines the reverse diffusion process, which is modeled as a sequence of conditional probability distributions that iteratively denoise the path.

$$p_\theta(\tau^{i-1} \mid \tau^i) = \mathcal{N}\left(\tau^{i-1}; \mu_\theta(\tau^i, i), \Sigma_\theta(\tau^i, i)\right) \tag{4.2}$$

where a neural network parameterized by θ predicts the mean and variance for sampling τ^{i-1} from τ^i. The goal is to train this network in a way that the reverse diffusion process accurately reconstructs the original data distribution by progressively denoising the input. As a result, the loss function used to train the diffusion model can be formulated as follows. This is the simplified training objective commonly used in DDPMs, where the model learns to predict the noise ϵ added to the original data τ^0 at timestep i.

$$L(\theta) = \mathbb{E}_{i,\tau^0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta\left(\sqrt{\bar{\alpha}_i}\,\tau^0 + \sqrt{1-\bar{\alpha}_i}\,\epsilon,\, i\right)\right\|^2\right] \tag{4.3}$$

here ϵ is Gaussian noise, ϵ_θ is the neural network which predict the noise, and $\bar{\alpha}_i = \prod_{j=1}^{i}(1-\beta_j)$ is the cumulative product of the noise schedule. It is important to note that this work adopts the cosine noise schedule proposed by Nichol and Dhariwal [190], and assume a fixed, precomputed variance for the reverse process, i.e., $\Sigma_\theta(\tau^i, i) = \Sigma^i$.

While standard diffusion models are effective at generating diverse and high-quality trajectories, they are typically formulated as unconditional or weakly conditioned generative processes. However, in the context of motion planning, it is often essential to guide path generation to satisfy specific task constraints, such as reaching a goal state, avoiding obstacles, or ensuring safety. *Guided diffusion* addresses this limitation by steering the reverse denoising process using auxiliary information or task-specific objective functions. Two primary approaches have been developed for

guiding the diffusion process:

Classifier-Guided Diffusion: Classifier guidance is a widely adopted technique that influences the diffusion model during inference without modifying its training procedure [191]. It employs an external classifier $C(\cdot \mid y)$ that estimates the likelihood of a trajectory τ satisfying a desired condition y (for example, obstacle avoidance or goal reaching). From Bayes' rule, the reverse process on y is conditioned:

$$p(\tau^{i-1} \mid \tau^i, y) \propto p(\tau^{i-1} \mid \tau^i) \cdot C(y \mid \tau^{i-1}) \tag{4.4}$$

During sampling, the gradient of the classifier's log-probability is used to bias the denoising path toward regions of higher task relevance. The guided mean is computed as:

$$\mu_{\text{guided}}(\tau^i, i) = \mu_\theta(\tau^i, i) + \lambda \cdot \Sigma^i \nabla_{\tau^i} \log C(y \mid \tau^i) \tag{4.5}$$

where:

- $\mu_\theta(\tau^i, i)$ is the mean predicted by the diffusion model at timestep i,
- Σ^i is the corresponding variance,
- $C(y \mid \tau^i)$ is the classifier's estimate of the likelihood that τ^i satisfies condition y,
- λ is a scaling factor that controls the strength of the guidance signal.

This classifier-guided reverse process encourages the diffusion model to generate trajectories that not only adhere to the learned data distribution but also conform to task-specific constraints. Such guidance is particularly beneficial in motion planning, where task success critically depends on satisfying geometric and kinematic constraints.

Classifier-Free Guidance: Classifier-free guidance is an alternative to classifier-based diffusion that eliminates the need for a separate, external classifier during inference. Instead, it integrates the guidance mechanism directly into the generative model by training it to handle both conditional and unconditional generation within the same architecture. This approach was introduced by Ho and Salimans [192] and has become a widely adopted technique due to its simplicity and effectiveness.

During training, the model $\epsilon_\theta(\tau^i, i, y)$ is trained to predict the noise ϵ added to the clean path τ^0, conditioned on both the diffusion timestep i and some conditioning variable y. Moreover, with some probability p, the condition y is randomly dropped (replaced with a null token), allowing the model to learn both conditional and unconditional denoising behaviors.

At inference time, the final noise prediction is computed as a weighted combination of the conditional and unconditional outputs:

$$\epsilon_{\text{guided}} = (1 + \lambda) \cdot \epsilon_\theta(\tau^i, i, y) - \lambda \cdot \epsilon_\theta(\tau^i, i, \emptyset) \tag{4.6}$$

where:

- $\epsilon_\theta(\tau^i, i, y)$ is the noise prediction conditioned on the desired target or context y,
- $\epsilon_\theta(\tau^i, i, \emptyset)$ is the unconditional noise prediction (i.e., no conditioning),
- λ is a guidance scale hyperparameter that adjusts the strength of the conditioning.

This linear interpolation amplifies the influence of the conditioning signal while maintaining sample diversity. Classifier-free guidance is particularly appealing for motion planning tasks, enables a unified training and inference process and has been shown to produce high-quality, goal-directed trajectories without sacrificing generative flexibility.

Advantages of Diffusion Models for Motion Planning

Motion planning is inherently multi-modal; multiple valid trajectories may exist for the same task and environment. Traditional learning architectures, such as RNNs, autoregressive models, or VAEs, often struggle to capture such trajectory diversity and suffer from issues like error accumulation or limited conditioning flexibility.

Diffusion models, by contrast, offer a powerful generative framework for modeling and sampling entire trajectories in a way that naturally aligns with the needs of robotic planning. Specifically, diffusion models bring the following advantages:

- **Trajectory-Level Generation:** Diffusion models predict all timesteps of a trajectory jointly (non-autoregressively), enabling coherent long-horizon behavior synthesis without accumulating compounding errors.
- **Multi-Modality and Generalization:** Diffusion models can capture rich, multi-modal distributions over trajectories, enabling the generation of diverse solutions for the same task. Through flexible conditioning mechanisms they generalize effectively to novel start-goal pairs and previously unseen environments with varying layouts.
- **Guided Planning:** The iterative denoising process enables guidance via auxiliary gradient signals (such as cost functions, goal constraints, or control barrier functions) allowing task objectives to be composed or swapped without retraining.
- **Temporal Compositionality:** Diffusion-based planner models entire trajectories as structured sequences, enabling the learning of temporally coherent behaviors. By iteratively denoising complete joint trajectories, the model captures long-range dependencies and composes motions that are globally consistent.

This temporal structure allows the planner to generalize across environments by recombining learned sub-trajectories (reaching, avoiding, aligning) into new motion plans. Moreover, the use of guidance signals, such as cost-based gradients, enables temporal corrections that propagate across the planning horizon, enhancing safety and feasibility without retraining.

These properties make diffusion models especially compelling for motion planning in robotics, where safety, generalization, and path diversity are essential for deploying agents in complex and long-horizon settings.

4.1.2 Control Barrier Functions

CBFs [193] provide a formal mechanism for enforcing safety constraints in control systems by ensuring forward invariance of a designated safe set. Let us consider a control-affine nonlinear dynamical system of the form:

$$\dot{x} = f(x) + g(x)u \tag{4.7}$$

where $x \in \mathbb{R}^n$ is the system state, $u \in \mathbb{R}^m$ is the control input, $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$ are locally Lipschitz continuous functions representing the system dynamics.

To encode safety, a continuously differentiable scalar function is defined $h : \mathbb{R}^n \to \mathbb{R}$, known as the control barrier function, which characterizes the forward-invariant safe set:

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\} \tag{4.8}$$

The goal is to ensure that if the system starts within $\mathcal{C}$, it remains in $\mathcal{C}$ for all future time: $\mathcal{C}$ is forward invariant. This is guaranteed if the time derivative of $h(x)$

along the system trajectories satisfies the following inequality:

$$\sup_{u \in \mathbb{R}^m} \left[\nabla h(x)^\top (f(x) + g(x)u) \right] \geq -\alpha(h(x)) \tag{4.9}$$

for a chosen extended class-$\mathcal{K}$ function α, which is typically chosen as a linear function $\alpha(h) = \gamma h$ for some $\gamma > 0$. This inequality imposes a constraint on the allowable control inputs u such that the system does not violate the safety condition $h(x) \geq 0$.

Equation (4.9) defines a set of control inputs $\mathcal{K}_{\text{cbf}}(x) \subseteq \mathbb{R}^m$ that are admissible for maintaining forward invariance of $\mathcal{C}$. CBFs can be integrated into control and optimization frameworks such as quadratic programs (QPs) or reinforcement learning policies to ensure constraint satisfaction in real time [194, 195, 196].

In the context of generative motion planning, the CBF condition can be used in the path generation process as a differentiable constraint or guidance signal. When applied to diffusion models, this enables the integration of safety-critical knowledge, such as obstacle avoidance or joint limit enforcement, directly into the denoising process, without sacrificing the generative flexibility of the model. Using the strengths of diffusion models and CBFs, the following section introduces a motion planning framework titled GADGET, Generalizable and Adaptive Diffusion-Guided Environment-aware Trajectory generation, designed for safe, efficient, and generalizable robotic manipulation.

4.2 GADGET framework

The GADGET framework is designed to perform generalizable and collision-free path planning for robotic manipulators across diverse environments. It combines voxel-based scene encoding, conditional generative modeling via denoising diffusion, and safety guidance using CBFs. The modular design enables transferability across unseen

environments and different robot platforms. Figure 4.2 depicts the structure of this framework.

4.2.1 Scene Perception via Voxel Carving

GADGET employs a voxel carving pipeline to reconstruct the 3D occupancy of the workspace. Multiple depth cameras placed around the robot capture views of the environment. Each depth image is back-projected into 3D space to obtain a point cloud of occupied regions, which is then discretized into a fixed-resolution voxel grid. Voxels along free-space rays between the camera and visible surfaces are marked free. The result is a binary occupancy grid from which the coordinates of occupied voxels are extracted and processed through a scene encoder inspired by [197]. Shared MLP layers embed each voxel coordinate, and voxel-wise max pooling aggregates these into a global latent representation $\phi_\psi(s)$. This embedding captures the geometry and spatial distribution of obstacles, providing geometric context to the diffusion model for planning collision-free trajectories.

Effect of Voxel Resolution. The resolution of the voxel grid determines the granularity of spatial representation. Finer voxels capture subtle geometric details which is useful for accurate obstacle boundaries and narrow passages; however, it increases the dimensionality of the occupancy tensor and the encoder's computational load. Conversely, coarser grids reduce memory and inference time at the cost of geometric precision, potentially blurring thin obstacles or introducing aliasing in free-space boundaries.

4.2.2 Conditional Embedding

To guide path generation, the diffusion model is conditioned on both the encoded scene and the robot's task specification. The condition vector is defined as $y =$

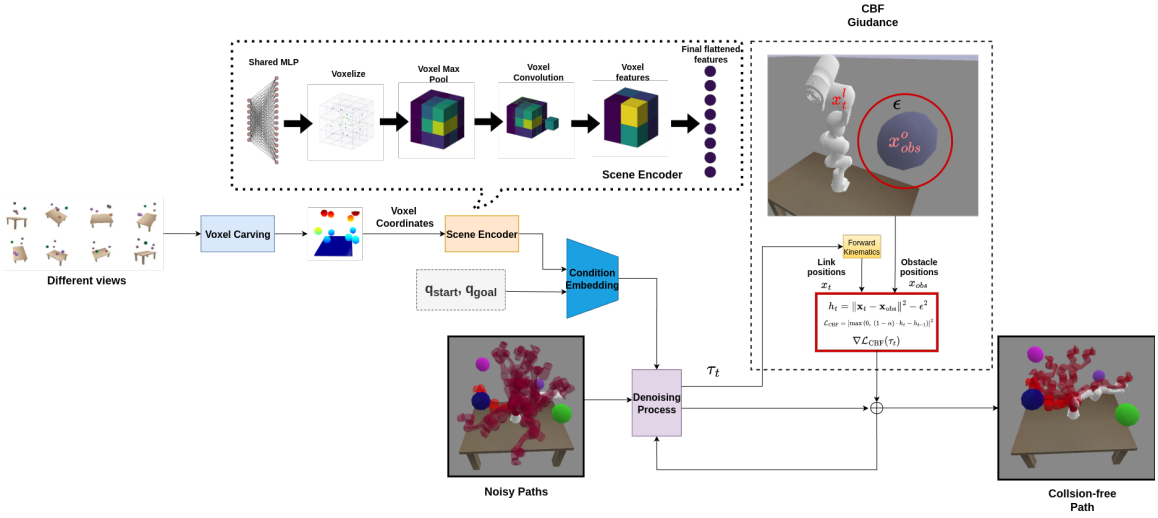

Figure 4.2: Generalizable and Adaptive Diffusion-Guided Environment-aware Trajectory generation framework.

$\{q_s, q_g, \phi_\psi(s)\}$, where q_s and q_g are the start and goal joint configurations, and $\phi_\psi(s)$ is the latent scene embedding. These elements are fused into a single context vector, enabling the diffusion model to generate trajectories aligned with the environment geometry and task objectives.

4.2.3 Diffusion-Based Path Planning with CBF Guidance

GADGET employs a dual-phase guidance methodology that combines task conditioning via classifier-free guidance with CBF-inspired noise perturbations for safety shaping. This method incorporates geometry-aware safety constraints into the denoising process while allowing for a wide variety of condition-driven trajectory generation.

A path denoted by τ is represented as $\tau \in \mathbb{R}^{T \times d}$, with T being the time horizon and d the configuration space dimension. The trajectory generator, which is a conditional denoising diffusion model, is represented as $\epsilon_\theta(\tau_t, t, y)$ that forecasts the noise to be eliminated from the noisy trajectory τ_t at time t conditioned on scene/task y. The

model is trained using expert demonstrations $\tau_0 \sim \mathcal{D}$ by minimizing:

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{\tau_0, t, \epsilon} \left[\|\epsilon - \epsilon_\theta(\tau_t, t, y)\|^2 \right], \tag{4.10}$$

where y may be randomly masked with probability p_{drop} [192] to support classifier-free guidance during inference.

$$\epsilon_{\text{guided}}(\tau_t, t, y) = (1 + \lambda_{\text{cfg}})\epsilon_\theta(\tau_t, t, y) - \lambda_{\text{cfg}}\epsilon_\theta(\tau_t, t, \emptyset), \tag{4.11}$$

with guidance coefficient $\lambda_{\text{cfg}} \geq 0$. The reverse process mean is then:

$$\mu_t(\tau_t, t, y) = \frac{1}{\sqrt{\alpha_t}} \left(\tau_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_{\text{guided}}(\tau_t, t, y) \right), \tag{4.12}$$

where α_t and $\bar{\alpha}_t$ follow a standard diffusion noise schedule.

Guided Diffusion as Conditional Sampling. Sampling is biased toward desirable, safe behaviors by employing a guidance signal C:

$$\tilde{p}(\tau \mid y) = p(\tau \mid y, C) \propto p_\theta(\tau \mid y) p(\tau \mid C), \tag{4.13}$$

where C denotes arbitrary inference-time constraints (e.g., safety constraints). The reverse diffusion process is sampled from:

$$p(\tau_0 \mid y, C) = p(\tau_N | y, C) \prod_{t=1}^{N} p(\tau_{t-1} \mid \tau_t, y, C) \tag{4.14}$$

with $p(\tau_N | y, C)$ standard Gaussian, so it suffices to recursively sample

$$p(\tau_{t-1} \mid \tau_t, y, C) \propto p_\theta(\tau_{t-1} \mid \tau_t, y) p(C \mid \tau_{t-1}) \tag{4.15}$$

where $p_\theta(\tau_{t-1} \mid \tau_t, y)$ is modeled with a classifier-free diffusion model. Since the reverse transition is Gaussian,

$$\log p_\theta(\tau_{t-1} \mid \tau_t, y) \propto -\frac{1}{2}(\tau_{t-1} - \mu_t)^T \Sigma_t^{-1}(\tau_{t-1} - \mu_t) \tag{4.16}$$

a first-order Taylor expansion of $\log p(C \mid \tau_{t-1})$ around μ_t [124]:

$$\log p(C \mid \tau_{t-1}) \approx \log p(C \mid \mu_t) + (\tau_{t-1} - \mu_t)^\top g \tag{4.17}$$

with $g = \nabla_\tau \log p(C|\tau)|_{\tau=\mu_t}$. Substituting Eq. 4.16 and Eq. 4.17 into Eq. 4.15 shows that the new reverse transition has mean $\mu_t + \Sigma_t g$.

Assuming independent constraints, the guidance likelihood expresses the probability that a trajectory τ satisfies all desired constraints as an exponential-family distribution is given by:

$$p(C|\tau) \propto \prod_i \exp(-J_i(\tau)), \tag{4.18}$$

where each $J_i(\tau)$ represents the penalty associated with violating the i-th constraint or safety requirement. The resulting guidance direction simplifies to:

$$g = -\sum_i \nabla_\tau J_i(\tau)|_{\tau=\mu_t} \tag{4.19}$$

Discrete-Time Dynamical System and CBF Definition. The denoising process can be viewed as a discrete-time dynamical system with state τ_t and update rule:

$$\tau_{t-1} = \tau_t + \delta_t, \tag{4.20}$$

where δ_t denotes the learned denoising direction.

For this discrete system, a CBF $h(\tau)$ is defined to encode safety constraints.

Specifically, for each robot link ℓ and obstacle o, the following safety function is defined:

$$h_t^{(\ell,o)} = \left\| \mathbf{x}_t^{(\ell)} - \mathbf{x}_{\text{obs}}^{(o)} \right\|^2 - \epsilon^2, \tag{4.21}$$

where $\mathbf{x}_t^{(\ell)} = f_{\text{FK}}^{(\ell)}(\tau_t)$ is the Cartesian position of link ℓ at step t, obtained via forward kinematics, and $\epsilon > 0$ is the safety margin specifying minimum allowable clearance Moreover, $f_{\text{FK}}^{(\ell)}(\tau_t)$ is a differentiable function.

In classical CBF theory, the time derivative of the safety function satisfies:

$$\dot{h}(\tau) = \nabla h(\tau)^T \dot{\tau} \geq -\alpha(h(\tau)) \tag{4.22}$$

In our discrete-time setting, where $\dot{\tau} \approx \delta_t$, a finite-difference approximation is used.

$$\Delta h(\tau_t) = h_{t-1} - h_t \geq -\alpha(h_t) \tag{4.23}$$

As a result, the discrete-time CBF condition requires that safety does not degrade too quickly between steps:

$$h_{t-1}^{(\ell,o)} \geq (1 - \alpha_{\text{cbf}}) h_t^{(\ell,o)}, \tag{4.24}$$

where $\alpha_{\text{cbf}} \in (0, 1)$ controls the temporal conservativeness of the barrier. Rather than enforcing Eq. (4.24) as a hard constraint, it is incorporated as a soft penalty during sampling: violations of Eq. (4.24) are penalized, and their gradients bias the denoising updates away from unsafe configurations. As a result, the CBF term acts as a soft safety-shaping mechanism: once iterates approach the safe set (i.e., $h_t^{(\ell,o)} \geq 0$), the guidance term tends to preserve or increase the safety margin, while still allowing occasional constraint violations due to the stochastic nature of diffusion. In our formulation, $h_t^{(\ell,o)}$ measures the signed squared distance between robot link ℓ and

obstacle o at denoising step t. Positive values correspond to maintaining at least a clearance of ϵ from obstacles, whereas negative values indicate constraint violations.

Soft Penalty for CBF Violations. To enforce this constraint softly during sampling, a differentiable penalty over all link–obstacle pairs is defined:

$$\mathcal{L}_{\text{CBF}}(\tau_t) = \frac{1}{LO} \sum_{\ell=1}^{L} \sum_{o=1}^{O} \left[\max\left(0, (1-\alpha_{\text{cbf}}) h_t^{(\ell,o)} - h_{t-1}^{(\ell,o)}\right) \right]^2, \tag{4.25}$$

where L and O denote the number of links of the robot and obstacles respectively. Since each $h_t^{(\ell,o)}$ is dependent on the forward kinematics, the gradient $\nabla_{\tau_t} \mathcal{L}_{\text{CBF}}(\tau_t)$ transmits the geometric feedback throughout the kinematic chain to guide the safe denoising process.

This CBF penalty function fits naturally into the framework of constraint guidance used in diffusion sampling. Each cost term $J_i(\tau)$ corresponds to an individual link–obstacle penalty:

$$J_i(\tau_t) \equiv \left[\max\left(0, (1-\alpha_{\text{cbf}}) h_t^{(\ell,o)} - h_{t-1}^{(\ell,o)}\right) \right]^2, \tag{4.26}$$

where the index i maps uniquely to (ℓ, o). The total penalty is a sum over i,

$$\mathcal{L}_{\text{CBF}}(\tau_t) = \frac{1}{LO} \sum_i J_i(\tau_t),$$

which guides the diffusion process via gradients of these penalties.

Final Reverse Update. The overall guided denoising update combines classifier-free guidance with the CBF perturbation:

$$\tau_{t-1} = \mu_t(\tau_t, t, y) - \lambda_{\text{cbf}} \nabla_{\tau_t} \mathcal{L}_{\text{CBF}}(\tau_t), \tag{4.27}$$

where λ_{cbf} is the parameter that determines the effect of safety shaping. The guidance acts as a soft barrier at each denoising step, preventing the trajectory from entering unsafe areas and at the same time allowing for task compliance.

Algorithm 7 shows the training and inference process of GADGET. The combined use of classifier-free and CBF-based guidance can be interpreted as energy shaping: the diffusion model parameterizes the conditional distribution $p_\theta(\tau_t \mid y)$, and the CBF term reshapes it toward safety-compliant regions of the trajectory space. This unified formulation provides:

- **Multi-Modality and Flexibility:** The stochastic diffusion process samples diverse, feasible trajectories, capturing multiple path hypotheses for a given task.
- **Environment Awareness:** Classifier-free conditioning on $(q_s, q_g, \phi_\psi(s))$ ensures that generated trajectories are well aligned with cluttered and previously unseen environments.
- **Safety Enforcement and Cross-Embodiment Transfer:** The CBF-based loss penalizes proximity to obstacles at every denoising step. Because this safety shaping relies solely on geometric distances computed from robot-specific forward kinematics, the same model transfers seamlessly across different manipulators without retraining.

4.3 Experimental Results

To evaluate the proposed framework, the effectiveness of GADGET through the following key questions will be investigated:

- **Q1** Can it generalize to unseen environments without retraining?

Algorithm 7 GADGET: Diffusion-Based Path Planning with CBF Guidance

TRAINING

Input: Expert dataset $\mathcal{D}$ (collision-free trajectories), diffusion model ϵ_θ, noise schedule $\{\bar{\alpha}_t\}_{t=1}^N$, learning rate η, classifier-free dropout p_{drop}

while training not converged **do**

Sample expert path $\tau_0 \sim \mathcal{D}$, timestep $t \sim \mathcal{U}(1, N)$, noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

Forward diffuse path:

$$\tau_t = \sqrt{\bar{\alpha}_t}\,\tau_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$$

With probability p_{drop}, set $y = \emptyset$, else $y = \{q_s, q_g, \phi_\psi(s)\}$

Compute denoising loss:

$$\mathcal{L}(\theta) = \|\epsilon - \epsilon_\theta(\tau_t, t, y)\|^2$$

Update network parameters: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$

INFERENCE

Input: Trained model ϵ_θ, start/goal (q_s, q_g), safety weight λ_{cbf}, safety margin ϵ, temporal factor α_{cbf}, classifier-free weight λ_{cfg}

Acquire multi-view depth images $\{\mathcal{D}_i\}_{i=1}^M$

Reconstruct occupied voxel coordinates $P = \texttt{VoxelCarve}(\{\mathcal{D}_i\})$ ▷ $P \in \mathbb{R}^{N\times 3}$

Encode scene: $\phi_\psi(s) = \texttt{SceneEncoder}(P)$

Initialize noisy path: $\tau_N \sim \mathcal{N}(0, \mathbf{I})$

for $t = N, \ldots, 1$ **do**

Compute conditional and unconditional noise:

$$\epsilon_{\text{cond}} = \epsilon_\theta(\tau_t, t, y), \quad \epsilon_{\text{uncond}} = \epsilon_\theta(\tau_t, t, \emptyset)$$

Apply classifier-free guidance:

$$\epsilon_{\text{guided}} = (1 + \lambda_{\text{cfg}})\,\epsilon_{\text{cond}} - \lambda_{\text{cfg}}\,\epsilon_{\text{uncond}}$$

Predict reverse-process mean:

$$\mu_t = \frac{1}{\sqrt{\alpha_t}}\left(\tau_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_{\text{guided}}\right)$$

Compute signed distance using Eq. 4.21

Compute safety gradient: $\nabla_{\tau_t}\mathcal{L}_{\text{CBF}}(\tau_t)$ using Eq. 4.25

Update path with CBF-guided correction:

$$\tau_{t-1} = \mu_t - \lambda_{\text{cbf}} \nabla_{\tau_t}\mathcal{L}_{\text{CBF}}(\tau_t)$$

end for

Output: Final path τ_0

- **Q2** Can it transfer to different robot arms with distinct kinematics?
- **Q3** Does CBF-guided diffusion improve safety in cluttered environments?
- **Q4** How does GADGET compare to state-of-the-art learning-based and sampling-based planners?

To answer these questions, GADGET is evaluated across a variety of simulated scenarios designed to test its generalization, safety, and planning performance.

Training Phase:

Environments: 10,000 randomized training scenes populated with 10–16 spherical obstacles placed in the robot's workspace is generated (Figure 4.3).

Expert Planner: Ground-truth collision-free trajectories are computed using the BiRRT algorithm, ensuring high-quality supervision during training.

Robot: A 7-DoF Franka Emika Panda arm is used for data collection and path supervision.

4.3.1 Answering Key Questions

To evaluate the effectiveness of GADGET, a series of experiments targeting the four key questions that mentioned in previous section is designed. The following results, summarized in Tables 4.1 and 4.2, directly address these questions.

To thoroughly assess the generalization capability of GADGET, it is compared against a diverse set of classical and learning-based motion planners, each chosen for their complementary strengths and widespread adoption in the field.

Bi-RRT is a classical sampling-based planner that incrementally grows two trees from the start and goal configurations, attempting to connect them. It is probabilistically complete and is often employed as an expert oracle; however, it typically produces suboptimal paths, especially in cluttered environments.

BIT* (Batch Informed Trees) [107] is an asymptotically optimal sampling-based planner that integrates the efficiency of informed graph search (e.g., A*) with the flexibility of incremental sampling. It is well-suited for high-dimensional spaces; however, its performance strongly depends on heuristic guidance and it can become computationally expensive, often leading to large planning times in real-time settings.

MPNet [114] is a learning-based planner that employs a point cloud encoder together with a neural planner to directly predict motion trajectories. It offers fast planning performance; however, its generalization is limited and lacks the ability to adapt to unseen environments during inference.

MPD (Motion Planning Diffusion) [124] is a recent method that leverages diffusion models to learn priors over robot trajectories. It effectively captures path multi-modality and enables posterior-guided sampling; however, since it does not incorporate environment information, its performance degrades significantly when the environment changes at inference time.

DP3 [127] is a visuomotor imitation learning method that combines point cloud representations with diffusion policies to generate robot actions. It generalizes across tasks, but like other imitation-based methods, it remains dependent on demonstration data and may struggle in unseen environments.

To quantitatively compare GADGET against baseline planners, the following metrics is used:

- **Success Rate (SR)**: The percentage of planning queries for which at least one generated path is entirely collision free and reaches the goal configuration within a fixed distance. In our experiments, for each start-goal query, 30 trajectories is sampled, and the query is considered successful if at least one of them satisfies the criteria. This definition is based on [124], and reflects the multi-modal nature of diffusion models, which can generate diverse candidate solutions for

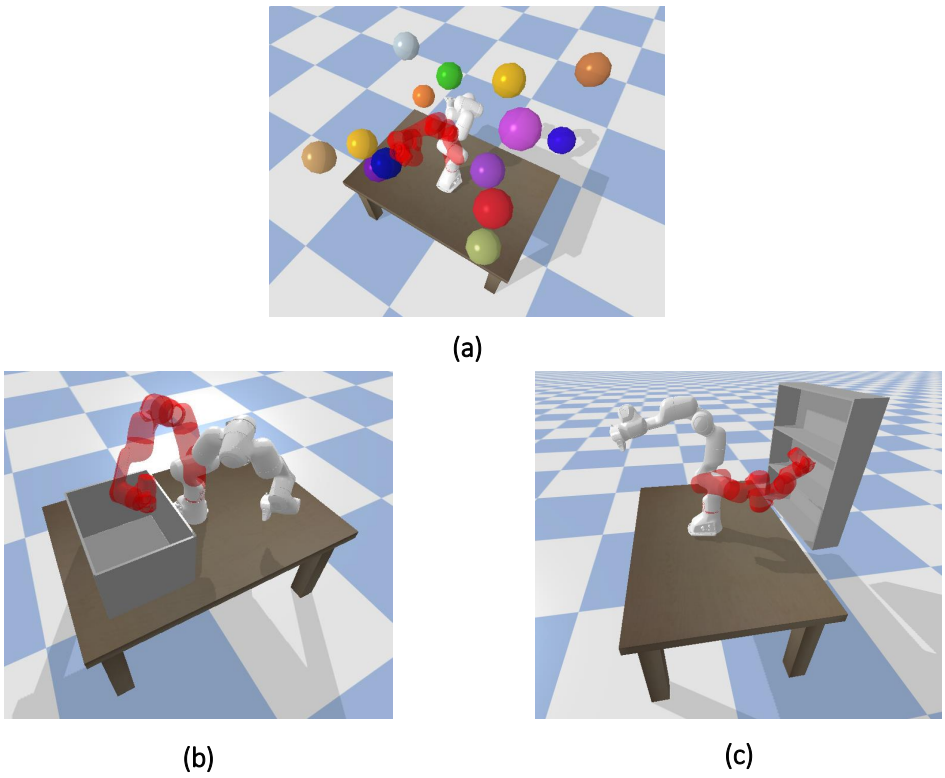

Figure 4.3: Simulation environments. (a) Spherical Obstacles: Used for training; during testing, obstacle positions are randomized to assess generalization. (b) Bin Picking: Another test-only environment not encountered during training, involving cluttered object retrieval. (c) Shelf Manipulation: A test-only environment unseen during training, representing structured manipulation tasks. In all scenes, the red shadow of the robot denotes the target (goal) configuration.

the same query.

- **Collision Intensity (CI)**: The percentage of the waypoints that are in collision with obstacles. This metric reflects the safety of the generated trajectories.

- **Planning Time (T)**: The amount of time (in seconds) required to generate a collision-free path for a random start-goal query. In case of the diffusion-based planners, 30 trajectories sampled for each query and collision checking is is performed sequentially. The planning time is recorded at the point when the first collision-free path is found.

- **Path Length (PL):** The length of the generated paths (in radians) which is measured as the sum of Euclidean distances between consecutive waypoints in joint space.

Tables 4.1 and 4.2 highlight the effectiveness of GADGET across both environment and robot generalization benchmarks (It is worth mentioning that the system used for training and testing has a 3.500 GHz AMD Ryzen 9 processor with 64 GB RAM and NVIDIA 3090 GPU). 1000 different scenarios (random start/goal configurations and random obstacle positions) were used in each environment. Although GADGET is being trained just on randomized spherical environment, it demonstrates zero-shot generalization to previously unseen test scenarios. In the evaluation, the position of obstacles are randomized and the test scenes are visually and geometrically distinct from those encountered during training (Bin and shelf). GADGET succes rate consistently remains high with low collision intensity (Figure 4.6). This confirms that GADGET generalizes effectively to novel environments without retraining (a clear answer to **Q1**).

To demonstrate the performance of GADGET with different previously unseen robots, the model is deployed on manipulators with distinct kinematics such as Franka Panda, Gen3 (7DoF/6DoF), and UR5. GADGET sustains high SR and low CI (Cross-robot transfer **(Q2)**). This transfer is driven primarily by the CBF-guided diffusion, which imposes geometry-aware safety constraints at inference time using the forward kinematics of the deployed robot, thereby providing corrective gradients that adapt to variations in link lengths, joint limits, and Jacobians. While the diffusion model learns joint-space priors implicitly conditioned on the training robot's morphology (Franka Panda), the CBF layer acts as a morphology-specific adapter that steers trajectories toward collision-free regions for the new embodiment. In contrast, MPD, DP3, and MPNet which lack such runtime geometric feedback degrade more noticeably when

transferred to unseen robots. This transferability is most effective among commonly used manipulators in industry with similar kinematic topologies (serial chains with comparable DOFs); transfer to radically different morphologies (e.g., parallel robots, soft manipulators) remains an open challenge.

To demonstrate the effect of CBF guidance, the performance of GADGET with and without this guidance signal was compared. The results show that the addition of safety constraints markedly reduces collision intensity and raises success rates, particularly in bin-picking and shelf-manipulation environments where narrow passages often cause other planners to fail (**Q3**). This indicates that CBF-guided guidance improves the safety and enhances reliability in challenging environments.

Finally, in benchmarking against state-of-the-art planners (**Q4**), GADGET demonstrates a favorable performance in all of the metrics. Classical sampling-based methods such as Bi-RRT and BIT* achieve high success rates, but only under generous search budgets. Specifically, BIT* is capped at 1 second to match the real-time budget; this naturally limits its optimality compared to longer runs. Moreover, Bi-RRT tends to produce excessively long trajectories, since it grows trees locally and connects samples opportunistically, often resulting in detours that are not globally efficient. In contrast, GADGET conditions on the full set of occupied voxel coordinates representing the scene and learns global spatial priors, enabling it to generate more direct and corridor-following paths. Interestingly, although GADGET is trained on Bi-RRT demonstrations, its diffusion-based denoising acts as a form of implicit shortcutting, capturing common structural patterns from many examples and often producing trajectories that are smoother and shorter than the teacher itself. Unlike MPNet, MPD, and DP3 which suffer from limited generalization and dependence on demonstrations, GADGET achieves consistently higher success rates, lower collision intensity, and competitive planning times across diverse environments and robot platforms, all

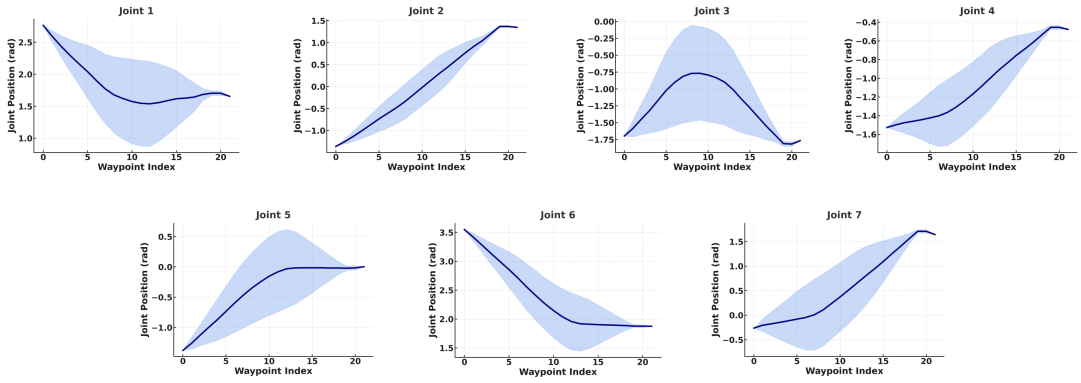

Figure 4.4: Mean and variance of joint trajectories generated by GADGET for a single start-goal query. The shaded area represents the standard deviation across 100 diffusion samples, demonstrating multi-modal planning behavior.

while maintaining path lengths comparable to existing methods.

4.3.2 Multi-Modality of Diffusion-Based Planning

In addition to quantitative performance, one of the most valuable properties of GADGET is its ability to generate diverse trajectories for the same task. This diversity arises from the stochastic sampling process of the underlying diffusion model, which models a distribution over feasible paths rather than predicting a single deterministic solution.

To illustrate this behavior, the joint-wise variance across 100 sampled trajectories was analyzed for a fixed start and goal configuration. Figure 4.4 shows the mean and variance for each joint over the planning horizon. The visible variance confirms that GADGET explores a rich, multi-modal solution space, allowing it to adapt to different local minima in the trajectory space.

4.3.3 Design Principles for Generalization

The primary goal of GADGET is to achieve zero-shot generalization to out-of-distribution (OOD) environments and unseen robot embodiments without the need for fine-tuning. This capability is crucial for real-world deployment, where robots often encounter novel workspace layouts, obstacle geometries, and kinematic structures. GADGET attains this generalization by combining a geometry-centric scene encoding, end-to-end conditional training, and inference-time safety shaping through CBFs.

Geometry-Centric Scene Representation. GADGET employs a sparse voxel encoding, a product of multi-view voxel carving, to represent the workspace. The planner takes as input the coordinates of occupied voxels. The representation, although it only encodes occupied areas, indirectly informs about the structure of free space through its geometric context. Thus, the diffusion model is made capable of reasoning about geometry in a category-agnostic manner, which means it can be deployed in shelves or bins, for instance, that were never seen during the training phase. The generated scene embeddings (Figure 4.5) show clear latent clusters corresponding to different workspace types (spherical, bin, shelf). Despite these differences, the planner generalizes well, indicating that the diffusion model leverages geometric structure rather than memorizing specific environment categories.

End-to-End Conditional Training. The occupied voxel coordinates s are mapped into a latent embedding $z = \phi_\psi(s)$ by a scene encoder ϕ_ψ. This embedding, together with the start and goal joint configurations (q_s, q_g), forms the conditioning input $y = \{q_s, q_g, \phi_\psi(s)\}$ for the denoising diffusion model ϵ_θ. During training, the model acquires the skill of predicting and removing noise from the trajectories τ_t following

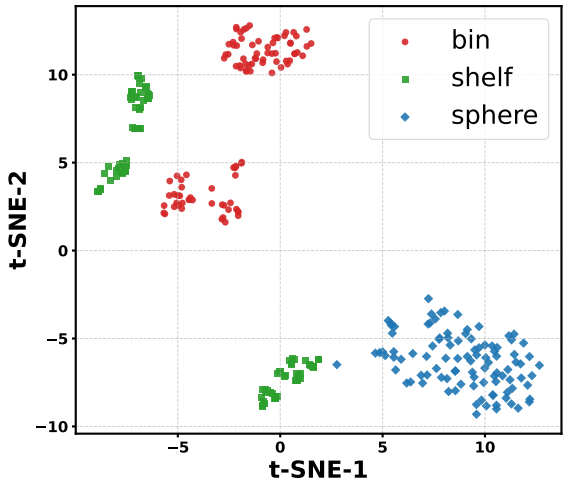

Figure 4.5: t-SNE visualization of scene embeddings for spherical, bin, and shelf environments. The embeddings form distinct clusters, confirming that the latent distributions differ across environment types while still supporting generalization in planning.

the standard diffusion objective:

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{\tau_0 \sim \mathcal{D},\, t \sim \mathcal{U}(1,N),\, \epsilon \sim \mathcal{N}(0,I)} \Big[\|\epsilon - \epsilon_\theta(\tau_t, t, y)\|^2 \Big], \tag{4.28}$$

where τ_0 is an expert path from the dataset and τ_t is its noisy version at diffusion step t. Importantly, there are no direct penalties for collisions that are applied in the training phase. Instead, the encoder and denoiser are jointly optimized end-to-end purely through this denoising objective, which enforces the encoder to produce geometry-aware, semantically meaningful embeddings that remain consistent across environments.

Inference-Time Safety Guidance. At inference, GADGET denoises a randomly initialized path over N steps while applying two complementary forms of guidance. First, classifier-free guidance sharpens the conditional distribution on $(q_s, q_g, \phi_\psi(s))$, improving the model's sensitivity to task and scene context. Second, a differentiable

CBF-inspired loss $\mathcal{L}_{\text{CBF}}$ is computed dynamically at each step based on the robot's forward kinematics and the voxel-derived scene structure. The gradient of this loss biases the reverse diffusion update away from obstacles (Eq. 4.27), steering the trajectory toward safety-compliant regions of the state space. Although the training objective is agnostic to safety constraints, this energy-based safety shaping at inference significantly enhances robustness and feasibility in unseen environments.

Robot-Adaptable Formulation. GADGET's generalization ability across robots relies on its separation of learned joint-space path priors from the robot-specific geometric constraints enforced during inference. While the diffusion model learns collision-free joint-space distributions grounded in the kinematic structure of the training robot (e.g., Franka Panda), it does not explicitly encode robot kinematics. Instead, at inference time, the robot-dependent forward kinematics are incorporated within the CBF loss to compute link-to-obstacle distances and gradients, enabling geometry-aware safety corrections. This modular design allows the same trained diffusion model to be adapted online to new robot manipulators without retraining, provided the new morphology shares fundamental kinematic characteristics. However, as the learned distribution reflects the morphology of the training robot, the transfer is not universal: trajectories suitable for one kinematic chain may be suboptimal or infeasible for radically different robots. Thus, the CBF layer serves as a vital morphology-specific safety interface, steering trajectories for safe cross-embodiment transfer within a constrained family of manipulators.

The combination of these principles provides an explanation for GADGET's success in non-training environments. The encoder–denoiser duo, tuned through end-to-end optimization, acquires a latent coordinate system associated with geometry instead of learned layouts. Although unseen environments produce different latent distributions that are disjoint from those seen in training, their codes remain inter-

Table 4.1: Performance of GADGET and baseline planners across diverse test environments.

Method	**Spherical Obs**				**Bin Picking**				**Shelf Manipulation**			
	T (s)	CI (%)	PL (rad)	SR (%)	T (s)	CI (%)	PL (rad)	SR (%)	T (s)	CI (%)	PL (rad)	SR (%)
Bi-RRT	0.24±0.12	_	11.1±4.4	99.9	0.22± 0.14	_	10.7±3.9	100	0.37±0.15	_	10.7±4.1	100
BIT*	1.02±0.02	_	8.1±2.6	94.9	1.01±0.01	_	7.6±2.1	90.1	1.01±0.02	_	8.0±2.3	77.8
MPNet	0.22±0.14	44.1	7.9±1.7	53.8	0.26±0.15	12.9	7.6±1.7	38.7	0.31±0.20	22.3	8.1±1.9	35.1
MPD	1.15±0.04	10.3	7.2±1.5	90.0	1.14±0.04	14.5	7.6±1.6	52.8	1.14±0.09	20.5	7.8±1.5	26.2
DP3	0.42±0.11	14.6	8.4±2.3	79.4	0.41±0.13	5.4	8.2±2.3	76.3	0.48±0.19	6.9	8.2±2.3	67.8
GADGET w.o CBF	0.43±0.09	13.9	8.1±2.2	79.9	0.45±0.12	5.2	8.5±2.5	85.2	0.62±0.23	6.5	8.4±2.3	74.2
GADGET with CBF	0.59±0.08	**8.2**	8.3±2.2	**95.5**	0.60±0.10	**4.6**	8.7±2.3	**92.2**	0.77±0.23	**5.9**	8.7±2.4	**80.1**

Table 4.2: Cross-robot generalization performance of GADGET compared to baselines without retraining.

Method	**Franka**				**Gen3 (7DoF)**				**Gen3 (6DoF)**				**UR5**			
	T(s)	CI (%)	SR (%)	PL (rad)	T(s)	CI (%)	SR (%)	PL (rad)	T(s)	CI (%)	SR (%)	PL (rad)	T(s)	CI (%)	SR(%)	PL (rad)
MPD	1.15±0.10	10.3	90.0	7.3±1.5	1.08±0.04	17.8	55.8	9.8±1.9	1.01±0.05	21.2	46.8	7.7±1.6	1.04±0.03	29.4	38.9	7.9±2.7
DP3	0.42±0.12	14.6	79.4	8.4±2.3	0.42±0.11	15.7	72.6	9.4±2.7	0.38±0.08	14.1	75.4	8.6±2.6	0.39±0.09	22.4	52.2	8.2±2.4
GADGET	0.59±0.08	**8.2**	**95.5**	8.3±2.2	0.61±0.07	**7.1**	**97.7**	9.8±2.8	0.71±0.17	**6.7**	**97.9**	8.7±2.4	0.67±0.09	**11.4**	**89.7**	8.2±2.0

pretable by the denoiser because the semantics of the latent space are consistent. Additionally, inference-time safety guidance reinforces feasibility further that makes zero-shot planning across new environments and robots possible.

4.3.4 Ablation on CBF Guidance Parameters

To determine the influence of the CBF parameters (the safety margin ϵ and the temporal constraint factor α_{cbf}) on the planner's behavior, an ablation study is conducted in the spherical environment. The safety margin ϵ defines the minimum required distance from the obstacles. The increasing of ϵ leads to a more conservative safety buffer, thereby reducing the risk of collision but at the same time prolonging the path as the planner has to go further away from the obstacles. The temporal factor α_{cbf} determines the aggressiveness of the planner in moving towards the safety boundary

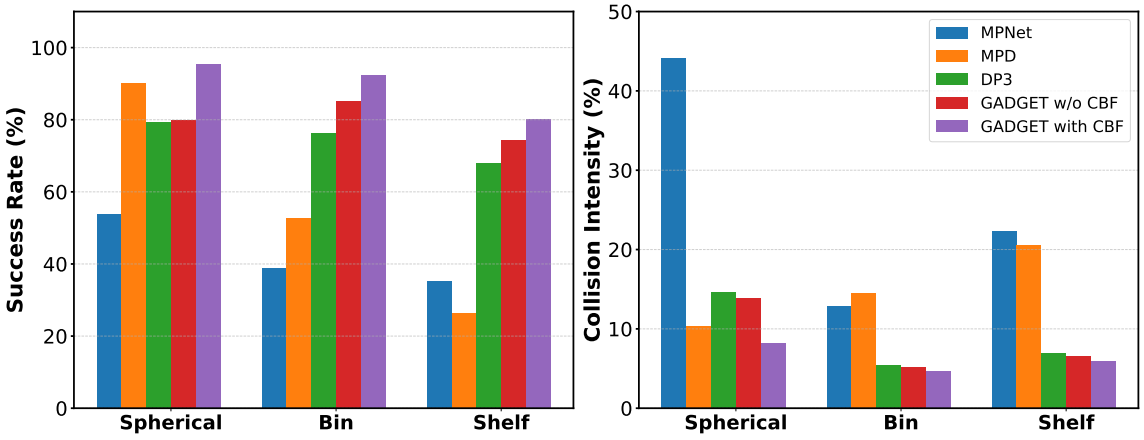

Figure 4.6: Success rate (left) and collision intensity (right) across spherical, bin, and shelf environments. GADGET outperforms learning-based baselines (MPNet, MPD, DP3), achieving higher success with lower collision rates, while CBF guidance further enhances safety.

between the denoising steps. Small α_{cbf} values demand careful, slow updates, while large α_{cbf} values relax the constraint, which can lead to faster convergence but also overshooting during the passage near obstacles.

The impact of these parameters on the safety–performance was assessed by analyzing four main metrics. The success rate (SR) reflects the portion of queries that managed to attain the goal without any collisions and higher values indicate more reliable performance. The PL captures the total joint-space distance of the planned path. The minimum clearance (d_{min}) indicates the closest distance between any robot link and the nearest obstacle along the path, with larger values indicating safer motion; this metric naturally grows as ϵ increases due to the enforced wider safety margin. Finally, CI represents the fraction of path waypoints in collision, where lower values reflect stricter adherence to hard safety constraints. It should be acknowledged that the occasional invasion of the soft safety buffer (determined by ϵ) does not invariably signify unsafe motion since the planner might still steer clear of collisions while taking advantage of the broader feasible region to enhance success rate.

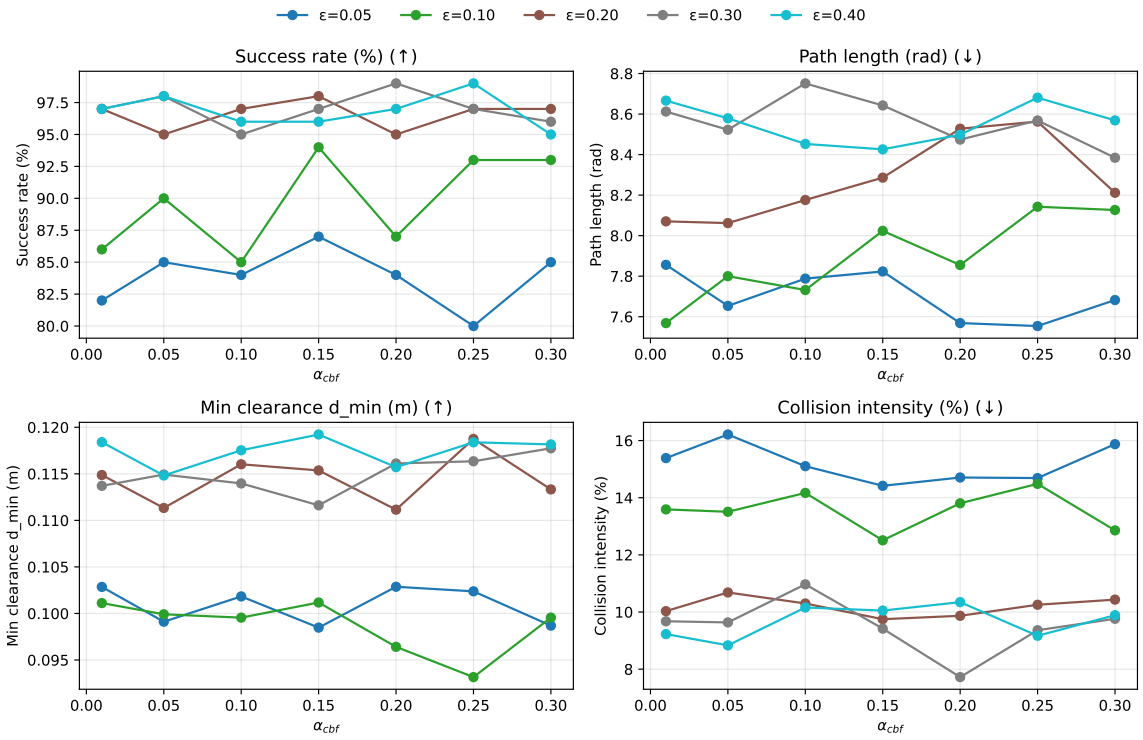

Figure 4.7: **Ablation on CBF guidance parameters.** Each curve shows the effect of α_{cbf} for a fixed safety margin ϵ. **(Top-left)** Success rate (SR): moderate α_{cbf} (typically 0.10–0.20) yields the best feasibility; very small α_{cbf} can overconstrain updates and very large α_{cbf} can destabilize denoising near obstacles. **(Top-right)** Path length (PL): increases with ϵ because larger margins shrink the feasible space and force detours, illustrating an SR–PL trade-off. **(Bottom-left)** Minimum clearance (d_{min}): higher is better; confirms that trajectories remain collision-free even when soft margins are occasionally entered. **(Bottom-right)** Collision intensity (CI): lower is better; near-zero CI with high SR indicates robust safety in practice.

The impact of the CBF guidance parameters α_{cbf} and ϵ on performance and safety is shown in Figure 4.7. The success rate shows a characteristic inverted–U dependence on α_{cbf}. Very small α_{cbf} values overly constrain updates and reduce feasibility, whereas very large values relax safety constraints too aggressively and can destabilize the denoising process near obstacles. Apart from that, SR most of the time increases as ϵ gets larger, mainly because the planner benefits a wider feasible region and stronger safety constraints.

The increment in path length that comes with the increase in ϵ is due to the

reduction of the free configuration space as a result of wider safety margins and the necessity of detours around the obstacles. The implication is that there is a fundamental trade-off; the larger the value of ϵ, the better the SR and the safer the operation but at the cost of longer and less efficient trajectories. However, this trade-off can be adjusted according to the application. For example, in the case of a safety-critical task, priority can be given to the success rate, whereas for highly efficiency-oriented tasks, shorter trajectories may be preferred. The average minimum clearance d_{min} is a direct indicator of how much the robot approaches the obstacles. As it was expected, d_{min} is larger with the larger ϵ, thus it proves that the planner keeps larger clearances when this is necessary. However, at the same time, the collision intensity remains near zero for most parameters. This implies that even when the trajectories go into the soft safety zone, they do not violate the hard collision constraints very often. These patterns confirm the strength and flexibility of GADGET over a very wide range of parameter settings and without the need for intensive fine-tuning.

4.4 Real-world Implementation

The real-world execution of GADGET follows a five-stage pipeline:

1. **Environment Scanning:** An end-effector-mounted RGB-D camera captures multiple views of the workspace from different angles. These observations provide depth data critical for scene understanding.

2. **3D Reconstruction:** The captured multi-view depth images are fused using open3D library to reconstruct a 3D model of the workspace.

3. **Digital Twin Generation in PyBullet:** To generate a digital twin of the real workspace the reconstructed voxel map is imported into PyBullet (Figure 4.9). This simulated replica will be used for collision evaluation.

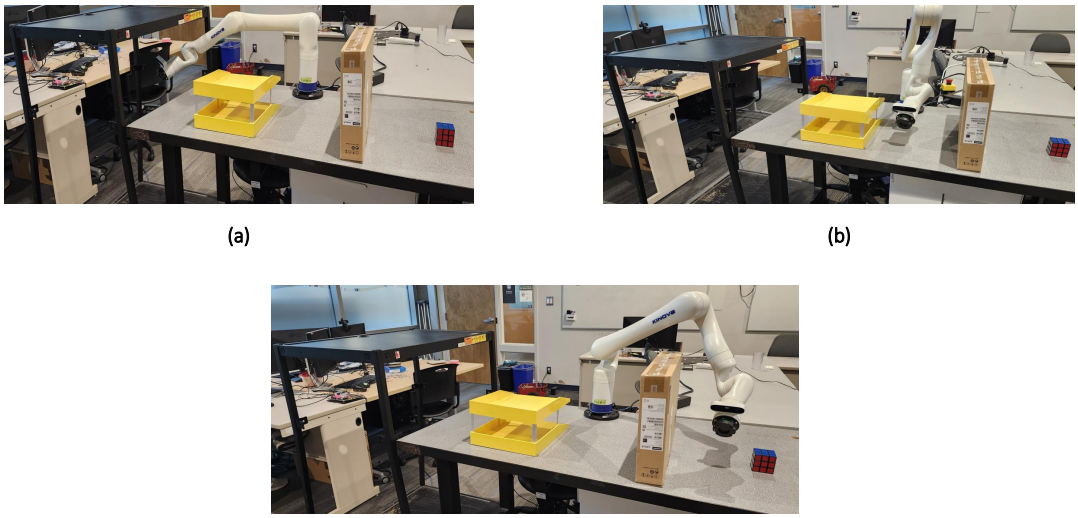

Figure 4.8: Practical implementation setup on the Kinova Gen3. The robot starts from configuration (a), moves to an intermediate goal (b), and finally proceeds to target (c). GADGET is tasked with generating collision-free trajectories between these waypoints.

4. **Path Generation via GADGET:** Given the voxel-encoded scene and goal condition, GADGET smaples a set of joint-space path using its conditional diffusion model, guided at inference time by CBF-based safety constraints.

5. **Robot Execution:** The generated path is streamed to the physical robot controller for execution. Since planning occurs directly in joint space, no additional inverse kinematics is required.

This pipeline is used to demonstrate GADGET's ability to real-world scenarios without any retraining. By combining scene perception with CBF-guided generative planning, the framework acts as a robust zero-shot path planner, capable of adapting to arbitrary changes in workspace geometry. Figure 4.8 shows our real-world setup. 50 trajectories were sampled from GADGET for this task, of which 38 were successful. those were implemented on the real robot using the Kortex API [185]. The high-level control mode were used and the maximum velocity of all six joints was limited

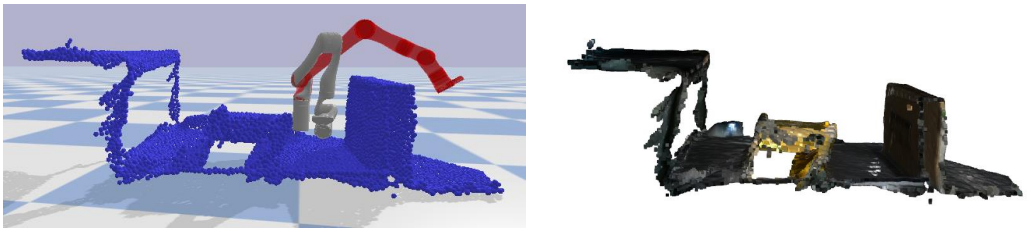

Figure 4.9: (a) Digital twin of the environment in PyBullet used for planning and evaluation. (b) 3D reconstruction of the same environment obtained from the depth camera.

to 35°/s during hardware deployment to ensure safety. This velocity cap was also used to compute the trajectory duration for each path segment. 30/38 collision-free paths were obtained, demonstrating GADGET's efficacy of generating diverse success solutions. The remaining 8 trajectories that collided with the obstacles are due to the inaccurate 3D reconstruction and digital twin generation.

The success of this deployment highlights the practical potential of diffusion-based planners for industrial robotic manipulation. Notably, the real-world implementation employs the same model trained entirely in simulation using spherical obstacle environments and the Franka Emika Panda arm, demonstrating strong generalization. To further illustrate the deployment, Figure 4.10 presents representative real-world execution data recorded during experiments. These results confirm that the trajectories generated by GADGET in the digital twin can be reliably executed on hardware[1].

[1]A video demonstrating the simulation results and real-world deployment is available at the following link: https://youtu.be/6ocxqQTG2d0

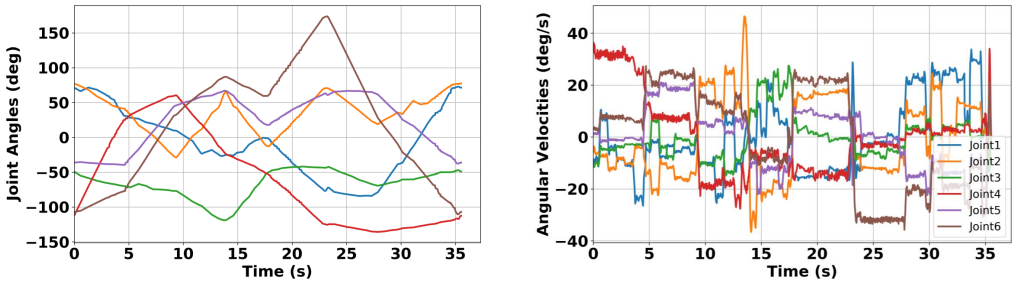

Figure 4.10: Experimental dataset samples collected from the real Kinova Gen3 setup. These data are used to assess GADGET's generalization under practical conditions.

4.5 Conclusion

This chapter introduced **GADGET**, a generalizable and adaptive motion planning framework that leverages conditional diffusion models for generating collision-free paths in unseen environments and across different robot platforms. By integrating voxel-based scene encoding, classifier-free guidance, and CBF-inspired safety mechanisms, GADGET addresses core limitations of prior learning-based planners that struggle with generalization and constraint satisfaction.

Extensive evaluations demonstrated that GADGET:

- Generalizes to novel environments, such as bin-picking and shelf manipulation scenes, without retraining.
- Successfully transfers to different robotic arms with distinct but similar kinematics, including Kinova Gen3 and UR5, thanks to its modular and robot-agnostic design.
- Outperforms both classical and modern state-of-the-art planners (Bi-RRT, BIT*,

MPNet, MPD, DP3) in terms of success rate and safety.

- Benefits significantly from CBF-guided inference, which improves safety and path feasibility.

Overall, GADGET bridges the gap between generative trajectory synthesis and practical robot motion planning by unifying geometric scene understanding, safety-aware inference, and task-driven conditioning into a single, flexible framework. These results position GADGET as a promising solution for safe and adaptive robotic planning in dynamic and unstructured real-world environments.

Chapter 5

Concluding remarks

5.1 Summary of Contributions

Motion planning plays a pivotal role in enabling intelligent, autonomous behavior for robotic manipulators, particularly within industrial environments that demand speed, precision, safety, and adaptability. The contributions of this thesis address long-standing limitations in classical and learning-based motion planning approaches by introducing two novel frameworks: **PPCNet** and **GADGET**. These models not only advance real-time planning performance and safety but also introduce mechanisms for generalization to unseen environments and robotic arms which is a critical requirement for practical, large-scale deployment in modern factories, warehouses, and service robotics.

This dissertation advances the state of the art in robotic motion planning through the following key contributions:

PPCNet: A deep learning-based framework composed of two coordinated neural networks: a waypoint planner and a learned collision checker. PPCNet accelerates path planning while maintaining safety, particularly in repetitive tasks such as indus-

trial bin-picking. Its use of DAgger-based imitation learning and population-based labeling ensures robustness, and its architecture achieves substantial gains in planning time without compromising trajectory quality.

GADGET: A conditional diffusion model–based planner that generates safe, feasible trajectories conditioned on task-specific embeddings (start/goal states and scene representations). GADGET employs CBF–based guidance during inference to explicitly inject safety constraints, thereby bridging generative trajectory sampling and real-time collision avoidance.

Generalization: Both frameworks were evaluated across different robotic platforms and diverse obstacle configurations. GADGET, in particular, demonstrated generalization to new environments and robot arms without retraining which is a crucial step toward scalable and flexible deployment.

Safety-Aware Learning: This work incorporates safety directly into the learning process. PPCNet introduces a fast, learned collision checker; GADGET uses a differentiable CBF-guided loss to enforce obstacle avoidance during trajectory generation.

5.2 Limitations

Despite demonstrating strong generalization capabilities and real-time performance, the GADGET framework presents several limitations that merit discussion.

Computational Overhead. Diffusion models operate through an iterative denoising process, often requiring hundreds of refinement steps to generate a single trajectory. While this allows for sampling from complex trajectory distributions, it introduces latency that can hinder responsiveness in high-dimensional spaces or long-horizon planning tasks. Although GADGET remains suitable for real-time applica-

tions under powerful hardware, its inference time may pose challenges for deployment on resource-constrained embedded systems.

Limited Adaptability to Dynamic Environments. At present, GADGET operates only in static environments. The lack of real-time reactivity constrains its applicability in dynamic or collaborative scenarios, such as those involving human interaction or moving obstacles, where continuous adaptation to changing conditions is essential.

Heuristic Safety Guarantees via CBF Guidance. To improve collision avoidance, GADGET employs CBF guidance into the diffusion process. While effective in steering trajectories away from obstacles, the CBF signals are integrated as soft, heuristic gradients rather than hard constraints. As such, there is no formal guarantee that every sampled trajectory will be collision-free, particularly in cluttered or highly constrained environments where gradient-based guidance may be insufficient.

Absence of Kinodynamic Constraints. The GADGET planner generates joint-space trajectories without using robot dynamics such as velocity, acceleration, or torque limits. The lack of time-parameterization or kinodynamic feasibility checks during planning means that additional post-processing is required to produce executable trajectories. This limitation restricts the applicability of GADGET in scenarios where dynamic constraints play a critical role, such as high-speed manipulation or interaction with fragile objects.

In summary, while GADGET presents a promising step toward generalizable and safe motion planning, addressing the above limitations is essential for broader adoption in real-world industrial and collaborative robotic applications.

5.3 Future Work

While the proposed methods mark a substantial step forward in generalizable motion planning, several avenues remain for future research to further enhance adaptability, safety, and automation:

1. **Online Adaptation for Dynamic Environments:** A critical direction is enabling GADGET to handle dynamic scenes where obstacle configurations may change during planning or execution. This could involve incorporating real-time scene updates into the voxel encoding pipeline and leveraging fast environment encoding techniques. Furthermore, integrating predictive models or recurrent architectures could help anticipate future changes and adapt trajectories on-the-fly. Another promising approach is to combine GADGET with online replanning strategies or streaming diffusion variants that allow for partial trajectory refinement in response to environmental changes without restarting the entire sampling process.

2. **Stronger Safety Guarantees Through Constrained Sampling:** To overcome the heuristic nature of current CBF-guided sampling, future work could explore constrained generative modeling approaches. For instance, projecting samples back into the feasible set using differentiable optimization layers or hard barrier certificates could enforce strict safety. Alternatively, combining diffusion with reachability analysis or control-theoretic verification may provide formal safety guarantees during inference.

3. **Integration of Kinodynamic Constraints:** Enhancing GADGET to generate time-parameterized trajectories that account for velocity, acceleration, and torque limits would extend its applicability to high-speed and force-sensitive

tasks. This can be achieved by incorporating dynamics-aware priors or conditioning the diffusion model on kinodynamic feasibility metrics.

4. **Automated Forward Kinematics via LLMs:** A major bottleneck in adapting motion planners to new robots lies in constructing forward kinematics (FK) modules. Large Language Models (LLMs) can automate this process by parsing URDF files and programmatically extracting the Denavit–Hartenberg (DH) parameters. Such an LLM-based DH extractor could translate URDF joint and link specifications into symbolic FK chains, accelerating deployment on new robotic platforms and reducing the need for manual modeling. Coupling this capability with automated code generation would enable plug-and-play motion planning for arbitrary robot arms.

5. **Hybrid Learning-Classical Planning Systems:** Future systems could combine the strengths of learning-based planners like GADGET with classical algorithms such as RRT*, CHOMP, or TrajOpt. For instance, GADGET's sampled trajectories could serve as informed priors or initial guesses for optimization-based planners, enabling faster convergence while preserving optimality and constraint satisfaction. Similarly, in multi-query settings, classical methods could act as global solvers with GADGET providing local trajectory refinements.

6. **Multi-Robot and Multi-Arm Coordination:** Extending the proposed frameworks to multi-robot or multi-arm systems introduces new research challenges in distributed planning, collision avoidance, and joint task execution. GADGET can be augmented with scene-level awareness that accounts for the presence of other agents, or with shared latent spaces for synchronized trajectory generation. Incorporating shared safety constraints and communication protocols will be essential for safe and scalable deployment in collaborative settings.

7. **Task and Motion Planning Integration:** To enable fully autonomous behavior in complex tasks, future planners should integrate low-level motion generation with high-level task reasoning. This can be achieved by conditioning diffusion-based planners not only on geometric and goal constraints, but also on symbolic task representations which enables planning over sequences of actions with both logical preconditions and continuous feasibility requirements.

5.4 Closing Remarks

This thesis has explored the intersection of deep learning, generative modeling, and safety in robotic motion planning. By developing and evaluating PPCNet and GADGET, it has provided concrete pathways toward planners that are real-time, safe, and adaptable which are properties that are critical for widespread deployment in industrial and service robotic systems.

The research demonstrates that it is now feasible to generate feasible and safe robot trajectories not only faster than traditional methods but also more flexibly across tasks, robots, and environments. With further integration of techniques such as LLM-based automation and hybrid planning the foundations laid in this work can be expanded to power a new generation of intelligent, autonomous robotic systems capable of operating safely and efficiently in the complex, unstructured environments of the real world.

Bibliography

[1] Kuo-Ching Ying, Pourya Pourhejazy, Chen-Yang Cheng, and Zong-Ying Cai. Deep learning-based optimization for motion planning of dual-arm assembly robots. *Computers & Industrial Engineering*, 160:107603, 2021.

[2] Ahmed H Qureshi and Michael C Yip. Deeply informed neural sampling for robot motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6582–6588. IEEE, 2018.

[3] David A Duque, Flavio A Prieto, and Jose G Hoyos. Trajectory generation for robotic assembly operations using learning by demonstration. *Robotics and Computer-Integrated Manufacturing*, 57:292–302, 2019.

[4] Mehran Ghafarian Tamizi, Marjan Yaghoubi, and Homayoun Najjaran. A review of recent trend in motion planning of industrial robots. *International Journal of Intelligent Robotics and Applications*, pages 1–22, 2023.

[5] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, United Kingdom, 2006.

[6] Jan Rosell and Pedro Iniguez. Path planning using harmonic functions and probabilistic cell decomposition. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 1803–1808. IEEE, 2005.

[7] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. *Motion and operation planning of robotic systems*, pages 3–27, 2015.

[8] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles*, pages 396–404. Springer, 1986.

[9] Giacomo Palmieri and Cecilia Scoccia. Motion planning and control of redundant manipulators for dynamical obstacle avoidance. *Machines*, 9(6):121, 2021.

[10] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, and Robin De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, 2016.

[11] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE access*, 2:56–77, 2014.

[12] Rosen Diankov and James Kuffner. Randomized statistical path planning. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6. IEEE, 2007.

[13] Dave Ferguson and Anthony Stentz. Anytime rrts. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5369–5375. IEEE, 2006.

[14] Rogerio R Santos, Domingos A Rade, and Ijar M da Fonseca. A machine learning strategy for optimal path planning of space robotic manipulator in on-orbit servicing. *Acta Astronautica*, 191:41–54, 2022.

[15] Qisong Song, Shaobo Li, Qiang Bai, Jing Yang, Ansi Zhang, Xingxing Zhang, and Longxuan Zhe. Trajectory planning of robot manipulator based on rbf neural network. *Entropy*, 23(9):1207, 2021.

[16] M Aruna Devi, Praveen D Jadhav, Nepal Adhikary, Prajwal S Hebbar, Mohammed Mohsin, and S Krishna Shashank. Trajectory planning & computation of inverse kinematics of scara using machine learning. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)*, pages 170–176. IEEE, 2021.

[17] Mayur J Bency, Ahmed H Qureshi, and Michael C Yip. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3965–3972. IEEE, 2019.

[18] Victor Parque. Learning motion planning functions using a linear transition in the c-space: Networks and kernels. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1538–1543. IEEE, 2021.

[19] Vijay Bhaskar Semwal and Yash Gupta. Performance analysis of data-driven techniques for solving inverse kinematics problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 85–99. Springer, 2021.

[20] Vijay Bhaskar Semwal, Meghana Reddy, and Aditya Narad. Comparative study of inverse kinematics using data driven and fabrik approach. In *Advances in Robotics-5th International Conference of The Robotics Society*, pages 1–6, 2021.

[21] Richard Cheng, Krishna Shankar, and Joel W Burdick. Learning an optimal sampling distribution for efficient motion planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7485–7492. IEEE, 2020.

[22] Yang Li, Rongxin Cui, Zhijun Li, and Demin Xu. Neural network approximation based near-optimal motion planning with kinodynamic constraints using rrt. *IEEE Transactions on Industrial Electronics*, 65(11):8718–8729, 2018.

[23] Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

[24] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, 2019.

[25] Quang-Cuong Pham. Trajectory planning. *Handbook of Manufacturing Engineering and Technology*, pages 1873–1887, 2015.

[26] Eyad Almasri and Mustafa Kemal Uyguroğlu. Trajectory optimization in robotic applications, survey of recent developments. 2021.

[27] Aurelio Piazzi and Antonio Visioli. Global minimum-time trajectory planning of mechanical manipulators using interval analysis. *International journal of Control*, 71(4):631–652, 1998.

[28] Corrado Guarino Lo Bianco and Aurelio Piazzi. Minimum-time trajectory planning of mechanical manipulators under dynamic constraints. *International Journal of Control*, 75(13):967–980, 2002.

[29] Tie Zhang, Meihui Zhang, and Yanbiao Zou. Time-optimal and smooth trajectory planning for robot manipulators. *International Journal of Control, Automation and Systems*, 19(1):521–531, 2021.

[30] Giovanni Carabin and Lorenzo Scalera. On the trajectory planning for energy efficiency in industrial robotic systems. *Robotics*, 9(4):89, 2020.

[31] Glen Field and Yury Stepanenko. Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators. In *Proceedings of IEEE international conference on robotics and automation*, volume 3, pages 2755–2760. IEEE, 1996.

[32] Aurelio Piazzi and Antonio Visioli. Global minimum-jerk trajectory planning of robot manipulators. *IEEE transactions on industrial electronics*, 47(1):140–149, 2000.

[33] Song Lu, Bingxiao Ding, and Yangmin Li. Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation. *Advances in Mechanical Engineering*, 12(3):1687814020913667, 2020.

[34] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[35] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, and Wolfram Burgard. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[36] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[37] Philip M Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics (TOG)*, 15(3):179–210, 1996.

[38] Michal Kleinbort, Oren Salzman, and Dan Halperin. Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning. *arXiv preprint arXiv:1607.04800*, 2016.

[39] Jia Pan and Dinesh Manocha. Efficient configuration space construction and optimization for motion planning. *Engineering*, 1(1):046–057, 2015.

[40] Jinwook Huh and Daniel D Lee. Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–69. IEEE, 2016.

[41] Jia Pan and Dinesh Manocha. Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing. *The International Journal of Robotics Research*, 35(12):1477–1496, 2016.

[42] Nikhil Das and Michael Yip. Learning-based proxy collision detection for robot motion planning applications. *IEEE Transactions on Robotics*, 36(4):1096–1114, 2020.

[43] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[44] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[45] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International journal of robotics research*, 32(9-10):1164–1193, 2013.

[46] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential

convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Berlin, Germany, 2013.

[47] Emile Aarts, Jan Korst, and Wil Michiels. Simulated annealing. In *Search methodologies*, pages 187–210. Springer, 2005.

[48] Marcos de Sales Guerra Tsuzuki, Thiago de Castro Martins, and Fabio Kawaoka Takase. Robot path planning using simulated annealing. *IFAC Proceedings Volumes*, 39(3):175–180, 2006.

[49] Hyeyun Yang and Antoine Vigneron. A simulated annealing approach to coordinated motion planning (cg challenge). In *37th International Symposium on Computational Geometry (SoCG 2021)*, pages 65–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[50] Saša Singer and John Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

[51] Burak Boyacioglu and Seniz Ertugrul. Time-optimal smoothing of rrt-given path for manipulators. In *ICINCO (2)*, pages 406–411, 2016.

[52] Richard Volpe and Pradeep Khosla. Manipulator control with superquadric artificial potential functions: Theory and experiments. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1423–1436, 1990.

[53] M Haghshenas Jaryani. An effective manipulator trajectory planning with obstacles using virtual potential field method. In *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 1573–1578. IEEE, 2007.

[54] X Xu, Y Hu, JM Zhai, LZ Li, and PS Guo. A novel non-collision trajectory planning algorithm based on velocity potential field for robotic manipulator.

International Journal of Advanced Robotic Systems, 15(4):1729881418787075, 2018.

[55] Ming Zhao and Xiaoqing Lv. Improved manipulator obstacle avoidance path planning based on potential field method. *Journal of Robotics*, 2020, 2020.

[56] Tianying Xu, Haibo Zhou, Shuaixia Tan, Zhiqiang Li, Xia Ju, and Yichang Peng. Mechanical arm obstacle avoidance path planning based on improved artificial potential field method. *Industrial Robot: the international journal of robotics research and application 2021*, 2021.

[57] S Liu, Q Zhang, and D Zhou. Obstacle avoidance path planning of space manipulator based on improved artificial potential field method. *Journal of The Institution of Engineers (India): Series C*, 95(1):31–39, 2014.

[58] Ahmed Badawy. Dual-well potential field function for articulated manipulator trajectory planning. *Alexandria Engineering Journal*, 55(2):1235–1241, 2016.

[59] Zhang Long. Virtual target point-based obstacle-avoidance method for manipulator systems in a cluttered environment. *Engineering Optimization*, 52(11):1957–1973, 2020.

[60] Sheng Nan Gai, Rui Sun, Shun Jun Chen, and ShuTing Ji. 6-dof robotic obstacle avoidance path planning based on artificial potential field method. In *2019 16th International Conference on Ubiquitous Robots (UR)*, pages 165–168. IEEE, 2019.

[61] Ning Zhang, Yong Zhang, Chao Ma, and Bin Wang. Path planning of six-dof serial robots based on improved artificial potential field method. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 617–621. IEEE, 2017.

[62] Hao Li, Zhiyang Wang, and Yongsheng Ou. Obstacle avoidance of manipulators based on improved artificial potential field method. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 564–569. IEEE, 2019.

[63] Hsien-I Lin and Ming-Feng Hsieh. Robotic arm path planning based on three-dimensional artificial potential field. In *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, pages 740–745. IEEE, 2018.

[64] EJ Pires and JA Tenreiro Machado. Trajectory optimization for redundant robots using genetic algorithms with heuristic operators. In *Genetic and evolutionary computation conference*, pages 1–9, 2000.

[65] Devendra P Garg and Manish Kumar. Optimization techniques applied to multiple manipulators for path planning and torque minimization. *Engineering applications of artificial intelligence*, 15(3-4):241–252, 2002.

[66] Lianfang Tian and Curtis Collins. An effective robot trajectory planning method using a genetic algorithm. *Mechatronics*, 14(5):455–470, 2004.

[67] Maria da Graça Marcos, JA Tenreiro Machado, and T-P Azevedo-Perdicoúlis. Trajectory planning of redundant manipulators using genetic algorithms. *Communications in nonlinear science and numerical simulation*, 14(7):2858–2869, 2009.

[68] Maria da Graça Marcos, JA Tenreiro Machado, and T-P Azevedo-Perdicoúlis. A multi-objective approach for the motion planning of redundant manipulators. *Applied Soft Computing*, 12(2):589–599, 2012.

[69] Rinku Roy, Manjunatha Mahadevappa, and CS Kumar. Trajectory path planning of eeg controlled robotic arm using ga. *Procedia Computer Science*, 84:147–151, 2016.

[70] Yi Liu, Chen Guo, and Yongpeng Weng. Online time-optimal trajectory planning for robotic manipulators using adaptive elite genetic algorithm with singularity avoidance. *IEEE Access*, 7:146301–146308, 2019.

[71] Riad Menasri, Amir Nakib, Boubaker Daachi, Hamouche Oulhadj, and Patrick Siarry. A trajectory planning of redundant manipulators based on bilevel optimization. *Applied Mathematics and Computation*, 250:934–947, 2015.

[72] Stanislav Števo, Ivan Sekaj, and Martin Dekan. Optimization of robotic arm trajectory using genetic algorithm. *IFAC Proceedings Volumes*, 47(3):1748–1753, 2014.

[73] Tieliang Qiao, Daoguo Yang, Weidong Hao, Jingsen Yan, and Ruiqing Wang. Trajectory planning of manipulator based on improved genetic algorithm. In *Journal of Physics: Conference Series*, volume 1576, page 012035. IOP Publishing, 2020.

[74] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[75] Xiaoman Cao, Hansheng Yan, Zhengyan Huang, Si Ai, Yongjun Xu, Renxuan Fu, and Xiangjun Zou. A multi-objective particle swarm optimization for trajectory planning of fruit picking manipulator. *Agronomy*, 11(11):2286, 2021.

[76] Luchuan Yu, Kaiqiang Wang, Qinhe Zhang, and Jianhua Zhang. Trajectory planning of a redundant planar manipulator based on joint classification and particle swarm optimization algorithm. *Multibody System Dynamics*, 50(1):25–43, 2020.

[77] Javier Alexis Abdor-Sierra, Emmanuel Alejandro Merchán-Cruz, Flavio Arturo Sánchez-Garfias, Ricardo Gustavo Rodríguez-Cañizo, Edgar Alfredo Portilla-Flores, and Valentin Vázquez-Castillo. Particle swarm optimization for inverse kinematics solution and trajectory planning of 7-dof and 8-dof robot manipulators based on unit quaternion representation. *Journal of Applied Engineering Science*, 19(3):592–599, 2021.

[78] Serdar Kucuk. Optimal trajectory generation algorithm for serial and parallel manipulators. *Robotics and Computer-Integrated Manufacturing*, 48:219–232, 2017.

[79] Hongxin Cao, Shouqian Sun, Kejun Zhang, and Zhichuan Tang. Visualized trajectory planning of flexible redundant robotic arm using a novel hybrid algorithm. *Optik*, 127(20):9974–9983, 2016.

[80] Ahmed T Sadiq, Firas A Raheem, and Noor Alhuda F Abbas. Ant colony algorithm improvement for robot arm path planning optimization based on d* strategy. *International Journal of Mechanical and Mechatronics Engineering*, 2021.

[81] O. Hamdoun, L. El Bakkali, and F. Z. Baghli. Optimal trajectory planning of 3rrr parallel robot using ant colony algorithm. In Saïd Zeghloul, Med Amine Laribi, and Jean-Pierre Gazeau, editors, *Robotics and Mechatronics*, pages 131–139, Cham, 2016. Springer International Publishing.

[82] Fatima Zahra Baghli, Larbi El bakkali, and Yassine Lakhal. Optimization of arm manipulator trajectory planning in the presence of obstacles by ant colony algorithm. *Procedia Engineering*, 181:560–567, 2017. 10th International Conference Interdisciplinarity in Engineering, INTER-ENG 2016, 6-7 October 2016, Tirgu Mures, Romania.

[83] Li-sang Liu, Jia-feng Lin, Jin-xin Yao, Dong-wei He, Ji-shi Zheng, Jing Huang, and Peng Shi. Path planning for smart car based on dijkstra algorithm and dynamic window approach. *Wireless Communications and Mobile Computing*, 2021, 2021.

[84] Syed Abdullah Fadzli, Sani Iyal Abdulkadir, Mokhairi Makhtar, and Azrul Amri Jamal. Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries. In *2015 2nd International Conference on Information Science and Security (ICISS)*, pages 1–4. IEEE, 2015.

[85] Akshay Kumar Guruji, Himansh Agarwal, and DK Parsediya. Time-efficient a* algorithm for robot path planning. *Procedia Technology*, 23:144–149, 2016.

[86] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Theta^*: Any-angle path planning on grids. In *AAAI*, volume 7, pages 1177–1183, 2007.

[87] Chee Sheng Tan, Rosmiwati Mohd-Mokhtar, and Mohd Rizal Arshad. A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access*, 2021.

[88] Anthony Stentz. Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*, pages 203–220. Springer, 1997.

[89] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[90] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and automation*, 14(1):166–171, 1998.

[91] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[92] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.

[93] Carlos Rodríguez, Andrés Montaño, and Raúl Suárez. Planning manipulation movements of a dual-arm system considering obstacle removing. *Robotics and Autonomous Systems*, 62(12):1816–1826, 2014.

[94] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[95] Tomasz Rybus and Karol Seweryn. Application of rapidly-exploring random trees (rrt) algorithm for trajectory planning of free-floating space manipulator. In *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 91–96. IEEE, 2015.

[96] Tomasz Rybus. Point-to-point motion planning of a free-floating space manipulator using the rapidly-exploring random trees (rrt) method. *Robotica*, 38(6):957–982, 2020.

[97] Xinda Wang, Xiao Luo, Baoling Han, Yuhan Chen, Guanhao Liang, and Kailin Zheng. Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm. *Applied Sciences*, 10(4):1381, 2020.

[98] Dong-Hyung Kim, Sung-Jin Lim, Duck-Hyun Lee, Ji Yeong Lee, and Chang-Soo Han. A rrt-based motion planning of dual-arm robot for (dis) assembly tasks. In *IEEE ISR 2013*, pages 1–6. IEEE, 2013.

[99] Chengren Yuan, Wenqun Zhang, Guifeng Liu, Xinglong Pan, and Xiaohu Liu. A heuristic rapidly-exploring random trees method for manipulator motion planning. *IEEE Access*, 8:900–910, 2019.

[100] Ameer Tamoor Khan, Shuai Li, Seifedine Kadry, and Yunyoung Nam. Control framework for trajectory planning of soft manipulator using optimized rrt algorithm. *IEEE Access*, 8:171730–171743, 2020.

[101] Xin Gao, Haoxin Wu, Lin Zhai, Hanxu Sun, Qingxuan Jia, Yifan Wang, and Likai Wu. A rapidly exploring random tree optimization algorithm for space robotic manipulators guided by obstacle avoidance independent potential field. *International Journal of Advanced Robotic Systems*, 15(3):1729881418782240, 2018.

[102] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[103] Kris Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2951–2957. IEEE, 2015.

[104] Ahmed Hussain Qureshi and Yasar Ayaz. Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, 40(6):1079–1093, 2016.

[105] Ahmed Hussain Qureshi, Saba Mumtaz, Khawaja Fahad Iqbal, Badar Ali, Yasar Ayaz, Faizan Ahmed, Mannan Saeed Muhammad, Osman Hasan, Whoi Yul Kim, and Moonsoo Ra. Adaptive potential guided directional-rrt. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1887–1892. IEEE, 2013.

[106] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[107] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 3067–3074. IEEE, 2015.

[108] Weiying Ding, Yubin Liu, Hongbo Zhang, Mohd Asif Shah, and Mohammad Asif Ikbal. Research on manipulator motion planning for complex systems based on deep learning. *International Journal of System Assurance Engineering and Management*, pages 1–10, 2021.

[109] Dmitry Berenson, Pieter Abbeel, and Ken Goldberg. A robot path planning framework that learns from experience. In *2012 IEEE International Conference on Robotics and Automation*, pages 3671–3678. IEEE, 2012.

[110] David Coleman, Ioan A Şucan, Mark Moll, Kei Okada, and Nikolaus Correll. Experience-based planning with sparse roadmap spanners. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 900–905. IEEE, 2015.

[111] Peter Lehner and Alin Albu-Schäffer. Repetition sampling for efficiently planning similar constrained manipulation tasks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2851–2856. IEEE, 2017.

[112] Peter Lehner and Alin Albu-Schäffer. The repetition roadmap for repetitive constrained motion planning. *IEEE Robotics and Automation Letters*, 3(4):3884–3891, 2018.

[113] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

[114] Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1):48–66, 2020.

[115] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Sampling-based methods for motion planning with constraints. *Annual review of control, robotics, and autonomous systems*, 1:159–185, 2018.

[116] Ahmed H Qureshi, Jiangeng Dong, Austin Choe, and Michael C Yip. Neural manipulation planning on constraint manifolds. *IEEE Robotics and Automation Letters*, 5(4):6089–6096, 2020.

[117] Ahmed Hussain Qureshi, Jiangeng Dong, Asfiya Baig, and Michael C Yip. Constrained motion planning networks x. *IEEE Transactions on Robotics*, 2021.

[118] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[119] Kun Zhang, Peng Yun, Jun Cen, Junhao Cai, Didi Zhu, Hangjie Yuan, Chao Zhao, Tao Feng, Michael Yu Wang, Qifeng Chen, et al. Generative artificial intelligence in robotic manipulation: A survey. *arXiv preprint arXiv:2503.03464*, 2025.

[120] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.

[121] Chia-Man Hung, Shaohong Zhong, Walter Goodwin, Oiwi Parker Jones, Martin Engelcke, Ioannis Havoutis, and Ingmar Posner. Reaching through latent space: From joint statistics to path planning in manipulation. *IEEE Robotics and Automation Letters*, 7(2):5334–5341, 2022.

[122] Takayuki Osa. Motion planning by learning the solution manifold in trajectory optimization. *The International Journal of Robotics Research*, 41(3):281–311, 2022.

[123] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[124] Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[125] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[126] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy

learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[127] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. In *ICRA 2024 Workshop on 3D Visual Representations for Robot Manipulation*.

[128] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023.

[129] Lars-Peter Ellekilde and Henrik Gordon Petersen. Motion planning efficient trajectories for industrial bin-picking. *The International Journal of Robotics Research*, 32(9-10):991–1004, 2013.

[130] Kimberly N McGuire, Guido CHE de Croon, and Karl Tuyls. A comparative study of bug algorithms for robot navigation. *Robotics and Autonomous Systems*, 121:103261, 2019.

[131] Shirin Chehelgami, Erfan Ashtari, Mohammad Amin Basiri, Mehdi Tale Masouleh, and Ahmad Kalhor. Safe deep learning-based global path planning using a fast collision-free path generator. *Available at SSRN: https://ssrn.com/abstract=4170011r*, 2022.

[132] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[133] Ignazio Aleo, Paolo Arena, and Luca Patané. Sarsa-based reinforcement learning for motion planning in serial manipulators. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2010.

[134] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.

[135] Michael Everett, Yu Fan Chen, and Jonathan P How. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018.

[136] Xi Chen, Ali Ghadirzadeh, John Folkesson, Mårten Björkman, and Patric Jensfelt. Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3110–3116. IEEE, 2018.

[137] Ming Cong, Hang Dong, and Dong Liu. Reinforcement learning and ega-based trajectory planning for dual robots yi liu. *International Journal of Robotics and Automation*, 33(4), 2018.

[138] Mihai Duguleana, Florin Grigore Barbuceanu, Ahmed Teirelbar, and Gheorghe Mogan. Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning. *Robotics and Computer-Integrated Manufacturing*, 28(2):132–146, 2012.

[139] Evan Prianto, MyeongSeop Kim, Jae-Han Park, Ji-Hun Bae, and Jung-Su Kim. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. *Sensors*, 20(20):5911, 2020.

[140] Kapil Katyal, I Wang, Philippe Burlina, et al. Leveraging deep reinforcement learning for reaching robotic tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–19, 2017.

[141] Zhuang Li, Hongbin Ma, Yazhe Ding, Chen Wang, and Ying Jin. Motion planning of six-dof arm robot based on improved ddpg algorithm. In *2020 39th Chinese Control Conference (CCC)*, pages 3954–3959. IEEE, 2020.

[142] Kaveh Kamali, Ilian A Bonev, and Christian Desrosiers. Real-time motion planning for robotic teleoperation using dynamic-goal deep reinforcement learning. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pages 182–189. IEEE, 2020.

[143] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[144] Jiexin Xie, Zhenzhou Shao, Yue Li, Yong Guan, and Jindong Tan. Deep reinforcement learning with optimized reward functions for robotic trajectory planning. *IEEE Access*, 7:105669–105679, 2019.

[145] Gang Peng, Jin Yang, Xinde Lia, and Mohammad Omar Khyam. Deep reinforcement learning with a stage incentive mechanism of dense reward for robotic trajectory planning. *arXiv preprint arXiv:2009.12068*, 2020.

[146] Yinkang Li, Xiaolong Hao, Yuchen She, Shuang Li, and Meng Yu. Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning. *Aerospace Science and Technology*, 109:106446, 2021.

[147] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[148] Debasmita Mukherjee, Kashish Gupta, Li Hsin Chang, and Homayoun Najjaran. A survey of robot learning strategies for human-robot collaboration in industrial settings. *Robotics and Computer-Integrated Manufacturing*, 73:102231, 2022.

[149] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[150] Sylvain Calinon and Aude Billard. Active teaching in robot programming by demonstration. In *RO-MAN 2007-The 16th IEEE International Symposium on Robot and Human Interactive Communication*, pages 702–707. IEEE, 2007.

[151] Sylvain Calinon. A tutorial on task-parameterized movement learning and retrieval. *Intelligent service robotics*, 9(1):1–29, 2016.

[152] Sylvain Calinon. Robot learning with task-parameterized generative models. In *Robotics Research*, pages 111–126. Springer, 2018.

[153] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[154] Jeong-Jung Kim, So-Youn Park, and Ju-Jang Lee. Adaptability improvement of learning from demonstration with sequential quadratic programming for motion planning. In *2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1032–1037. IEEE, 2015.

[155] RB Ashith Shyam, Zhou Hao, Umberto Montanaro, Shilp Dixit, Arunkumar Rathinam, Yang Gao, Gerhard Neumann, and Saber Fallah. Autonomous robots for space: Trajectory learning and adaptation using imitation. *Frontiers in Robotics and AI*, 8, 2021.

[156] Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. Learning constrained distributions of robot configurations with generative adversarial network. *IEEE Robotics and Automation Letters*, 6(2):4233–4240, 2021.

[157] Meishan Guo, Yao Wang, Binyan Liang, Zhihong Chen, Junqin Lin, and Kui Huang. Robot path planning via deep reinforcement learning with improved reward function. In *Proceedings of 2021 Chinese Intelligent Systems Conference*, pages 672–680. Springer, 2022.

[158] Kashish Gupta and Homayoun Najjaran. Exploiting abstract symmetries in reinforcement learning for complex environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3631–3637. IEEE, 2022.

[159] Gian Paolo Incremona, Nikolas Sacchi, Bianca Sangiovanni, and Antonella Ferrara. Experimental assessment of deep reinforcement learning for robot obstacle avoidance: A lpv control perspective. *IFAC-PapersOnLine*, 54(8):89–94, 2021.

[160] Shengjie Wang, Yuxue Cao, Xiang Zheng, and Tao Zhang. An end-to-end trajectory planning strategy for free-floating space robots. In *2021 40th Chinese Control Conference (CCC)*, pages 4236–4241. IEEE, 2021.

[161] Jin Yang and Gang Peng. Ddpg with meta-learning-based experience replay separation for robot trajectory planning. In *2021 7th International Conference on Control, Automation and Robotics (ICCAR)*, pages 46–51. IEEE, 2021.

[162] Tom Jurgenson and Aviv Tamar. Harnessing reinforcement learning for neural motion planning. *arXiv preprint arXiv:1906.00214*, 2019.

[163] Pengzhan Chen, Jiean Pei, Weiqing Lu, and Mingzhen Li. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing*, 497:64–75, 2022.

[164] Claudia Pérez-D'Arpino and Julie A Shah. C-learn: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4058–4065. IEEE, 2017.

[165] Sakmongkon Chumkamon, Umaporn Yokkampon, Eiji Hayashi, and Ryusuke Fujisawa. Robot motion generation by hand demonstration. In *Proceedings of International Conference on Artificial Life & Robotics (ICAROB2021)*, pages 768–771, 2021.

[166] Linjun Li, Yinglong Miao, Ahmed H Qureshi, and Michael C Yip. Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints. *IEEE Robotics and Automation Letters*, 6(3):4496–4503, 2021.

[167] Wanxin Jin, Todd D Murphey, Dana Kulić, Neta Ezer, and Shaoshuai Mou. Learning from sparse demonstrations. *IEEE Transactions on Robotics*, 2022.

[168] Simon Zimmermann, Ghazal Hakimifard, Miguel Zamora, Roi Poranne, and Stelian Coros. A multi-level optimization framework for simultaneous grasping and motion planning. *IEEE Robotics and Automation Letters*, 5(2):2966–2972, 2020.

[169] Gregory Kahn, Peter Sujan, Sachin Patil, Shaunak Bopardikar, Julian Ryde, Ken Goldberg, and Pieter Abbeel. Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4783–4790. IEEE, 2015.

[170] Naresh Marturi, Marek Kopicki, Alireza Rastegarpanah, Vijaykumar Rajasekaran, Maxime Adjigble, Rustam Stolkin, Aleš Leonardis, and Yasemin Bekiroglu. Dynamic grasp and trajectory planning for moving objects. *Autonomous Robots*, 43(5):1241–1256, 2019.

[171] Joan Fontanals, Bao-Anh Dang-Vu, Oliver Porges, Jan Rosell, and Máximo A Roa. Integrated grasp and motion planning using independent contact regions. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 887–893. IEEE, 2014.

[172] Jean Bosco Mbede, Xinhan Huang, and Min Wang. Fuzzy motion planning among dynamic obstacles using artificial potential fields for robot manipulators. *Robotics and autonomous Systems*, 32(1):61–72, 2000.

[173] Mahmoud Tarokh and Xiaomang Zhang. Real-time motion tracking of robot manipulators using adaptive genetic algorithms. *Journal of Intelligent & Robotic Systems*, 74(3):697–708, 2014.

[174] Bianca Sangiovanni, Gian Paolo Incremona, Marco Piastra, and Antonella Ferrara. Self-configuring robot path planning with obstacle avoidance via deep reinforcement learning. *IEEE Control Systems Letters*, 5(2):397–402, 2020.

[175] Dongxu Zhou, Ruiqing Jia, Haifeng Yao, and Mingzuo Xie. Robotic arm motion planning based on residual reinforcement learning. In *2021 13th International Conference on Computer and Automation Engineering (ICCAE)*, pages 89–94. IEEE, 2021.

[176] Evan Prianto, Jae-Han Park, Ji-Hun Bae, and Jung-Su Kim. Deep reinforcement learning-based path planning for multi-arm manipulators with periodically moving obstacles. *Applied Sciences*, 11(6):2587, 2021.

[177] Dirk Buchholz, Daniel Kubus, Ingo Weidauer, Alexander Scholz, and Friedrich M Wahl. Combining visual and inertial features for efficient grasping and bin-picking. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 875–882. IEEE, 2014.

[178] Yukiyasu Domae, Haruhisa Okuda, Yuichi Taguchi, Kazuhiko Sumi, and Takashi Hirai. Fast graspability evaluation on single depth maps for bin picking with general grippers. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1997–2004. IEEE, 2014.

[179] Dirk Buchholz. Bin-picking. *Studies in Systems, Decision and Control*, 44:3–12, 2016.

[180] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[181] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[182] Caelan Reed Garrett. Pybullet planning, 2020.

[183] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[184] J Kuffner and S LaValle RRT-Connect. An efficient approach to single-query path planning ieee international conference on robotics and automation. *San Francisco*, pages 473–479, 2000.

[185] Kinova Kortex API Github repository, 2023.

[186] Paolo Boscariol, Roberto Caracciolo, and Dario Richiedei. Energy optimal design of jerk-continuous trajectories for industrial robots. In *The International Conference of IFToMM ITALY*, pages 318–325. Springer, 2020.

[187] Federico Lozer, Lorenzo Scalera, Paolo Boscariol, and Alessandro Gasparetto. An experimental setup to test time-jerk optimal trajectories for robotic manipulators. In *International Conference on Robotics in Alpe-Adria Danube Region*, pages 309–316. Springer, 2023.

[188] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. Gomp: Grasp-optimized motion planning for bin picking. In *2020 IEEE*

International Conference on Robotics and Automation (ICRA), pages 5270–5277. IEEE, 2020.

[189] Mehran Ghafarian Tamizi, Homayoun Honari, Aleksey Nozdryn-Plotnicki, and Homayoun Najjaran. End-to-end deep learning-based framework for path planning and collision checking: bin-picking application. *Robotica*, 42(4):1094–1112, 2024.

[190] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[191] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[192] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[193] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. Ieee, 2019.

[194] Aaron D Ames, Xu Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[195] Q Nguyen and K Sreenath. Exponential control barrier functions for enforcing high relative-degree safety-critical constraints. In *American Control Conference (ACC)*. IEEE, 2016.

[196] A Agrawal, B Zhao, and AD Ames. Discrete-time control barrier functions with application to bipedal robot locomotion. *Robotics: Science and Systems (RSS)*, 2017.

[197] Michael Danielczuk, Arsalan Mousavian, Clemens Eppner, and Dieter Fox. Object rearrangement using learned implicit collision functions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6010–6017. IEEE, 2021.