

Towards Generalist Neural Motion Planners for Robotic Manipulators: Challenges and Opportunities

Davood Soleymanzadeh¹, Ivan Lopez-Sanchez², Hao Su², Yunzhu Li³, Xiao Liang⁴, and Minghui Zheng¹

Abstract— State-of-the-art generalist manipulation policies have enabled the deployment of robotic manipulators in unstructured human environments. However, these frameworks struggle in cluttered environments primarily because they utilize auxiliary modules for low-level motion planning and control. Motion planning remains challenging due to the high dimensionality of the robot's configuration space and the presence of workspace obstacles. Neural motion planners have enhanced motion planning efficiency by offering fast inference and effectively handling the inherent multi-modality of the motion planning problem. Despite such benefits, current neural motion planners often struggle to generalize to unseen, out-of-distribution planning settings. This paper reviews and analyzes the state-of-the-art neural motion planners, highlighting both their benefits and limitations. It also outlines a path toward establishing generalist neural motion planners capable of handling domain-specific challenges. For a list of the reviewed papers, please refer to <https://davoodsz.github.io/planning-manip-survey.github.io/>.

Note to Practitioners— The paper reviews and summarizes rapidly evolving studies that leverage deep learning for motion planning of robotic manipulators. As robotic manipulators continue to transition from controlled laboratory environments to real-world settings, the demand for efficient, robust, and adaptable motion planning algorithms grows significantly. Thanks to characteristics such as fast inference time and inherent inductive bias, deep learning has been leveraged to facilitate this transition. This paper extensively reviews state-of-the-art deep learning methods used for motion planning of robotic manipulators, and outlines promising avenues and challenges for future research. It specifically evaluates and summarizes the performance of the most commonly used deep learning methods on various key components of motion planning, such as informed sampling, warm-starting trajectory optimization, and collision checking. This paper can serve as a resource for both experts and newcomers in high-DoF robotic motion planning using deep learning.

Index Terms—Robotic Manipulators, Motion Planning, Neural Motion Planners, Generalist Neural Motion Planners, Deep Learning

Received: September, 20, 2025; Revised December, 11, 2025; Accepted January, 24, 2026. This article was recommended for publication by Editor H. Moon upon evaluation of reviewers' comments. This work was partially supported by the US National Science Foundation under Grants 2422826. (Corresponding author: Minghui Zheng.)

¹Davood Soleymanzadeh and Minghui Zheng are with the J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University, College Station, TX, USA (e-mail: davoodso@tamu.edu; mhzheng@tamu.edu).

²Ivan Lopez-Sanchez and Hao Su are with the Lab of Biomechanics and Intelligent Robotics, Department of Biomedical Engineering, Tandon School of Engineering, New York University, NY, USA (e-mail: hao.su@nyu.edu).

³Yunzhu Li is with the Department of Computer Science, Columbia University, NY, USA (e-mail: yunzhu.li@columbia.edu).

⁴Xiao Liang is with the Zachry Department of Civil and Environmental Engineering, Texas A&M University, College Station, TX, USA (e-mail: xliang@tamu.edu).

Digital Object Identifier (DOI): see top of this page.

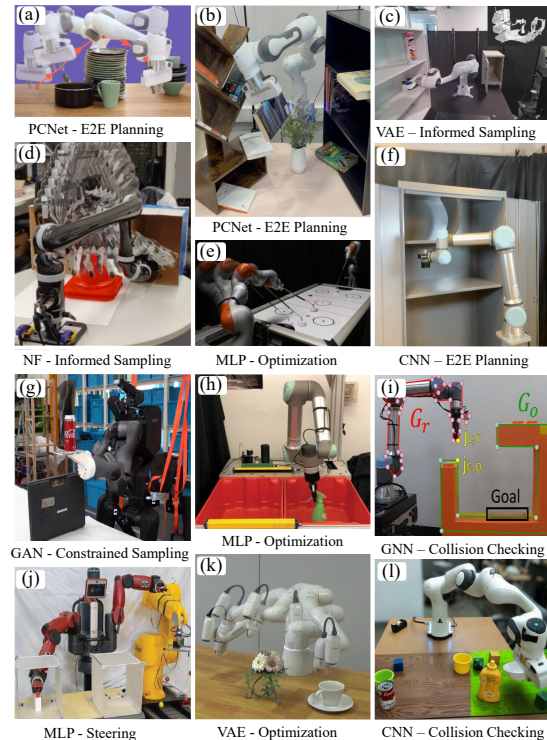


Fig. 1: Deep learning for robotic manipulator motion planning. (a, b) Point cloud networks (PCNets) for End-to-end (E2E) planning [1], [2] (Section V-A). (c) Variational autoencoders (VAEs) [3], and (d) normalizing flows [4] for informed sampling (Section V-C). (e) Multilayer perceptions (MLPs) for trajectory optimization [5] (Section V-D). (f) Convolutional neural networks (CNNs) for E2E planning [6] (Section V-A). (g) Generative adversarial networks (GANs) for constraint manifold learning [7] (Section V-C). (h) MLPs for trajectory optimization [8] (Section V-D). (i) Graph neural networks (GNNs) for collision checking [9] (Section V-E). (j) MLPs for steering [10] (Section V-B2). (k) VAEs for trajectory optimization [11] (Section V-D). (l) CNNs for collision checking [12] (Section V-E).

I. INTRODUCTION

THE emergence of robotics for humans and society [13], has positioned robotic manipulators as fundamental elements across various applications, such as the evolution of medical services [14], and manufacturing settings [15], [16]. A ubiquitous challenge in the deployment of robotic manipulators is planning motions for seamless operation in dynamic, cluttered environments [15], [17], [18].

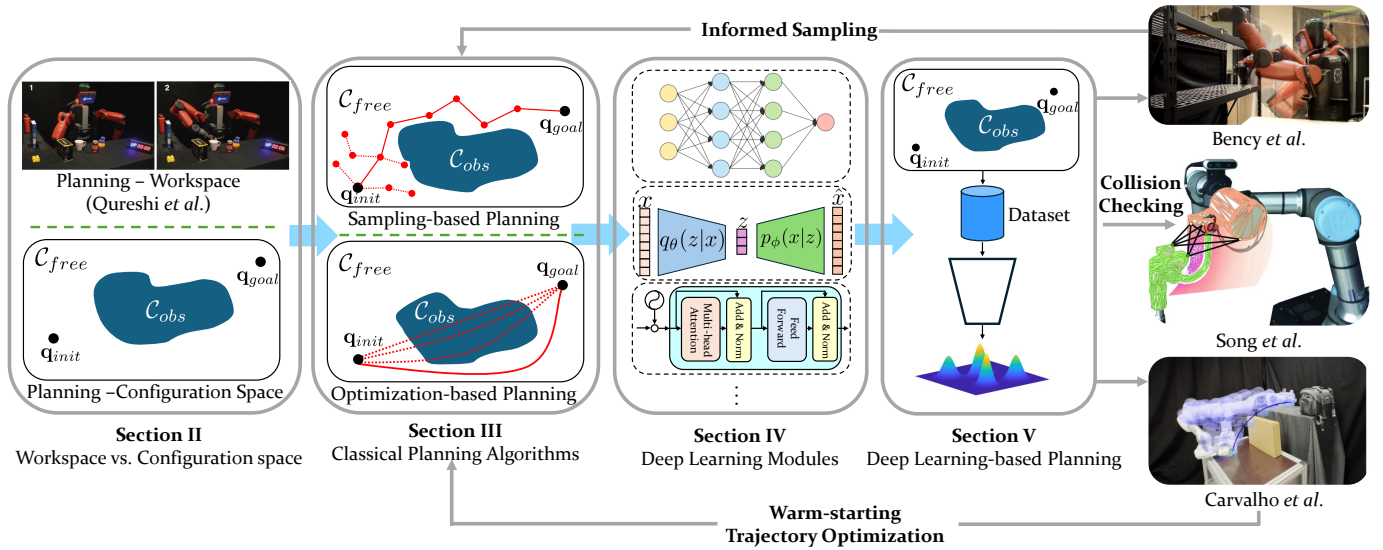


Fig. 2: Overview of this survey paper. Figures are adopted from Carvalho et al. [19], Qureshi et al. [20], Bency et al. [21], and Song et al. [22]

Planning for a robotic system involves finding a feasible collision-free path between a pre-defined start and goal within its configuration space. For robotic manipulators, the process is difficult due to high dimensionality and expensive collision checking. Robotic manipulators that operate in dynamic everyday environments need fast planning algorithms. Classical planning algorithms are often slow, computationally expensive, and require extensive knowledge about the manipulator's environment and its operational capabilities [23], [24]. Deep learning methods have been leveraged to address these challenges by encoding the underlying similarities between planning problems for efficient planning, thanks to their fast inference, and ease of implementation [25].

Deep learning techniques have significantly enhanced various components of the manipulator's autonomy stack [26]. Figure 3 demonstrates various components of the manipulator's autonomy stack. Deep learning methods for vision [27] and language processing [28] have been utilized to convert high-dimensional environmental sensory data (e.g., images, videos, voices, texts) into low-dimensional interpretable embeddings [29]. Furthermore, deep learning methods have been leveraged for task specifications, transforming high-level input instructions to low-level sequences of actions that robotic manipulators can execute while imposing temporal specifications [30]. Additionally, deep learning methods have been efficiently utilized to solve the inverse kinematics problem for redundant robotic manipulators [31].

Deep learning methods have also enhanced the motion planning sub-stack, either by improving specific components of classical motion planners or by functioning as end-to-end planners [18], [24], as illustrated in Figure 1. Despite vast utilization, there are challenges associated with the utilization of deep learning methods for manipulator motion planning. These challenges include:

- **Data scarcity:** The first step in training and utilizing deep learning methods is dataset collection. Although founda-

tion models [33] are pre-trained on internet-scale datasets, they are not suitable for the motion planning sub-stack. In the literature, almost all the deep learning models utilized for planning are trained on limited datasets specifically gathered for that specific problem [24]. Creating internet-scale datasets for low-level planning poses significant challenges, including the need for high-fidelity physics simulators for large-scale data generation.

- **Generalization:** The main drawback of deep learning tools is their excellent performance on in-distribution problems during inference time, while struggling to generalize to out-of-distribution problems. This limitation is particularly pronounced in planning due to the inherent discontinuities in the planning problem [34]. Small changes in the planning problem (workspace) can lead to significant changes in configuration space that were not present in the training dataset [35]. Addressing these challenges is crucial for the efficient utilization of deep learning methods in motion planning.
- **Real-time applications:** Although more complex deep learning methods and deeper networks may excel at encoding complex similarities within planning problems, the relatively slow inference time of these models limits their deployment in real-world dynamic environments [36]. Further research is required to reduce the inference time of these planners for real-time deployment.
- **Safety guarantees:** It is challenging to analytically ensure the safety and stability of deep learning models. Additional considerations and constraints should be included to guarantee the required safety criteria [37]. Rigorously checking for the safety and stability of these methods is challenging and requires further exploration.

In this survey paper, we explored the state-of-the-art literature on the utilization of deep learning methods in robotic manipulator motion planning. We identified current challenges and provided future perspectives and research directions to

TABLE I: Overview of existing survey papers on robotic motion planning, their scope, focus, and limitations compared to this paper.

Reference	Robot Types	Scope and Focus	Notes
Wang <i>et al.</i> , 2021 [24]	Not specific to manipulators	Machine learning for improving/replacing classical planning algorithms	Up to 2021
McMahon <i>et al.</i> , 2022 [18]	Not specific to manipulators	Machine learning for improving classical sampling-based planning algorithms	Not covering other planning algorithms
Noroozi <i>et al.</i> , 2023 [23]	Not specific to manipulators	All types of planning algorithms (including both classical and learning-based)	Not specific to deep learning
Tamizi <i>et al.</i> , 2023 [15]	Robotic manipulators	All types of planning algorithms (classical, learning-based)	Not specific to deep learning
Carvalho <i>et al.</i> , 2025 [32]	Not specific to manipulators	Data-driven planning algorithms	Not specific to robotic manipulators
Ours	Robotic manipulators	Focused on deep learning for improving/replacing classical planning algorithms	Since 2018

Note: The main criteria for selecting recent survey papers were twofold. First, the survey needed to contain planning for robotic manipulators. Second, it had to consider recent papers that utilize deep learning methods for planning.

address these challenges accordingly. Table I lists related survey papers on robotic motion planning. In comparison with these papers, our focus is to provide the promises and limitations of using deep learning methods to enhance classical motion planning algorithms for robotic manipulators.

Our criteria for paper selection are as follows. (1) We focused on the state-of-the-art literatures that utilize *deep learning* for *robotic manipulator planning* since 2018; (2) We focused solely on papers that apply *deep learning* to enhance *global planning algorithms*. (3) We *excluded* papers that utilize deep learning for task planning within task and motion planning (TAMP) scenarios and papers that utilize deep learning for low-level motion control of robotic manipulators. (4) We also surveyed *classical planning algorithms*-related research papers to identify their components that can be enhanced by deep learning methods.

The key contributions of this survey paper are as follows:

- **Robotic manipulator motion planning:** We review the state-of-the-art literature that has utilized deep learning for robotic manipulator planning. Robotic manipulators are increasingly deployed within critical applications (e.g., healthcare, re-manufacturing, and agriculture), which necessitate safe and efficient motion planning algorithms. However, motion planning for robotic manipulators remains challenging due to their high DOF and the complexity of real-world environments.
- **Systematic mapping from deep learning frameworks to motion planning algorithmic primitives:** We provide a systematic mapping from various deep learning architectures (e.g., convolutional neural networks, deep generative models, large language models) to core algorithmic primitives of classical motion planning algorithms (e.g., sampling and steering primitives of sampling-based planning algorithms).
- **Road to generalist neural motion planners:** We outline a path toward generalist neural motion planners capable of end-to-end planning for robotic manipulators. We summarize the progress that has been made in this

direction, identify how far the research community has advanced, and highlight key considerations necessary to achieve this goal. Particularly, we emphasize the need for standardized benchmarks, large-scale planning datasets, explicit handling of safety constraints, generalization to out-of-distribution scenarios, and robustness to planning uncertainties for reliable deployment within unstructured real-world environments. Additionally, we discuss how large-scale foundation models can be established and leveraged to facilitate traversing this path.

Given the recent advances in computational power and the emergence of new deep-learning methods, we believe it is essential to revisit the recent applications of these methods in motion planning for robotic manipulators. This paper can provide a unified, comprehensive overview of the challenges and promises of utilizing deep learning methods for robotic manipulator motion planning.

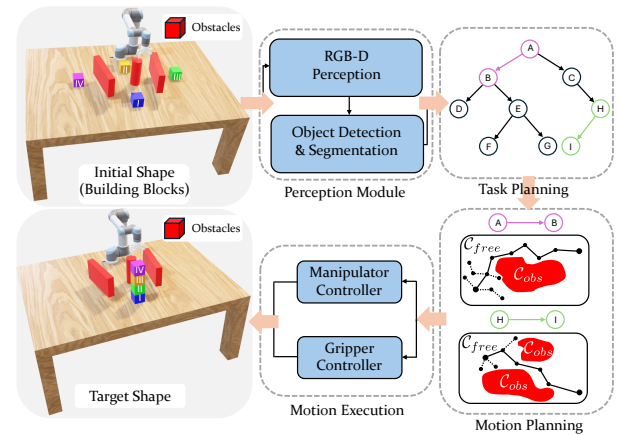


Fig. 3: An example of a robotic manipulator's autonomy stack.

This survey paper is organized as follows. Section II defines the planning preliminaries for robotic manipulators, explaining foundational concepts essential to the planning problem. Section III reviews classical planning algorithms widely used

TABLE II: Robotic manipulator planning terminologies, and their definitions used in this manuscript.

Term	Definition
Configuration space	An N -dimensional space spanned by robotic manipulator joint values, where N is the number of DOF.
Workspace	The 3-dimensional space in which the robotic manipulator's end-effector operates.
Planning constraints	Include geometric constraints; kinematic constraints; and dynamic constraints.
Global planning	Planning algorithms that first plan a path from a start to a goal configuration, and then execute it.

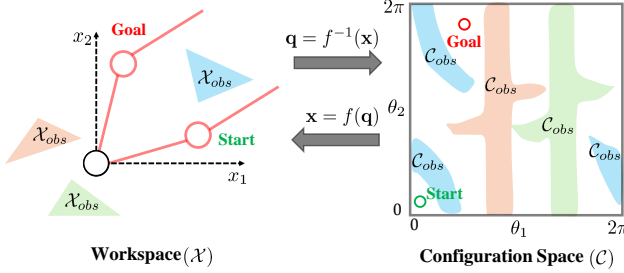


Fig. 4: An example of a 2-DOF planar manipulator: workspace vs. configuration space [39].

for robotic manipulators, summarizing their strengths and limitations. Section IV introduces various deep learning methods and their potential application for robotic manipulator motion planning. Section V reviews the state-of-the-art literature on neural motion planners for robotic manipulators. Section VI outlines the current challenges and presents future perspectives associated with the application of neural motion planners. Section VII provides domain-specific challenges and necessary considerations for effective deployment of neural motion planners. Finally, section VIII summarizes our findings and conclusions based on the reviewed works. Figure 2 illustrates the structure of the survey paper.

II. ROBOTIC MANIPULATORS PLANNING PRELIMINARIES

In this section, we introduce commonly used terminologies and planning problems for robotic manipulators. Table II provides robotic manipulators' planning terminologies and their definitions used in this paper, and Table III summarizes the mathematical notations used throughout this section.

TABLE III: Mathematical notations and definitions for motion planning of robotic manipulators utilized throughout this paper.

Symbol	Definition	Symbol	Definition
C	Configuration space	C_{free}	Free configuration space
C_{obs}	Obstacle configuration space	\mathcal{X}	Workspace
\mathcal{X}_{free}	Free workspace	\mathcal{X}_{obs}	Obstacle workspace
\mathbf{q}	Configuration	\mathbf{x}	Workspace pose
$f(\cdot)$	Forward kinematics	\mathbf{q}_{init}	Initial configuration
\mathbf{q}_{goal}	Goal configuration	σ	Path
$c(\cdot)$	Planning cost	τ	Trajectory

A. Terminologies

Common terms and terminologies related to planning for robotic manipulators [38] are as follows. Figure 4 illustrates these terms for a 2-DOF robotic manipulator.

Configuration Space: The *configuration* of a robotic manipulator is defined by its joint values, represented as $\mathbf{q} = (q_1, \dots, q_N)$ where N is the number of DOF of the manipulator [40]. The *configuration space* (C) is an N -dimensional space spanned by these joint values, which describe the manipulator's configuration. Using configuration space simplifies the planning problem, as the manipulator is represented by a point (instead of bodies and volumes), and obstacles are represented as forbidden regions (C_{obs}) in this space. The planning problem is then reduced to moving the point representing the manipulator's configuration from the initial configuration to the goal configuration while avoiding the forbidden regions (i.e., staying within the free space $C_{free} = C \setminus C_{obs}$). However, computing these forbidden regions is nontrivial: inverse kinematics for high-DOF (especially redundant) manipulators generally lacks closed-form solutions except for special kinematic structures, and the exact boundary separating C_{obs} from C_{free} is typically high-dimensional and nonconvex, with no simple closed-form representation [41].

Workspace: The workspace \mathcal{X} is the space in which a manipulator's end-effector operates. In this space, the end-effector position is specified by Cartesian coordinates in \mathbb{R}^3 , while its orientation is represented by the rotation group $SO(3)$ ($\mathcal{X} \in \mathbb{R}^3 \times SO(3)$).

Forward & Inverse Kinematics: These operations enable mapping between the joint configuration and the Cartesian pose of all geometries attached to the manipulator. *Forward kinematics* (f) maps a configuration (\mathbf{q}) from the configuration space to a workspace pose (\mathbf{x}), i.e., $\mathbf{x} = f(\mathbf{q})$. Conversely, *inverse kinematics* maps a pose (\mathbf{x}) from the workspace (\mathcal{X}) back to a configuration (\mathbf{q}) in the configuration space (C). i.e., $\mathbf{q} = f^{-1}(\mathbf{x})$.

B. Planning Definitions

Various approaches that have been utilized to solve the planning problem are as follows. Refer to Table IV for an overview of the characteristics of these approaches. This Table highlights key characteristics and differences among various planning methods for robotic manipulators.

Path Planning: The path planning problem can be defined as $M = \{\mathbf{q}_{init}, \mathbf{q}_{goal}, C_{free}\}$ where \mathbf{q}_{init} , and \mathbf{q}_{goal} represent initial and goal configurations, respectively. The objective of path planning is to find a feasible path connecting the start and the goal configurations that lies entirely within the free space. The primary goal is to find a feasible path

TABLE IV: Three main approaches to solve the planning problem for robotic manipulators, their distinguishing characteristics.

	Geometric constraints	Kinematics constraints	Dynamic constraints	Velocity evolution
Path planning	✓	✗	✗	✗
Motion planning	✓	✓	✓	✗
Trajectory planning	✓	✓	✓	✓

Note: In this manuscript, we use the term motion planning to broadly describe low-level planning within the robotic manipulation stack. Low-level planning can be further categorized into *path planning*, *motion planning*, and *trajectory planning*, each with distinct meanings, although these terms are often used interchangeably in the literature.

$\sigma = [\mathbf{q}_1, \dots, \mathbf{q}_t, \dots, \mathbf{q}_T]$ such that:

$$\begin{aligned} \sigma(0) &= \mathbf{q}_{\text{init}}, \\ \sigma(t) &\in \mathcal{C}_{\text{free}}, \\ \sigma(T) &= \mathbf{q}_{\text{goal}}, \end{aligned} \quad (1)$$

where the primary constraints for path planning are *geometric constraints*. Therefore, path planning provides a geometric description of the robotic manipulator's movement by generating collision-free waypoints between the start and goal configurations.

Motion Planning: Path planning, as defined in Eq. 1, focuses solely on geometric constraints. However, for effective planning, other types of constraints must also be considered, including *kinematic constraints*, *dynamic constraints*, and the robotic manipulator's evolution over time [38]. The motion planning problem incorporates all these constraints.

To address the motion planning problem, a set of costs c_i are considered. These costs include all constraints - geometric, kinematic, and dynamic - as soft constraints (i.e., costs to be optimized). The motion planning problem can be defined as follows:

$$\sigma^* = \arg \min_{\sigma} \sum_i \lambda_i c_i(\sigma), \quad (2)$$

where $\lambda_i > 0$ is a hyperparameter weight assigned to each cost function $c_i(\sigma)$, and σ and σ^* are the planned and optimal motion, respectively. Therefore, motion planning generates a collision-free path between the start and goal configuration while satisfying kinematics and dynamics constraints.

Trajectory Planning: In the literature, the motion planning problem and the trajectory planning problem are often treated as synonyms. However, from a technical perspective, trajectory planning differs in that it also considers the time evolution of velocity as part of the trajectory [38]. Let $\mathbf{s} = [\mathbf{q}^T, \dot{\mathbf{q}}^T]^T \in \mathbb{R}^{2d}$ represent the state of the robotic manipulator, where \mathbf{q} is the robotic manipulator configuration, and $\dot{\mathbf{q}}$ is the velocity of robotic manipulator. A Trajectory $\tau = [\mathbf{s}_1, \dots, \mathbf{s}_t, \dots, \mathbf{s}_T]$ is defined as a sequence of states over a time horizon T . The trajectory planning problem can be defined as follows [42]:

$$\tau^* = \arg \min_{\tau} \sum_i \lambda_i c_i(\tau). \quad (3)$$

The main difference between the motion planning problem (Eq. 2) and the trajectory planning problem (Eq. 3) lies in the inclusion of the robotic manipulator velocity evolution into the problem formulation. Therefore, trajectory planning generates a collision-free path between the start and goal configuration while satisfying kinematics and dynamics constraints, and includes the evolution of robotic manipulator velocity.

C. Metrics

The common metrics to evaluate the performance of planning algorithms for robotic manipulators are: *planning time*, *planning cost*, and *success rate*.

- **Planning Time:** Planning time (T) is the average planning time the planner takes to find a solution.
- **Planning Cost:** Planning cost (C) refers to the average length of the planned paths in the configuration space or the workspace.
- **Success Rate:** Success rate (S) represents the percentage of successfully planned paths.

III. CLASSICAL PLANNING ALGORITHMS

In this section, we introduce classical motion planning algorithms for robotic manipulators. Over the years, a diverse array of algorithms has been developed to address the planning problem. These algorithms are broadly categorized into two groups: sampling-based, and optimization-based algorithms.

A. Sampling-based Planning Algorithms

Sampling-based planning algorithms utilize random sampling to create a tree or roadmap within the configuration space [43], and are broadly categorized into unconstrained and constrained algorithms.

1) **Unconstrained Planning:** Sampling-based motion planners can be divided into two main categories: multi-query probabilistic roadmaps (graph-based planners) [44]–[48], [68], and single-query rapidly-exploring random trees (tree-based planners) [49]–[61]. Table V summarizes representative planners from each category with their primitives and limitations.

Probabilistic Roadmaps (PRMs): PRMs [44] operates in two phases: graph construction and path-finding. During the graph construction phase, the algorithm randomly generates samples in the robotic manipulator's configuration space and adds them as a new node to the graph. Then the algorithm tries to connect the new node to the existing nodes of the graph using a specific distance metric and collision avoidance. This process will create a graph in the free configuration space ($\mathcal{C}_{\text{free}}$). In the path-finding phase, the algorithm inserts the start and goal configurations of the planning problem into the constructed graph and uses graph search algorithms to find a path connecting these two nodes.

Tree-based Planners: In the context of probabilistic roadmaps, “multi-query” means that the graph constructed during the first phase can be used for various start and goal configurations to solve different path planning problems. This also can be achieved by simply solving a series of “single-query” problems for different start and goal configurations

TABLE V: The two main categories of sampling-based planning algorithms for motion planning of robotic manipulators, their advanced variants, algorithmic primitives, and limitations.

Category	Basic Algorithm	Variants	Algorithmic Primitives	Limitations
Graph-based (multi-query)	PRM [44]	LazyPRM [45], PRM* [46], LazyPRM* [46], SPARS [47], Improved SPARS [48].	<ul style="list-style-type: none"> Graph Construction Path Finding 	<ul style="list-style-type: none"> Computationally complex graph construction.
Tree-based (single-query)	RRT [49]	Connect-RRT [50], RRT* [46], LBT-RRT [51], Stable-sparse RRT [52], Transition-based RRT [53], Lazy-RRG* [54], Quotient-space RRT (QRRT) [55], Quotient-space RRT* (QRRT*) [56], RRT# [57], RRTX [58], P-RRT* [59], [60], Informed-RRT* [61], FMT* [62], BFMT* [63], Hierarchical FMT* [64], BIT* [65], ABIT* [66], AIT* [67].	<ul style="list-style-type: none"> Sampling Local Planning Collision Checking 	<ul style="list-style-type: none"> Struggle to scale up to high-dimensional configuration spaces. Non-smooth paths Low convergence rate Hardly applicable to dynamic environments Inefficient sampling primitive

[46]. In “single-query” planners, a tree is incrementally built from the start configuration to the goal configuration, which reduces the planning time compared to “multi-query” planners. This results in a widespread application of single-query planners compared to probabilistic roadmaps.

Tree-based planners, such as Rapidly-exploring Random Tree (RRT) algorithm [49], have three algorithmic primitives: First, random sampling within the configuration space (*Sampling*). Second, steering from existing sampled configurations to the new sample (*Local Planning*). Third, collision checking the steering connections (*Collision Checking*). Figure 5 illustrates the algorithmic primitives of sampling-based planning algorithms. These algorithmic steps enable an implicit representation of the configuration space, making tree-based planners an effective choice for path-planning problems within the high-dimensional configuration space of robotic manipulators.

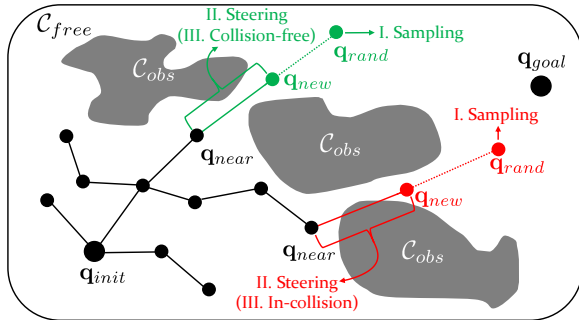


Fig. 5: Algorithmic primitives (I. Sampling, II. Steering, and III. Collision Checking) of sampling-based planning algorithms.

Sampling-based Planning Algorithms Variants and Extensions: Given that basic sampling-based planning algorithms are effective in finding the feasible path within high-dimensional configuration spaces due to its probabilistic completeness, several variants and extensions of the sampling-based planning algorithms have been developed to further enhance their performance. Some of the variants and extensions are as follows:

- **Informed sampling:** Sampling functions in sampling-based planners can be either random sampling or informed sampling. Random sampling functions indiscrim-

inately select points from the entire configuration space to construct the tree. In contrast, informed sampling functions adaptively sample the configuration space. Informed samplers commonly utilize hand-crafted heuristics to direct the sampling process towards configuration space regions with high success rates and low planning costs [61].

- **Lazy collision checking:** This technique performs collision checking in the path construction phase, rather than during the initial graph or tree construction stage of the path planning algorithm. By postponing collision checking, the algorithm can focus on rapidly generating a graph or tree structure without being overwhelmed by the computational overhead of immediate collision evaluations [45], [54].
- **Bi-directionality:** In this module, two sampling-based planners are developed simultaneously: one starts from the start node, and the other from the goal node. This module enhances the efficiency of the path planning algorithm by concurrently expanding these planners towards each other [50].
- **Optimality:** This module makes it possible for the sampling-based planner to find an asymptotically optimal path by utilizing tree rewiring operations after each sampling iteration. The rewiring process continually looks for ways to reduce the overall path length, which eventually leads to an asymptotically optimal solution [46].

Advanced Variants: There are several families of sampling-based planning algorithms that extend beyond the basic framework of probabilistic roadmaps and tree-based planners. The first family is Fast Marching Tree (FMT*) algorithms [62]–[64]. These algorithms combine features of both PRM and RRT algorithms by using a lazy dynamic programming approach on samples to grow a tree of paths. The FMT* algorithm leverages the strength of both PRM and RRT by incorporating elements of PRM’s multi-query and RRT’s single-query approach. Another family is Batch Informed Trees (BIT*) algorithms [65]–[67], [69]. The BIT* algorithm combines search-based planners (e.g., A* [70]), and sampling-based planners (e.g., RRT* [46]) to approximate and search the configuration space. The BIT* algorithm efficiently explores the configuration space by prioritizing regions with a higher probability of containing an optimal path.

Summary: Sampling-based planning algorithms, have three algorithmic primitives, *sampling*, *steering*, and *collision checking*.

- **Sampling:** There are two types of sampling functions in sampling-based planners: uniform sampling and informed sampling. Uniform sampling indiscriminately samples from the configuration space, leading to computational complexity. Informed sampling functions sample within promising areas of the configuration space to reduce computational complexity. Classical planners commonly use hand-crafted heuristics for this purpose, which is challenging to implement for robotic manipulators. In Section V-B1 we explore how various deep learning frameworks can improve this part of sampling-based planning algorithms for robotic manipulators.
- **Steering:** Classical sampling-based algorithms steer toward newly sampled configurations using straight-line paths, which can violate motion constraints of robotic manipulators. Recently, deep learning frameworks have been employed to learn efficient steering functions to improve sampling-based planning algorithms (Section V-B2).
- **Collision Checking:** Classical sampling-based algorithms perform fine-grained geometric collision queries to validate each edge of the constructed tree, which is computationally intensive. Recently, deep learning frameworks have been utilized as proxy collision checkers to enable efficient, continuous collision checking (Section V-E).

2) *Constrained Planning:* Geometric task constraints, such as maintaining a certain pose for a robotic manipulator end-effector, are types of constraints that require modifications to the sampling-based planning algorithms [71]. These constraints are prevalent in real-world applications of robotic manipulators, which require modified sampling-based planning algorithms.

Definition: In constrained planning, the planner must not only avoid collisions but also satisfy hard geometric task constraints. Thus, the constrained planning problem involves finding a path between any specified initial and goal configuration that lies entirely within the union of free configuration space and constraint manifold.

There are several methods to incorporate constraint adherence within sampling-based planning algorithms. The first approach is *projection*, where a projection operator is utilized to project a given configuration onto the constraint manifold [72], [73]. Another method involves *tangent spaces*, which calculate the tangent space of the constraint manifold, allowing for generation of nearby samples [74]. These tangent spaces can be stored and integrated into an *atlas*, providing a linear piecewise approximation of the constraint manifold [75]. For more details on constrained sampling-based planning, refer to the survey paper by Kingston *et al.* [76].

Summary: Classical constrained sampling-based planning algorithms utilize *projection* and *continuation* operators to enforce kinematic constraints. Recently, deep learning frame-

works have been employed to learn the constraint manifold for direct constraint-aware sampling (Section V-C).

3) *Limitations of Sampling-based Planning Algorithms:*

Although sampling-based motion planners are widely used for robotic manipulators' path planning, they come with several limitations and drawbacks, including:

- **Curse of dimensionality:** The number of samples for path planning may increase exponentially with the dimension of the configuration space. Sometimes the sampling-based planners need to cover the configuration space with discrete samples.
- **Path smoothness:** The output of sampling-based planners often contains many unnecessary nodes which result in jerky motions, and takes longer to execute. The probabilistic nature of sampling-based planners results in paths with unnecessary motions and discontinuities, which necessitate post-processing steps to improve the quality of the generated path. For more details, refer to Section III-B.
- **Low convergence rate:** Sampling-based planning algorithms demonstrate low convergence rates in the high-dimensional environment, due to random sampling.
- **Hardly applicable to dynamic environments:** Planning in dynamic environments necessitates regular re-planning for collision avoidance [77]. The sampling-based motion planning approaches incur a significant computational cost, rendering them unsuitable for real-time computation and reactive motion planning.
- **Sample inefficiency:** Due to the sampling primitives of these algorithms, these planners often struggle to provide enough samples in important areas within the configuration space (e.g., near obstacles and in narrow spaces). This reduces their capability to model the configuration space [78] explicitly. In highly complex workspaces and narrow passages, these algorithms may reach a singularity such that the constructed graph becomes disconnected and planning fails.

B. *Optimization-based Planning Algorithms*

Sampling-based planners are effective and fast for path planning in high-dimensional configuration spaces. However, these planners can not guarantee the smoothness and local optimality of the planned path. Optimization-based planners address these drawbacks by post-processing the planned path to eliminate redundant or jerky motions [79]. There are two primary methods for this post-processing: gradient-free and gradient-based optimization methods [17]. Table VI summarizes trajectory optimization planners with their characteristics and limitations.

Gradient-free Optimization Methods: A family of gradient-free methods aims to shorten the path by removing redundant nodes. These methods typically employ (partial) short-cutting [44], [80], [81], which involves tree pruning, or hybridization [82]. Another family focuses on smoothing the path by utilizing smooth curves to interpolate between the path's waypoints. Techniques such as B-splines [83], Bezier curves [84], and cubic polynomials [85] are commonly utilized for this purpose.

TABLE VI: The two main categories of trajectory optimization planning algorithms for robotic manipulators, their constituent algorithms, characteristics, and limitations.

Category	Algorithms	Characteristics	Limitations
Gradient-free	Short-cutting [44], [80], [81], Path Smoothing by Utilizing Smooth Curves (i.e., B-splines [83], Bezier Curves [84], Cubic Polynomials [85])	<ul style="list-style-type: none"> • Post-processing the output of sampling-based planning algorithms (Removing redundant nodes, smoothing) 	<ul style="list-style-type: none"> • Computationally complex
Gradient-based	Elastic Band [86], Elastic Strips [87], CHOMP [79], [88], TrajOpt [89], [90], GPMP [91], GPMP2 [92], STOMP [93], GCS [94]–[97]	<ul style="list-style-type: none"> • Refining an initial path subject to kinematic and task-specific constraints. 	<ul style="list-style-type: none"> • Getting stuck in local minima. • Computationally complex. • Hand-crafted cost functions. • Requires collision gradient. • Dependent on the initial guess

Gradient-based Optimization Methods: Gradient-based optimization methods leverage optimization techniques from the field of optimal control to iteratively refine the initial path subject to planning constraints. These methods formulate planning as an optimization problem subject to kinematic and task-specific constraints [17] (cf. motion planning definition in Section II).

One approach to optimizing the initial path involves utilizing *elastic band* [86] or *elastic strips* [87], [98], [99] methods to locally deform the initial path for smoothness and collision avoidance. These methods model the initial path as an elastic band subjected to two types of artificial forces: an internal force that maintains the connectivity of the path, and an external repulsion force that steers the path away from obstacles.

Another gradient-based approach is global optimization, which also refines the initial trajectory utilizing a numerical optimization method, subject to planning and task-specific constraints [79], [88]–[92], [96], [100]–[114]. An early method proposed for this approach involves subjecting the initial trajectory to repulsive and attractive artificial potential fields [115] to improve its quality [100].

The Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [79], [88] utilizes covariant gradient descent [116] as the numerical optimization technique to optimize the initial, in-collision trajectory. CHOMP discretizes the initial trajectory into waypoints and subjects them to trajectory smoothness and obstacle avoidance, modeled as pre-computed signed distance fields [117]. In cluttered environments, a fine-grained trajectory discretization is needed for optimization, which increases computational complexity [110].

Trajectory Optimization (TrajOpt) employs continuous-time collision checking [90] to ensure continuous-time safety, thereby reducing the need for a large number of trajectory states [89], [90]. TrajOpt utilizes sequential convex optimization [118] for global optimization, which addresses the non-convexity problem by solving a series of convex optimization problems. Like CHOMP, TrajOpt also requires a densely parameterized trajectory in clutter environments.

The main drawback of CHOMP and TrajOpt is the need for a finely discretized trajectory for optimization. Continuous-trajectory representation, like radial basis functions (FBS) [104], and Gaussian process (GP) [91], has helped alleviate this problem. Gaussian Process Motion Planning (GPMP) [91] addresses this issue by parameterizing the initial trajectory via

a few states and utilizing a continuous Gaussian process for interpolation to query the initial trajectory.

GPMP2 [92], [119] improves upon GPMP by considering the trajectory optimization problem as probabilistic inference [120], [121] on a factor graph [122]. It leverages Incremental Smoothing and Mapping (iSAM2) [123] optimization algorithm for trajectory optimization.

The main drawback of the above-mentioned gradient-based optimization methods is that both cost and constraint functions need to be differentiable. Stochastic trajectory optimization methods relax the requirement for differentiable constraints by utilizing sampling methods [93], [105], [124]–[131]. The Stochastic Trajectory Optimization for Motion Planning (STOMP) [93], [124] improves upon the aforementioned trajectory optimization methods by employing a derivative-free stochastic optimization approach. This algorithm starts by generating a set of random trajectories around the initial candidate solution. Then, these trajectories are evaluated using planning cost functions to update the candidate solution [132].

Another drawback of optimization-based planning algorithms comes from the non-convexity of the planning problem. Graphs of Convex Sets (GCS) [94]–[97] address this limitation by modeling non-convex obstacle-avoidance constraints as a collection of safe convex regions [133]–[135]. This approach simplifies the planning process into two steps: selecting which convex region to traverse (the discrete component), and optimizing the parameterized robot trajectory within each region (the continuous component). This formulation reduces the planning problem to a Shortest-Path Problem (SPP) [136] over the GCS.

Although optimization-based trajectory planners have been widely used in the research community for trajectory planning, there are several limitations and drawbacks associated with them, including:

- **Local minima - lack of formal completeness guarantees:** Optimization-based planners can easily get stuck in local minima due to the inherent non-convexity of the optimization problem.
- **Collision gradients:** The collision detection module in the optimization-based motion planning algorithms is required to provide collision gradients. The gradient helps the optimal trajectory steer away from the obstacles. The efficiency of optimization-based algorithms depends on the smoothness and continuity of the collision gradients. However, existing classical collision detection methods

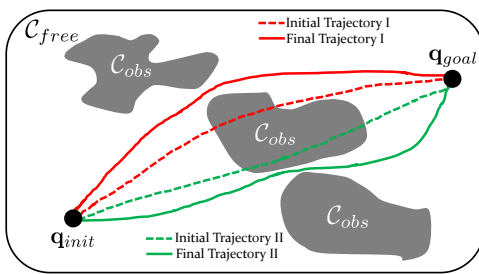


Fig. 6: An abstract illustration demonstrating how global trajectory optimization techniques rely on the initial trajectory to warm-start the optimization process. This highlights that even slightly different initializations can lead to distinct final trajectories.

only provide numerical [90] or stochastic gradients [93].

- **Optimization warm-starting:** As demonstrated in Figure 6, the output of optimization-based trajectory planning methods is dependent on the initial trajectory. Therefore, a feasible initialization is needed to avoid getting stuck in local minima.
- **Long-horizon, global optimization:** Trajectory optimization methods solve the optimization problem over a long temporal horizon, which is challenging for dynamic planning and real-time obstacle avoidance.
- **Requiring hand-crafted cost functions:** Trajectory optimization methods often require hand-crafted cost functions to encode task-specific constraints, and desired behaviors [137]. Within complex planning environments, the cost function can become ill-conditioned and lead to exploding and vanishing gradients. In turn, this will lead to algorithmic singularity and the optimization problem cannot converge to a solution.

Summary: One class of optimization-based algorithms uses gradient-free methods, such as short-cutting [81], B-Splines [83], and Bezier Curves [84], to post-process the output of sampling-based algorithms such that the resulting path is smooth and jerk-free.

Another class of optimization-based algorithms, known as trajectory optimization (TO), uses gradient-based optimization techniques to refine an initial straight-line path such that it satisfies motion planning constraints. A key challenge with these methods is their dependence on the initial trajectory, which can lead to convergence to local minima. Recently, deep learning frameworks have been utilized to warm-start the optimization process to improve the efficiency of these algorithms (Section V-D).

C. Collision Checking

Collision avoidance is an important component of motion planning for robotic manipulators, which ensures the robot avoids self-collision and collisions with the environment for efficient plan execution [138], [139].

Collision and proximity queries are important algorithmic primitives of sampling-based and optimization-based planning algorithms. In sampling-based planning algorithms, 90% of

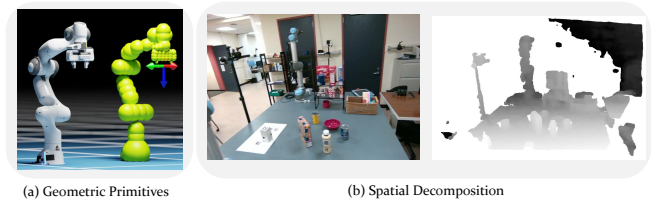


Fig. 7: Common practices for collision checking: (a) Convex geometric primitives [138]. (b) Spatial decomposition [140].

the computation time is consumed for collision queries. The primary goal of collision avoidance is to predict potential collisions and redirect the planned path toward the free configuration space.

Geometric Collision Checking: Geometric collision checking methods utilize geometric primitives (e.g., spheres, ellipsoids), and employ the Gilbert-Johnson-Keerthi (GJK) algorithm and its variants for collision detection [141], [142]. A major limitation of the geometric primitive method is its dependency on the number of obstacles within the configuration space; as the number of obstacles increases, so does the computational complexity of the collision-checking method. To address this issue, hierarchical representations are utilized [143], where a coarse representation is utilized for initial fast collision checking, and a finer representation is utilized when necessary (e.g., near or in-collision scenarios).

There are generally two methods for hierarchical representation of the workspace, as illustrated in Figure 7: bounding volume hierarchy (BVH), and spatial decomposition (SD) [144]. Bounding volume hierarchy is an object-centric method that utilizes bounding volumes, such as spheres, discrete oriented polytope (k-DOP) [145], and oriented bounding box (OBB) [146], for hierarchical representation of the environment [147]. Spatial decomposition, on the other hand, is a space-centric method, that employs partitioning techniques such as K-d trees [148], octrees [149], and space-time bounds to decompose the workspace into cells. Cells are considered occupied if an object occupies that cell [143]. Both hierarchical representations are utilized for continuous [150] and point cloud collision detection [140]. Continuous collision checking is most suitable for local steering in sampling-based planning algorithms, while point cloud collision checking is capable of performing collision queries between point clouds and point clouds and other geometric primitives. Furthermore, these collision-checking algorithms can be parallelized to leverage the capabilities of GPUs [151] and multi-core CPUs [152] for enhanced parallel computations.

Signed Distance Fields (SDFs): The Signed distance function is an alternative methodology for representing collision distances. It computes the distance between a point in the workspace and a surface, assigning a zero value to points on the surface. Other points receive a signed value indicating the distance and direction relative to the surface. The resulting Signed Distance Field (SDF) can be used as collision avoidance constraints in various motion planning algorithms [79]. These methods provide a distance field and its gradient

for proper collision checking within trajectory optimization methods [153]–[155].

Swept Volume: The swept volume between two manipulator configurations represents the total volume occupied by the manipulator as it moves from one configuration to another. It is commonly used for continuous collision detection. However, computing the exact swept volume for articulated objects, such as robotic manipulators, is computationally complex. To address this, several approximation methods have been proposed in the literature, including polyhedra-based [156], occupation grid-based [157], and boundary-based [158] algorithms.

Machine learning-based Proxy Collision Checkers: Proxy collision checkers have been developed to improve the computational efficiency of motion planning algorithms for robotic manipulators by replacing conventional geometric collision-checking methods [41]. These proxy collision checkers are utilized during the tree construction phase, while final path validity is checked by an exact geometric collision checker. Proxy collision checkers must maintain high accuracy to avoid false negatives and scale well to high-dimensional configuration spaces. Various machine learning-based binary classifiers, including parametric methods like Support Vector Machines (SVMs), and K-Nearest Neighbors (KNNs), as well as non-parametric approaches like Gaussian Mixture Models (GMMs), and Gaussian Processes (GPs)) have been utilized as proxy collision checkers [159]–[166].

Collision Detection for pHRI: In addition to proactive collision avoidance for motion planning, another line of research focuses on utilizing sensors for contact detection [167] in physical human-robot interaction (pHRI) scenarios where human and robotic manipulators operate in close proximity. In pHRI scenarios, proprioceptive sensors primarily detect collisions, as exteroceptive sensors may not adequately predict potential contacts between the robot and its environment. Collision detection involves monitoring changes in the electrical currents of manipulator drives [168], the differences between measured torques and the nominal control law [169], [170], and utilizing tactile sensors [171]. After a contact is detected, the collision needs to be identified, and processed to determine the intentionality, locality, duration, and severity of it for efficient pHRI [172]. For a detailed review of contact detection and collision management, refer to the survey paper by Haddadin *et al.* [173].

Summary: Collision checking takes up to 90% of the computation time in motion planning algorithms. Classical approaches rely on methods like geometric collision detection [140], [145], signed distance fields [79], swept volume estimation [156], and machine learning-based proxy collision checkers [41]. In recent years, the parallelization, auto-differentiation, and fast inference capabilities of deep neural networks have been leveraged to improve the efficiency of collision checking algorithms (Section V-E).

IV. DEEP LEARNING BASICS AND POTENTIAL FOR ROBOTIC MANIPULATORS MOTION PLANNING

In this section, we introduce various building blocks of deep learning methods used for manipulator motion planning.

A. Basic Deep Learning Frameworks

This section provides a brief introduction to basic deep learning frameworks used in motion planning for robotic manipulators. Table VII provides an overview of these frameworks and their applications in robotic manipulator motion planning.

Multi-layer Perceptrons (MLPs): Multi-layer perceptron is an architecture that consists of stacking several fully connected layers. Each layer in an MLP has two parts: an affine transformation and an activation function. The activation function introduces nonlinearity into the network such that MLPs are realized as universal function approximations [174].

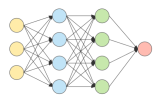
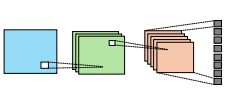
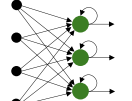
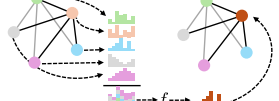
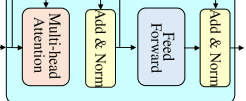
The universal function approximation property of multi-layer perceptrons (MLPs) has enabled their utilization for learning complex mappings in robotic manipulators' motion planning. When trained on an oracle planning dataset, these structures rely on hierarchical nonlinear transformations to capture global planning patterns. This allows MLPs to generate end-to-end plans, replace the sampling primitive in sampling-based algorithms for informed sampling, warm-start trajectory optimization with high-quality initial guesses, or serve as a proxy collision checker to accelerate planning.

Convolutional Neural Networks (CNNs): Although MLPs are efficient in learning interactions between various features in tabular data [175], they are agnostic toward existing patterns within the data. In contrast, Convolutional Neural Networks leverage the translation invariance [176] and locality properties of convolutional kernels to capture spatial hierarchies and patterns within grid-like topology data (e.g., images). Translation invariance ensures that all regions of an image are treated uniformly, while locality focuses on small neighborhoods to encode hidden representations. These properties make CNNs highly effective for image recognition, and object detection, with applications in robotic manipulator autonomy stack [177].

In real-world motion planning, the exact geometry of obstacles is often unknown, and planners often rely on partial or noisy sensed representations obtained from cameras or depth sensors. The locality and translation invariance of CNNs enable them to extract spatially consistent features from sensed representations such as occupancy grids, depth maps, or RGB images. CNNs can capture both local obstacle geometry and global workspace structure by hierarchically composing local features into higher-level representations. This allows CNNs to be integrated into motion planning for end-to-end planning, to guide sampling in sampling-based algorithms, or serve as a proxy collision checker to accelerate planning.

Recurrent Neural Networks (RNNs): MLPs and CNNs are well-suited for handling fixed-length data, i.e., structured tabular data and grid-like data structures. However, these frameworks fall short for applications in domains such as time

TABLE VII: Schematic of basic Deep learning frameworks, their characteristics, and their potential for improving various components of classical planning algorithms for robotic manipulators.

	MLPs	CNNs	RNNs	GNNs	Transformers
Schematic					
Characteristics	• Universal approximator due to nonlinear activation function.	• Structural encoding via local, translation-invariant convolution operator.	• Spatiotemporal encoding via recurrent connections.	• spatiotemporal encoding via convolution operator and permutation-invariant operators.	• Long-horizon dependencies via multi-head attention and scaled dot-product.
E2E Planning	• planning policy representation.	• Workspace encoding.	• Temporal dependencies of planning policy.	• Spatio-temporal dependencies within planning policy.	• Long spatiotemporal relations within planning policy.
SBMP	• Sampling distribution for informed sampling.	• Planning space encoding for informed sampling.	• Encode temporal dependencies for neural sampling	• Encode spatiotemporal dependencies for neural sampling	• Encode long-horizon spatio-temporal dependencies for neural sampling
TO	• Provide initial trajectory for TO.	-	-	-	-
Collision Checking	• (Binary) collision checker.	• Point cloud collision checker.	-	-	• Distance-to-collision estimator.

Note: “E2E Planning” denotes end-to-end planning, “SBMP” denotes sampling-based motion planning algorithms, and “TO” denotes trajectory optimization algorithms.

series prediction and language processing, where inputs are sequential and vary in length [178].

Recurrent Neural Networks (RNNs) [179], [180] feature recurrent connections that capture the dynamics within sequence data by maintaining a form of memory across the input. This memory is achieved by incorporating cycles into the network’s architecture, which allows RNNs to process data sequences [181].

Motion planning for robotic manipulators is inherently a long-horizon problem that requires reasoning over temporal dependencies between successive configurations. RNNs allow information from earlier steps in the planning to influence later decisions by maintaining a hidden state that is updated sequentially, which enables RNNs to capture temporal correlations. Therefore, RNNs can be applied to generate end-to-end motion plans by encoding sequential dependencies, or to bias samples in sampling-based algorithms toward consistent regions of the configuration space.

Graph Neural Networks (GNNs): MLPs, CNNs, and RNNs are specialized in handling structured data, i.e., tabular, grid-like, and sequential. Graph Neural Networks (GNNs) extend beyond these limitations by being powerful tools for operating on both structured and unstructured data [182].

Graphs and GNNs have been used for the representation and encoding of data structures in a variety of applications, including robotic tactile recognition [183], robotic grasping [184], surface defect recognition [185], and water quality prediction [186], showcasing their capability in representing complex data structures and the relationships between their elements.

The workspace and configuration space in robotic manipulator motion planning are inherently high-dimensional, unstructured, and exhibit spatiotemporal dependencies between workspace entities. GNNs have the potential to encode these

dependencies by representing the planning problem as a graph, where nodes encode workspace entities such as robots or obstacles, and edges capture their relationships. The GNN aggregates local and global dependencies with iterative message passing while preserving permutation invariance. As a result, GNNs are capable of encoding both spatial correlations (e.g., robot-to-obstacle proximity) and temporal dependencies within planning waypoints. Given these properties, GNNs can be employed for end-to-end planning, to guiding the sampling primitive in sampling-based algorithms toward feasible regions, or serve as a proxy collision checker to accelerate planning.

Transformers: The early boom in deep learning was powered by foundational architectures like MLPs, CNNs, and RNNs. However, the current wave of progress in deep learning is predominantly driven by the transformer architecture. Central to the transformer architecture is the multi-head attention mechanism (scaled-dot product), which was originally proposed by Vaswani *et al.* [187].

Transformers have become central to natural language processing tasks [188], and have also emerged as the default model for numerous vision tasks, including image recognition and object detection [189]. Moreover, transformers have demonstrated impressive performance in other domains such as reinforcement learning [190] and GNNs [191].

The multi-head attention mechanism operates on a collection of embedding vectors that represent candidate waypoints or states along a planning trajectory. Within each head, the scaled dot-product attention computes similarity between a query vector (e.g., the current waypoint) and all key vectors (other waypoints), to produce attention coefficients. These coefficients weight the corresponding value vectors to selectively attend to past, present, or future waypoints that are most relevant to the current planning state. Subsequently, multi-

head attention encodes diverse dependency patterns within the planning dataset (e.g., local smoothness vs. global feasibility).

Within the motion planning framework, the multi-head attention mechanism encodes long-range spatiotemporal dependencies across the planning trajectories, such that early waypoints can directly influence the prediction of later ones to improve the consistency of long-horizon trajectories. Consequently, this framework can be leveraged to learn and encode these dependencies for end-to-end planning, guiding sampling primitive of sampling-based algorithms, and performing collision checking.

Large Language Models (LLMs): Large language models are built upon the transformer architecture and are extensively pre-trained on internet-scale datasets and can be fine-tuned for specific tasks [192]. These models hold the potential to significantly enhance various aspects of the robotics domain [193]–[195].

LLMs and vision-language models (VLMs) have been widely applied across the manipulation stack for robot policy learning, high-level task planning, and code generation [26]. In policy learning, language-conditioned imitation learning leverages the semantic understanding of LLMs to enable vision-based manipulation guided by language instructions [196]. These models also help decompose complex manipulation tasks into simpler sub-tasks to assist reinforcement learning agents to interact with the environment [197]. For task planning, LLMs have been utilized to generate high-level task plans for long-horizon, complex manipulation tasks [30]. Additionally, their code generation capabilities reduce the need for extensive domain knowledge in task planning and manipulation [198]. These models can also be integrated with classical planning algorithms to form a modular framework for motion planning.

Although LLMs improve the generalizability of manipulation algorithms, they still face challenges such as distribution shift during policy deployment. Furthermore, their high computational demands and long inference times limit their deployment on real-world robotic systems. Additionally, these models are restricted to processing language without the capability to reason about the physical world.

B. Generative Models

Discriminative and generative models are two fundamental approaches for data modeling and prediction in deep learning. These models serve different purposes and are based on distinct principles for learning from data [174].

- **Discriminative models:** A discriminative model is primarily applicable to tasks such as regression or classification, where the objective is to distinguish between different classes or predict a specific value. Essentially, a discriminative model aims to model the conditional probability of the output given the input, focusing on differences between classes.
- **Generative models:** A generative model learns to model the underlying distribution of the training data and uses this model for generating new data. For instance, an image generation algorithm can generate new images by

learning the underlying statistical properties and patterns of an image dataset.

Deep Generative Models (DGMs) [200] are neural networks designed to approximate complex, high-dimensional probability distributions within datasets. Once trained, a DGM can estimate the likelihood of observations and generate samples from the learned underlying distribution. These models have been widely used in diverse applications such as anomaly detection [201], wind farm control [202], and surface defect recognition [203]. Some of the most popular approaches in DGMs include Variational Auto-encoders (VAEs) [204], Generative Adversarial Networks (GANs) [205], Normalizing Flows [206], Energy-based Models (EBMs) [207], Diffusion Models [208], and Flow Matching [209]. Table VIII provides an overview of generative models, their characteristics, and their application in robotic manipulator motion planning.

Motion planning datasets are inherently multi-modal where multiple feasible trajectories may exist for a given planning problem. Deep generative models capture this inherent multi-modality by learning a distribution over trajectories and generate diverse candidates during inference. For instance, VAEs can encode trajectories into a latent space and decode them to generate new samples, GANs can learn to generate trajectories by matching the distribution of the planning dataset, and diffusion models can iteratively refine noisy trajectories via learned denoising steps. This generative capability allows deep generative models to be utilized to generate end-to-end plans, to generate informed samples within sampling-based planning algorithms, and to warm-starting trajectory optimization with high-quality initial guesses.

Variational Auto-encoders (VAEs): VAEs are a class of deep generative models designed to encode the distribution within a dataset by mapping it into a latent space characterized by a Gaussian distribution [210]–[216]. The learned underlying distribution is then used to generate new data points.

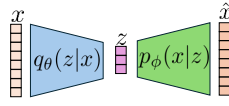
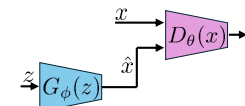
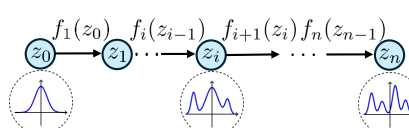
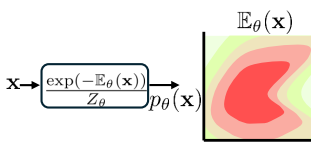
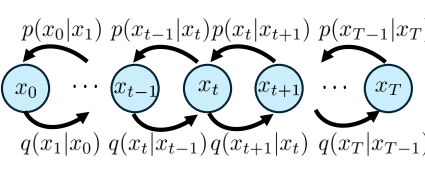
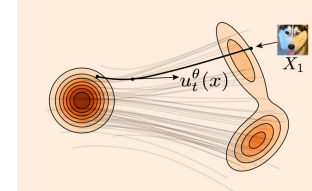
VAEs consist of two components: an encoder and a decoder. The encoder is a neural network that maps a data point to a hidden representation in the latent space. The decoder is also a neural network that reconstructs the input data from the latent space representation.

Generative Adversarial Networks (GANs): GANs consist of two components: a generator, and a discriminator [217], [218]. These components engage in a competitive interaction, where the generator attempts to create data similar to the original data, while the discriminator tries to distinguish between the generated and the actual data. This competition allows GANs to generate samples that closely resemble the original training data.

Normalizing Flows (NFs): NF principle [219]–[221] is to transform a simple distribution into a more complex distribution that better represents the underlying characteristics of the dataset. This transformation is achieved through a sequence of invertible, and differentiable mappings [222].

Energy-Based Models (EBMs): EBMs capture complex relationships within a dataset by defining a scalar energy function over the data space [223]. The non-normalized probabilistic

TABLE VIII: Schematic of various deep generative models, their characteristics, and their applications in improving various components of classical planning algorithms for robotic manipulators. The flow matching figure is adopted from [199].

Deep Generative Models		
 Variational Auto-encoders (VAEs)	 Generative Adversarial Networks (GANs)	 Normalizing Flows (NFs)
 Energy-based Models (EBMs)	 Diffusion Models (DMs)	 Flow Matching (FMs)
Characteristics	<ul style="list-style-type: none"> • Probability density estimation. 	<ul style="list-style-type: none"> • Sample generation during inference.
E2E Planning	<ul style="list-style-type: none"> • Task and planning space encoding for E2E planning. 	
SBMP	<ul style="list-style-type: none"> • Capture the multi-modality of planning problem for neural informed sampler. 	<ul style="list-style-type: none"> • Learning constraint manifold for constraint-aware sampling.
TO	<ul style="list-style-type: none"> • Learning prior for trajectory optimization algorithm. 	<ul style="list-style-type: none"> • Solve the optimization problem via maximum a-posterior (MAP) formulation.

Note: “E2E Planning” denotes end-to-end planning, “SBMP” denotes sampling-based motion planning algorithms, and “TO” denotes trajectory optimization algorithms.

property of EBMs allows them to be compatible with various deep learning frameworks for energy function representation. However, this property makes training EBMs challenging. Additionally, in practical applications, selecting an appropriate energy function is non-trivial and requires careful consideration.

Diffusion Models: Diffusion models operate by sequentially adding noise to the input data in a forward process and then removing this noise in a backward process known as the denoising process [224]. The effectiveness of diffusion models relies on their ability to predict the noise added during the forward process.

Flow Matching: Flow matching is an alternative to diffusion models for generative modeling. Unlike diffusion models, which rely on an iterative denoising process, flow matching [199], [209] directly learns a time-dependent vector field that transports the base distribution to the target distribution.

C. Point Cloud Neural Networks

3D point clouds, derived from 3D LIDAR and RGB-D cameras, provide more information about the environment compared to RGB images [225]. However, encoding 3D point clouds with image processing neural networks is challenging because they lack the inherent spatial structure found in image data. Consequently, novel deep learning frameworks are

required for processing unstructured 3D point cloud data. Notable among proposed point cloud neural networks (PCNets) are PointNet [226], PointNet++ [227], and Point Transformer [228], which combine basic neural network frameworks and permutation invariance operators (e.g., sum, max) to operate on 3D point cloud data [229].

In real-world settings, planners often have access to partial and noisy 3D point cloud observations of the workspace. Point cloud neural networks can encode such unstructured data, where the number and ordering of points may vary across observations. These models embed each point into a feature space and then aggregate features across neighborhoods to extract meaningful workspace representations. In the context of motion planning, these models can encode obstacle geometry, free-space structure, and robot–workspace interactions directly from sensor data. As a result, they can be leveraged for end-to-end planning, to guide informed sampling in sampling-based algorithms, and to accelerate collision checking with perception-driven embeddings.

D. Neural Radiance Field

Neural Radiance Field (NeRF) frameworks learn 3D scene representations from 2D images, and provide a promising method for encoding perception and motion in robotics [230]. These frameworks take camera rays as input and output a volumetric rendering of the scene. This volumetric rendering has been integrated into classical path planning algorithms as

occupancy representations for navigation [231]. Additionally, NeRFs' 3D structural bias makes them useful for object pose estimation [232] and integration into manipulation policies [233].

The success of motion planning algorithms for robotic manipulators largely depends on accurate scene representation for collision checking. NeRF's ability to reconstruct 3D scenes from 2D images can be used for dynamic collision checking between the manipulator and the neural field. Also, the sampling primitive of sampling-based algorithms can be conditioned on the reconstructed 3D scene for informed sampling [234].

E. Neural SDFs

Neural SDFs are signed distance functions learned using deep neural networks [235], [236]. They are commonly used for scene reconstruction, where the zero-level set of the neural SDF is extracted to represent clear geometric surfaces within the scene. Collision checking is the major bottleneck in motion planning for robotic manipulators. Neural SDFs also can be considered as the collision avoidance constraint in motion planning for robotic manipulators (Section V-E).

F. Robotic Foundation Models

LLMs are trained on large-scale internet datasets and have been applied to text generation and open-vocabulary visual recognition tasks. Their semantic reasoning and visual interpretation capabilities have been leveraged for high-level robotic planning. However, since these models primarily reason about semantics and textual prompts, they require additional auxiliary components to handle low-level motion planning and control tasks [26].

Robotic foundation models, also known as vision-language-action (VLA) models, jointly process visual input, language commands, and directly output executable actions for task-conditioned control in an end-to-end manner [237], [238]. These models combine LLMs' ability to encode text and images with physical interactions learned from a robotic-specific dataset [239] to establish generalist robotic systems for manipulation.

V. DEEP LEARNING IN PLANNING FOR ROBOTIC MANIPULATORS

In this section, we survey the state-of-the-art in utilizing deep learning methods for robotic manipulator motion planning. Given that robots work within similar settings and tackle similar motion planning problems, leveraging past planning experiences expedites the search for future plans. Figure 8 provide an overview of the state-of-the-art research that utilized deep learning frameworks for motion planning in robotic manipulators. Additionally, Figure 9 presents the performance metrics of a neural motion planner compared to benchmark planners across varying levels of task complexity for a robotic manipulator. This section is organized based on the deep learning frameworks used to improve each component of classical planning algorithms. Figure 10 provides an overview of data representation, task representation, and training neural motion planners.

A. End-to-end Planning

Deep learning frameworks have been utilized for end-to-end planning to capture the complex spatio-temporal dependencies and multi-modality inherent in motion planning for robotic manipulators. Table IX overviews the state-of-the-art of utilizing deep learning for end-to-end planning, their contribution, and performance compared to benchmark planners.

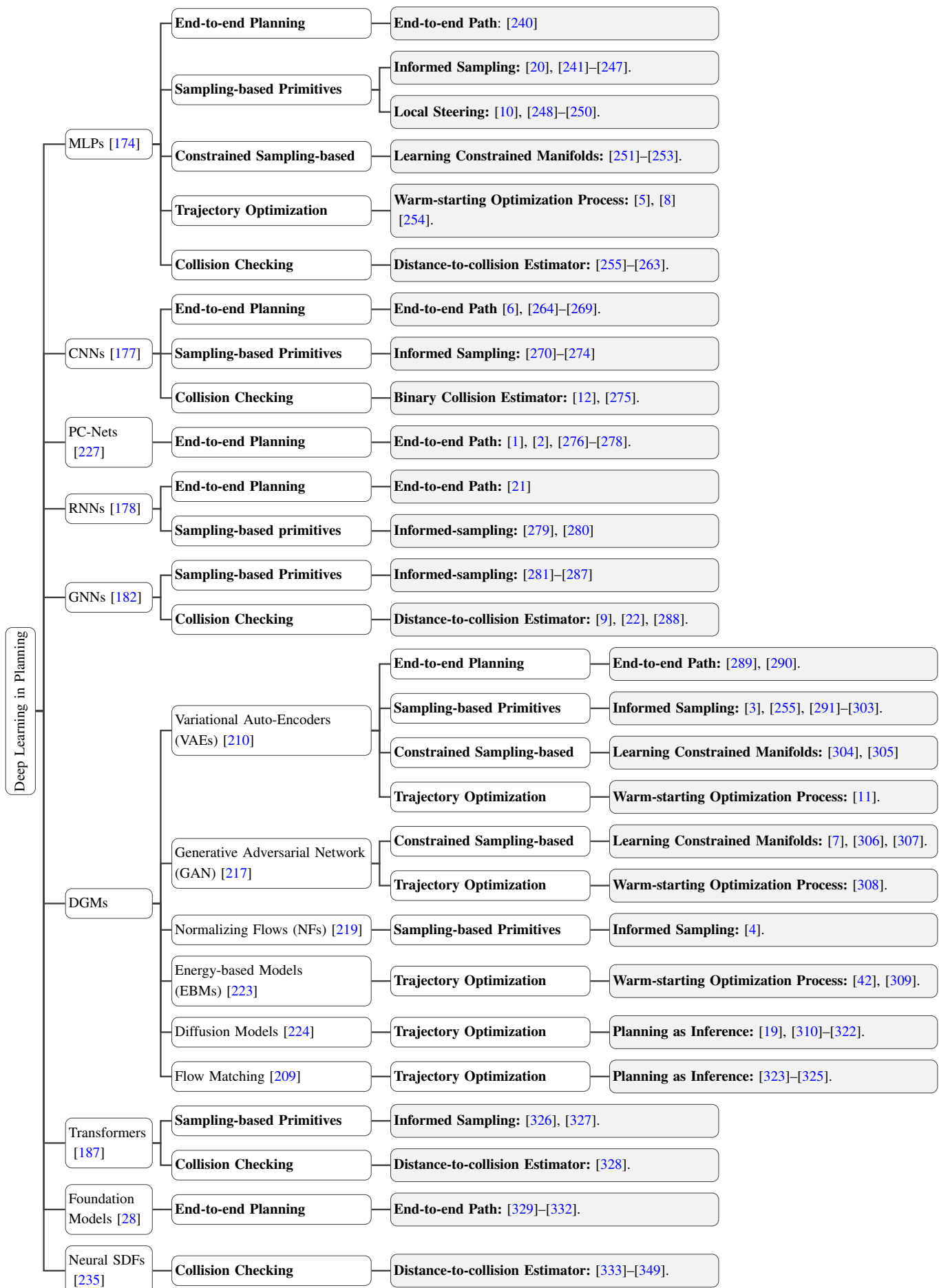
Multi-Layer Perceptrons (MLPs): Deep neural networks (MLPs) have been utilized as universal approximations to learn the motion policy of robotic manipulators. Pandey *et al.* [240] employ an MLP consisting of a fully connected layer and ten highway layers [350] to learn the motion policy of the robotic manipulator. This framework processes a parameterized description of the workspace and generates an end-to-end trajectory. The network was trained using a customized cost (loss) function that aims to minimize the path length and avoid collision with workspace obstacles, such that the output waypoints satisfy the planning constraints.

Convolutional Neural Networks (CNNs): Ota *et al.* [264] also employ an MLP structure for short-horizon path waypoints generation. The proposed framework uses a CNN-based encoder-decoder framework to encode the depth information of the workspace and an MLP framework to generate waypoints for a downstream reactive motion generation algorithm. This module processes the depth image of the environment and the robot's state to output waypoints leading to the goal, complemented by a low-level reinforcement learning (RL)-based action controller for navigation between these waypoints.

Neural Time Fields (NTFields) [265], a demonstration-free deep learning planner, utilizes a ResNet-style deep neural structure and 3D CNNs to generate the factorized time field from the start/goal configurations and workspace embeddings. Then, the speed model of the planning space is derived by solving the Eikonal equation [351], [352] and used as the gradient step to move from the start configuration to the end configuration in a bi-directional manner. Progressive NTFields (P-NTFields) *et al.* [266] improve upon NTFields to address its generalizability and improve its success rate within clutter environments. The proposed framework adds a viscosity term (Laplacian of time field) to the Eikonal equation, which guarantees the smoothness of the predicted time field. Additionally, they deploy a progressive speed scheduling technique to address optimization difficulty near obstacles.

Constrained Neural Time Fields (C-NTFields) [6], [267] modifies the expert speed model of progressive NTFields to incorporate kinematic constraints into the estimated time fields. Task Space Regions (TSRs) [73] are leveraged to calculate the distance between robot configurations and constraint manifolds. NTFields, P-NTFields, and C-NTFields train time fields estimator network offline and assume that the environment is known a priori. Active NTFields (A-NTFields) [268] relaxes this assumption by training the time field estimator network on the fly. The proposed framework processes the incoming sensor data to calculate the ground truth speed values, and utilizes an online learning framework for training the network.

Ni *et al.* [269] improve NTFields' learning convergence by introducing temporal difference loss, normal alignment loss,



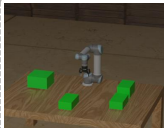
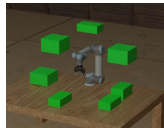
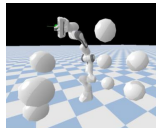
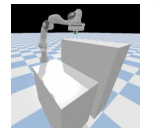
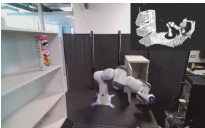
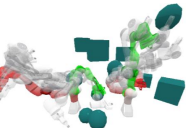
Planning Time (T(s) ↓)	Average planning time the planner takes to find a solution.			Planning Cost (C ↓)	Refers to length of the planned path within configuration space or workspace.			Success Rate (S [%] ↑)	Represents the percentage of successfully planned paths.																																																																																																																																		
(a) Planning Metrics																																																																																																																																											
																																																																																																																																											
Simple Environment		Complex Environment		Spheres		Shelf		7D		14D																																																																																																																																	
<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>Bi-RRT</td><td>1.22</td><td>10.1</td><td>97</td></tr><tr><td>IRRT*</td><td>1.3</td><td>5.3</td><td>84</td></tr><tr><td>MPNet</td><td>0.94</td><td>5.7</td><td>92</td></tr><tr><td>KG-Planner</td><td>0.91</td><td>5.8</td><td>97</td></tr><tr><td>SIMPNet</td><td>0.67</td><td>5.9</td><td>97</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	Bi-RRT	1.22	10.1	97	IRRT*	1.3	5.3	84	MPNet	0.94	5.7	92	KG-Planner	0.91	5.8	97	SIMPNet	0.67	5.9	97	<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>Bi-RRT</td><td>4.9</td><td>9.9</td><td>68</td></tr><tr><td>IRRT*</td><td>6.4</td><td>4</td><td>34</td></tr><tr><td>MPNet</td><td>20.1</td><td>4.4</td><td>37</td></tr><tr><td>KG-Planner</td><td>6.8</td><td>5.7</td><td>38</td></tr><tr><td>SIMPNet</td><td>6.3</td><td>5.9</td><td>65</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	Bi-RRT	4.9	9.9	68	IRRT*	6.4	4	34	MPNet	20.1	4.4	37	KG-Planner	6.8	5.7	38	SIMPNet	6.3	5.9	65	<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>Bi-RRT</td><td>42.9</td><td>12.1</td><td>100</td></tr><tr><td>GPMP</td><td>194.4</td><td>5.1</td><td>42</td></tr><tr><td>MDP</td><td>1.1</td><td>9.9</td><td>100</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	Bi-RRT	42.9	12.1	100	GPMP	194.4	5.1	42	MDP	1.1	9.9	100	<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>Bi-RRT</td><td>29.9</td><td>11.2</td><td>100</td></tr><tr><td>GPMP</td><td>192.1</td><td>5.1</td><td>88</td></tr><tr><td>MDP</td><td>1.1</td><td>9.3</td><td>100</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	Bi-RRT	29.9	11.2	100	GPMP	192.1	5.1	88	MDP	1.1	9.3	100	<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>RRT*</td><td>49.35</td><td>683</td><td>52.8</td></tr><tr><td>IRRT*</td><td>54</td><td>63</td><td>89</td></tr><tr><td>BIT*</td><td>7.58</td><td>826</td><td>72.2</td></tr><tr><td>MPNet</td><td>5.18</td><td>147</td><td>94.2</td></tr><tr><td>VQ-MPT</td><td>0.92</td><td>45</td><td>97.4</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	RRT*	49.35	683	52.8	IRRT*	54	63	89	BIT*	7.58	826	72.2	MPNet	5.18	147	94.2	VQ-MPT	0.92	45	97.4	<table><tr><th></th><th>T (s) ↓</th><th>C ↓</th><th>S [%] ↑</th></tr><tr><td>RRT*</td><td>1.8</td><td>9</td><td>11.8</td></tr><tr><td>IRRT*</td><td>52.8</td><td>45</td><td>21.8</td></tr><tr><td>BIT*</td><td>9.56</td><td>384</td><td>30.8</td></tr><tr><td>MPNet</td><td>17.46</td><td>117</td><td>92.2</td></tr><tr><td>VQ-MPT</td><td>2.62</td><td>18</td><td>99.2</td></tr></table>			T (s) ↓	C ↓	S [%] ↑	RRT*	1.8	9	11.8	IRRT*	52.8	45	21.8	BIT*	9.56	384	30.8	MPNet	17.46	117	92.2	VQ-MPT	2.62	18	99.2
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
Bi-RRT	1.22	10.1	97																																																																																																																																								
IRRT*	1.3	5.3	84																																																																																																																																								
MPNet	0.94	5.7	92																																																																																																																																								
KG-Planner	0.91	5.8	97																																																																																																																																								
SIMPNet	0.67	5.9	97																																																																																																																																								
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
Bi-RRT	4.9	9.9	68																																																																																																																																								
IRRT*	6.4	4	34																																																																																																																																								
MPNet	20.1	4.4	37																																																																																																																																								
KG-Planner	6.8	5.7	38																																																																																																																																								
SIMPNet	6.3	5.9	65																																																																																																																																								
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
Bi-RRT	42.9	12.1	100																																																																																																																																								
GPMP	194.4	5.1	42																																																																																																																																								
MDP	1.1	9.9	100																																																																																																																																								
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
Bi-RRT	29.9	11.2	100																																																																																																																																								
GPMP	192.1	5.1	88																																																																																																																																								
MDP	1.1	9.3	100																																																																																																																																								
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
RRT*	49.35	683	52.8																																																																																																																																								
IRRT*	54	63	89																																																																																																																																								
BIT*	7.58	826	72.2																																																																																																																																								
MPNet	5.18	147	94.2																																																																																																																																								
VQ-MPT	0.92	45	97.4																																																																																																																																								
	T (s) ↓	C ↓	S [%] ↑																																																																																																																																								
RRT*	1.8	9	11.8																																																																																																																																								
IRRT*	52.8	45	21.8																																																																																																																																								
BIT*	9.56	384	30.8																																																																																																																																								
MPNet	17.46	117	92.2																																																																																																																																								
VQ-MPT	2.62	18	99.2																																																																																																																																								
(b) Environment Complexity		(b) Obstacle Complexity		(c) Manipulator Complexity																																																																																																																																							

Fig. 9: A comparison of neural motion planners with benchmark planners across various planning complexities for out-of-distribution planning scenarios. (a) Planning metrics were used to quantify the performance of motion planners. (b) Environment complexity: SIMPNet [281] performance compared with benchmark planners (Bi-RRT [50], IRRT* [61], MPNet [20], KG-Planner [282]) across different scenarios. (c) Degree of freedom: VQ-MPT [3] performance compared with benchmark planners (RRT*, IRRT* [61], BIT* [65], MPNet [20]) across different scenarios. (d) Obstacle type: MPD [19] performance compared with benchmark planners (Bi-RRT [50], GPMP [91]) across different scenarios.

and causality preservation alongside the original loss function. The framework also employs a PirateNet structure [353], an MLP with residual gates, to estimate the time field while ensuring non-negativity, symmetry, and triangle inequality. Furthermore, it incorporates an attention mechanism to condition the field estimation network on workspace embeddings, enabling generalization to unseen workspaces.

Point Cloud Neural Networks (PC-Nets): Their ability to process unstructured 3D data and remain invariant to input permutations makes them well-suited for encoding workspace/configuration space with motion planning problems. Motion Policy Networks (M π Nets) [1], [276] features a point-cloud encoder that leverages PointNet++ [227] to encode the input point-cloud, which consists of the manipulator's current geometry, scene geometry, and manipulator's target pose, effectively representing the workspace. Additionally, an MLP-based configuration encoder is used to capture the robot's current configuration, while another MLP serves as a decoder to generate the next planning way-point guiding the manipulator toward the goal configuration.

Neural MP [2] improves upon M π Nets by incorporating more realistic scenes for data generation. This framework also utilizes an LSTM-based configuration encoder and implements a stochastic learning framework based on Gaussian Mixture Models (GMMs) to address the multi-modality inherent in the planning dataset, and integrates a light-weight real-time optimization module to improve both the efficiency and success rate of the planning problem. Deep Reactive Policy (DRP) [277] extends Neural MP by integrating a point cloud encoder with an action-chunking transformer architecture for end-to-end motion planning. DPR further fine-tunes the neural planner using the Geometric Fabrics [354] framework and privileged workspace information in simulation to resolve minor collisions. In addition, it leverages point-cloud-aware

Riemannian Motion Policies [36] to enable reactive planning in dynamic environments.

Motion Policy with LLM-Powered Dataset Synthesis and Fusion Action-Chunking Transformers (PerFACT) [278] leverages the planning capabilities of LLMs [28] and procedural primitive generation [2] to create a diverse, and semantically feasible set of workspaces for large-scale planning data collection. It then combines a point cloud encoder with a fusion action-chunking transformer architecture to intelligently attend to various planning sensing modalities for efficient planning.

Recurrent Neural Networks: The recurrent connections within RNNs have been leveraged to learn the temporal dependencies within the motion planning problem. OracleNet [21] employs stacked LSTM layers to preserve the temporal information inherent in oracle paths. This framework generates waypoints directed toward the goal configuration by incorporating the goal configuration as an auxiliary input to the network.

Deep Generative Networks (DGNs) are powerful frameworks for end-to-end motion planning. For instance, auto-encoders (AEs) are utilized for end-to-end planning for robotic manipulators.

DGNs - Auto Encoders (AE): Latent Space Path Planning (LSPP) [289] employs a VAE to encode both the configuration space (joint angles) and task space (end-effector positions) based on a dataset of randomly generated configurations. This specific method alleviates the need for explicit mapping between configuration space and task space. For optimization, Activation Maximization (AM) [355] is used to refine the goal-reaching process by back-propagating the goal-reaching error and the output of a binary collision checker network to update the latent vector. This probabilistic approach allows the planner to exploit areas of the latent space with higher prob-

abilities, enhancing efficiency. However, a major limitation is its requirement for predefined obstacle shapes for collision checking, which is impractical for real-world applications. To overcome this, Activation Maximization Planning in Latent Space (AMP-LS) [290] integrates a neural collision predictor that operates directly on environment point clouds.

Transformers and Large Language Models: The attention mechanism in transformers and LLMs enables them to capture the spatio-temporal dependencies inherent in motion planning problems. Roco [329] utilizes an LLM agent (GPT-4) [28] to generate workspace waypoints for multi-robot collaboration scenarios. These waypoints are subsequently fed into a centralized RRT-based multi-arm motion planner for coordinated planning across all manipulators. The model can generate a new set of waypoints in each iteration by adjusting the temperature of the GPT-4 model, which controls the stochasticity of the language model.

Kwon *et al.* [330] investigate the use of LLMs for zero-shot dense trajectory generation for robotic manipulators. They show that, with well-designed task-agnostic prompts, LLMs combined with off-the-shelf perception models can generate sequences of end-effector poses without relying on auxiliary low-level planning components [356]. However, since these models are not trained on physical interaction data, they struggle in handling low-level planning for robotic manipulators. LATTE [331], [332] leverages pre-trained large language models to modify initial workspace manipulator trajectories given planning context. It takes in multiple data modalities - scene geometry, images, user language command, and initial trajectory - and processes them through a transformer encoder-decoder structure to generate a modified trajectory. However, post-processing is applied to ensure the modified trajectory satisfies planning constraints since the output may not always be valid.

The main challenge in using LLMs for end-to-end manipulator motion planning is that the problem is inherently spatiotemporal rather than textual. As a result, LLM-based manipulation planning often relies on pre-defined motion primitives to carry out physical interactions within the environment and plan motions. This is due to the lack of large-scale robotic data [30], [198], [356], [357].

B. Improving Unconstrained Sampling-based Planners' Algorithmic Primitives

This section reviews how deep learning methods have improved the performance of these types of planners. As mentioned in Section III-A, sampling-based planning algorithms are built upon three algorithmic primitives: sampling, steering, and collision checking. Deep learning frameworks can be employed to improve each of these primitives to increase both the efficiency and success rate of the algorithms. Table X overviews deep learning application in unconstrained sampling-based planning algorithms. Since collision checking is a common component across all planning algorithms, the application of deep learning frameworks for collision checking will be discussed in detail in Section V-E.

1) *Sampling Primitive:* Deep learning frameworks can help to address the low convergence rate of classic sampling-based motion planners by leveraging historical data to generate collision-free informed samples within complex planning workspaces and narrow passages. Table XI overviews the state-of-the-art of utilizing deep learning for informed sampling, their contribution, and performance compared to benchmark planners.

Multi-Layer Perceptrons: Deep neural networks can encode the underlying distribution of planning datasets, making them well-suited for learning the underlying sampling distributions for informed sampling within sampling-based planning algorithms. Deep learning frameworks have been used as a sampling distribution to explicitly generate informed samples in place of the original sampling primitive of sampling-based planners. Motion Planning Networks (MPNet) [20], [241], [242] incorporates an MLP-based framework to serve as the sampling primitive within the structure of sampling-based planning algorithms. These frameworks maintain an active Dropout layer [359] during inference (sampling), which introduces stochastic behavior in the MLP-based sampler for encoding the multi-modality within the planning problem. Contractive autoencoders (CAEs) [213] are used for workspace embedding, ensuring robust representation. Additionally, Path Planning and Collision Checking Network (PPCNet) [243] consists of two neural networks: a planning network and a collision checking network for repetitive bin-picking tasks. The planning network processes the current and goal configuration to output the next time-step configuration, while the collision-checking network checks the validity of steering toward this new configuration.

MLP models have also been employed to implicitly encode and learn the planning space for informed sample generation. Parque *et al.* [244] employed MLP networks to learn the linear motion planning functions of robotic manipulators. These deep networks map a given pair of start and goal configurations of the planning problem to parameters that encode the linear transition of robot joints within the configuration space. Parallelized Diffeomorphic Sampling-based Motion Planning (PDMP) [245] combines the benefits of sampling-based and trajectory optimization planning methods. This framework utilizes an MLP structure to implicitly learn the occupancy probability distribution of the workspace and transform it through a differentiable bijection to configuration space for informed sampling.

Bhardwaj *et al.* [246] formulate edge selection in lazy graph search algorithms as a Markov Decision Process (MDP) and apply imitation learning to train a deep neural network. This network is designed to map the features of each edge to the probability of its validity, enhancing the efficiency of motion planning where edge collision checking is computationally expensive. Points-Guided Sampling Net (PGSN) [247] employs a VAE framework to encode the workspace point cloud into a linearly interpolatable latent space to enhance generalizability to unseen environments. This framework employs an MLP-based multi-modal sampling net to switch between different mode of the planning problem to encode the inherent multi-

TABLE IX: Overview of the state-of-the-art of utilizing deep learning for end-to-end planning, including the planning metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
Pandy <i>et al.</i> [240]	95 ms	PL-W : 25.2	73	<ul style="list-style-type: none"> • A planning-specific cost (loss) function. • MLPs plus highway layers [350] as neural motion planner.
	100 ms (RRT*)	PL-W : 16.98 (RRT*)	1.45 (RRT*)	
Ota <i>et al.</i> [264]	-	-	-	<ul style="list-style-type: none"> • MLP-based neural waypoints generation for RL-based reactive planning.
P-NTFields [266]	0.03	PL-C : 0.43	92	<ul style="list-style-type: none"> • Utilizes ResNet-style deep network for configuration encoding. • Utilizes 3D CNNs for workspace encoding.
	0.05 (NTFields [265])	PL-C : 0.38 (NTFields [265])	84 (NTFields [265])	
C-NTFields [6]	0.05	PL-C : 1.32	100	<ul style="list-style-type: none"> • Utilizes TSR [73] to measure the distance to constraint manifolds.
	0.06 (CBiRRT [358])	PL-C : 1.30 (CBiRRT [358])	100 (CBiRRT [358])	
A-NTFields [268]	0.03	PL-C : 2.25	91	<ul style="list-style-type: none"> • Calculates ground truth speed values on the fly. • Estimates the time field on the fly.
	1.36 (LazyPRM)	PL-C : 3.05 (LazyPRM)	87 (LazyPRM)	
Ni <i>et al.</i> [269]	0.074	PL-C : 1.95	87	<ul style="list-style-type: none"> • Utilizes PirateNet [353] for the time field estimation. • Leverage the attention mechanism to achieve generalizability.
	0.063 (NTFields [265])	PL-C : 1.63 (NTFields [265])	74 (NTFields [265])	
OracleNet [21]	1.24	PL-C : 0.85	-	<ul style="list-style-type: none"> • Utilizes RNNs to iteratively generated end-to-end paths. • LSTM for encoding temporal dependencies.
	29.32 (RRT*)	PL-C : 1 (RRT*)	-	
M π Net [1]	0.33	-	82.78	<ul style="list-style-type: none"> • PointNet++ [226] as workspace and planning space encoder. • Geometric, task-space loss for training.
	16.46 (AIT*)	-	100 (AIT*)	
Neural MP [2]	-	-	95.83	<ul style="list-style-type: none"> • PointNet++ [226] as workspace and planning space encoder. • Encoding multi-modality via GMMs.
	-	-	16.67 (M π Net [1])	
DRP [277]	-	-	84.60	<ul style="list-style-type: none"> • Geometric Fabrics [354] for fine-tuning and RMPs [36] for reactive planning.
	-	-	50.59 (Neural MP [2])	
PerFACT [278]	-	-	51.2	<ul style="list-style-type: none"> • LLMs [28] for large-scale dataset generation. • Planning modality-aware end-to-end planning.
	-	-	14.6 (Neural MP [2])	
LSPP [289]	179 ms	PL-W: 1.52	85	<ul style="list-style-type: none"> • Planning with a latent space of a VAE. • Using activation maximization [355] for updating latent vectors.
	128 ms (BiRRT)	PL-W: 2.33 (BiRRT)	84 (BiRRT)	
Roco [329]	-	-	-	<ul style="list-style-type: none"> • GPT-4 [28] for generating workspace intermediate goal states for multi-arm motion planning.
Kwon <i>et al.</i> [330]	-	-	-	<ul style="list-style-type: none"> • LLMs for zero-shot planning for robotic manipulators.
LATTE [332]	-	-	-	<ul style="list-style-type: none"> • Reshaping trajectories via language commands.

Note: “T” denotes *planning time*, “C” denotes *planning cost*, and “S” refers to *success rate* (Section II-C). “↓” indicates lower is better, and “↑” indicates higher is better. “PL-W” refers to path length measured in the workspace, and “PL-C” denotes path length in the configuration space.

modality.

Convolutional Neural Networks (CNNs): Transition invariance and locality properties of CNNs allow encoding the inherent distribution within planning data for informed sample generation, and effective workspace encoding. These models account for spatial information and can capture both local and global structures. Hierarchical Abstraction guided Robot Planner (HARP) [270] employs a U-net [360] architecture to identify critical regions (abstractions) for end-effector and other DOFs that are not defined by the end-effector's position.

Then, it utilizes a customized high-level planner to generate high-level plans that are further refined by a low-level planner for trajectory generation between start and goal configurations. Similarly, Bottleneck guided RRT* [271] utilizes a 3D CNN for encoding the bottleneck regions (narrow passages) in the workspace, biasing the sampling of the sampling-based planners towards these regions. Abdi *et al.* [272] leverage an object detection model (YOLO [361]) coupled with a deep neural network to get the coordinate of objects within the workspace. Then, a graph search algorithm is leveraged over

TABLE X: State-of-the-art literature on utilizing various deep learning frameworks to improve various components of sampling-based motion planning algorithms for robotic manipulator motion planning.

Primitive	Papers	MLPs	CNNs	RNNs	GNNs	Generative Models	Transformers	Foundation Models
Sampling	[20], [241]–[247]	✓	✗	✗	✗	✗	✗	✗
	[270]–[274]	✗	✓	✗	✗	✗	✗	✗
	[279], [280]	✗	✗	✓	✗	✗	✗	✗
	[281]–[287]	✗	✗	✗	✓	✗	✗	✗
	[3], [4], [255], [291]–[303]	✗	✗	✗	✗	✓	✗	✗
	[326], [327]	✗	✗	✗	✗	✗	✓	✗
Steering	[10], [248]–[250]	✓	✗	✗	✗	✗	✗	✗

Note: “MLPs” denotes multi-layer perceptrons, “CNNs” denotes convolutional neural networks, “RNNs” denotes recurrent neural networks, and “GNNs” denotes graph neural networks.

a grid in the workspace to find a path between the start and goal poses.

FIRE (Fast retrieval of Relevant Experiences) [273] retrieves relevant location representations from past planning instances to guide the planning problem. This method utilizes a self-supervising method to find pairs of similar-dissimilar location representations. Then, a similarity function through the latent space of Siamese network [362] - a CNN-based neural network architecture with two identical encoders - is trained to retrieve relevant experiences to guide the planning problem and ensure generalizability to out-of-distribution problems. **Heuristic Map Network (HMNet)** [274] employs a CNN-based framework to embed and encode workspace information, to learn a heuristic map (cost-to-go) for guiding the sampling process of the planning algorithm to perform guided exploration towards the planning goal.

Recurrent Neural Networks (RNNs): recognized for their capability to encode inherent temporal sequences, have been effectively employed to enhance the sampling module within sampling-based planners and to encode dynamic dependencies in dynamic task spaces. The recurrent connection within RNNs effectively encodes spatial and temporal relationships within planning datasets. **LSTM-BiRRT** [279] incorporates an LSTM sampler to guide the BiRRT algorithm toward the goal configuration by leveraging past planning experiences in dual-arm planning scenarios. Similarly, **Hou *et al.*** [280] utilizes an RNN-based encoder to capture the temporal dependencies within dynamic/static environments. The proposed framework also implements a deep neural network to learn the feasible solution space from past experiences, enhancing sampling efficiency.

Graph Neural Networks (GNNs): The capability of GNNs to accurately characterize and learn the structure of Euclidean and non-Euclidean data makes them highly effective for encoding the planning space and biasing the sampling distribution sampling-based planning algorithms towards the planning goal. **Spatial-Informed Motion Planning Network (SIMPNet)** [281] and **KG-Planner** [282], [283] construct graphs representing the kinematic structure of the robotic manipulator. Then, a graph neural network is trained on the constructed graph to generate informed, kinematic aware samples towards

the planning goal. **Yu *et al.*** [284], [285] accelerated classical sampling-based motion planners by utilizing a trained GNN for path exploration and path smoothing. This framework generates a random graph (RGG) by randomly sampling within the configuration space and including start and goal configurations. Then, the GNN determines the validity of RGG’s edges to reduce collision queries. This framework also incorporates a cross-attention mechanism to integrate the obstacle embeddings from the workspace into the configuration space for environment-aware motion planning.

Zhang *et al.* [286] utilizes a GNN framework to encode the spatio-temporal structure within dynamic motion planning. The network uses an RGG and an attention-based temporal encoding of the dynamic obstacles to determine which edges to steer to minimize unnecessary collision queries. In this framework, the trajectory of the dynamic obstacles needs to be known a priori, which is not necessarily the case in real-world examples. In subsequent work, **Zhang *et al.*** [287] introduces **DynGMP** to mitigate the need for prior knowledge of the movement of the dynamic obstacles and plan in unpredictable dynamic environments. In this framework, a GNN trained on an RGG determines edge priority. **DynGMP** preserves collision-free parts of the initially constructed tree for reuse in subsequent tree constructions, performing collision-checking on nodes and edges that intersect with the geometries of the obstacles and the robot to reduce the computational overhead of the replanning sub-module.

Deep generative networks (DGNs) are powerful deep learning frameworks utilized to capture the multimodal nature within planning datasets. DGN models such as auto-encoders (AEs), generative adversarial networks (GANs), and normalizing flows (NFs) are commonly used to enhance the sampling primitive of sampling-based planning algorithms.

DGNs - Auto-Encoders (AEs): AEs are effective for learning sampling distributions in sampling-based planning algorithms by encoding the dataset into a latent space. **Ichter *et al.*** [291] employs a conditional variational autoencoder (CVAE) conditioned on planning information (i.e., obstacles, start and goal configurations) to bias the sampling process to promising regions of configuration space. This framework learns the underlying structure within the planning dataset generated by

TABLE XI: Overview of the state-of-the-art of utilizing deep learning frameworks for informed sampling, including the planning metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
MPNet [241]	0.81	PL-C : 6.98	78.6	<ul style="list-style-type: none"> MLP framework as informed sampler. Dropout [359] for encoding planning multi-modality.
	9.2 (BIT*)	10.78 (BIT*)	56 (BIT*)	
PCCNet [243]	0.1	PL-C : 5.88	90	<ul style="list-style-type: none"> MLP-based informed sampler and collision-checker for bin-picking tasks.
	0.38 (MPNet [241])	7.3 (MPNet [241])	94.4 (MPNet [241])	
Hou <i>et al.</i> [280]	0.6	-	-	<ul style="list-style-type: none"> MLP-based informed sampling distribution.
	0.81 (RRT)	-	-	
PDMP [245]	5.99	-	96	<ul style="list-style-type: none"> MLP to learn occupancy distribution. Transforms distribution to configuration space for sampling.
	13.83 (RRT*)	-	43 (RRT*)	
HARP [270]	-	-	100	<ul style="list-style-type: none"> U-net for critical region identification for the end effector as well as other DOFs.
	-	-	~ 50 (BiRRT)	
HMNet [274]	149 ms	PL-W : 0.99	100	<ul style="list-style-type: none"> Learning heuristic map through 3D CNN.
	2466 ms (BiRRT)	3.8 (BiRRT)	76 (BiRRT)	
SIMPNet [274]	6.3	PL-C : 5.9	65	<ul style="list-style-type: none"> GNNs for kinematic-aware informed sampling. Cross-attention mechanism for workspace-aware sampling.
	20 (MPNet [241])	4.4 (MPNet [241])	37 (MPNet [241])	
SERA [293]	9.1	-	95.3	<ul style="list-style-type: none"> VAE for mapping the planning space into a low-dimensional latent space.
	22.6 (MPNet [241])	-	82.5 (MPNet [241])	
VQ-MPT [3]	0.9	#V: 45	97	<ul style="list-style-type: none"> VQ-VAEs [215] for encoding feasible sampling regions. Transformer-based sampling within feasible regions.
	5.18 (MPNet [241])	#V: 147 (MPNet [241])	94 (MPNet [241])	
G-WAE [303]	4.36	PL-W : 1.86	80	<ul style="list-style-type: none"> WAEs [216] for encoding free configuration space. GGNNs [363] for spatial-aware encoding.
	9.74 (RRT*)	PL-W : 2.21 (RRT*)	37 (RRT*)	
P-Flows [4]	-	#V: 11.6 × 10³	-	<ul style="list-style-type: none"> Utilizes normalizing flows to learn sampling distribution.
	-	#V : 32.7 × 10 ³ (MPNet [241])	-	
NEXT [326]	-	PL-C : 17.5	70.6	<ul style="list-style-type: none"> Leverages attention mechanism to encode configuration space into a discrete latent space.
	-	PL -C : 29.6 (BIT*)	47.5 (BIT*)	
TEMP [326]	0.741	#V : 195	-	<ul style="list-style-type: none"> Utilizes attention mechanism for informed sample generation.
	17.87 (RRT*)	#V : 1486 (RRT*)	-	

Note: “T” denotes *planning time*, “C” denotes *planning cost*, and “S” refers to *success rate* (Section II-C). “↓” indicates lower is better, and “↑” indicates higher is better. “PL-W” refers to path length measured in the workspace, “PL-C” denotes path length in the configuration space, and “#V” indicates the number of vertices the planner explores to plan.

an oracle planner and generates informed samples within the framework of sampling-based planning algorithms. Dastider *et al.* [292]–[294] introduced a motion planning framework that transforms the high-dimensional configuration space into a low-dimensional latent space using a CVAE framework to generate adaptable motion policies. Then, a graph search algorithm is deployed to plan in the learned low-dimensional latent space. They further enhance their approach by implementing a diffusion variational autoencoder (D-VAE) to encode and reduce the dimensionality of the planning space for motion planning in a low-dimensional space [295]. The D-VAE encodes the motion dynamics of the robotic manipulator and moving obstacles into a unified framework. This approach utilizes an extended Kalman filter (EKF) [364] to predict the

movement of dynamic obstacles.

Latent Sampling-Based Motion Planning (L-SBMP) [255] employs an encoder-decoder framework to encode high-dimensional planning spaces into a low-dimensional latent space and performs sampling-based motion planning within it. This allows sampling, steering, and collision checking to be conducted more efficiently within the low-dimensional latent space, thus enhancing the efficiency and reducing the computational complexity of motion planning for high DOF robotic manipulators. Leveraging Experience with Graph Oracles (LEGO) [296] trains a CVAE on an RGG to generate a roadmap which contains bottleneck nodes within the planning space. The work [297], employed LEGO-CVAE [296] to identify and learn the critical bottleneck regions within the

planning space, generating informed samples that serve as roots for RRT motion planning algorithm.

Gaebert *et al.* [298] employed a CVAE to encode the planning space and generate informed samples conditioned on the motion planning problem (start and goal configuration and workspace encoding). The CVAE-based sampler is incorporated into any sampling-based planning algorithm for informed sample generation. Kobashi *et al.* [299] proposed decomposing the task space of the motion planning problem to handle the high variance implicit within such problems. This framework utilizes Binary Space Partitioning (BSP) [365] to partition the workplace and identify challenging regions and assign nodes to them. Then, the assigned nodes are used to train a neural network for generating nodes for new workspaces. And, a CVAE - conditioned on environment embeddings - is utilized to generate samples around key nodes for planning.

The Neural Randomized Planner (NRP) [300] trains a discriminative MLP and a generative CVAE neural local sampler to learn local sampling distributions. Then, these local samplers are used within global sampling-based motion planners for motion planning. Planning with Learned Subgoals (PLS) [301] incorporates the planning problem's temporal information to account for temporal constraints. This model feeds the temporal and spatial information of the planning problem to a CVAE to generate sub-goal configurations for planning, effectively handles time-constrained, reactive motion planning scenarios in dynamic environments.

Vector Quantized-Motion Planning Transformers (VQ-MPT) [3], [302] utilizes Vector Quantized-Variational Autoencoders (VQ-VAEs) [215] to learn and encode the configuration space, aiming to reduce its dimensionality and identify areas likely to contain feasible paths. VQ-VAEs help to avoid the issue of posterior collapse, a common problem in traditional variational autoencoders, making them particularly effective for encoding high-dimensional configuration spaces. VQ-MPT also incorporates a cross-attention mechanism to condition the planning problem on workspace embedding, and start and goal configurations. Additionally, an auto-regressive transformer is employed to capture long-horizon correlations and sample within promising regions within the configuration space. This facilitates informed sampling in downstream sampling-based motion planning algorithms, enhancing the efficiency of these planners.

Graph Wasserstein Autoencoder (GraphWAE) [303], utilizes a variant of the Wasserstein Autoencoder (WAE) [216] with Gated Graph Neural Networks (GGNNs) [363] as encoder and decoder, to encode the collision-free region within the configuration space. The WAEs, compared to VAEs, offer enhanced stability during training and demonstrate sufficient representation capabilities, which are crucial for encoding high-dimensional and complex configuration spaces. This framework is trained on a successful path dataset, and the decoder (graph generative model) is leveraged as a neural informed sampler for a downstream sampling-based planning algorithm.

DGNs - Normalizing flows (NFs): NFs have also been employed to effectively encode the configuration space for in-

formed sampling within sampling-based planning algorithms. A distinctive property of NF is that they don't experience mode collapse, thanks to the inclusion of the mode collapse loss in the optimization problem. PlannerFlows [4] learns the sampling distribution of sampling-based planning algorithms through normalizing flows. This framework is conditioned on planning information (i.e., workspace embedding, start and goal configurations) to generate informed samples towards the planning goal.

Transformers: Transformers and attention mechanisms have been leveraged to encode the long-horizon spatiotemporal dependency within planning problems. Neural Exploration-Exploitation Tree (NEXT) algorithm [326] utilizes an attention-based network to embed the configuration space into a discrete latent representation. Then, a neuralized value iteration [366] is applied in the discretized latent space for informed sample generation. Transformer Enhanced Motion Planner (TEMP) [327] leverages the transformer architecture's ability to encode long-horizon dependencies and inter-relationships for generating next state configuration. This framework gets workspace embeddings, planning history, and planning objectives, and outputs informed samples to expand the constructed tree towards the planning goal. This approach has enhanced both planning time and planning efficiency compared to classical planners.

2) *Steering Primitive:* Neural network architectures can also be utilized for steering in sampling-based planning algorithms. Typically, classical sampling-based planners steer towards the sampled configuration along a straight line, which requires fine-grained collision queries. Learning-based customized steering function can be applied to minimize the early termination of the expansion and improve the efficiency of the planner. Table XII provides an overview of the state-of-the-art of utilizing deep learning for steering in sampling-based planning algorithms.

Multi-Layer Perceptrons: MLPs are utilized to learn the steering primitive in sampling-based planning algorithms. One approach to implementing customized steering functions involves using control barrier function (CBF)-based [367] steering controllers. These controllers steer the robot towards the sampled configuration while avoiding collision with the obstacles. However, hand-crafted CBF-based controllers often struggle to generalize to unseen configuration spaces, and different robots, particularly those with higher DOF. Control Barrier Function-Induced Neural Controller (CBF-INC) [248] addresses this limitation by designing a neural control barrier function for collision avoidance and informed steering within sampling-based motion planners. By reducing the number of collision checks and enhancing the steering function, CBF-INC increases the success rate and efficiency of the sampling-based planning algorithms.

Chiang *et al.* [10], [249] utilize deep neural networks to estimate the manipulator's swept volume between two configurations to use it as the steering primitive to improve sampling-based planning algorithms. This process leverages explicit neural representations to enhance the process. This approach benefits from the local steering module having prior

TABLE XII: Overview of the state-of-the-art of deep learning for improving the steering primitive, including the planning metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
CBF-INC [248]	345.8	#V : 162.5	76.1	• Utilizes a neural control barrier function for effective steering.
	287.6 (RRT)	#V: 252.5 (RRT)	62.5 (RRT)	
Chiang <i>et al.</i> [10]	-	PL - L : 90	80	• Deep neural network for implicit swept volume estimation for efficient steering.
	-	PL - L: 100 (PRM)	40 (PRM)	

Note: “T” denotes *planning time*, “C” denotes *planning cost*, and “S” refers to *success rate* (Section II-C). “↓” indicates lower is better, and “↑” indicates higher is better. “PL-L” refers to the swept volume of the robotic manipulator in liters, and “#V” indicates the number of vertices the planner explores to plan. Please note that the main difference between the listed methods and respective benchmarks is the steering function within the sampling-based planning algorithm.

knowledge and proper representation of obstacle distributions within the configuration space. However, classic measures like configuration space Euclidean distance do not account for obstacle distributions, and weighted Euclidean metrics are hard to tune. To address this, Their framework utilizes an MLP-based network to approximate swept volume, though steering with the trained Deep Neural Network (DNN) remains complex due to the frequent need for inference throughout the planning process. To mitigate this, the authors suggest using a weighted Euclidean estimator (a single-layer neural network) for pre-filtering configurations before applying the trained MLP-based swept estimator model for steering, optimizing the steering in sampling-based motion planning algorithms. However, the offline data collection required for training these deep neural networks is time-consuming, as are the pre-processing steps needed for distance calculations in generating ground truth labels. To mitigate these challenges, Sugaya *et al.* [250] apply transfer learning [368] to the learning of the swept volume function, aiming to capture geometrical similarities among the same type of robotic manipulators.

C. Constrained Sampling-based Planning

In real-world manipulation settings, the output of the motion planning algorithms must satisfy certain task-specific and manipulator-specific constraints (Section III). Deep learning methods have been effectively utilized to encode constraint representations in constrained motion planning. Table XIII provides an overview of the state-of-the-art of utilizing deep learning for constrained sampling-based planning.

Multi-Layer Perceptrons: MLPs are utilized to facilitate constraint-aware sampling and projecting informed samples on constraint manifolds. Constrained Motion Planning networks (CoMPNet) [251] extends the MPNet framework [241] to handle constrained motion planning. Using a projection operator, CoMPNet uses MPNet’s neural stochastic sampler to generate informed samples, which are then projected onto the constraint manifold. Constraint manifolds are defined through Task Space Regions (TSRs) [73], and task constraints are implicitly embedded within task descriptions. CoMPNetX [252] extends CoMPNet by introducing a discriminator neural network that estimates the distance between generated samples and the constraint manifolds. The discriminator network gradient is

leveraged to project the configuration onto the constraint manifold if a generated configuration lies outside a defined threshold.

Also, deep learning frameworks have been utilized to learn constraint manifolds and directly sample on them during planning. Equality Constraint Manifold Neural Network (ECoMaNN) [253] leverages a deep learning framework to learn equality constraint manifolds from demonstrations, which can then be integrated into constrained sampling-based planners. This trained framework evaluates whether a robot configuration satisfies a given constraint and, if not, determines the distance from the constraint manifold.

Deep Generative Networks (DGNs): DGNs’ ability to learn the underlying distribution of datasets has been leveraged to directly learn constrained manifolds for constraint-aware sampling. DGN models such as auto-encoders (AEs), and generative adversarial networks (GANs) are commonly used for this purpose.

DGNs - Auto-Encoders (AEs): AEs have been used for learning equality constraint manifolds in constrained motion planning. Learning-Assisted Constrained RRT (LAC-RRT) [304] utilizes Configuration Transfer Model (CTM) - a self-supervised, encoder-decoder architecture - to map configurations from configuration space to a feature space for constraint-aware sampling. In this feature space, a custom feature composer is utilized to impose planning equality constraints on the encoder’s output. This is advantageous as handling equality constraints in the feature space becomes more straightforward. Additionally, Park *et al.* [305] use a conditional variational autoencoder (CVAE) to learn constraint manifolds. Their framework maps the planning problem into a latent space conditioned on planning constraints and performs planning within this space. A validity network is employed to verify constraint satisfaction in the latent space. After planning, the path is mapped back to the configuration space, where a numerical projector projects the generated waypoints onto the constraint manifolds.

DGNs - Generative Adversarial networks (GANs): GANs also have been used to learn constraint manifolds due to their scalability to high-DOF spaces and encoding conditional distributions. Lembono *et al.* [306], [307] utilized a GANs architecture to generate samples that are already close to

TABLE XIII: Overview of the state-of-the-art of deep learning for improving constrained sampling-based planning, including the planning metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
CoMPNet [251]	4.92 54.81 (CBiRRT [358])	-	-	<ul style="list-style-type: none"> Utilizes MPNet's sampler for informed sampling. Projects samples on constraint manifolds via projection operator.
CoMPNetX [252]	15.05 19.77 (CoMPNet [251])	-	-	
LAC-RRT [304]	5.37 47.2 (CBiRRT [358])	#V 203.11 #V 196.6 (CBiRRT [358])	100 100 (CBiRRT [358])	<ul style="list-style-type: none"> Utilizes an encoder-decoder structure for configuration encoding. Equality constraints are enforced within the latent space.
Park <i>et al.</i> [305]	2.19 2.42 (CoMPNetX [252])	-	100 100 (CoMPNetX [252])	
Lembono <i>et al.</i> [307]	0.74 1.44 (CBiRRT2 [73])	#V: 59.7 #V: 116.5 (CBiRRT2 [73])	99 100 (CBiRRT2 [73])	<ul style="list-style-type: none"> Utilizes GANs for constraint-aware informed sampling. Implements an ensemble of networks for the GANs generator to encode planning multi-modality.
Acar <i>et al.</i> [7]	0.116 0.57 (CBiRRT [358])	-	100 100 (CBiRRT [358])	

Note: "T" denotes *planning time*, "C" denotes *planning cost*, and "S" refers to *success rate* (Section II-C). "↓" indicates lower is better, and "↑" indicates higher is better. "#V" indicates the number of steps the planner explores on constraint manifolds to plan.

the constraints manifold. Their approach involves using an ensemble of neural networks in the GAN generator to encode the multi-modality inherent in the dataset and mitigate mode collapse. This framework decreases the number of projections and reduces planning time. Acar *et al.* [7] utilize two deep generative networks (DGNs)- CVAEs, and GANs - for constraint-aware sample generation. Their method learn task-specific constraint manifolds - such as end-effector pose, closed-kinematic chain, and balance - to facilitate sampling within the framework of constrained sampling-based planning algorithms.

D. Global Trajectory Optimization

Deep learning methods have increasingly been leveraged to learn prior trajectory distributions to guide the trajectory optimization algorithm. These frameworks combine the expressive power of deep learning frameworks with robust trajectory optimization algorithms for efficient trajectory planning. Table XIV provides an overview of the state-of-the-art of utilizing deep learning for warm-starting trajectory optimization algorithms.

Multi-Layer Perceptrons (MLPs): MLPs are used to learn the initial trajectories from planning data due to their universal approximation capability. Deep-learning Jerk-limited Grasp Optimized Motion Planner (DJ-GOMP) [5] incorporates an MLP-based deep-learning module to provide a robust initial approximation for the associated optimization-based motion planning algorithm. The proposed network has multiple output heads to generate initial paths with different horizon lengths. Then, a classification network predicts the optimal initial trajectory from the set of initial trajectories with various horizon lengths. Similarly, Banerjee *et al.* [254] employed

an MLP framework to warm-start their underlying trajectory optimization problem. The proposed MLP-based framework takes the start and goal states in the workspace and generates the coefficients for a polynomial parameterization of the initial trajectory.

Constrained Neural Motion Planning with B-splines (CNP-B) [8] employs a deep neural network for constrained kinodynamic motion planning. The initial trajectory is time-parameterized using B-splines, and the network predicts the control points of the spline. The training loss is defined based on the constraint manifold loss, allowing the model to be trained with supervision using only motion planning problem instances.

Deep Generative Networks (DGNs): DGNs are increasingly used in motion planning to learn distributions of successful paths from datasets, which can be used as priors in trajectory optimization problems. These learned priors can be used to generate the initial trajectories for gradient-based trajectory optimization problems, or within a maximum-a-posterior (MAP) formulation of trajectory optimization problems, typically conditioned on motion planning constraints and goals, for end-to-end trajectory optimization.

DGNs - Auto-Encoders (AEs): AEs' ability to encode the underlying distribution within datasets is utilized to generate the initial trajectory for the trajectory optimization algorithm. Motion Planning by Learning the Solution Manifold (MPSM) [11] learns the solution manifold of trajectories using a VAE architecture with a customized loss function. Trained on non-optimal trajectories sampled from a baseline proposal distribution, the framework generates trajectories that are further finetuned using the CHOMP [79] algorithm.

TABLE XIV: Overview of the state-of-the-art of deep learning for improving trajectory optimization, including the planning metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
DJ-GOMP [5]	0.75 1.25 (TrajOpt [90])	-	-	• Utilizes an MLP framework to estimate the initial trajectory for warm-starting trajectory optimization.
CNP-B [8]	8 ms 60 ms (BIT*)	-	100 100 (BIT*)	• Parametrizes the initial trajectory via B-spline. • utilizes an MLP to predict the control points of the B-spline.
MPSM [11]	0.21 1.8 (CHOMP [79])	-	-	• Encodes solution manifolds with a customized VAE structure. • Utilizes CHOMP [79] for fine-tuning the generated trajectories.
Ando <i>et al.</i> [308]	5.94 ms 125 ms (BiRRT)	PL - W: 3 PL - W: 3.14 (BiRRT)	81.8 100 (BiRRT)	• Encodes solution manifolds with a GAN architecture. • Trajectory optimization is done in the obstacle-free latent space.
GFCOP [309]	5.2 6 (BIT*)	PL - W: 1.32 PL - W: 3.71 (BIT*)	98 100 (BIT*)	• GFs [354] for local and CMA-ES [369] for global optimality. • An EBM for warm-starting CMA-ES optimization.
MPD [19]	1.1 194.4 (GPMP [91])	PL - C: 9.9 PL - C: 5.1 (GPMP [91])	100 42 (GPMP [91])	• Formulates motion planning as inference with a diffusion model. • Utilizes temporal U-net [370] for the denoising process.
M ² Diffuser [314]	3.89 0.46 (M π Nets [1])	-	30.49 3.24 (M π Nets [1])	• Diffusion process for trajectory distribution learning. • Task goals and planning costs guide the generative algorithm.
EDMP [320]	- -	-	85 32 (CHOMP [79])	• Diffusion-based trajectory generation with collision guidance. • Ensemble-of-collision-costs improves generalizability.
Sharma <i>et al.</i> [321]	2.74 1.94 (EDMP [320])	-	85.13 80.3 (EDMP [320])	• utilizes a hierarchy of cascaded diffusion models to encode global and local information of the planning problem.
DiffSeeder [310]	42.3 38.6 (cuRobo [138])	-	86 57 (cuRobo [138])	• Employs a conditional diffusion model for warm-starting the cuRobo [138] trajectory optimization algorithm.
Luo <i>et al.</i> [322]	0.13 0.22 (M π Nets [1])	-	99.7 88 (M π Nets [1])	• Trains EBMs of the planning space. • Leverages EMBs' gradient for the denoising process.
FlowMP [323]	0.13 5.29 (MPD [19])	-	-	• Incorporates acceleration and jerk fields for smooth and dynamically aware trajectory generation.

Note: "T" denotes *planning time*, "C" denotes *planning cost*, and "S" refers to *success rate* (Section II-C). "↓" indicates lower is better, and "↑" indicates higher is better. "PL-W" refers to path length measured in the workspace, and "PL-C" denotes path length in the configuration space.

DGNs - Generative Adversarial Networks (GANs): GANs have been utilized to encode and learn the solution manifolds for trajectory planning. Ando *et al.* [308] implemented a conditional GAN framework conditioned on obstacle information and using an RGG (comprising both in-collision and collision-free configurations). The framework encodes the free space of the configuration space into a latent space, which is entirely collision-free. This allows planning to be performed directly in the latent space, where additional constraints can also be applied. The planned path is then mapped back to the configuration space using the trained GAN.

DGNs - Energy-based Models (EBMs): EBMs' ability to implicitly represent un-normalized distributions makes them capable of encoding datasets' multimodality. Urain *et al.* [42] utilized EBMs, trained on demonstration trajectories, to generate prior distribution in a MAP inference framework to solve the planning problem. Instead of relying on a single monolithic prior, this framework employs a factored distribution to exploit composability to learn various aspects of the planning problem. Geometric Fabrics Command Optimization Problem (GFCOP) [309] uses Geometric Fabrics (GFs) [354]

to generate locally optimal motion trajectories and applies Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [369] over trajectory waypoints for reducing global planning cost. An EBM is used to warm-start the CMA-ES global optimization. The EBM is trained in a self-supervision manner, where solutions from CMA-ES are continuously added to a buffer, enabling a unified loop of learning and optimization.

DGNs - Diffusion Models: Diffusion models' ability in encoding multi-modal and high-dimensional data, as well as stable training, made them a good choice for learning motion policies from an expert dataset. One application of these models in trajectory optimization is generating the initial seed for the downstream optimization problem. DiffusionSeeder (DiffSeeder) [310] also utilizes a conditional diffusion model, conditioned on environment embedding, for seeding cuRobo [138] motion optimization algorithm.

Motion Planning Diffusion (MPD) [19], [311] formulates motion planning as an inference problem using a diffusion model. In this framework, a temporal U-net [370] is trained as a prior on expert demonstrations. Then, a task-conditioned posterior is defined based on task-specific costs such as col-

lision, self-collision, joint limits violations, enabling the sampling of optimal trajectories. Language-guided Object Centric 3D Diffusion Policy (Lan-o3dp) [312] incorporates collision avoidance as a constraint function applied to the end-effector of the robotic manipulator. This constraint is used to guide the generation of waypoints for the end-effector. Similarly, Nikken *et al.* [313] use a conditional diffusion model to generate trajectories for the end-effector of a robotic manipulator. Instead of relying on expert demonstrations, the framework generates training data through linear interpolation between start and goal poses and conditions on obstacle information during training.

Mobile Manipulation Diffuser (M²Diffuser) [314] learns scene-conditioned goal-directed trajectory-level distributions [371] using a diffusion process trained on expert demonstrations for mobile manipulators. Then, a guided trajectory optimization is performed for finding the optimal trajectory. During inference, the trajectory optimization problem is guided by task objective energy functions (e.g., grasping, placement) as well as planning constraint functions (e.g., smoothness and collision avoidance). APEX [315] introduces an obstacle-guided diffusion-based trajectory planner for dual-arm robotic manipulators. This platform first transforms the planning problem into a latent space with a variational autoencoder, then employs an obstacle-guidance classifier to perform planning and replanning in dynamic environments.

Li *et al.* [316], [317] utilize a constraint-aware diffusion model for trajectory optimization. In this framework, planning constraints are incorporated into the training loss function for accurate trajectory representation. Planning with Environment Representation, Sampling, and Trajectory Optimization (PRESTO) [318] employs a conditional diffusion process to generate initial trajectories for the downstream trajectory optimization. The framework constructs an environmental embedding by identifying key configurations, which are then input into a Diffusion Transformer (DiT) [372] for the denoising process. Planning constraints such as collision avoidance, and smoothness are incorporated into the training loss to enhance generalizability to new planning instances.

Ensemble-of-costs-guided Diffusion for Motion Planning (EDMP) [320] runs multiple guided diffusion models in parallel for trajectory generation. Each diffusion process is guided by a specific cost function and corresponding guidance hyperparameter. The use of an ensemble of collision costs improves the planner's generalization across different planning environments. Sharma *et al.* [321] propose a hierarchical cascaded diffusion planner for global planning in complex environments. The framework employs a high-level diffusion model to generate a coarse plan from the start to the goal configuration, and uses lower-level diffusion models to satisfy local constraints.

Power *et al.* [319] propose a constraint-composable diffusion model to generate initial trajectories for warm-starting the Constrained Stein Variational Trajectory Optimization (CSVTO) [113] algorithm. The model is a classifier-free diffusion model trained on a dataset generated by CSVTO for individual constraints. During inference, planning constraints are composed to improve generalization to new planning

scenarios. Luo *et al.* [322] introduce a diffusion potential field for trajectory generation, where EBMs are trained on successful paths to encode promising regions of the planning space. The gradients of learned energy functions are utilized to guide the denoising process during trajectory sampling. In this framework, different EBMs are associated with different environments and can be combined to enable generalization to out-of-distribution trajectory generation scenarios.

DGNs - Flow Matching (FM): Flow matching's ability to directly learn a time-dependent transport from a source distribution to a target distribution makes it well-suited for trajectory optimization through sampling. FlowMP [323] improves upon MPD [19] by utilizing flow matching to learn the trajectory prior. It extends the flow matching framework by incorporating acceleration and jerk fields, enabling the generation of smoother and more dynamically feasible trajectories. Safe Flow Matching (SafeFlow) [324] combines the flexibility of flow matching methods with control barrier functions (CBFs) to introduce Flow Matching Barrier Functions (FMBFs), which provide formal safety guarantees for the generated trajectories. FMBFs incorporate dynamic control inputs as a regularization term to guide the flow toward the safe manifold (i.e., collision-free planning space). Tian *et al.* [325] Leverages a flow matching framework to learn the planning distribution of feasible motion plans to warm-start cuRobo [138] for efficient motion planning.

E. Collision and Proximity Querying

Collision checking is the main bottleneck in motion planning algorithms, accounting for up to 90% of the computation time [373]. Deep neural networks have been utilized as proxy collision checkers to mitigate this limitation. Table XV provides an overview of the state-of-the-art of utilizing deep learning for improving collision checking.

Multi-Layer Perceptrons (MLPs): One line of research has used MLP frameworks for binary collision checking. Ichter *et al.* [255] employ an MLP-based binary collision checker within a sampling-based planning framework. Trained on data generated by a classical collision checker, the model acts as a binary classifier that takes consecutive states and a workspace embedding as input to determine whether the path between the two states is in collision.

Liu *et al.* [256] propose an MLP-based framework for binary self-collision detection in redundant manipulators. The network is trained on randomly sampled configurations labeled with their collision status and, during inference, predicts the probability of a given configuration being in self-collision. Additionally, Self-imitation Learning by Planning Plus (SLIP+) [257] trains a deep neural network to assess the collision probability of a given robot configuration. This network takes the robot's configuration and the workspace embedding as input and outputs the collision probability of the state. Also, Krawczyk *et al.* [258] utilize an MLP-based framework to predict the self-collision status of a mobile manipulator. Trained on data generated by the Flexible Collision Library (FCL) [147], the model takes the mobile manipulator configuration as input and outputs its collision status.

TABLE XV: Overview of the state-of-the-art of deep learning for improving collision checking, including the metrics reported relative to evaluated benchmark methods, and each approach's primary contribution to robotic manipulator motion planning.

Paper	Benchmarks & Metrics			Contributions
	T [s] ↓	C ↓	S [%] ↑	
SLIP+* [257]	0.13	-	96	• Utilizes an MLP framework to determine the collision status of robotic configurations.
	1.02 (PRM)	-	77 (PRM)	
Krawczyk et al. [258]	0.16 ms	-	-	• Utilizes an MLP framework to determine the collision status of robotic configurations.
	0.31 ms (FCL [147])	-	-	
CN-RRT* [262]	2.0	#V: 964	97	• Utilizes an MLP framework to determine the separation distance between the manipulator and the surrounding environment.
	2.7 (GJK-RRT [141])	#V: 1214 (GJK-RRT [141])	95.2 (GJK-RRT [141])	
SCN [12]	0.01 ms	-	93.2	• Utilizes 3D CNN to process workspace point cloud. • Processes target object through PointNet++ [227].
	0.49 ms (FCL [147])	-	75.4 (FCL [147])	
CabiNet [275]	6.41 μ s	-	89	• Improves upon SceneCollisionNet [12] by considering diverse workspaces.
	7.03 μ s (SCN [12])	-	69.9 (SCN [12])	
GraphDistNet [9]	3.2	-	70	• utilizes two graphs to encode the geometrical relation between the manipulator and workspace obstacles.
	1.9 (ClearanceNet [262])	-	30 (ClearanceNet [262])	
GDN-R [22]	0.18	-	-	• utilizes Gumbel top-k relaxation to identify highly probable interconnections between graph geometries.
	0.38 (GraphDistNet [9])	-	-	
PairwiseNet [288]	-	-	99	• Focuses on pairwise collision distance estimation. • Utilizes DGCNN [374] for encoding the point cloud of objects.
	-	-	96 (ClearanceNet [262])	
DistFormer [328]	31458 ms	-	99.1	• Utilizes bounding sequence to retain geometrical properties. • Leverages transformers to estimate distance to collision.
	14797.2 ms (GraphDistNet [9])	-	97.76 (GraphDistNet [9])	
CompositeSDF* [337]	4.78	-	-	• Utilizes MLP frameworks to learn an SDF network for each link.
	241 (GJK-RRT [141])	-	-	
SE3NN [342]	0.189	-	90	• Utilizes a link SE(3) representation for planning embedding. • Leverages continual learning to handle dynamic environments.
	0.148 (ClearanceNet [262])	-	73 (ClearanceNet [262])	

Note: Please note that * indicates that the collision checking framework is embedded within a planning algorithm for benchmark comparison. “T” denotes *planning time* for rows marked with *, and *average collision query time for the others*, “C” denotes *planning cost*, and “S” refers to *success rate* (Section II-C). “↓” indicates lower is better, and “↑” indicates higher is better. “#V” indicates the number of steps the planner explores on constraint manifolds to plan.

Tran *et al.* [259] integrated a contractive auto-encoder (CAE) with an MLP to assess the collision status (i.e., collision-free or in-collision) of the sampled configurations in sampling-based motion planning frameworks. The CAE encodes the robot task space, and the output, along with a robot configuration, is fed into the MLP to determine the collision status of the configuration. DeepCollide [260] employs an MLP-based structure combined with a forward kinematics kernel to determine the collision status of a robotic manipulator. This framework takes joint angles as input and outputs a collision score, enabling efficient evaluation of potential collisions.

MLPs also have been utilized to determine the collision distance for collision and self-collision queries [261]. ClearanceNet (CN) [262] employs an MLP framework to accurately predict the minimum separation distance between a manipulator and surrounding obstacles. The model takes as input an environment embedding and the manipulator pose, and is trained on data generated using the GJK algorithm [141]. By leveraging the fast inference capabilities of MLPs, Clear-

anceNet facilitates batch collision checking, which significantly improves planning speed by reducing the computational complexity traditionally associated with geometrical collision checking. However, since it uses point cloud representations of obstacles, ClearanceNet faces challenges in accurately estimating distances for non-convex geometries and exhibits limited generalization to unseen environments.

Liu *et al.* [263] introduced an MLP-based hierarchical self-collision detection method consisting of a classifier and a regressor designed for binary collision detection and estimating the distance-to-collision for collision-free configurations. To ensure the accuracy of the predictions, a robust geometric collision detector is used to double-check collision-free states when the distance to collision falls below a pre-defined threshold.

Convolutional Neural Networks (CNNs): CNNs are well-suited for collision checking over point clouds due to their ability to encode the local and global structures of 3D scenes. SceneCollisionNet (SCN) [12] is a framework designed for real-time collision checking between two point clouds - the

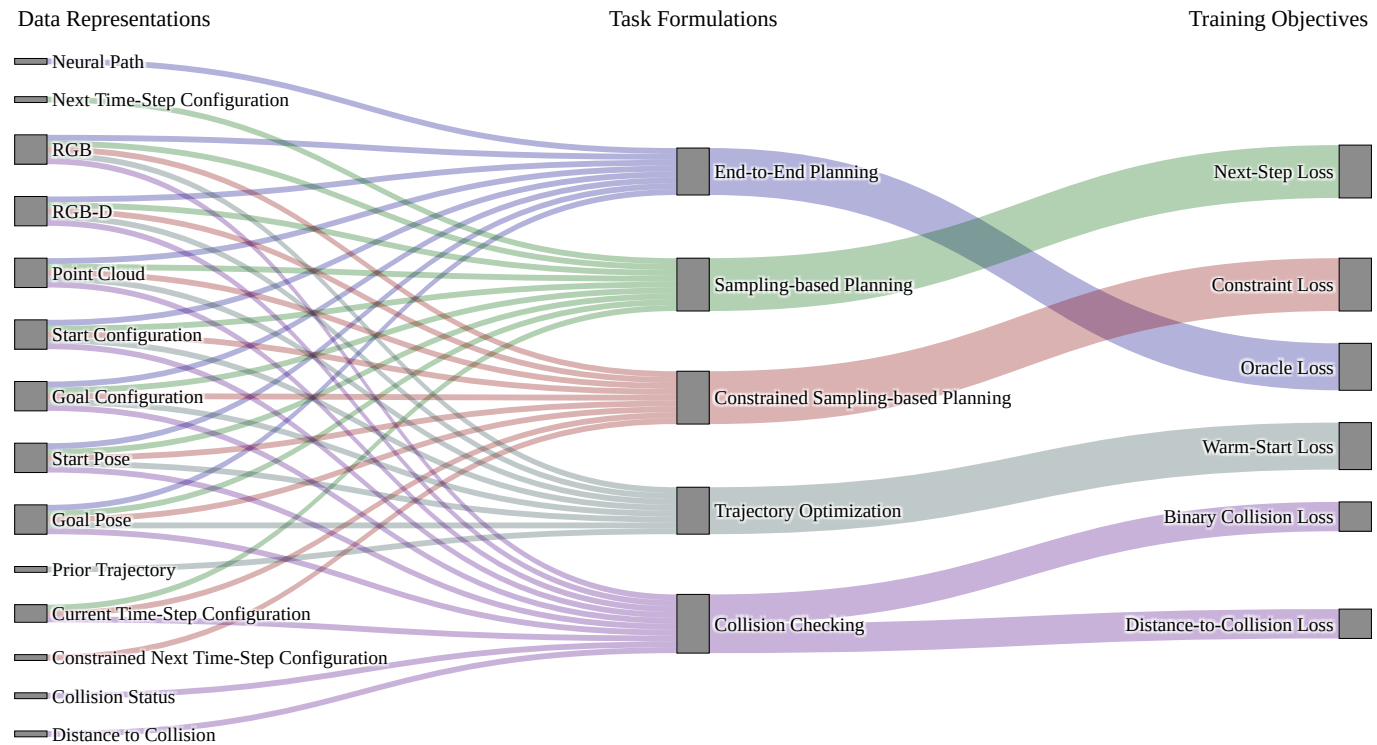


Fig. 10: Data representation and training objectives of neural motion planners for robotic manipulators. The left column illustrates the input and output modalities (data representation) and the right column demonstrates the corresponding loss function (training objective) for training and deploying neural motion planners. Classical motion planners such as cuRobo [138] and planners from OMPL [375], along with collision checkers from FCL [147], are mainly used to generate oracle datasets for training neural motion planners.

manipulator and workspace within a motion planning context - even under partial observability. The workspace is voxelized, and a shared MLP encodes the latent features within each voxel. These voxel-wise representations are then processed through 3D convolutional layers to learn an implicit 3D embedding of the workspace. Additionally, the target object is encoded using the set abstraction layer of PointNet++ [227], enabling the downstream MLP-based binary classifier to perform collision checking between the workspace embedding and the target object. CabiNet [275] improves upon SceneCollisionNet capabilities to scale and generalize to various clutter environments. This is achieved by augmenting the workplaces with various common objects, such as shelves and cabinets.

Graph Neural Networks (GNNs): GNNs' scalability and generalizability are utilized for collision querying in planning problems. GraphDistNet [9] aims to encode geometrical and topological relationships between a manipulator and workspace obstacles to estimate both collision distance and collision gradients. The framework constructs two graphs to encode the manipulator and the workspace and employs Graph Attention Networks [376] to encode interactions between them. Trained on data generated with FCL [147], GraphDistNet enables accurate distance and gradient estimation and generalizes to unseen workspaces without requiring retraining. However, the cost of message passing in GraphDistNet is proportional to the graph size, which can impede its appli-

cation to 3D complex geometries. To address this, GDN-R [22] with layer-wise probabilistic graph rewiring is proposed for distance-to-collision estimation, utilizing Gumbel top-k relaxation [377] to identify high probable interconnections between graph geometries, thereby, increasing connectivity between graphed objects. The input to GDN-R consists of a graphed representation of workspace geometries, which undergo iterative message passing to produce updated embeddings.

PairwiseNet [288] focuses on estimating pairwise collision distance, instead of focusing of global collision distance. This framework employs EdgeConv layers from Dynamic Graph Convolutional Neural Networks (DGCNN) [374] to encode the point-cloud of workspace shape geometries into shape feature vectors. Subsequently, a fully connected neural network processes these vectors of paired geometries to determine the minimum distance between them. Then the minimum of these pairwise distances will be treated as the global collision distance. Thanks to the pairwise collision-distance estimation, this framework easily generalizes to a workspace with new but similarly shaped geometries.

Transformers: Transformers, known for their ability to learn long-horizon dependencies, have also been utilized for collision querying in manipulator motion planning. DistFormer [328] is a distance-to-collision estimator that leverages the attention mechanism. Trained on a dataset generated by FCL

TABLE XVI: A compact and concise overview of state-of-the-art literature that leverages various deep learning frameworks to improve various components of classical planning algorithms for robotic manipulator motion planning. *less than 5 papers*, *5 ~ 10 papers*, *more than 10 papers*.

	E2E Planning	U-SBMP		C-SBMP	TO	Collision Checking
		Sampling	Steering			
MLPs	[240]	[20], [241]–[247]	[10], [248]–[250]	[251]–[253]	[5], [8], [254]	[255]–[263]
CNNs	[6], [264]–[269]	[270]–[274]	-	-	-	[12], [275]
PC-Nets	[1], [2], [276]–[278]	-	-	-	-	-
RNNs	[21]	[279], [280]	-	-	-	-
GNNs	-	[281]–[287]	-	-	-	[9], [22], [288]
DGMs	VAEs	[289], [290]	[3], [255], [291]–[303]	[304], [305]	[11]	-
	GANs	-	-	[7], [306], [307]	[308]	-
	NFs	-	[4]	-	-	-
	EBMs	-	-	-	[42], [309]	-
	DMs	-	-	-	[19], [310]–[322]	-
	FM	-	-	-	[323]–[325]	-
Transformers	-	[326], [327]	-	-	-	[328]
Foundation Models	[329]–[332]	-	-	-	-	-
Neural SDFs	-	-	-	-	-	[333]–[349]

Note: “E2E Planning” denotes end-to-end planning, “SBMP” denotes unconstrained sampling-based motion planning algorithms, “C-SBMP” denotes constrained sampling-based motion planning algorithms, and “TO” denotes trajectory optimization algorithms.

[147], this framework utilizes bounding sequences to retain the shape of the manipulator and obstacles, employing a self-attention mechanism to transform these bounding sequences into feature sequences. A cross-attention module then fuses the feature sequences of the manipulator and obstacles, ensuring that the manipulator is implicitly aware of the obstacle’s locations. The augmented feature sequence of the manipulator is ultimately used to estimate the collision distance in this framework.

Large Language Models (LLMs): LLMs can be used to encode the workspace of a robotic manipulator by generating a voxel value map, which can serve as input for downstream collision checking algorithms. VoxPoser [378] utilizes LLMs to generate a 3D voxel value map based on task descriptions. It then applies a greedy search over a collision avoidance map to identify collision-free end-effector positions.

Neural SDFs: Neural SDFs learn a continuous signed distance function using a neural network, which can be integrated into motion planning algorithms for robotic manipulators. Neural Joint Signed Distance Field (Neural JSDF) [333]–[335] is an MLP-based implicit signed distance function to estimate the distance between robot links to any point in the workspace. The network is trained on a dataset containing both collision-free and in-collision samples generated randomly. The proposed network provides a smooth and differentiable distance field that can be used as a collision constraint in optimization-based trajectory planning algorithms. Neural JSDF implicitly learns the manipulator’s kinematics, which leads to the accumulation of error along the forward kinematics chain. Regularized Deep Sign Distance Field (ReDSDF) [336] introduces a neural signed distance function for articulated objects, taking robot configuration and a 3D point in the workspace as input to the signed distance, which is then used to compute the

repulsive velocities in motion planning.

Composite SDF [337] is a neural signed distance function for predicting the minimum distance between an articulated robot and any 3D point in the workspace. Instead of learning an SDF for the whole body of manipulator, this framework learns individual SDF networks for each robot link to handle the complexity of high-dimensional configuration spaces. Composite SDF utilizes the Open3D library [379] to sample on/near manipulator meshes for training the proposed framework. Zhao *et al.* [338] enhance the Composite SDF framework by incorporating the gradient of the distance function, enabling its integration into optimization-based trajectory planning algorithms. The composite neural SDF frameworks need to follow the kinematic chain step-by-step, which leads to computational complexity. To address this issue, Robot Neural Distance Function (RNDF) [339] incorporates the manipulator’s forward kinematics chain implicitly within a neural framework. The model includes a regression head for each robot link, with each head conditioned on the others based on the arm’s kinematic chain. This design reduces the computational complexity of explicitly computing the forward kinematics while also minimizing error accumulation associated with implicit representation.

Quintero-Pena *et al.* [340] extend the Neural JSDF *et al.* [334] to stochastic environments by learning a distribution over signed distance function. An MLP-based framework is utilized to learn and predict the mean and variance of the stochastic neural signed distance function, capturing uncertainty in distance estimation. Configuration Space Distance Field (CDF) [341] directly measures the distance between joint configurations and workspace obstacles in the configuration space, where planning and control are performed. This distance field preserves the Euclidean property and ensures a unit-norm gradient, eliminating the need for repeated inverse

kinematics computations to map between the workspace and configuration space.

One major limitation of neural collision checkers is their poor adaptability to minor workspace changes, such as the addition of new obstacles or robots, which often necessitates complete retraining. SE3NN [342] addresses this issue by utilizing an active learning method to augment the dataset with near-boundary configurations. Instead of using joint configurations as input to the neural collision checker, the framework adopts a redundant SE(3) representation of planning space for effective collision checking. Markov Chain Monte Carlo (MCMC) sampling [380] is then employed to continuously sample boundary data for continuous training of the network.

Another approach to collision detection involves estimating the manipulator's swept volume between start and goal configurations. Deep neural networks have been widely used to approximate the geometry of this swept volume, enabling continuous collision checking. Baxter *et al.* [343] propose an MLP-based framework that takes the start and goal configurations of a manipulator as input and outputs a voxel grid representation of the swept volume. The V-REP simulation framework [381] is used to generate training data by collecting the robot's swept volumes. This approach, however, results in low spatial resolution and inherent discretization errors. To address this issue, Lee *et al.* [344]–[346] utilize a neural network to approximate the boundary surfaces of the manipulator's swept volume. The framework employs a high fidelity neural signed SDF [236] for surface reconstruction to represent the manipulator's boundary surface along the trajectory, enabling accurate construction of the swept volume.

Reachability-based Signed Distance Functions (RDFs) [347], an implicit neural representation, computes the distance between manipulator's parameterized swept volume and workspace objects. This framework utilizes a polynomial zonotope-based representation of the robotic arm and box obstacles to create the training dataset, and follows the network structure of [236] as the implicit representation. Joho *et al.* [348] propose a neural continuous implicit swept volume representation that takes as input start and goal configurations along with workspace query points, and outputs the signed distance between the query points and the manipulator's swept volume. To enhance reliability, the neural collision checker is interleaved with a geometric collision checker.

Comformalized Reachable Sets for Obstacle Avoidance with Spheres (CROWS) [349] uses MLP-based frameworks to model the swept volume of a robotic manipulator. One network learns the centers and radii of Spherical Forward Occupancy (FSO) [382] of the robotic manipulator, while another provides the derivative of the SFO with respect to the sphere centers. The SFO approximates the swept volume of the robotic manipulator using a collection of spheres, and this implicit representation is used as a collision constraint in optimization-based trajectory planning.

VI. CHALLENGES AND FUTURE PERSPECTIVES

In this section, we explore the challenges of employing deep learning frameworks for motion planning in robotic

manipulators, and explore future avenues to address some of these challenges. Table XVI provides a guide for researchers interested in using deep learning techniques for robotic manipulator motion planning. It classifies more than 100 research papers published since 2018 according to motion planning primitives and deep learning frameworks.

A. Generalizability

1) **Challenges:** While various deep learning frameworks incorporate different types of inductive biases to handle data not encountered during training, they often struggle to generalize to out-of-distribution settings. This is because robot skills are limited to the movement distribution learned from planning data. This challenge is particularly pronounced in motion planning for robotic manipulators, where small changes in the workspace significantly alter the planning problem [34]. The poor scalability and generalizability of neural motion planners stem from the fact that motion planning-specific datasets are scarce.

2) **Possible Method 1: LLMs as Generalist Motion Planners:** LLMs possess a vast actionable knowledge that can be leveraged to plan motions for robotic manipulators [26]. As shown in Figure 11-(a), these models can be prompted for zero-shot or few-shot motion planning for robotic manipulators. Utilizing LLMs for end-to-end motion planning is challenging because these models are not well-suited for encoding spatiotemporal dependencies within motion planning problems, as they are not trained on physical interaction datasets [330].

A more practical application of LLMs in robotic manipulator motion planning is their potential to improve specific algorithmic primitives of classical planning algorithms. In sampling-based planning algorithms, LLMs with high temperature settings, conditioned on language instructions and visual representations of the workspace, can function as informed samplers. Moreover, these models can be instructed to learn and encode the constraint manifold, enabling constraint-aware informed sampling.

In optimization-based planning algorithms, LLMs can be used to warm-start the optimization process based on language instructions and visual representations of the workspace. For collision checking, these models can generate the workspace value map from visual and language inputs to enable real-time collision checking [378].

One limitation of using LLMs within motion planning algorithms for real-time motion planning is their high inference time, as these models contain millions of parameters [383]. To mitigate this, utilizing powerful computational resources such as dedicated GPUs and TPUs can significantly reduce inference time.

3) **Possible Method 2: Large-scale Synthetic Data Generation:** The generalizability of neural motion planners can be improved with large-scale planning datasets. Also, the methodology of robot foundation models [237] can be leveraged to fine-tuning LLMs on this large-scale, planning-specific dataset.

State-of-the-art high-fidelity physics-based simulators [385]–[387] can be leveraged to generate datasets for motion

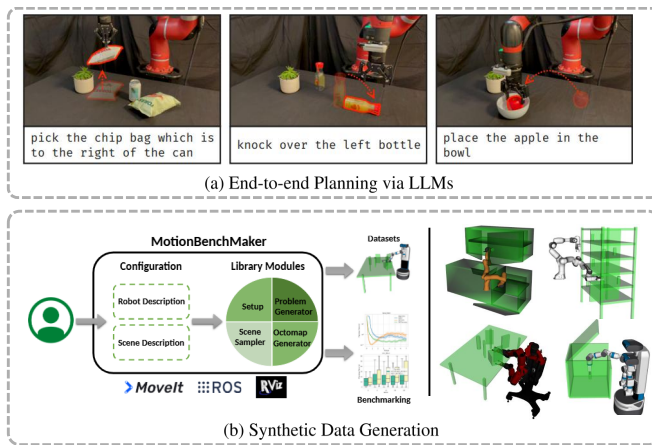


Fig. 11: Possible methods to improve the generalizability of neural motion planners: (a) LLMs for end-to-end planning [330]. (b) Large-scale synthetic dataset generation [384].

planning. However, a major limitation is the lack of variability and realism in the simulated planning workspaces. There are two possible methods to address this limitation: *procedural workspace generation* and *generative AI-based workspace generation*.

Procedural workspace generation methods programmatically generate cluttered workspaces for motion planning. MotionBenchMaker [384] generates diverse workspaces by procedurally generating assets with URDF sampling (Figure 11-(b)). These assets can be combined in a physics-based simulator to generate various workspaces. Classical motion planners from the Open Motion Planning Library (OMPL) [375] or advanced planners such as cuRobo [138] can then be leveraged to generate motion planning datasets. However, MotionBenchMaker produces simple workspaces with only one major asset and large gaps between smaller ones. As a result, the generated workspaces are not realistic, and the trained neural motion planner struggle to plan in real-world settings.

Neural MP [2] addresses MotionBenchmaker's limitations by generating more realistic and cluttered workspaces for data collection. This framework utilizes procedural assent generation and sampled everyday objects from a 3D object dataset [388] to create diverse workspaces. Then, a classical planning algorithm from OMPL is utilized to collect planning data to train a generalist neural motion planner. The resulting motion planner is capable of planning within out-of-distribution scenarios, enhancing adaptability and robustness in robotic manipulator motion planning. However, randomly creating workspaces is not trivial, and the resulting workspaces may not necessarily resemble real-world scenarios.

Generative AI-based workspace generation can leverage advanced AI tools to generate realistic and diverse workspaces for data collection. These methods can utilize generative methods (e.g., text-to-3D [390], 2D-to-3D [391]), or advanced NeRFs [230] to generate articulated 3D assets from language instructions or images for workspace generation. Classical motion planners can then be utilized within these workspaces

to generate a large-scale dataset. However, Current generative models struggle to generate a wide variety of articulated 3D assets with diverse configurations and articulations, which limits their effectiveness for training generalist neural motion planners. Existing 3D object datasets, such as Objaverse [388] and PartNet-Mobility [392], can also be used to augment generative AI workspace generation methods by directly providing articulated objects. However, the diversity of articulated objects in these datasets is limited, making them insufficient for training generalist neural motion planners.

One promising direction is to combine procedural workspace generation with generative AI methods to create realistic and diverse workspaces to collect large-scale datasets for training generalist neural motion planners.

B. Safety

1) **Challenges:** Neural motion planners can unlock the potential of robotic manipulators for deployment in vast real-world settings and applications. However, these planners do not provide completeness or optimality guarantees, which may result in planning failures [393]. On the other hand, although struggling with generalizability and scalability, classical planning algorithms do offer a certain level of completeness (e.g., probabilistic completeness of sampling-based algorithms) and theoretical optimality. Table XVII provides the advantages and limitations of neural motion planners compared to classical planning algorithms.

TABLE XVII: A qualitative comparison between neural motion planning and classical motion planning for robotic manipulators.

Planners	Advantages	Limitations
SBMP	<ul style="list-style-type: none"> Probabilistic completeness 	<ul style="list-style-type: none"> Non-smooth paths
Global Optimization	<ul style="list-style-type: none"> Smooth paths. Incorporates equality and non-equality constraints 	<ul style="list-style-type: none"> Susceptible to local minima Computationally complex
Neural Planning	<ul style="list-style-type: none"> Fast inference Encoding high-dimensional, complex distributions Leverages past experience for new problems 	<ul style="list-style-type: none"> Data scarcity Safety and reliability

Note: "SBMP" denotes sampling-based motion planning, and "Neural Planning" refers to methods that leverage deep learning methods for planning.

Additionally, task-specific safety concerns require careful consideration. For example, in collaborative environments, the neural motion planner must always prioritize avoiding collisions with human operators to ensure safe operation.

Existing research on motion planning for robotic manipulators often focuses on improving efficiency and success rates using deep learning frameworks. However, many of these methods overlook critical safety aspects, even though planned trajectories must satisfy specific constraints to ensure safe operation of the robotic manipulator [181].

For instance, one important safety challenge arises when a robotic manipulator encounters a singularity configuration. At such configurations, the geometric Jacobian becomes ill-conditioned and loses rank, which results in the loss of one

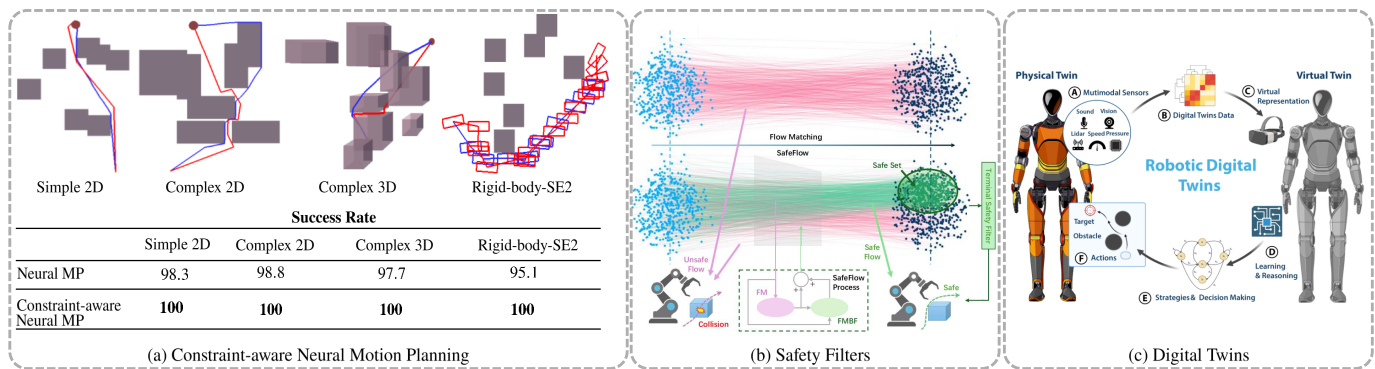


Fig. 12: Possible methods to enhance the safety of neural motion planners: (a) Constraint-aware neural motion planning [241]. (b) Safety filters [324]. (c) Digital twins [389].

or more degrees of freedom. As a result, the end-effector movement can become unpredictable and uncontrollable [394]. However, most state-of-the-art neural motion planners overlook the singularity issue and instead focus on generalization within the workspace.

2) **Possible Method 1: Constraint-aware Neural Motion Planning:** Constraint-aware motion planners combine the strengths of classical and neural motion planners for safe deployment in real-world settings [241]. These planners retain the completeness and optimality guarantees of classical algorithms for safe performance. At the same time, they can leverage the semantic reasoning and scene understanding capabilities of neural networks and LLMs for better generalization to domain-specific, dynamic planning scenarios (Figure 12-(a)).

For sampling-based planning algorithms, combining their inherent probabilistic completeness [241] and constraint-aware sampling [252] with an implicit neural-informed sampler can enhance both the safety and adaptability of neural motion planners (Figure 12-(a)). Global trajectory optimization methods can integrate the safety and flexibility of optimization-based planning with the expressive power of neural networks. In these algorithms, deep generative models can capture the multi-modal distribution of the planning dataset to efficiently warm-start the optimization algorithm, while safety constraints can be incorporated as soft constraints within the optimization framework [113]. For collision checking, fast neural collision checkers can be combined with safe geometric collision checkers for safe and efficient collision checking.

Regarding singularity, one possible direction is to treat singularity as a planning cost. Prior work has mitigated the singularity through Jacobian matrix processing techniques like Damped Least Squares (DLS) [395] and its variants [396], as well as maximizing manipulability indices [397]. Manavalan *et al.* [398] utilize kinesthetic task demonstrations to learn the manipulator's manipulability index, given that constraints are state-dependent, and the manipulability varies during task execution. For neural motion planners, one possible approach is to identify workspace areas with low manipulability measures and guide the planner to avoid these areas. Singularity-related costs can also be incorporated into learning-based trajectory optimization methods to keep trajectories away from

singular configurations. However, integrating these methods with neural motion planners for deployment in everyday unstructured environments remains challenging. Further research is necessary to combine learning-based singularity detection with neural motion planners to ensure safe operation near singular configurations.

3) **Possible Method 2: Safety Filters:** The generalizability and scalability of neural motion planners can be combined with model-based safety filters to avoid catastrophic failures [399]. Safety filters monitor the operation of a robotic system and intervene when necessary [400].

One class of safety filters is control barrier functions (CBFs), which provide smooth intervention through real-time control optimization [401]. These methods generate a safe control signal by modifying the original task signal to satisfy the safety filter's required decrease rate. CBFs [401] can be applied during neural path planning and/or execution to enforce safety constraints (Figure 12-(b)) [324]. However, designing valid CBFs for neural motion planners is challenging and requires careful consideration. Moreover, these filters mainly address geometric constraints, and additional work is needed to translate semantically defined constraints into analytical forms that can be handled by such methods [402].

Another type of safety filter is trajectory optimization at runtime. For example, Model Predictive Shielding (MPS) evaluates the task control within a prediction horizon [403]. If the task control satisfies the safety constraints, it is executed; otherwise, a fallback policy is executed. This approach can be combined with neural motion planners to locally enforce task-specific trajectory constraints. However, MPS requires a complete dynamic model of the robotic manipulator and well-defined safety constraints, which are often difficult to obtain for real-world applications.

4) **Possible Method 3: Digital Twins:** High-fidelity, cost-effective simulation environments are widely used for training and optimizing embodied AI systems, including neural motion planners. However, the sim-to-real gap limits the transfer of models from simulation to the real world due to the complexity and unpredictability of real-world environments [404]. Digital twins can bridge this gap by providing real-time simulations that closely mirror their physical counterparts [389].

The ability of digital twins to ensure consistency and

synchronization has advanced research in waste management [405] and human-robot collaboration [406]. These models enable real-time monitoring, optimization, and prediction, facilitating a seamless connection between physical and virtual environments [407]. High-fidelity, task-specific digital twins have the potential to improve the safety of neural motion planners by allowing motion planning and execution to be tested in the digital twin before deployment in the real world (Figure 12-(c)). However, the underlying sim-to-real gap within physics engines still limits the reliability of digital twins for providing safety guarantees.

VII. DOMAIN-SPECIFIC CHALLENGES AND POTENTIALS

Deep learning methods hold significant promise to advance robotic manipulator planning in complex, domain-specific applications. We outline the benefits and potential risks of applying neural motion planners for robotic manipulators across various domains.

A. Healthcare

State-of-the-art surgical [408]–[412] and assistive robots [413]–[420] have achieved limited, task-specific autonomy. These systems may reduce operating time and standardize technique, potentially lowering costs and widening access, but robust generalization to complex, variable clinical conditions remains challenging. As shown in Figure 13-(a), most surgical robots still operate in master–slave teleoperation, where a surgeon commands instruments via a console or joysticks [421], [422].

Early surgical path planning methods relied on trajectory optimizers and visual servoing under surgeon supervision, which are brittle to deformable anatomy, occlusions, and nonstationary scenes [423]. Deep learning re-frames planning as closed-loop visuo-motor proposal generation from multi-modal observations (endoscope video, robotic proprioception, and force/tactile feedback) learned via imitation, (inverse) RL, and sequence modeling [424], [425]. However, motion planning for surgical robots is challenging due to the complexity of surgical environments. These environments exhibit high variability across patients, continuous changes during the procedure, a limited field of view (FoV), and partial observability due to occlusions. In addition, strict safety and precision requirements further complicate the development and deployment of motion planning algorithms in surgical settings [426], [427].

Recently, surgical foundation models frame planning as a closed-loop trajectory proposal conditioned on intraoperative perception. Hierarchical Surgical Transformer (SRT-H) is a hierarchical structure developed for autonomous ex vivo cholecystectomy [426]. This framework couples a language-conditioned high-level policy that issues intent instructions with a low-level controller that generates end-effector motions. The high-level policy also provides correction instructions to adjust the surgical process if needed. Experimental results show a 100% success rate across eight unseen surgical scenarios, demonstrating the potential for deploying autonomous surgical systems in clinical settings.

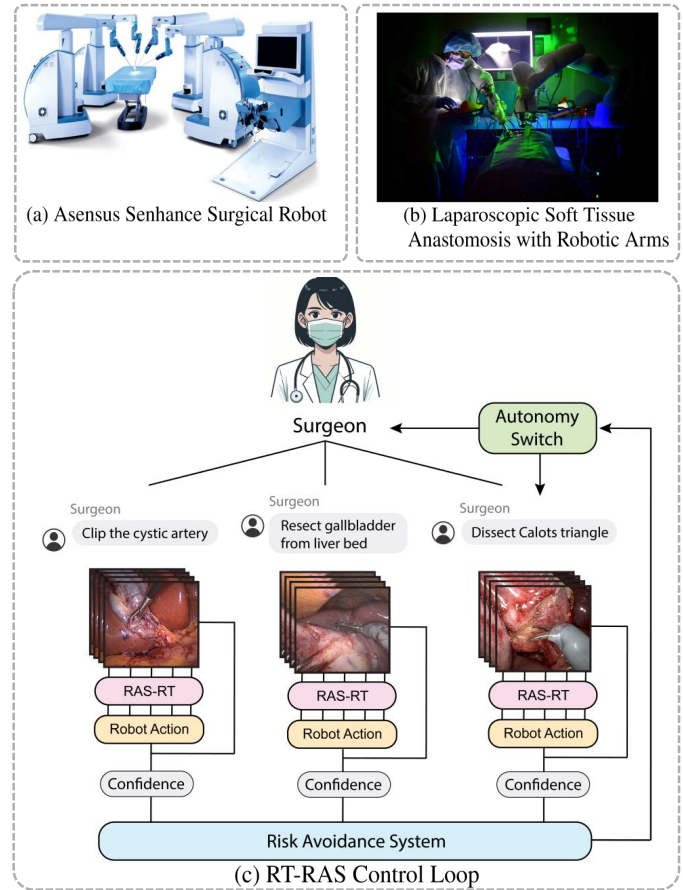


Fig. 13: Robot-assisted surgery. (a) Asensus Senhance surgical robot platform [421]. (b) Autonomous robot for autonomous laparoscopic soft tissue anastomosis [425]. (c) Robot-assisted surgery robot transformer (RT-RAS) control loop [424].

SRT-H robotic system [426] was evaluated in ex vivo settings, and additional considerations and risk mitigation strategies are needed for safe deployment within in vivo surgical environments. Risk avoidance methods such as conservative Q-learning [428] and conformal methods [429] can provide the robot with confidence estimates when performing out-of-distribution tasks. Figure 13-(c) demonstrates an example of incorporating a risk avoidance system for surgical robots. Patient-specific digital twins can facilitate in vivo deployment by validating surgical trajectories on preoperative anatomy and stress-testing failure cases before real-world execution [430].

Robot-assisted surgical frameworks [424], [425] like SRT-H can benefit from fast, collision-aware neural motion planners that propose smooth, collision-free tool paths to achieve smoother task execution and establish fully autonomous surgical robots for end-to-end procedures. Integrating safety filters into the neural motion planner [324] helps ensure safe tool movements, preventing accidental collisions with or damage to vital organs.

Surgical foundation models combined with neural motion planners necessitate large-scale surgical datasets, which are challenging to obtain due to the scarcity and variability of demonstrations. Moreover, they must undergo rigorous safety

certification to comply with stringent medical regulations and clinical protocols before deployment in real-world settings.

Field of view (FoV) limitations represent a fundamental barrier to robotic-assisted surgery (RAS) automation [431], [432]. These limitations impede both large-scale data collection for training generalist neural motion planners and accurate collision checking during deployment. Clinical evidence demonstrates that surgeons operate under suboptimal visual conditions for approximately 40% of minimally invasive procedures, contributing to nearly 20% of surgical complications [433], while impaired vision affects 58% of robotic surgery duration [432]. These FoV constraints stem from multiple sources: trocar diameter restrictions limiting camera dimensions, rapid scene changes from camera movements, lighting fluctuations, instrument occlusion, and organ deformations [427], [434].

Recently, specialized frameworks have been developed for specific interventions with critical FoV challenges. SafeRPlan [435] implements Safe Deep Reinforcement Learning for robotic spine surgery, integrating neural networks pre-trained on preoperative images with uncertainty-aware safety filters to encode anatomical knowledge. For laparoscopic procedures, Iyama *et al.* [436] developed a neural network combining point cloud CNN with DRL for autonomous forceps positioning, directly optimizing tissue surface planarity and visibility to address FoV limitations. In neurosurgery, Segato *et al.* [437] introduced an Inverse Reinforcement Learning framework for steerable needle navigation in keyhole procedures, learning from expert demonstrations to adapt to dynamic tissue deformations in real-time, achieving exceptional precision with sub-millimeter targeting errors and 0.02-second re-planning capabilities while maintaining 100% success rates under challenging deformable conditions.

Recent advances in neural networks have demonstrated significant progress in addressing FoV limitations. Neural radiance fields (NeRF)-based approaches have emerged as particularly promising solutions. Qin *et al.* [438] developed a NeRF-driven network that reconstructs highly realistic endoscopic scenes from multi-view images. To address the impracticality of extensive multi-view image collection during surgery, Neri *et al.* [439] proposed a method enabling NeRF reconstruction from single intraoperative images combined with preoperative data, employing neural style transfer for efficient alignment and training. These approaches have the potential to be integrated into the neural motion planner's pipeline for efficient training and reliable deployment.

B. Re-manufacturing

Re-manufacturing involves full disassembly and reassembly of end-of-life (EOL) products to enable their sustainable recovery [440]–[442]. Human-robot collaborative (HRC) disassembly combines the robot's strength and the human's dexterity to enable flexible and efficient disassembly of EOL products [443]–[445].

HRC disassembly requires dynamic and collision-free collaboration, which relies on efficient human motion prediction and reliable motion planning for the robotic manipulator [446]. As demonstrated in Figure 14, ensuring collision-free

collaboration is challenging due to the proximity of human operators and the complexity of the disassembly environment. Deep learning methods, have the potential to enhance the adaptability and flexibility of HRC disassembly by offering fast inference, ease of implementation, and strong generalization capabilities [447].

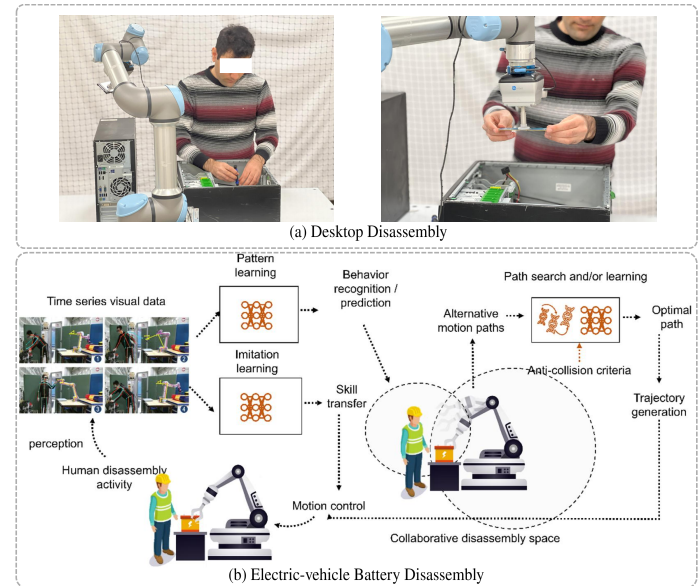


Fig. 14: Human-robot intelligent collaboration for disassembly. (a) Desktop disassembly. (b) Electric-vehicle battery disassembly [447].

Deep learning methods have enhanced the HRC disassembly process by facilitating reliable human motion prediction [448] and efficient motion planning for robotic manipulators [446]. Regarding human motion prediction, sequence modeling network architectures such as recurrent neural networks (RNNs) have been used to capture temporal patterns in human motion for efficient prediction [448]. Regarding reliable motion planning, various deep learning methods have been utilized for collision-free and precise manipulator motion planning [281]. More specifically, constraint-aware neural motion planners, combined with safety filters, have the potential to generate optimal, collision-free, and safe trajectories for robotic manipulators in HRC disassembly settings. These planners are designed to efficiently handle planning constraints [251], which can be deployed to handle disassembly constraints, such as managing fragile or hazardous components, and facilitate fast planning in dynamic environments. Predicted human motion could also be incorporated as an additional constraint to ensure safe collaboration. Integrating safety filters [400] with these planners prevents collision with the human operator and guarantees safety. Validating the disassembly process in a high-fidelity digital twin further ensures safe and efficient planning execution within HRC scenarios [449].

Large-scale re-manufacturing still relies on manual labor, while intelligent disassembly remains limited to laboratory environments [446]. Deploying neural motion planners in HRC disassembly settings involves several challenges. Firstly, large-

scale datasets and extensive training are required. However, EOL products lack standardized designs, and exhibit high structural variability, and have uncertain physical conditions, which complicates large-scale data collection. Moreover, task- and operator-specific safety filters are essential for safe HRC disassembly. However, the complexity of workspaces and variability among human operators make it difficult to design and integrate such safety filters into neural motion planners.

C. Other Domains

Manufacturing: The deployment of robotic manipulators in existing manufacturing environments is predominantly optimized for rigid processes [177], [450]–[452]. Deep learning and neural motion planners have huge potential for flexible manufacturing, which offers greater adaptability for various assembly tasks and products. However, their widespread adoption is hindered by the requirement for large-scale, real-world data collection and the stringent safety-certification procedures necessary for human-robot collaboration. Moreover, the ability to generalize learned motion policies to novel, unseen manufacturing tasks remains an open challenge.

Agriculture: Precision agriculture still relies heavily on manual labor [453]. Deploying robotic manipulators for agriculture automation requires adaptability to environmental variations, coordination under limited connectivity, and the ability for crop-specific harvesting. Deep learning and neural motion planners offer the potential to provide end-to-end policy networks for effective harvesting in cluttered environments. However, guaranteeing the robustness of these methods is challenging due to the less-structured nature of agricultural environments.

Construction: Construction environments are typically unstructured and constantly changing [454]. Deploying robotic manipulators in such environments necessitates real-time re-planning and strict adherence to safety regulations. Neural motion planners can enable robotic manipulators to autonomously adapt to dynamic conditions, navigate cluttered workspaces, avoid obstacles such as scaffolding, and coordinate with other agents such as humans and cranes. However, their limitations include the scarcity of labeled on-site data, the need for real-time domain adaptation as conditions evolve, and the difficulty of certifying safety when heavy loads or human workers are involved.

Warehouse: Warehouse robotics is essential for logistics and working alongside human operators [455]. Current warehouses typically utilize conveyors and sorting machines to handle various packages. Deploying robotic manipulators within warehouses requires high mobility, manipulability, and effective human collaboration. Moreover, they often need to operate in highly constrained, confined spaces. To address these challenges, neural motion planners have the potential to generate real-time, collision-free paths to enable fast pick-and-place operations within these environments. However, these planners need to explicitly consider kinodynamic constraints and smoothness for effective deployment. Ensuring operational

safety and collecting large-scale, domain-specific data are also critical for the widespread deployment of warehouse robotics.

The deployment of neural motion planners in real-world environments remains limited. Most existing studies demonstrate these methods primarily in simulation or controlled laboratory settings, with practical validation and widespread real-world adoption still underdeveloped.

VIII. CONCLUSIONS

Through a comprehensive examination of state of the art, we analyzed how various deep learning architectures have improved classical motion planning algorithms for robotic manipulators. In this examination:

- We delved into classical planning algorithms to identify their core components, and we discussed how various deep learning characteristics, such as fast inference, inductive biases, parallelization, and multi-modal feature encoding capabilities, have improved these components, and provided a systematic map from various deep learning frameworks to specific algorithmic primitives of classical motion planning algorithms.
- We also outlined the essential considerations towards developing generalist neural motion planners capable of end-to-end planning and robust deployment within unstructured real-world environments.

In addition to highlighting the improvements that deep learning methods lend to motion planning algorithms, we also identified challenges and considerations that need to be addressed before these frameworks can be safely deployed within a broad range of unstructured real-world environments. Particularly, we emphasized the need for standardized benchmarks, large-scale planning datasets, explicit handling of safety constraints, generalization to out-of-distribution scenarios, and robustness to planning uncertainties for reliable deployment within unstructured real-world environments. Additionally, we discussed and emphasized how recent large-scale foundation models can be established and leveraged to facilitate reaching this goal.

This review aims to serve as a foundational resource for researchers interested in exploring deep learning applications in motion planning for robotic manipulators.

REFERENCES

- [1] A. Fishman, A. Murali, C. Eppner, B. Peele, B. Boots, and D. Fox, "Motion policy networks," in *Conference on Robot Learning*. PMLR, 2023, pp. 967–977.
- [2] M. Dalal, J. Yang, R. Mendonca, Y. Khaky, R. Salakhutdinov, and D. Pathak, "Neural mp: A generalist neural motion planner," *arXiv preprint arXiv:2409.05864*, 2024.
- [3] J. J. Johnson, A. H. Qureshi, and M. C. Yip, "Learning sampling dictionaries for efficient and generalizable robot motion planning with transformers," *IEEE Robotics and Automation Letters*, 2023.
- [4] T. Lai and F. Ramos, "Plannerflows: Learning motion samplers with normalising flows," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2542–2548.
- [5] J. Ichnowski, Y. Avigal, V. Satish, and K. Goldberg, "Deep learning can accelerate grasp-optimized motion planning," *Science Robotics*, vol. 5, no. 48, p. eabd7710, 2020.
- [6] R. Ni and A. H. Qureshi, "Physics-informed neural motion planning on constraint manifolds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 12 179–12 185.

- [7] C. Acar and K. P. Tee, "Approximating constraint manifolds using generative models for sampling-based constrained motion planning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8451–8457.
- [8] P. Kicki, P. Liu, D. Tateo, H. Bou-Ammar, K. Walas, P. Skrzypczyński, and J. Peters, "Fast kinodynamic planning on the constraint manifold with deep neural networks," *IEEE Transactions on Robotics*, 2023.
- [9] Y. Kim, J. Kim, and D. Park, "Graphdistnet: A graph-based collision-distance estimator for gradient-based trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11118–11125, 2022.
- [10] H.-T. L. Chiang, J. E. Baxter, S. Sugaya, M. R. Yousefi, A. Faust, and L. Tapia, "Fast deep swept volume estimator," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1068–1086, 2021.
- [11] T. Osa, "Motion planning by learning the solution manifold in trajectory optimization," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 281–311, 2022.
- [12] M. Danielczuk, A. Mousavian, C. Eppner, and D. Fox, "Object rearrangement using learned implicit collision functions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6010–6017.
- [13] S. Proia, R. Carli, G. Cavone, and M. Dotoli, "Control techniques for safe, ergonomic, and efficient human-robot collaboration in the digital industry: A survey," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1798–1819, 2021.
- [14] G. P. Moustris, S. C. Hiridis, K. M. Deliparaschos, and K. M. Konstantinidis, "Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature," *The international journal of medical robotics and computer assisted surgery*, vol. 7, no. 4, pp. 375–392, 2011.
- [15] M. G. Tamizi, M. Yaghoubi, and H. Najjaran, "A review of recent trend in motion planning of industrial robots," *International Journal of Intelligent Robotics and Applications*, vol. 7, no. 2, pp. 253–274, 2023.
- [16] H. Lasi, P. Fettke, H. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0. bus inf syst eng 6: 239–242," 2014.
- [17] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-based motion planning: A comparative review," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.
- [18] T. McMahon, A. Sivaramakrishnan, E. Granados, and K. E. Bekris, "A survey on the integration of machine learning with sampling-based motion planning," *Foundations and Trends® in Robotics*, vol. 9, no. 4, p. 266–327, 2022. [Online]. Available: <http://dx.doi.org/10.1561/23000000063>
- [19] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. 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)*. IEEE, 2023, pp. 1916–1923.
- [20] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, "Motion planning networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2118–2124.
- [21] M. J. Bency, A. H. Qureshi, and M. 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)*. IEEE, 2019, pp. 3965–3972.
- [22] M. Song, Y. Kim, and D. Park, "Graph-based 3d collision-distance estimation network with probabilistic graph rewiring," *arXiv preprint arXiv:2310.04044*, 2023.
- [23] F. Noroozi, M. Daneshmand, and P. Fiorini, "Conventional, heuristic and learning-based robot motion planning: Reviewing frameworks of current practical significance," *Machines*, vol. 11, no. 7, p. 722, 2023.
- [24] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021.
- [25] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: a review of recent research," *Advanced Robotics*, vol. 31, no. 16, pp. 821–835, 2017.
- [26] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman *et al.*, "Foundation models in robotics: Applications, challenges, and the future," *arXiv preprint arXiv:2312.07843*, 2023.
- [27] M. Dehghani, J. Djolonga, B. Mustafa, P. Padlewski, J. Heek, J. Gilmer, A. P. Steiner, M. Caron, R. Geirhos, I. Alabdulmohsin *et al.*, "Scaling vision transformers to 22 billion parameters," in *International Conference on Machine Learning*. PMLR, 2023, pp. 7480–7512.
- [28] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.
- [29] A. Iosifidis and A. Tefas, *Deep learning for robot perception and cognition*. Academic Press, 2022.
- [30] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.
- [31] M. Kang, Y. Cho, and S.-E. Yoon, "Rcik: Real-time collision-free inverse kinematics using a collision-cost prediction network," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 610–617, 2021.
- [32] G. P. De Carvalho, T. Sawanobori, and T. Horii, "Data-driven motion planning: A survey on deep neural networks, reinforcement learning, and large language model approaches," *IEEE Access*, 2025.
- [33] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," *arXiv preprint arXiv:2303.04129*, 2023.
- [34] Farber, "Topological complexity of motion planning," *Discrete & Computational Geometry*, vol. 29, pp. 211–221, 2003.
- [35] C. Chamzas, Z. Kingston, C. Quintero-Peña, A. Shrivastava, and L. E. Kavraki, "Learning sampling distributions using local 3d workspace decompositions for motion planning in high dimensions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1283–1289.
- [36] N. D. Ratliff, J. Issac, D. Kappler, S. Birchfield, and D. Fox, "Riemannian motion policies," *arXiv preprint arXiv:1801.02854*, 2018.
- [37] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, and N. Ratliff, "Learning reactive motion policies in multiple task spaces from human demonstrations," in *Conference on Robot Learning*. PMLR, 2020, pp. 1457–1468.
- [38] T. Sandakalum and M. H. Ang Jr, "Motion planning for mobile manipulators—a systematic review," *Machines*, vol. 10, no. 2, p. 97, 2022.
- [39] K. Lynch, *Modern Robotics*. Cambridge University Press, 2017.
- [40] R. K. Malhan, S. Thakar, A. M. Kabir, P. Rajendran, P. M. Bhatt, and S. K. Gupta, "Generation of configuration space trajectories over semi-constrained cartesian paths for robotic manipulators," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 193–205, 2022.
- [41] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1096–1114, 2020.
- [42] J. Urain, A. T. Le, A. Lambert, G. Chaltatzaki, B. Boots, and J. Peters, "Learning implicit priors for motion optimization," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7672–7679.
- [43] L. Zhang, K. Cai, Z. Sun, Z. Bing, C. Wang, L. Figueredo, S. Haddadin, and A. Knoll, "Motion planning for robotics: A review for sampling-based planners," *Biomimetic Intelligence and Robotics*, p. 100207, 2025.
- [44] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.
- [45] R. Bohlin and L. 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)*, vol. 1. IEEE, 2000, pp. 521–528.
- [46] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.
- [47] A. Dobson, A. Krontiris, and K. E. Bekris, "Sparse roadmap spanners," in *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2013, pp. 279–296.
- [48] A. Dobson and K. E. Bekris, "Improving sparse roadmap spanners," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 4106–4111.
- [49] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.
- [50] J. J. Kuffner and S. 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)*, vol. 2. IEEE, 2000, pp. 995–1001.

- [51] O. Salzman and D. Halperin, "Asymptotically near-optimal rrt for fast, high-quality motion planning," *IEEE Transactions on Robotics*, vol. 32, no. 3, pp. 473–483, 2016.
- [52] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.
- [53] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.
- [54] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2951–2957.
- [55] A. Orthey and M. Toussaint, "Rapidly-exploring quotient-space trees: Motion planning using sequential simplifications," in *The International Symposium of Robotics Research*. Springer, 2019, pp. 52–68.
- [56] A. Orthey, S. Akbar, and M. Toussaint, "Multilevel motion planning: A fiber bundle formulation," *arXiv preprint arXiv:2007.09435*, 2020.
- [57] O. Arslan and P. Tsiotras, "The role of vertex consistency in sampling-based algorithms for optimal motion planning," *arXiv preprint arXiv:1204.6453*, 2012.
- [58] M. Otte and E. Frazzoli, "Rrt^x x rrt^x: Real-time motion planning/replanning for environments with unpredictable obstacles," in *Algorithmic foundations of robotics XI: selected contributions of the eleventh international workshop on the algorithmic foundations of robotics*. Springer, 2015, pp. 461–478.
- [59] A. H. Qureshi, S. Mumtaz, K. F. Iqbal, B. Ali, Y. Ayaz, F. Ahmed, M. S. Muhammad, O. Hasan, W. Y. Kim, and M. Ra, "Adaptive potential guided directional-rrt," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2013, pp. 1887–1892.
- [60] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Autonomous Robots*, vol. 40, pp. 1079–1093, 2016.
- [61] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2014, pp. 2997–3004.
- [62] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International journal of robotics research*, vol. 34, no. 7, pp. 883–921, 2015.
- [63] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2072–2078.
- [64] W. Reid, R. Fitch, A. H. Göktoğan, and S. Sukkarieh, "Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover," *Journal of Field Robotics*, vol. 37, no. 5, pp. 786–811, 2020.
- [65] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, "Batch informed trees (bit*): Informed asymptotically optimal anytime search," *The International Journal of Robotics Research*, vol. 39, no. 5, pp. 543–567, 2020.
- [66] M. P. Strub and J. D. Gammell, "Advanced bit (abit): Sampling-based planning with advanced graph-search techniques," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 130–136.
- [67] Strub, Marlin P and Gammell, Jonathan D, "Adaptively informed trees (ait): Fast asymptotically optimal path planning through adaptive heuristics," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3191–3198.
- [68] N. Haghtalab, S. Mackenzie, A. Procaccia, O. Salzman, and S. Srinivasa, "The provable virtue of laziness in motion planning," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 106–113.
- [69] C. Li, H. Ma, P. Xu, J. Wang, and M. Q.-H. Meng, "Biait*: Symmetrical bidirectional optimal path planning with adaptive heuristic," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [70] T. Goto, T. Kosaka, and H. Noborio, "On the heuristics of a* or a algorithm in its and robot path-planning," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 2. IEEE, 2003, pp. 1159–1166.
- [71] A. Pettinger, J. Panthi, F. Alambeigi, and M. Pryor, "Efficient constrained motion planning using direct sampling of screw-constraint manifolds," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [72] Z. Kingston, M. Moll, and L. E. Kavraki, "Exploring implicit spaces for constrained sampling-based planning," *The International Journal of Robotics Research*, vol. 38, no. 10–11, pp. 1151–1178, 2019.
- [73] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.
- [74] B. Kim, T. T. Um, C. Suh, and F. C. Park, "Tangent bundle rrt: A randomized algorithm for constrained motion planning," *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.
- [75] L. Jaillet and J. M. Porta, "Path planning with loop closure constraints using an atlas-based rrt," in *Robotics Research: The 15th International Symposium ISRR*. Springer, 2017, pp. 345–362.
- [76] Z. Kingston, M. Moll, and L. E. Kavraki, "Sampling-based methods for motion planning with constraints," *Annual review of control, robotics, and autonomous systems*, vol. 1, no. 1, pp. 159–185, 2018.
- [77] Y. Pi, X. Liu, Z. Yang, Y. Zhong, T. Huang, H. Pu, and J. Luo, "Omepp: Online multi-population evolutionary path planning for mobile manipulators in dynamic environments," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [78] R. Cheng, K. Shankar, and J. W. Burdick, "Learning an optimal sampling distribution for efficient motion planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7485–7492.
- [79] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 489–494.
- [80] P. C. Chen and Y. K. Hwang, "Sandros: a dynamic graph search algorithm for motion planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 390–403, 1998.
- [81] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," *The international journal of robotics research*, vol. 26, no. 8, pp. 845–863, 2007.
- [82] B. Raveh, A. Enosh, and D. Halperin, "A little more, a lot better: Improving path quality by a path-merging algorithm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 365–371, 2011.
- [83] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K.-i. Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Computer-Aided Design*, vol. 42, no. 4, pp. 350–359, 2010.
- [84] K. Yang, D. Jung, and S. Sukkarieh, "Continuous curvature path-smoothing algorithm using cubic b zier spiral curves for non-holonomic robots," *Advanced Robotics*, vol. 27, no. 4, pp. 247–258, 2013.
- [85] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [86] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 802–807.
- [87] O. Brock and O. Khatib, "Elastic strips: Real-time path modification for mobile manipulation," in *Robotics Research: The Eighth International Symposium*. Springer, 1998, pp. 5–13.
- [88] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International journal of robotics research*, vol. 32, no. 9–10, pp. 1164–1193, 2013.
- [89] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: science and systems*, vol. 9, no. 1. Berlin, Germany, 2013, pp. 1–10.
- [90] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.
- [91] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 9–15.
- [92] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1319–1340, 2018.
- [93] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in

- 2011 *IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.
- [94] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, “Motion planning around obstacles with convex optimization,” *Science robotics*, vol. 8, no. 84, p. ead7843, 2023.
- [95] T. Cohn, M. Petersen, M. Simchowitz, and R. Tedrake, “Non-euclidean motion planning with graphs of geodesically convex sets,” *The International Journal of Robotics Research*, p. 02783649241302419, 2023.
- [96] T. Cohn, S. Shaw, M. Simchowitz, and R. Tedrake, “Constrained bimanual planning with analytic inverse kinematics,” in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6935–6942.
- [97] S. Garg, T. Cohn, and R. Tedrake, “Planning shorter paths in graphs of convex sets by undistorting parametrized configuration spaces,” *arXiv preprint arXiv:2411.18913*, 2024.
- [98] O. Brock and O. Khatib, “Elastic strips: A framework for motion generation in human environments,” *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 1031–1052, 2002.
- [99] Y. Yang and O. Brock, “Elastic roadmaps—motion generation for autonomous mobile manipulation,” *Autonomous Robots*, vol. 28, pp. 113–130, 2010.
- [100] C. W. Warren, “Global path planning using artificial potential fields,” in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 316–317.
- [101] A. D. Dragan, N. D. Ratliff, and S. S. Srinivasa, “Manipulation planning with goal sets using constrained trajectory optimization,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4582–4588.
- [102] K. He, E. Martin, and M. Zucker, “Multigrid chomp with local smoothing,” in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 315–322.
- [103] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, “Space-time functional gradient optimization for motion planning,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6499–6506.
- [104] Z. Marinho, A. Dragan, A. Byravan, B. Boots, S. Srinivasa, and G. Gordon, “Functional gradient motion planning in reproducing kernel hilbert spaces,” *arXiv preprint arXiv:1601.03648*, 2016.
- [105] T. Osa, “Multimodal trajectory optimization for motion planning,” *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 983–1001, 2020.
- [106] A. Lambert and B. Boots, “Entropy regularized motion planning via stein variational inference,” *arXiv preprint arXiv:2107.05146*, 2021.
- [107] K. V. Alwala and M. Mukadam, “Joint sampling and trajectory optimization over graphs for online motion planning,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4700–4707.
- [108] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, “Motion planning as probabilistic inference using gaussian processes and factor graphs,” in *Robotics: Science and Systems*, vol. 12, no. 4, 2016.
- [109] E. Huang, M. Mukadam, Z. Liu, and B. Boots, “Motion planning with graph-based trajectories and gaussian process inference,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5591–5598.
- [110] C. Wang, H.-C. Lin, S. Jin, X. Zhu, L. Sun, and M. Tomizuka, “Bpomp: A bilevel path optimization formulation for motion planning,” in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 1891–1897.
- [111] H. Yu and Y. Chen, “A gaussian variational inference approach to motion planning,” *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2518–2525, 2023.
- [112] L. Barcelos, T. Lai, R. Oliveira, P. Borges, and F. Ramos, “Path signatures for diversity in probabilistic trajectory optimisation,” *The International Journal of Robotics Research*, p. 02783649241233300, 2024.
- [113] T. Power and D. Berenson, “Constrained stein variational trajectory optimization,” *IEEE Transactions on Robotics*, 2024.
- [114] H. Yu and Y. Chen, “Stochastic motion planning as gaussian variational inference: Theory and algorithms,” *arXiv preprint arXiv:2308.14985*, 2023.
- [115] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.
- [116] J. A. Bagnell and J. Schneider, “Covariant policy search,” 2003.
- [117] S. Quinlan, *Real-time modification of collision-free paths*. Stanford University, 1995.
- [118] A. Werner, R. Lampariello, and C. Ott, “Optimization-based generation and experimental validation of optimal walking trajectories for biped robots,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4373–4379.
- [119] J. Meng, Y. Liu, R. Bucknall, W. Guo, and Z. Ji, “Anisotropic gmp2: A fast continuous-time gaussian processes based motion planner for unmanned surface vehicles in environments with ocean currents,” *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3914–3931, 2022.
- [120] M. Toussaint, “Robot trajectory optimization using approximate inference,” in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 1049–1056.
- [121] M. Toussaint and C. Goerick, “A bayesian view on motor control and planning,” in *From Motor Learning to Interaction Learning in Robots*. Springer, 2010, pp. 227–252.
- [122] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.
- [123] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “isam2: Incremental smoothing and mapping using the bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.
- [124] C. Park, J. Pan, and D. Manocha, “Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments,” in *Proceedings of the international conference on automated planning and scheduling*, vol. 22, 2012, pp. 207–215.
- [125] Park, Chonhyon and Pan, Jia and Manocha, Dinesh, “Real-time optimization-based planning in dynamic environments using gpus,” in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 4090–4097.
- [126] J.-J. Kim and J.-J. Lee, “Trajectory optimization with particle swarm optimization for manipulator motion planning,” *IEEE transactions on industrial informatics*, vol. 11, no. 3, pp. 620–631, 2015.
- [127] L. Petrović, J. Peršić, M. Seder, and I. Marković, “Stochastic optimization for trajectory planning with heteroscedastic gaussian processes,” in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–6.
- [128] Petrović, Luka and Peršić, Juraj and Seder, Marija and Marković, Ivan, “Cross-entropy based stochastic optimization of robot trajectories using heteroscedastic continuous-time gaussian processes,” *Robotics and Autonomous Systems*, vol. 133, p. 103618, 2020.
- [129] L. Petrović, I. Marković, and I. Petrović, “Mixtures of gaussian processes for robot motion planning using stochastic trajectory optimization,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 12, pp. 7378–7390, 2022.
- [130] A. T. Le, G. Chalvatzaki, A. Biess, and J. R. Peters, “Accelerating motion planning via optimal transport,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [131] J. Mainprice and D. Berenson, “Human-robot collaborative manipulation planning using early prediction of human motion,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 299–306.
- [132] E. Theodorou, J. Buchli, and S. Schaal, “Reinforcement learning of motor skills in high dimensions: A path integral approach,” in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2397–2403.
- [133] A. Amice, H. Dai, P. Werner, A. Zhang, and R. Tedrake, “Finding and optimizing certified, collision-free regions in configuration space for robot manipulators,” in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2022, pp. 328–348.
- [134] M. Petersen and R. Tedrake, “Growing convex collision-free regions in configuration space using nonlinear programming,” *arXiv preprint arXiv:2303.14737*, 2023.
- [135] H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake, “Certified polyhedral decompositions of collision-free configuration space,” *The International Journal of Robotics Research*, vol. 43, no. 9, pp. 1322–1341, 2024.
- [136] T. Marcucci, J. Umenberger, P. Parrilo, and R. Tedrake, “Shortest paths in graphs of convex sets,” *SIAM Journal on Optimization*, vol. 34, no. 1, pp. 507–532, 2024.
- [137] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, “Guiding trajectory optimization by demonstrated distributions,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 819–826, 2017.
- [138] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos *et al.*,

- "Curobo: Parallelized collision-free minimum-jerk robot motion generation," *arXiv preprint arXiv:2310.17274*, 2023.
- [139] W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8749–8756.
- [140] C. W. Ramsey, Z. Kingston, W. Thomason, and L. E. Kavraki, "Collision-affording point trees: Simd-amenable nearest neighbors for fast collision checking," *arXiv preprint arXiv:2406.02807*, 2024.
- [141] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.
- [142] G. v. d. Bergen, "A fast and robust gjk implementation for collision detection of convex objects," *Journal of graphics tools*, vol. 4, no. 2, pp. 7–25, 1999.
- [143] M. C. Lin, D. Manocha, and Y. J. Kim, "Collision and proximity queries," in *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017, pp. 1029–1056.
- [144] Q. Fan, B. Tao, Z. Gong, X. Zhao, and H. Ding, "Fast global collision detection method based on feature-point-set for robotic machining of large complex components," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 470–481, 2022.
- [145] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-dops," *IEEE transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.
- [146] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtrees: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 171–180.
- [147] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3859–3866.
- [148] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [149] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.
- [150] S. Redon, M. C. Lin, D. Manocha, and Y. J. Kim, "Fast continuous collision detection for articulated models," 2005.
- [151] J. Pan and D. Manocha, "Gpu-based parallel collision detection for real-time motion planning," in *Algorithmic Foundations of Robotics IX: Selected Contributions of the Ninth International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2010, pp. 211–228.
- [152] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "Interactive continuous collision detection between deformable models using connectivity-based culling," in *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, 2008, pp. 25–36.
- [153] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1366–1373.
- [154] Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro, C. Stachniss, and M. Chli, "Voxfield: Non-projective signed distance fields for online planning and 3d reconstruction," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5331–5338.
- [155] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart, "nvblox: Gpu-accelerated incremental signed distance field mapping," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2698–2705.
- [156] A. Gaschler, R. Petrick, T. Kröger, O. Khatib, and A. Knoll, "Robot task and motion planning with sets of convex polyhedra," in *Robotics: Science and Systems (RSS) Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*, 2013.
- [157] H. Täubig, B. Bäuml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 1585–1592.
- [158] M. Campen and L. Kobbelt, "Polygonal boundary evaluation of minkowski sums and swept volumes," in *Computer Graphics Forum*, vol. 29, no. 5. Wiley Online Library, 2010, pp. 1613–1622.
- [159] M. Vergheze, N. Das, Y. Zhi, and M. Yip, "Configuration space decomposition for scalable proxy collision checking in robot planning and control," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3811–3818, 2022.
- [160] Y. Zhi, N. Das, and M. Yip, "Diffco: Autodifferentiable proxy collision detection with multiclass labels for safety-aware trajectory optimization," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2668–2685, 2022.
- [161] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning," *Engineering*, vol. 1, no. 1, pp. 046–057, 2015.
- [162] J. Huh and D. 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)*. IEEE, 2016, pp. 63–69.
- [163] J. Pan and D. Manocha, "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1477–1496, 2016.
- [164] J. Huh, B. Lee, and D. D. Lee, "Adaptive motion planning with high-dimensional mixture models," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3740–3747.
- [165] Y. Han, W. Zhao, J. Pan, Z. Ye, R. Yi, and Y.-J. Liu, "A configuration-space decomposition scheme for learning-based collision checking," *arXiv preprint arXiv:1911.08581*, 2019.
- [166] N. Das and M. C. Yip, "Forward kinematics kernel for improved proxy collision checking," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2349–2356, 2020.
- [167] E. B. Küçüktabak, J. L. Pons, K. M. Lynch, and R. S. Zarrin, "Physical human-robot-human interaction with hierarchical safety constraints," *IEEE Access*, 2024.
- [168] Y. Yamada, Y. Hirasawa, S. Huang, Y. Umetani, and K. Suita, "Human-robot contact in the safeguarding space," *IEEE/ASME transactions on mechatronics*, vol. 2, no. 4, pp. 230–236, 1997.
- [169] S. Takakura, T. Murakami, and K. Ohnishi, "An approach to collision detection and recovery motion in industrial robot," in *15th Annual conference of IEEE industrial electronics society*. IEEE, 1989, pp. 421–426.
- [170] Z. Zhang, K. Qian, B. W. Schuller, and D. Wollherr, "An online robot collision detection and identification scheme by supervised learning and bayesian decision theory," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1144–1156, 2020.
- [171] X. Fan, D. Lee, L. Jackel, R. Howard, D. Lee, and V. Isler, "Enabling low-cost full surface tactile skin for human robot interaction," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1800–1807, 2022.
- [172] S. Golz, C. Osendorfer, and S. Haddadin, "Using tactile sensation for learning contact knowledge: Discriminate collision from physical interaction," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 3788–3794.
- [173] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.
- [174] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023.
- [175] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey. arxiv 2021," *arXiv preprint arXiv:2110.01889*.
- [176] W. Zhang, J. Tanida, K. Itoh, and Y. Ichioka, "Shift-invariant pattern recognition neural network and its optical architecture," in *Proceedings of annual conference of the Japan Society of Applied Physics*, vol. 564. Montreal, CA, 1988.
- [177] Q. Yu, W. Shang, Z. Zhao, S. Cong, and Z. Li, "Robotic grasping of unknown objects using novel multilevel convolutional neural networks: From parallel gripper to dexterous hand," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 4, pp. 1730–1741, 2020.
- [178] B. Lu, X. Yu, J. Lai, K. Huang, K. C. Chan, and H. K. Chu, "A learning approach for suture thread detection with feature enhancement and segmentation for 3-d shape reconstruction," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 2, pp. 858–870, 2019.
- [179] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [180] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, vol. 27, 2014.
- [181] Z. Xu and Y. She, "Leto: Learning constrained visuomotor policy with differentiable trajectory optimization," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [182] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malininowski, A. Tacchetti, D. Raposo, A. Santoro,

- R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks. arxiv 2018," *arXiv preprint arXiv:1806.01261*, 2018.
- [183] Y. Zhang, X. Li, S. Fang, Y. Liu, J. Wang, B. Yuan, and Z. Yi, "Multi-branch multi-scale channel fusion graph convolutional networks with transfer cost for robotic tactile recognition tasks," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [184] M. Ding, Y. Liu, Y. Shi, X. Lan, and N. Zheng, "Dual graph attention networks for multi-view visual manipulation relationship detection and robotic grasping," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [185] Y. Wang, L. Gao, Y. Gao, and X. Li, "A graph guided convolutional neural network for surface defect recognition," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1392–1404, 2022.
- [186] J. Qiao, Y. Lin, J. Bi, H. Yuan, G. Wang, and M. Zhou, "Attention-based spatiotemporal graph fusion convolution networks for water quality prediction," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [187] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [188] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [189] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [190] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15 084–15 097, 2021.
- [191] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs. arxiv," *arXiv preprint arXiv:2012.09699*, 2020.
- [192] C. Zhou, Q. Li, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, "A comprehensive survey on pretrained foundation models: A history from bert to chatgpt," *arXiv preprint arXiv:2302.09419*, 2023.
- [193] D. Shah, B. Osifinski, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504.
- [194] Y. Song, P. Sun, P. Jin, Y. Ren, Y. Zheng, Z. Li, X. Chu, Y. Zhang, T. Li, and J. Gu, "Learning 6-dof fine-grained grasp detection based on part affordance grounding," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [195] Z. Li, J. Liu, Z. Li, Z. Dong, T. Teng, Y. Ou, D. Caldwell, and F. Chen, "Language-guided dexterous functional grasping by llm generated grasp functionality and synergy for humanoid manipulation," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [196] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.
- [197] N. Di Palo, A. Byravan, L. Hasenclever, M. Wulfmeier, N. Heess, and M. Riedmiller, "Towards a unified agent with foundation models," *arXiv preprint arXiv:2307.09668*, 2023.
- [198] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," *arXiv preprint arXiv:2209.07753*, 2022.
- [199] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat, "Flow matching guide and code," *arXiv preprint arXiv:2412.06264*, 2024.
- [200] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7327–7347, 2021.
- [201] Z. Chai, J. Wang, and C. Zhao, "Walk before you can run: Sampling-rate-aware sequential knowledge transfer for multirate process anomaly detection," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [202] Y. Huang and X. Zhao, "Wind farm control via offline reinforcement learning with adversarial training," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [203] Y. Tai, K. Yang, T. Peng, Z. Huang, and Z. Zhang, "Defect image sample generation with diffusion prior for steel surface defect recognition," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [204] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *2018 IEEE international conference on systems, man, and cybernetics (SMC)*. IEEE, 2018, pp. 415–419.
- [205] Z. Pan, W. Yu, X. Yi, A. Khan, F. Yuan, and Y. Zheng, "Recent progress on generative adversarial networks (gans): A survey," *IEEE access*, vol. 7, pp. 36 322–36 333, 2019.
- [206] I. Kobyzev, S. J. Prince, and M. A. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 11, pp. 3964–3979, 2020.
- [207] Y. Song and D. P. Kingma, "How to train your energy-based models," *arXiv preprint arXiv:2101.03288*, 2021.
- [208] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Surveys*, vol. 56, no. 4, pp. 1–39, 2023.
- [209] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.
- [210] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
- [211] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*. PMLR, 2014, pp. 1278–1286.
- [212] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.
- [213] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on international conference on machine learning*, 2011, pp. 833–840.
- [214] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2016.
- [215] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.
- [216] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein auto-encoders," *arXiv preprint arXiv:1711.01558*, 2017.
- [217] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.
- [218] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
- [219] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *Advances in neural information processing systems*, vol. 31, 2018.
- [220] R. v. d. Berg, L. Hasenclever, J. M. Tomczak, and M. Welling, "Sylvester normalizing flows for variational inference," *arXiv preprint arXiv:1803.05649*, 2018.
- [221] N. De Cao, W. Aziz, and I. Titov, "Block neural autoregressive flow," in *Uncertainty in artificial intelligence*. PMLR, 2020, pp. 1263–1273.
- [222] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.
- [223] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting structured data*, vol. 1, no. 0, 2006.
- [224] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [225] H. Lu and H. Shi, "Deep learning for 3d point cloud understanding: a survey," *arXiv preprint arXiv:2009.08920*, 2020.
- [226] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.
- [227] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.
- [228] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 259–16 268.

- [229] S. Lin, Z. Wang, S. Zhang, Y. Ling, and C. Yang, "Deep fusion for multi-modal 6d pose estimation," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 6540–6549, 2023.
- [230] G. Wang, L. Pan, S. Peng, S. Liu, C. Xu, Y. Miao, W. Zhan, M. Tomizuka, M. Pollefeys, and H. Wang, "Nerf in robotics: A survey," *arXiv preprint arXiv:2405.01333*, 2024.
- [231] T. Deng, Y. Wang, H. Xie, H. Wang, R. Guo, J. Wang, D. Wang, and W. Chen, "Neslam: Neural implicit mapping and self-supervised feature tracking with depth completion and denoising," *IEEE Transactions on Automation Science and Engineering*, 2025.
- [232] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield, "Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 606–617.
- [233] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint, "Learning multi-object dynamics with compositional neural radiance fields," in *Conference on robot learning*. PMLR, 2023, pp. 1755–1768.
- [234] M. Adamkiewicz, T. Chen, A. Caccavale, R. Gardner, P. Culbertson, J. Bohg, and M. Schwager, "Vision-only robot navigation in a neural radiance world," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606–4613, 2022.
- [235] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 165–174.
- [236] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.
- [237] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choremanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.
- [238] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mohtakuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhitlinsky, " π_0 : A vision-language-action flow model for general robot control," 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [239] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.
- [240] M. Pándy, D. Lenton, and R. Clark, "Learning to find shortest collision-free paths from images," *CoRR*, 2020.
- [241] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.
- [242] A. H. Qureshi and M. C. Yip, "Deeply informed neural sampling for robot motion planning, in 2018 IEEE," in *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6582–6588.
- [243] M. G. Tamizi, H. Honari, A. Nozdryn-Plotnicki, and H. Najjaran, "End-to-end deep learning-based framework for path planning and collision checking: bin-picking application," *Robotica*, vol. 42, no. 4, pp. 1094–1112, 2024.
- [244] V. Parquet, "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)*. IEEE, 2021, pp. 1538–1543.
- [245] T. Lai, W. Zhi, T. Hermans, and F. Ramos, "Parallelised diffeomorphic sampling-based motion planning," in *Conference on Robot Learning*. PMLR, 2022, pp. 81–90.
- [246] M. Bhardwaj, S. Choudhury, B. Boots, and S. Srinivasa, "Leveraging experience in lazy search," *Autonomous Robots*, vol. 45, no. 7, pp. 979–996, 2021.
- [247] E. Lyu, T. Liu, J. Wang, S. Song, and M. Q.-H. Meng, "Motion planning of manipulator by points-guided sampling network," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 821–831, 2022.
- [248] M. Yu, C. Yu, M. Naddaf-Sh, D. Upadhyay, S. Gao, C. Fan *et al.*, "Efficient motion planning for manipulators with control barrier function-induced neural controller," *arXiv preprint arXiv:2404.01184*, 2024.
- [249] H.-T. L. Chiang, A. Faust, S. Sugaya, and L. Tapia, "Fast swept volume estimation with deep learning," in *Algorithmic Foundations of Robotics XIII: Proceedings of the 13th Workshop on the Algorithmic Foundations of Robotics 13*. Springer, 2020, pp. 52–68.
- [250] S. Sugaya, M. R. Yousefi, A. R. Ferdinand, M. Morales, and L. Tapia, "Multitask and transfer learning of geometric robot motion," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 9071–9078.
- [251] A. H. Qureshi, J. Dong, A. Choe, and M. C. Yip, "Neural manipulation planning on constraint manifolds," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6089–6096, 2020.
- [252] A. H. Qureshi, J. Dong, A. Baig, and M. C. Yip, "Constrained motion planning networks x," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 868–886, 2021.
- [253] G. Sutanto, I. R. Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Conference on Robot Learning*. PMLR, 2021, pp. 2292–2305.
- [254] S. Banerjee, T. Lew, R. Bonalli, A. Alfaadhel, I. A. Alomar, H. M. Shageer, and M. Pavone, "Learning-based warm-starting for fast sequential convex programming and trajectory optimization," in *2020 IEEE Aerospace Conference*. IEEE, 2020, pp. 1–8.
- [255] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.
- [256] H. Liu, H. Wang, X. Guo, and L. Wang, "A reliable learning-based approach for self-collision checking on a redundant manipulator," *IEEE Transactions on Industrial Informatics*, 2025.
- [257] S. Luo and L. Schomaker, "Reinforcement learning in robotic motion planning by combined experience-based planning and self-imitation learning," *Robotics and Autonomous Systems*, vol. 170, p. 104545, 2023.
- [258] A. Krawczyk, J. Marciniak, and D. Belter, "Comparison of machine learning techniques for self-collisions checking of manipulating robots," in *2023 27th International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 2023, pp. 472–477.
- [259] T. Tran, J. Denny, and C. Ekenna, "Predicting sample collision with neural networks," *arXiv preprint arXiv:2006.16868*, 2020.
- [260] G. Guo, J. Goldfeder, A. Ray, T. Dear, and H. Lipson, "Deepcollide: Scalable data-driven high dof configuration space modeling using implicit neural representations," *arXiv preprint arXiv:2305.15376*, 2023.
- [261] D. Rakita, B. Mutlu, and M. Gleicher, "Relaxedik: Real-time synthesis of accurate and feasible robot arm motion," in *Robotics: Science and Systems*, vol. 14. Pittsburgh, PA, 2018, pp. 26–30.
- [262] J. Chase Kew, B. Ichter, M. Bandari, T.-W. E. Lee, and A. Faust, "Neural collision clearance estimator for batched motion planning," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 73–89.
- [263] H. Liu, H. Wang, and X. Guo, "Whole-body self-collision distance detection for a heavy-duty manipulator using neural networks," *IEEE Robotics and Automation Letters*, 2023.
- [264] K. Ota, D. Jha, T. Onishi, A. Kanazaki, Y. Yoshiyasu, Y. Sasaki, T. Mariyama, and D. Nikovski, "Deep reactive planning in dynamic environments," in *Conference on Robot Learning*. PMLR, 2021, pp. 1943–1957.
- [265] R. Ni and A. H. Qureshi, "Ntfields: Neural time fields for physics-informed robot motion planning," *arXiv preprint arXiv:2210.00120*, 2022.
- [266] Ni, Ruiqi and Qureshi, Ahmed H, "Progressive learning for physics-informed neural motion planning," *arXiv preprint arXiv:2306.00616*, 2023.
- [267] R. Ni and A. H. Qureshi, "Physics-informed neural networks for robot motion under constraints," in *RoboNerF: 1st Workshop On Neural Fields In Robotics at ICRA 2024*.
- [268] Y. Liu, R. Ni, and A. H. Qureshi, "Physics-informed neural mapping and motion planning in unknown environments," *arXiv preprint arXiv:2410.09883*, 2024.
- [269] R. Ni, A. H. Qureshi *et al.*, "Physics-informed temporal difference metric learning for robot motion planning," in *The Thirteenth International Conference on Learning Representations*.
- [270] N. Shah and S. Srivastava, "Using deep learning to bootstrap abstractions for hierarchical robot planning," *arXiv preprint arXiv:2202.00907*, 2022.
- [271] I. Patil, B. Rout, and V. Kalaichelvi, "Prediction of bottleneck points for manipulation planning in cluttered environment using a 3d convolutional neural network," in *2019 7th International Conference on Control, Mechatronics and Automation (ICCM)*. IEEE, 2019, pp. 358–364.

- [272] A. Abdi and J. H. Park, "A hybrid ai-based adaptive path planning for intelligent robot arms," *IEEE Access*, 2023.
- [273] C. Chamzas, A. Cullen, A. Shrivastava, and L. E. Kavraki, "Learning to retrieve relevant experiences for motion planning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7233–7240.
- [274] R. Terasawa, Y. Arik, T. Narihira, T. Tsuboi, and K. Nagasaka, "3d-cnn based heuristic guided task-space planner for faster motion planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9548–9554.
- [275] A. Murali, A. Mousavian, C. Eppner, A. Fishman, and D. Fox, "Cabinet: Scaling neural collision detection for object rearrangement with procedural scene generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1866–1874.
- [276] A. Fishman, A. Walsman, M. Bhardwaj, W. Yuan, B. Sundaralingam, B. Boots, and D. Fox, "Avoid everything."
- [277] J. Yang, J. J. Liu, Y. Li, Y. Khaky, K. Shaw, and D. Pathak, "Deep reactive policy: Learning reactive manipulator motion planning for dynamic environments," *arXiv preprint arXiv:2509.06953*, 2025.
- [278] D. Soleymanzadeh, X. Liang, and M. Zheng, "Perfact: Motion policy with llm-powered dataset synthesis and fusion action-chunking transformers," *arXiv preprint arXiv:2512.03444*, 2025.
- [279] K.-C. Ying, P. Pourhejazy, C.-Y. Cheng, and Z.-Y. Cai, "Deep learning-based optimization for motion planning of dual-arm assembly robots," *Computers & Industrial Engineering*, vol. 160, p. 107603, 2021.
- [280] S. Hou, M. Bdiwi, A. Rashid, S. Krusche, and S. Ihlenfeldt, "A data-driven approach for motion planning of industrial robots controlled by high-level motion commands," *Frontiers in Robotics and AI*, vol. 9, p. 1030668, 2023.
- [281] D. Soleymanzadeh, X. Liang, and M. Zheng, "Simpnet: Spatial-informed motion planning network," *IEEE Robotics and Automation Letters*, 2025.
- [282] Liu, Wansong and Eltouny, Kareem and Tian, Sibao and Liang, Xiao and Zheng, Minghui, "Kg-planner: Knowledge-informed graph neural planning for collaborative manipulators," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [283] W. Liu, K. Eltouny, S. Tian, X. Liang, and M. Zheng, "Integrating uncertainty-aware human motion prediction into graph-based manipulator motion planning," *IEEE/ASME Transactions on Mechatronics*, 2024.
- [284] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.
- [285] L. Huang, X. Zang, Y. Gong, and B. Yuan, "Hardware architecture of graph neural network-enabled motion planner," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–7.
- [286] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao, "Learning-based motion planning in dynamic environments using gnns and temporal encoding," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30003–30015, 2022.
- [287] W. Zhang, X. Zang, L. Huang, Y. Sui, J. Yu, Y. Chen, and B. Yuan, "Dyngmp: Graph neural network-based motion planning in unpredictable dynamic environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 858–865.
- [288] J. Kim and F. C. Park, "Pairwisenet: Pairwise collision distance learning for high-dof robot systems," in *Conference on Robot Learning*. PMLR, 2023, pp. 2863–2877.
- [289] C.-M. Hung, S. Zhong, W. Goodwin, O. P. Jones, M. Engelcke, I. Havoutis, and I. Posner, "Reaching through latent space: From joint statistics to path planning in manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5334–5341, 2022.
- [290] J. Yamada, C.-M. Hung, J. Collins, I. Havoutis, and I. Posner, "Leveraging scene embeddings for gradient-based motion planning in latent space," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5674–5680.
- [291] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7087–7094.
- [292] Dastider, Apan and Lin, Mingjie, "Damon: Dynamic amorphous obstacle navigation using topological manifold learning and variational autoencoding," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 251–258.
- [293] —, "Sera: Safe and efficient reactive obstacle avoidance for collaborative robotic planning in unstructured environments," *arXiv preprint arXiv:2203.13821*, 2022.
- [294] A. Dastider and M. Lin, "Reactive whole-body obstacle avoidance for collision-free human-robot interaction with topological manifold learning," *CoRR*, 2022.
- [295] A. Dastider, H. Fang, and M. Lin, "Unified control framework for real-time interception and obstacle avoidance of fast-moving objects with diffusion variational autoencoder," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 13 883–13 890.
- [296] R. Kumar, A. Mandalika, S. Choudhury, and S. Srinivasa, "Lego: Leveraging experience in roadmap generation for sampling-based planning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1488–1495.
- [297] R. K. Jenamani, R. Kumar, P. Mall, and K. Kedia, "Robotic motion planning using learned critical sources and local sampling," *arXiv preprint arXiv:2006.04194*, 2020.
- [298] C. Gaeberth and U. Thomas, "Learning-based adaptive sampling for manipulator motion planning," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 715–721.
- [299] K. Kobashi, C. Wang, Y. Zhao, H.-C. Lin, and M. Tomizuka, "Learning from local experience: Informed sampling distributions for high dimensional motion planning," *arXiv preprint arXiv:2306.09446*, 2023.
- [300] Y. Lu, Y. Ma, D. Hsu, and C. Pan, "Neural randomized planning for whole body robot motion," *arXiv preprint arXiv:2405.11317*, 2024.
- [301] X. Huang, G. Soti, C. Ledermann, B. Hein, and T. Kröger, "Planning with learned subgoals selected by temporal information," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9306–9312.
- [302] J. J. Johnson, A. H. Qureshi, and M. C. Yip, "Zero-shot constrained motion planning transformers using learned sampling dictionaries," *arXiv preprint arXiv:2309.15272*, 2023.
- [303] C. Xia, Y. Zhang, S. A. Coleman, C.-Y. Weng, H. Liu, S. Liu, and I.-M. Chen, "Graph wasserstein autoencoder-based asymptotically optimal motion planning with kinematic constraints for robotic manipulation," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 244–257, 2022.
- [304] C.-K. Ho and C.-T. King, "Lac-rrt: Constrained rapidly-exploring random tree with configuration transfer models for motion planning," *IEEE Access*, 2023.
- [305] S. Park, S. Jeon, and J. Park, "A constrained motion planning method exploiting learned latent space for high-dimensional state and constraint spaces," *IEEE/ASME Transactions on Mechatronics*, 2024.
- [306] Lembono, Teguh Santoso and Pignat, Emmanuel and Jankowski, Julius and Calinon, Sylvain, "Generative adversarial network to learn valid distributions of robot configurations for inverse kinematics and constrained motion planning," *CoRR*, abs/2011.05717, 2020.
- [307] T. S. Lembono, E. Pignat, J. Jankowski, and S. Calinon, "Learning constrained distributions of robot configurations with generative adversarial network," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4233–4240, 2021.
- [308] T. Ando, H. Iino, H. Mori, R. Torishima, K. Takahashi, S. Yamaguchi, D. Okanohara, and T. Ogata, "Learning-based collision-free planning on arbitrary optimization criteria in the latent space through cgans," *Advanced Robotics*, vol. 37, no. 10, pp. 621–633, 2023.
- [309] W. Zhi, I. Akinola, K. Van Wyk, N. D. Ratliff, and F. Ramos, "Global and reactive motion generation with geometric fabric command sequences," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 939–945.
- [310] H. Huang, B. Sundaralingam, A. Mousavian, A. Murali, K. Goldberg, and D. Fox, "Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning," *arXiv preprint arXiv:2410.16727*, 2024.
- [311] J. Carvalho, A. Le, P. Kicki, D. Koert, and J. Peters, "Motion planning diffusion: Learning and adapting robot motion planning with diffusion models," *arXiv preprint arXiv:2412.19948*, 2024.
- [312] H. Li, Q. Feng, Z. Zheng, J. Feng, and A. Knoll, "Language-guided object-centric diffusion policy for collision-aware robotic manipulation," *arXiv preprint arXiv:2407.00451*, 2024.
- [313] M. Nikken, N. Botteghi, W. Roozing, and F. Califano, "Denoising diffusion planner: Learning complex paths from low-quality demonstrations," *arXiv preprint arXiv:2410.21497*, 2024.
- [314] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M2diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes," *arXiv preprint arXiv:2410.11402*, 2024.

- [315] A. Dastider, H. Fang, and M. Lin, "Apex: Ambidextrous dual-arm robotic manipulation using collision-free generative diffusion models," *arXiv preprint arXiv:2404.02284*, 2024.
- [316] A. Li, Z. Ding, A. B. Dieng, and R. Beeson, "Efficient and guaranteed-safe non-convex trajectory optimization with constrained diffusion model," *arXiv preprint arXiv:2403.05571*, 2024.
- [317] Li, Anjian and Ding, Zihan and Dieng, Adji Bouso and Beeson, Ryne, "Constraint-aware diffusion models for trajectory optimization," *arXiv preprint arXiv:2406.00990*, 2024.
- [318] M. Seo, Y. Cho, Y. Sung, P. Stone, Y. Zhu, and B. Kim, "Presto: Fast motion planning using diffusion models based on key-configuration environment representation," *arXiv preprint arXiv:2409.16012*, 2024.
- [319] T. Power, R. Soltani-Zarrin, S. Iba, and D. Berenson, "Sampling constrained trajectories using composable diffusion models," in *IROS 2023 Workshop on Differentiable Probabilistic Robotics: Emerging Perspectives on Robot Learning*, 2023.
- [320] K. Saha, V. Mandadi, J. Reddy, A. Srikanth, A. Agarwal, B. Sen, A. Singh, and M. Krishna, "Edmp: Ensemble-of-costs-guided diffusion for motion planning," *arXiv preprint arXiv:2309.11414*, 2023.
- [321] M. Sharma, A. Fishman, V. Kumar, C. Paxton, and O. Kroemer, "Cascaded diffusion models for neural motion planning," *arXiv preprint arXiv:2505.15157*, 2025.
- [322] Y. Luo, C. Sun, J. B. Tenenbaum, and Y. Du, "Potential based diffusion motion planning," *arXiv preprint arXiv:2407.06169*, 2024.
- [323] K. Nguyen, A. T. Le, T. Pham, M. Huber, J. Peters, and M. N. Vu, "Flowmp: Learning motion fields for robot planning with conditional flow matching," *arXiv preprint arXiv:2503.06135*, 2025.
- [324] X. Dai, Z. Yang, D. Yu, S. Zhang, H. Sadeghian, S. Haddadin, and S. Hirche, "Safe flow matching: Robot motion planning with control barrier functions," *arXiv preprint arXiv:2504.08661*, 2025.
- [325] S. Tian, M. Zheng, and X. Liang, "Warm-starting optimization-based motion planning for robotic manipulators via point cloud-conditioned flow matching," *arXiv preprint arXiv:2510.03460*, 2025.
- [326] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song, "Learning to plan in high dimensions via neural exploration-exploitation trees," in *International Conference on Learning Representations*, 2019.
- [327] L. Zhuang, J. Zhao, Y. Li, Z. Xu, L. Zhao, and H. Liu, "Transformer-enhanced motion planner: Attention-guided sampling for state-specific decision making," *IEEE Robotics and Automation Letters*, 2024.
- [328] Y. Cao, J. Zhou, F. Cao, and F. Shu, "Distformer: A high-accuracy transformer-based collision distance estimator for robotic arms," in *2023 9th International Conference on Control Science and Systems Engineering (ICCSSE)*. IEEE, 2023, pp. 178–184.
- [329] Z. Mandi, S. Jain, and S. Song, "Roco: Dialectic multi-robot collaboration with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 286–299.
- [330] T. Kwon, N. Di Palo, and E. Johns, "Language models as zero-shot trajectory generators," *IEEE Robotics and Automation Letters*, 2024.
- [331] A. Buckner, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, and R. Bonatti, "Reshaping robot trajectories using natural language commands: A study of multi-modal data alignment using transformers," in *2022 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2022, pp. 978–984.
- [332] A. Buckner, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, S. Vemprala, and R. Bonatti, "Latte: Language trajectory transformer," in *2023 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 7287–7294.
- [333] M. Koptev, N. B. Figueroa Fernandez, and A. Billard, "Implicit distance functions: Learning and applications in control," in *International Conference on Robotics and Automation (ICRA 2022)*, 2022.
- [334] M. Koptev, N. Figueroa, and A. Billard, "Neural joint space implicit signed distance functions for reactive robot manipulator control," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 480–487, 2022.
- [335] Koptev, Mikhail and Figueroa, Nadia and Billard, Aude, "Reactive collision-free motion generation in joint space via dynamical systems and sampling-based mpc," *The International Journal of Robotics Research*, p. 02783649241246557, 2024.
- [336] P. Liu, K. Zhang, D. Tateo, S. Jauhari, J. Peters, and G. Chelvatzi, "Regularized deep signed distance fields for reactive motion generation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6673–6680.
- [337] B. Liu, G. Jiang, F. Zhao, and X. Mei, "Collision-free motion generation based on stochastic optimization and composite signed distance field networks of articulated robot," *IEEE Robotics and Automation Letters*, 2023.
- [338] G. Zhao, J. Wu, and Z. Xiong, "Perceptual local collision-free motion generation for manipulators," *IEEE Robotics and Automation Letters*, 2024.
- [339] Y. Chen, X. Gao, K. Yao, L. Niederhauser, Y. Bekiroglu, and A. Billard, "Implicit articulated robot morphology modeling with configuration space neural signed distance functions,"
- [340] C. Quintero-Pena, W. Thomason, Z. Kingston, A. Kyrillidis, and L. E. Kavraki, "Stochastic implicit neural signed distance functions for safe motion planning under sensing uncertainty," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2360–2367.
- [341] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, "Configuration space distance fields for manipulation planning," *arXiv preprint arXiv:2406.01137*, 2024.
- [342] J. Kim and F. C. Park, "Active learning of the collision distance function for high-dof multi-arm robot systems," *Robotica*, pp. 1–15, 2024.
- [343] J. Baxter, M. R. Yousefi, S. Sugaya, M. Morales, and L. Tapia, "Deep prediction of swept volume geometries: Robots and resolutions," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6665–6672.
- [344] M.-H. Lee and J.-S. Liu, "Single swept volume reconstruction by signed distance function learning: A feasibility study based on implicit geometric regularization," *IFAC-PapersOnLine*, vol. 55, no. 15, pp. 142–147, 2022.
- [345] —, "Reliable and accurate implicit neural representation of multiple swept volumes with application to safe human–robot interaction," *SN Computer Science*, vol. 5, no. 3, p. 313, 2024.
- [346] Lee, Ming-Hsiu and Liu, Jing-Sin, "Fast collision detection for robot manipulator path: an approach based on implicit neural representation of multiple swept volumes," in *2023 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*. IEEE, 2023, pp. 1–7.
- [347] J. Michaux, Q. Chen, Y. Kwon, and R. Vasudevan, "Reachability-based trajectory design with neural implicit safety constraints," *arXiv preprint arXiv:2302.07352*, 2023.
- [348] D. Joho, J. Schwin, and K. Safronov, "Neural implicit swept volume models for fast collision detection," *arXiv preprint arXiv:2402.15281*, 2024.
- [349] Y. Kwon, J. Michaux, S. Isaacson, B. Zhang, M. Ejakov, K. A. Skinner, and R. Vasudevan, "Conformalized reachable sets for obstacle avoidance with spheres," *arXiv preprint arXiv:2410.09924*, 2024.
- [350] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
- [351] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational physics*, vol. 378, pp. 686–707, 2019.
- [352] J. D. Smith, K. Azizzadenesheli, and Z. E. Ross, "Eikonet: Solving the eikonal equation with deep neural networks," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 12, pp. 10 685–10 696, 2020.
- [353] S. Wang, B. Li, Y. Chen, and P. Perdikaris, "Piratenets: Physics-informed deep learning with residual adaptive networks," *Journal of Machine Learning Research*, vol. 25, no. 402, pp. 1–51, 2024.
- [354] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, 2022.
- [355] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.
- [356] S. Vemprala, R. Bonatti, A. Buckner, and A. Kapoor, "Chatgpt for robotics: Design principles and model abilities. 2023," *Published by Microsoft*, 2023.
- [357] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," *arXiv preprint arXiv:2306.08647*, 2023.
- [358] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 625–632.
- [359] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

- [360] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* 18. Springer, 2015, pp. 234–241.
- [361] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
- [362] D. Chicco, "Siamese neural networks: An overview," *Artificial neural networks*, pp. 73–94, 2021.
- [363] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.
- [364] M. Khodarahmi and V. Maihami, "A review on kalman filter models," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 727–747, 2023.
- [365] H. Fuchs, Z. M. Kedem, and B. F. Naylor, "On visible surface generation by a priori tree structures," in *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, 1980, pp. 124–133.
- [366] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *Advances in neural information processing systems*, vol. 29, 2016.
- [367] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.
- [368] K. Weiss, T. M. Khoshgoufar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.
- [369] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.
- [370] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.
- [371] S. Huang, Z. Wang, P. Li, B. Jia, T. Liu, Y. Zhu, W. Liang, and S.-C. Zhu, "Diffusion-based generation, optimization, and planning in 3d scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 750–16 761.
- [372] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.
- [373] M. Kleinbort, O. Salzman, and D. Halperin, "Collision detection or nearest-neighbor search? on the computational bottleneck in sampling-based motion planning," in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 624–639.
- [374] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.
- [375] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.
- [376] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
- [377] W. Kool, H. Van Hoof, and M. Welling, "Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3499–3508.
- [378] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.
- [379] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.
- [380] C. J. Geyer, "Practical markov chain monte carlo," *Statistical science*, pp. 473–483, 1992.
- [381] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 1321–1326.
- [382] J. Michaux, A. Li, Q. Chen, C. Chen, B. Zhang, and R. Vasudevan, "Safe planning for articulated robots using reachability-based obstacle avoidance with spheres," *arXiv preprint arXiv:2402.08857*, 2024.
- [383] L. Huber, J.-J. Slotine, and A. Billard, "Fast obstacle avoidance based on real-time sensing," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1375–1382, 2022.
- [384] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "Motionbenchmark: A tool to generate and benchmark motion planning datasets," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, 2021.
- [385] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.
- [386] V. Makovychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.
- [387] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.
- [388] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. Vanderbilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 142–13 153.
- [389] J. Li and S. X. Yang, "Digital twins to embodied artificial intelligence: review and perspective," *Intelligence & Robotics*, vol. 5, no. 1, pp. 202–227, 2025.
- [390] H. Wang, X. Du, J. Li, R. A. Yeh, and G. Shakhnarovich, "Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 12 619–12 629.
- [391] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pigan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.
- [392] F. Xiang, Y. Qin, K. Mo, Y. Xia, H. Zhu, F. Liu, M. Liu, H. Jiang, Y. Yuan, H. Wang *et al.*, "Sapien: A simulated part-based interactive environment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 097–11 107.
- [393] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.
- [394] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control," *John Wiley & Sons*, 2020.
- [395] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Transactions on Robotics*, vol. 25, no. 3, pp. 670–685, 2009.
- [396] S. Guptasarma, M. Strong, H. Zhen, and M. Kennedy III, "J-parse: Jacobian-based projection algorithm for resolving singularities effectively in inverse kinematic control of serial manipulators," *arXiv preprint arXiv:2505.00306*, 2025.
- [397] T. Yoshikawa, "Analysis and control of robot manipulators with redundancy," in *Preprints of 1st Int. Symp. of Robotics Research, Boston, MA, 1983*, 1983, pp. 735–747.
- [398] J. Manavalan and M. Howard, "Learning singularity avoidance," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6849–6854.
- [399] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.
- [400] K.-C. Hsu, V. Rubies-Royo, C. J. Tomlin, and J. F. Fisac, "Safety and liveness guarantees through reach-avoid reinforcement learning," *arXiv preprint arXiv:2112.12288*, 2021.
- [401] A. Robey, L. Lindemann, S. Tu, and N. Matni, "Learning robust hybrid control barrier functions for uncertain systems," *IFAC-PapersOnLine*, vol. 54, no. 5, pp. 1–6, 2021.
- [402] L. Brunke, Y. Zhang, R. Römer, J. Naimier, N. Staykov, S. Zhou, and A. P. Schoellig, "Semantically safe robot manipulation: From semantic scene understanding to motion safeguards," *IEEE Robotics and Automation Letters*, 2025.
- [403] O. Bastani, "Safe reinforcement learning with nonlinear dynamics via model predictive shielding," in *2021 American control conference (ACC)*. IEEE, 2021, pp. 3488–3494.
- [404] Y. Lin, J. Humplik, S. H. Huang, L. Hasenclever, F. Romano, S. Saliceti, D. Zheng, J. E. Chen, C. Barros, A. Collister *et al.*, "Proc4gem: Foundation models for physical agency through procedural generation," *arXiv preprint arXiv:2503.08593*, 2025.
- [405] S. Su, C. Yu, Y. Jiang, and R. Y. Zhong, "Digital twin-enabled building demolition waste trading: A demonstrative case," *IEEE Transactions on Automation Science and Engineering*, 2025.

- [406] X. Li, B. He, Z. Wang, Y. Zhou, G. Li, and X. Li, "Toward cognitive digital twin system of human-robot collaboration manipulation," *IEEE Transactions on Automation Science and Engineering*, 2024.
- [407] V. Villani, M. Picone, M. Mamei, and L. Sabatini, "A digital twin driven human-centric ecosystem for industry 5.0," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 11 291–11 303, 2024.
- [408] M. Haiderbhai, R. Gondokaryono, A. Wu, and L. A. Kahrs, "Sim2real rope cutting with a surgical robot using vision-based reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 4354–4365, 2024.
- [409] M. Hwang, J. Ichnowski, B. Thananjeyan, D. Seita, S. Paradis, D. Fer, T. Low, and K. Goldberg, "Automating surgical peg transfer: Calibration with deep learning can exceed speed, accuracy, and consistency of humans," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 909–922, 2022.
- [410] H. Su, A. Mariani, S. E. Ovrur, A. Mencias, G. Ferrigno, and E. De Momi, "Toward teaching by demonstration for robot-assisted minimally invasive surgery," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 484–494, 2021.
- [411] H. Su, K.-W. Kwok, K. Cleary, I. Iordachita, M. C. Cavusoglu, J. P. Desai, and G. S. Fischer, "State of the art and future opportunities in mri-guided robot-assisted surgery and interventions," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 968–992, 2022.
- [412] Y. Liu, X. Wu, Y. Sang, C. Zhao, Y. Wang, B. Shi, and Y. Fan, "Evolution of surgical robot systems enhanced by artificial intelligence: a review," *Advanced Intelligent Systems*, vol. 6, no. 5, p. 2300268, 2024.
- [413] Ł. Kidziński, B. Yang, J. L. Hicks, A. Rajagopal, S. L. Delp, and M. H. Schwartz, "Deep neural networks enable quantitative movement analysis using single-camera videos," *Nature communications*, vol. 11, no. 1, p. 4054, 2020.
- [414] J. Bessler, G. B. Prange-Lasander, L. Schaaake, J. F. Saenz, C. Bidard, I. Fassi, M. Valori, A. B. Lassen, and J. H. Buurke, "Safety assessment of rehabilitation robots: A review identifying safety skills and current knowledge gaps," *Frontiers in Robotics and AI*, vol. 8, p. 602878, 2021.
- [415] S. Massardi, D. Rodriguez-Cianca, D. Pinto-Fernandez, J. C. Moreno, M. Lancini, and D. Torricelli, "Characterization and evaluation of human-exoskeleton interaction dynamics: A review," *Sensors*, vol. 22, no. 11, p. 3993, 2022.
- [416] T. H. Huang, S. Zhang, S. Yu, M. K. MacLean, J. Zhu, A. D. Lallo, and H. Su, "Modeling and stiffness-based continuous torque control of lightweight quasi-direct-drive knee exoskeletons for versatile walking assistance," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1442–1459, 2022.
- [417] S. Monteiro, J. Figueiredo, P. Fonseca, J. P. Vilas-Boas, and C. P. Santos, "Human-in-the-loop optimization of knee exoskeleton assistance for minimizing user's metabolic and muscular effort," *Sensors*, vol. 24, no. 11, p. 3305, 2024.
- [418] S. Luo, M. Jiang, S. Zhang, J. Zhu, S. Yu, I. Dominguez Silva, T. Wang, E. Rouse, B. Zhou, H. Yuk, X. Zhou, and H. Su, "Experiment-free exoskeleton assistance via learning in simulation," *Nature*, vol. 630, no. 8016, pp. 353–359, 2024.
- [419] V. Firouzi, A. Seyfarth, S. Song, O. von Stryk, and M. Ahmad Sharbafi, "Biomechanical models in the lower-limb exoskeletons development: A review," *Journal of NeuroEngineering and Rehabilitation*, vol. 22, no. 1, p. 12, 2025.
- [420] T.-H. Huang, S. Zhang, S. Yu, M. K. MacLean, J. Zhu, A. Di Lallo, C. Jiao, T. C. Bulea, M. Zheng, and H. Su, "Modeling and stiffness-based continuous torque control of lightweight quasi-direct-drive knee exoskeletons for versatile walking assistance," *IEEE Transactions on Robotics*, vol. 38, no. 3, pp. 1442–1459, 2022.
- [421] A. Sarin, S. Samreen, J. M. Moffett, E. Inga-Zapata, F. Bianco, N. A. Alkhamisi, J. D. Owen, N. Shahi, J. C. DeLong, D. Stefanidis *et al.*, "Upcoming multi-visceral robotic surgery systems: a sages review," *Surgical Endoscopy*, vol. 38, no. 12, pp. 6987–7010, 2024.
- [422] N. T. Kantu, W. Gao, N. Srinivasan, G. D. Buckner, and H. Su, "Portable and versatile catheter robot for image-guided cardiovascular interventions," *IEEE/ASME Transactions on Mechatronics*, 2025.
- [423] T. S. Perry, "Profile: Veebot drawing blood faster and more safely than a human can," pp. 23–23, 2013.
- [424] S. Schmidgall, J. W. Kim, A. Kuntz, A. E. Ghazi, and A. Krieger, "General-purpose foundation models for increased autonomy in robot-assisted surgery," *Nature Machine Intelligence*, vol. 6, no. 11, pp. 1275–1283, 2024.
- [425] H. Saeidi, J. D. Opfermann, M. Kam, S. Wei, S. Léonard, M. H. Hsieh, J. U. Kang, and A. Krieger, "Autonomous robotic laparoscopic surgery for intestinal anastomosis," *Science robotics*, vol. 7, no. 62, p. eabj2908, 2022.
- [426] J. W. Kim, J.-T. Chen, P. Hansen, L. X. Shi, A. Goldenberg, S. Schmidgall, P. M. Scheikl, A. Deguet, B. M. White, D. R. Tsai *et al.*, "Srt-h: A hierarchical framework for autonomous surgery via language-conditioned imitation learning," *Science Robotics*, vol. 10, no. 104, p. eadt5254, 2025.
- [427] H. Su, K.-W. Kwok, K. Cleary, I. Iordachita, M. C. Cavusoglu, J. P. Desai, and G. S. Fischer, "State of the art and future opportunities in mri-guided robot-assisted surgery and interventions," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 968–992, 2022.
- [428] Y. Chebotar, Q. Vuong, K. Hausman, F. Xia, Y. Lu, A. Irpan, A. Kumar, T. Yu, A. Herzog, K. Pertsch *et al.*, "Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions," in *Conference on Robot Learning*. PMLR, 2023, pp. 3909–3928.
- [429] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley *et al.*, "Robots that ask for help: Uncertainty alignment for large language model planners," *arXiv preprint arXiv:2307.01928*, 2023.
- [430] H. Su, G. Li, and G. S. Fischer, "Sensors, actuators, and robots for MRI-guided surgery and interventions," in *The Encyclopedia of Medical Robotics: Image-Guided Surgical Procedures and Interventions*. Hackensack, NJ: World Scientific, 2019, vol. 3, pp. 201–231.
- [431] Z. E. Gketsis, D. Tzagkas, P. V. Hatzilias, and M. E. Zervakis, "Laparoscopic image analysis for robotic arm guidance," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2006, pp. 148–151.
- [432] N. Venkatayogi, M. Parker, J. Uecker, A. A. Laviana, A. Cohen, S.-H. Belbina, S. Gereta, N. Ancha, S. Ravi, C. Idelson *et al.*, "Impaired robotic surgical visualization: archaic issues in a modern operating room," *Journal of Robotic Surgery*, vol. 17, no. 6, pp. 2875–2880, 2023.
- [433] J. Dhingra, N. Beinart, A. Ahmed, M. Patel, A. Ameerah, M. Srinivasan, C. R. Idelson, and J. M. Uecker, "Clear vision, clear savings: Enhancing efficiency in minimally invasive surgery," *JSLS: Journal of the Society of Laparoscopic & Robotic Surgeons*, vol. 29, no. 3, pp. e2025–00023, 2025.
- [434] L. R. Kennedy-Metz, P. Mascagni, A. Torralba, R. D. Dias, P. Perona, J. A. Shah, N. Padoy, and M. A. Zenati, "Computer vision in the operating room: Opportunities and caveats," *IEEE transactions on medical robotics and bionics*, vol. 3, no. 1, pp. 2–10, 2020.
- [435] Y. Ao, H. Esfandiari, F. Carrillo, C. J. Laux, Y. As, R. Li, K. Van Assche, A. Davoodi, N. A. Cavalcanti, M. Farshad *et al.*, "Saferplan: Safe deep reinforcement learning for intraoperative planning of pedicle screw placement," *Medical Image Analysis*, vol. 99, p. 103345, 2025.
- [436] Y. Iyama, Y. Takahashi, J. Chen, T. Noda, K. Hara, E. Kobayashi, I. Sakuma, and N. Tomii, "Autonomous countertraction for secure field of view in laparoscopic surgery using deep reinforcement learning," *International Journal of Computer Assisted Radiology and Surgery*, vol. 20, no. 4, pp. 625–633, 2025.
- [437] A. Segato, M. Di Marzo, S. Zucchelli, S. Galvan, R. Secoli, and E. De Momi, "Inverse reinforcement learning intra-operative path planning for steerable needle," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 6, pp. 1995–2005, 2021.
- [438] Z. Qin, K. Qian, S. Liang, Q. Zheng, J. Peng, and Y. Tai, "Neural radiance fields-based multi-view endoscopic scene reconstruction for surgical simulation," *International Journal of Computer Assisted Radiology and Surgery*, vol. 19, no. 5, pp. 951–960, 2024.
- [439] A. Neri, M. Fehrentz, V. Penza, L. S. Mattos, and N. Haouchine, "Surgical neural radiance fields from one image," *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–5, 2025.
- [440] K. Wang, L. Gao, X. Li, and P. Li, "Energy-efficient robotic parallel disassembly sequence planning for end-of-life products," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 2, pp. 1277–1285, 2021.
- [441] R. Li, D. T. Pham, J. Huang, Y. Tan, M. Qu, Y. Wang, M. Kerin, K. Jiang, S. Su, C. Ji *et al.*, "Unfastening of hexagonal headed screws by a collaborative robot," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1455–1468, 2020.
- [442] Y. Feng, Y. Gao, G. Tian, Z. Li, H. Hu, and H. Zheng, "Flexible process planning and end-of-life decision-making for product recovery optimization based on hybrid disassembly," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 1, pp. 311–326, 2018.
- [443] Y. Peng, W. Li, Y. Zhou, and D. T. Pham, "Dynamic disassembly planning of end-of-life products for human-robot collaboration enabled by multi-agent deep reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, 2025.

- [444] X. Guo, C. Fan, M. Zhou, S. Liu, J. Wang, S. Qin, and Y. Tang, "Human-robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [445] S. Qin, C. Xiang, J. Wang, S. Liu, X. Guo, and L. Qi, "Multiple product hybrid disassembly line balancing problem with human-robot collaboration," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 712–723, 2024.
- [446] M.-L. Lee, X. Liang, B. Hu, G. Onel, S. Behdad, and M. Zheng, "A review of prospects and opportunities in disassembly with human-robot collaboration," *Journal of Manufacturing Science and Engineering*, vol. 146, no. 2, p. 020902, 2024.
- [447] K. Meng, G. Xu, X. Peng, K. Youcef-Toumi, and J. Li, "Intelligent disassembly of electric-vehicle batteries: a forward-looking overview," *Resources, Conservation and Recycling*, vol. 182, p. 106207, 2022.
- [448] J. Zhang, H. Liu, Q. Chang, L. Wang, and R. X. Gao, "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly," *CIRP annals*, vol. 69, no. 1, pp. 9–12, 2020.
- [449] N. Nikolakis, V. Maratos, and S. Makris, "A cyber physical system (cps) approach for safe human-robot collaboration in a shared workplace," *Robotics and Computer-Integrated Manufacturing*, vol. 56, pp. 233–243, 2019.
- [450] C. Zhong, J. Li, Z. Sun, T. Li, Y. Guo, D. C. Jeong, H. Su, and S. Liu, "Real-time acoustic holography with physics-based deep learning for robotic manipulation," *IEEE Transactions on Automation Science and Engineering*, 2023.
- [451] J. Li, H. Shi, and K.-S. Hwang, "Using goal-conditioned reinforcement learning with deep imitation to control robot arm in flexible flat cable assembly task," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 4, pp. 6217–6228, 2023.
- [452] S. Yang, S. Song, S. Chu, R. Song, J. Cheng, Y. Li, and W. Zhang, "Heuristics integrated deep reinforcement learning for online 3d bin packing," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 1, pp. 939–950, 2023.
- [453] M. Wakchaure, B. Patle, and A. Mahindrakar, "Application of ai techniques and robotics in agriculture: A review," *Artificial Intelligence in the Life Sciences*, vol. 3, p. 100057, 2023.
- [454] V. Zada and K. Belda, "Structure design and solution of kinematics of robot manipulator for 3d concrete printing," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3723–3734, 2022.
- [455] R. Prakash, L. Behera, S. Mohan, and S. Jagannathan, "Dual-loop optimal control of a robot manipulator and its application in warehouse automation," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 262–279, 2020.

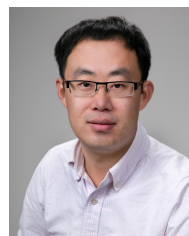


Davood Soleymanzadeh received the B.S. and M.S. degrees from Amirkabir University of Technology, Tehran, Iran, in 2016 and 2019, respectively. He is currently a Ph.D. student in the Department of Mechanical Engineering at Texas A&M University, College Station, TX, USA. His research interests lie at the intersection of automation and robotics, with a focus on motion planning and robotic manipulation for intelligent disassembly.

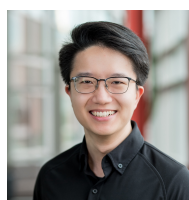


Ivan Lopez-Sanchez received the B.E. degree in aerospace engineering from the Autonomous University of Baja California (UABC), Tijuana, Mexico, in 2016. He earned the M.S. and D.Sc. degrees (Cum Laude) in digital systems with a specialization in control systems from the Digital Technology Research and Development Center of the National Polytechnic Institute (IPN-CITEDI), Tijuana, Mexico, in 2019 and 2023, respectively. From 2023 to 2025, he served as a Postdoctoral Research Scholar with the Biomechatronics and Intelligent Robotics

Lab at North Carolina State University and New York University. His research interests include the design and control of unmanned aerial vehicles, under-actuated systems, and adaptive neural network-based control for aerospace, biomechatronic, and electromechanical applications. Dr. Lopez-Sanchez was the recipient of the "Presea Lázaro Cárdenas 2023," the highest honor bestowed by the IPN for exceptional achievements in scientific research. He is a member of the National System of Researchers (SNII) in Mexico at the



Hao Su (Senior Member, IEEE) received the B.S. degree in automation from the Harbin Institute of Technology, Harbin, China, in 2006, the M.S. degree in mechanical engineering from the State University of New York University, Buffalo, NY, USA, in 2008, and the Ph.D. degree in mechanical engineering from Worcester Polytechnic Institute, Worcester, MA, USA, in 2013. He is currently an Associate Professor with the Tandon School of Engineering, New York University (NYU), New York, NY, USA. Prof. Su is a recipient of the National Science Foundation CAREER Award and Switzer Distinguished Fellowship of the U.S. Department of Health and Human Services. He is a Technical Editor of ASME/IEEE TRANSACTION ON MECHATRONICS, and an Associate Editor for IEEE ROBOTICS AND AUTOMATION MAGAZINE, IEEE International Conference on Robotics and Automation, and IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). He is keynote speaker of IROS. Dr. Su is reviewer of NIH Bioengineering, Technology and Surgical Sciences Study Section (BTSS) and NIH Musculoskeletal Rehabilitation Sciences Study Section (MRS).



Yunzhu Li is an Assistant Professor of Computer Science at Columbia University. Before joining Columbia, he was an Assistant Professor at UIUC CS and spent time as a Postdoc at Stanford, collaborating with Fei-Fei Li and Jiajun Wu. Yunzhu earned his PhD from MIT under the guidance of Antonio Torralba and Russ Tedrake. His work has been recognized with the Best Paper Award at ICRA, the Best Systems Paper Award, and as a Finalist for the Best Paper Award at CoRL. He is also a recipient of the AAAI New Faculty Highlights, the

Sony Faculty Innovation Award, the Amazon Research Award, the Adobe Research Fellowship, and the First Place Ernst A. Guillemin Master's Thesis Award in AI and Decision Making at MIT. His research has been published in top journals and conferences, including Nature and Science, and featured by major media outlets.



Xiao Liang received the B.S. degree from Hunan University, Changsha, China, in 2010, and the M.S. and Ph.D. degrees from the University of California at Berkeley, Berkeley, CA, USA, in 2011 and 2016, respectively, all in civil engineering. He is currently an Assistant Professor of the Zachry Department of Civil & Environmental Engineering, Texas A&M University, TX, USA. Before that, he was an Assistant Professor of the Department of Civil, Structural and Environmental Engineering, University at Buffalo, NY, USA. His research interests include health

monitoring and autonomous inspection of infrastructure systems through advanced data analytics, model-based and machine learning.



Minghui Zheng received the B.S. degree in engineering mechanics in 2008 and M.S. degree in control science and engineering in 2011, both from Beihang University, China, and the Ph.D. degree in mechanical engineering from University of California, Berkeley, CA, USA, in 2017. She is currently an Associate Professor of the J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University, TX, USA. Before that, she was an Associate Professor at the University at Buffalo, NY, USA. Her research interest lies in learning, planning,

and control with applications to several areas that are of vital importance to manufacturing and robotics. Dr. Zheng was the recipient of the NSF CAREER Award in 2021.