2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

TopAY: Efficient Trajectory Planning for Differential Drive Mobile Manipulators via Topological Paths Search and Arc Length-Yaw Parameterization

Long Xu[†,1,2], Choilam Wong[†,2], Mengke Zhang[1,2], Junxiao Lin[1,2], Jialiang Hou[1,2] and Fei Gao[1,2]

***Abstract*— Differential drive mobile manipulators combine the mobility of wheeled bases with the manipulation capability of multi-joint arms, enabling versatile applications but posing considerable challenges for trajectory planning due to their high-dimensional state space and nonholonomic constraints. This paper introduces TopAY, an optimization-based planning framework designed for efficient and safe trajectory generation for differential drive mobile manipulators. The framework employs a hierarchical initial value acquisition strategy, including topological paths search for the base and parallel sampling for the manipulator. A polynomial trajectory representation with arc length–yaw parameterization is also proposed to reduce optimization complexity while preserving dynamic feasibility. Extensive simulation and real-world experiments validate that TopAY achieves higher planning efficiency and success rates than state-of-the-art method in dense and complex scenarios. The source code is released at https://github.com/TopAY-Planner/TopAY.**

I. Introduction

Differential drive mobile manipulator (DDMoMa), comprising multi-joint manipulator(s) mounted on a differential drive base (DDB), integrates rich manipulation ability of manipulators and mobility of wheeled robots. This synergistic combination enables versatile application in diverse domains, including industrial manufacturing, healthcare, agriculture, etc [1]. For DDMoMa to perform complex tasks, autonomous navigation and obstacle avoidance are fundamental.

Prevailing methods [2]–[4] rely on numerical optimization to obtain safe, dynamically feasible, and optimal trajectories under certain criteria. The initial values for optimization are commonly acquired by a combination of sampling- and search-based methods [5], [6]. Despite demonstrating capability of real-time planning, their efficiency degrade significantly in complex scenarios, due to the difficulty of handling nonholonomic constraints of the DDB and high-dimensional state space of DDMoMa.

Due to the exponential growth of space volume with dimensionality, the combined state space of a DDB and manipulator(s) is significantly larger than their individual state spaces. This elevates the complexity of initial value acquisition and increases number of optimization variables for trajectory optimization. Critically, the presence of manipulator(s) necessitates 3D collision checking, rendering the simplified 2D planning methods common to indoor wheeled robots inadequate. Compounding this, the mobility of the base demands collision checking to be performed over a larger physical space.

Intuitively, the set of feasible paths for the DDB component within a DDMoMa is a subset of the feasible paths for a standalone DDB. Leveraging this, we propose a hierarchical initial value acquisition algorithm, introducing topological paths search for the DDB. Conditioned on the DDB paths found, parallelized state sampling for the manipulator is then conducted. In a decoupled manner, the computational burden associated with high dimensionality is largely mitigated, improving the efficiency and success rate for planning in

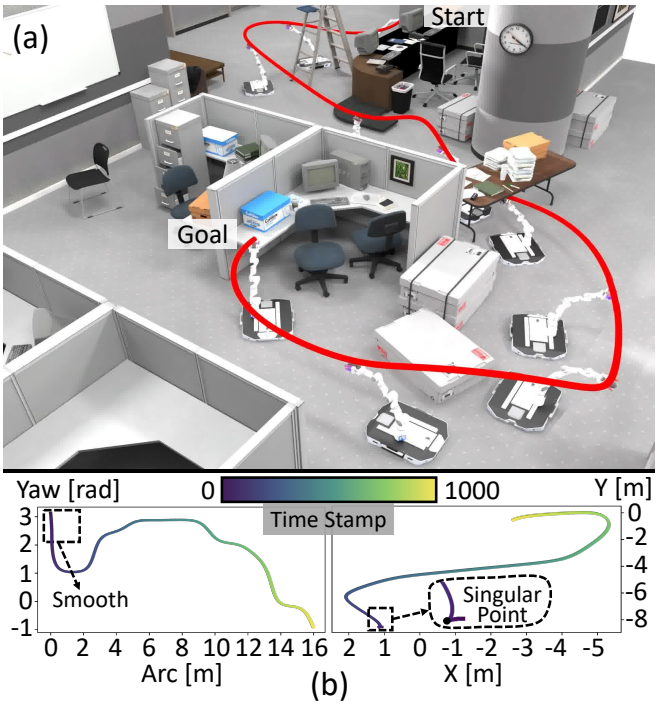

Fig. 1: A differential drive mobile manipulator delivers a mouse from the reception to a workstation in an office using the proposed planner. The red curve in Figure (a) represents the trajectory of the gripper. Figure (b) shows the motion curves of the base in arc length-yaw and Cartesian space.

This work was supported by the National Key RD Program of China under grant No. 2023YFB4706600, the Zhejiang Provincial Science and Technology Plan Project under Grant No. 2024C01170 and the National Natural Science Foundation of China under Grant No. 62322314.

[†]Indicates equal contribution.

[1]State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China. *Corresponding author: Jialiang Hou, Fei Gao.*

[2]Huzhou Institute of Zhejiang University, Huzhou 313000, China.
E-mail: {gaolon, jlhou25, fgaoaa}@zju.edu.cn

complex scenarios.

For handling the high dimension of optimization variables due to the high-dimensional state space, we propose a novel trajectory representation for DDMoMa inspired by polynomial trajectory parameterization. This parameterization has recently proven highly effective in robot motion planning, owing to its intrinsic smoothness and its substantially reduced dimensionality compared with finite-element discretizations [7]–[9]. To address nonholonomic constraints of the DDB, we introduce arc length-yaw parameterization, where arc length at time $t = \tau$ is defined as the signed distance from time $t = 0$ to $t = \tau$ along the trajectory, as shown in Fig. 1(b). This representation demonstrates higher efficiency than the state-of-the-art (SoTA) method [2] that uses differential flatness [10] in Cartesian space.

Integrating these components, we present TopAY, an efficient optimization-based trajectory planning framework for DDMoMa that integrates hierarchical initial value acquisition algorithm and proposed trajectory representation. We also implement parallel trajectory optimization to further improve efficiency. Comprehensive simulation and real-world experiments demonstrate the effectiveness and efficiency of our approach. The contributions of the paper are:

- We propose an efficient hierarchical path acquisition method for DDMoMa by introducing topological path search for the base and parallelized path sampling for the manipulator.
- We propose a novel polynomial-based trajectory representation for DDMoMa that handles the nonholonomic kinematics of the differential drive base with arc length-yaw parameterization.
- Integrating the above two modules, we present an efficient optimization-based trajectory planning framework for DDMoMa.

II. Related Works

A. Sampling-based Path Planning

Sampling-based methods [5] approximate the configuration space by sampling discrete waypoints uniformly or via heuristics-guided strategies [11]. They incrementally construct graphs where nodes represent sampled waypoints and edges denote collision-free paths. Feasible paths can then be derived by querying the graphs. This kind of method typically provides two key theoretical guarantees: probabilistic completeness and asymptotic optimality. However, the performance severely degrades in high-DoF systems, such as mobile manipulators [2]. The exponential growth of sampling space volume with dimensionality results in excessively long planning time in challenging scenarios, limiting the practicality.

One approach to improve efficiency involves employing alternative sampling strategies informed by prior knowledge of the workspace. Since only a subset of sampled waypoints contribute to the final path, it is natural to prioritize sampling in regions where these waypoints are more likely to be found with respect to the workspace and the planning task, i.e. using biased sampling distributions conditioned on the workspace [12]–[15]. An advantage of this approach is its ability to preserve both guarantees of completeness and optimality by combining uniform distribution with the biased distribution. The challenge lies in balancing the cost and frequency of sampling or iteration, as obtaining biased samples often requires more time.

The alternative approach [2], [3] mitigates the performance degradation induced by high DoF of the mobile manipulator by decomposing the manipulator and base motions, one conditioned on the other. Although this strategy sacrifices completeness, it greatly improves the efficiency of planning for DDMoMa and achieves real-time performance.

Recent work has also employed diffusion models to sample directly in the trajectory space [16], thereby bypassing the iterative graph-construction process. Its primary limitations include slow training and inference, as well as a strong dependence on the design of the objective function.

B. Optimization-based Trajectory Generation

When higher-order robot states, such as acceleration and angular acceleration, need to be considered to obtain better solutions, sampling-based methods need to search in a larger space, whereupon the problem of combinatorial explosion is magnified, severely affecting the efficiency. Optimization-based methods tackle this limitation by formulating motion planning as an optimization problem to generate smooth, optimal, and dynamically feasible trajectories. In practice, sampling-based path planning is often utilized as the front-end to initialize the back-end trajectory optimization.

The high DoF in DDMoMa renders optimization computationally expensive and poses a significant challenge to real-time planning. To ensure tractability, one strategy is receding horizon planning [3], [17]. For example, RAMPAGE [3] formulates motion generation as an integrated planning and control problem, planning over a receding horizon to substantially reduce computational cost. However, its inability to account for long-term consequences could lead to myopic behavior, such as getting trapped in dead ends.

To enable real-time global planning, REMANI [2] leverages the MINCO trajectory class [7] to reduce the number of optimization variables, greatly accelerating the trajectory planning for DDMoMa. However, the optimization process is affected by numerical issues arising from singularities in differential flatness they use for trajectory parameterization. Moreover, since a greedy strategy is adopted in path search, it fails to provide diverse initialization sets for trajectory optimization, harming its overall efficiency in complex scenarios.

Our method is also optimization-based. Unlike REMANI [2], we represent DDB trajectories in the arc length-yaw space to avoid singularities from differential flatness. We further apply topological path search for DDMoMa path sampling, which diversifies initializations and improves success rates in complex scenarios. Moreover, inspired by cuRobo's use of CUDA [18] for efficient manipulator trajectory generation and inverse kinematics, we parallelize both the manipulator and trajectory optimization modules to boost efficiency.

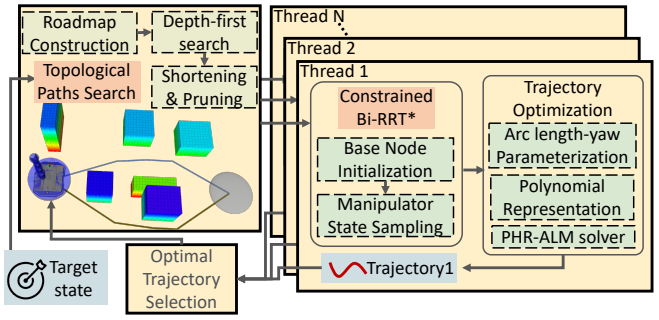

Fig. 2: Planning framework.

III. Planning Framework

Fig. 2 illustrates our planning framework. When the robot receives the target $SE(3)$ pose of the end effector, it performs inverse kinematics and topological paths search to obtain different 2D paths for the DDB. The planner then initializes the sequence of the base states in $SE(2)$ for these paths and samples the manipulator states to obtain whole-body paths of the robot in parallel, which are subsequently handed over to the optimizer to obtain dynamically feasible trajectories.

A. Hierarchical Path Acquisition

Hierarchical path acquisition for DDMoMa mainly consists of topological paths search and constrained bi-directional informed RRT*, as shown in the orange-marked parts in Fig. 2. The pipeline is shown in Algorithm 1. The idea of topological paths has been used in many works [19]–[22] for motion planning. In this paper, we adopt the *uniform visibility deformation* (UVD) class [20] to achieve efficient topological paths search in 2D environments (Lines 1-4). Specifically, a visibility probabilistic roadmap (PRM) [22] is first constructed. A depth-first search (DFS) is then applied to this graph to discover a diverse set of paths. These paths are shortened in parallel by checking the visibility between nodes. Further categorising them into different UVD classes, sorting and filtering them based on path length, we obtain the final topological paths $\mathcal{P}_b$.

Using the set interval Δl, each topological path in $\mathcal{P}_b$ will be discretized into a sequence of states in $SE(2)$ (i.e. $\boldsymbol{P}_b$), which is then used as constraints of path sampling for the manipulator (Lines 6-32). Here, we combine RRT-connect [23] and iRRT* [11] for sampling, node expansion, rewiring, tree switching, connection attempts, and merging. In this work, $\boldsymbol{P}_b$ imposes the following constraints on the algorithm: 1) The base states of the nodes from sampling or in trees $\mathcal{T}_a, \mathcal{T}_b$ are restricted to discrete sequence points $\boldsymbol{P}_b$. 2) When searching for neighbouring nodes (`Nearest`) and expanding nodes (`Steer`), the base state is selected from the adjacent $SE(2)$ states in the corresponding sequence.

B. Trajectory Representation

Let the $SE(2)$ state trajectory of the DDB be $[x(t), y(t), \theta(t)]^{\mathrm{T}}$. In work [2], Wu et al. use differential flatness to handle the nonholonomic kinematics of DDB, where $\dot{x}(t) = v(t)\cos\theta(t)$, $\dot{y}(t) = v(t)\sin\theta(t)$, $\dot{\theta}(t) = \omega(t)$.

Algorithm 1: Hierarchical Path Acquisition for differential drive mobile manipulator

Input: grid map $\mathcal{M}$, start and target state $\boldsymbol{s}_0, \boldsymbol{s}_f \in SE(2) \times \mathbb{R}^{\mathrm{N}}$, constants $\Delta l, t_{\mathrm{smax}}$.
Output: paths of DDMoMa $\mathcal{P}$

1 $\mathcal{G} \leftarrow \texttt{CreatePRM}(\boldsymbol{s}_0, \boldsymbol{s}_f, \mathcal{M})$;
2 $\mathcal{P}_b^{\mathrm{raw}} \leftarrow \texttt{DFS}(\boldsymbol{s}_0, \boldsymbol{s}_f, \mathcal{G})$;
3 $\mathcal{P}_b^{\mathrm{sc}} \leftarrow \texttt{ParallelShortCut}(\mathcal{P}_b^{\mathrm{raw}}, \mathcal{M})$;
4 $\mathcal{P}_b \leftarrow \texttt{PruneUVD}(\mathcal{P}_b^{\mathrm{sc}}, \mathcal{M})$; $\mathcal{P} \leftarrow \varnothing$;
5 **Parallel for** ***each*** $\boldsymbol{P}_r \in \mathcal{P}_b$ **do**
6 $\quad \boldsymbol{P}_b \leftarrow \texttt{DiscreteSE2}(\boldsymbol{P}_r, \Delta l)$;
7 $\quad \mathcal{T}_a.\texttt{init}(\boldsymbol{s}_0)$; $\mathcal{T}_b.\texttt{init}(\boldsymbol{s}_f)$;
8 $\quad \boldsymbol{n}_a, \boldsymbol{n}_b \leftarrow \texttt{NULL}$; $c_{max} \leftarrow +\infty$;
9 $\quad$ **while** $\neg\ \texttt{TimeOut}(t_{smax})$ **do**
10 $\quad\quad \texttt{TrySwap}(\mathcal{T}_a, \mathcal{T}_b)$; $\boldsymbol{n}_r \leftarrow \texttt{Sample}(\boldsymbol{P}_b, c_{max})$;
11 $\quad\quad \boldsymbol{n}_n \leftarrow \texttt{Nearest}(\mathcal{T}_a, \boldsymbol{n}_r)$;
12 $\quad\quad \boldsymbol{n}_s \leftarrow \texttt{Steer}(\boldsymbol{n}_n, \boldsymbol{n}_r)$;
13 $\quad\quad$ **if** $\texttt{IsInvalid}(\boldsymbol{n}_s)$ **then continue** ;
14 $\quad\quad$ **if** $\boldsymbol{n}_s \in \mathcal{T}_b$ **then**
15 $\quad\quad\quad$ // try updating c_{max}, i.e. `TryUpdateCost`
16 $\quad\quad\quad c = \texttt{Cost}(\boldsymbol{n}_n) + \texttt{Cost}(\boldsymbol{n}_s) + \texttt{Heu}(\boldsymbol{n}_n, \boldsymbol{n}_s)$;
17 $\quad\quad\quad$ **if** $c_{max} > c$ **then**
18 $\quad\quad\quad\quad c_{max} \leftarrow c$; $\boldsymbol{n}_a \leftarrow \boldsymbol{n}_n$; $\boldsymbol{n}_b \leftarrow \boldsymbol{n}_s$;
19 $\quad\quad\quad\quad$ **continue**;
20 $\quad\quad$ **if** $\texttt{IsExpanded}(\boldsymbol{n}_s) \vee (\texttt{Cost}(\boldsymbol{n}_s) > \texttt{Cost}(\boldsymbol{n}_n) + \texttt{Heu}(\boldsymbol{n}_n, \boldsymbol{n}_s))$ **then**
21 $\quad\quad\quad \texttt{Link}(\boldsymbol{n}_n, \boldsymbol{n}_s)$; $\texttt{Rewire}(\mathcal{T}_a, \boldsymbol{n}_s)$;
22 $\quad\quad\quad \boldsymbol{n}_{on} \leftarrow \texttt{Nearest}(\mathcal{T}_b, \boldsymbol{n}_s)$;
23 $\quad\quad\quad \boldsymbol{n}_{os} \leftarrow \texttt{Steer}(\boldsymbol{n}_{on}, \boldsymbol{n}_s)$;
24 $\quad\quad\quad$ **if** $\texttt{IsInvalid}(\boldsymbol{n}_{os})$ **then continue** ;
25 $\quad\quad\quad$ **if** $\boldsymbol{n}_{os} \in \mathcal{T}_a$ **then**
26 $\quad\quad\quad\quad \texttt{TryUpdateCost}(\boldsymbol{n}_{on}, \boldsymbol{n}_{os}, c_{max})$;
27 $\quad\quad\quad$ **if** $\texttt{IsExpanded}(\boldsymbol{n}_{os}) \vee (\texttt{Cost}(\boldsymbol{n}_{os}) > \texttt{Cost}(\boldsymbol{n}_{on}) + \texttt{Heu}(\boldsymbol{n}_{on}, \boldsymbol{n}_{os}))$ **then**
28 $\quad\quad\quad\quad \texttt{Link}(\boldsymbol{n}_{on}, \boldsymbol{n}_{os})$; $\texttt{Rewire}(\mathcal{T}_b, \boldsymbol{n}_{os})$;
29 $\quad\quad\quad\quad \texttt{TryConnect}(\boldsymbol{n}_{os}, \mathcal{T}_a)$;
30 $\quad$ **if** $\texttt{IsValid}(\boldsymbol{n}_a) \wedge \texttt{IsValid}(\boldsymbol{n}_b)$ **then**
31 $\quad\quad \texttt{MergeTree}(\mathcal{T}_a, \mathcal{T}_b, \boldsymbol{n}_a, \boldsymbol{n}_b)$;
32 $\quad\quad \mathcal{P}.\texttt{Push}(\texttt{GetPath}(\mathcal{T}_a, \mathcal{T}_b))$;
33 **return** $\mathcal{P}$.

They adopt $x(t)$ and $y(t)$ in Cartesian space to represent the DDB trajectory, explicitly deriving the velocity $v(t) = \eta\sqrt{\dot{x}^2 + \dot{y}^2}$, yaw $\theta(t) = \mathrm{atan2}(\eta\dot{y}, \eta\dot{x})$, and angular velocity $\omega(t) = (\dot{x}\ddot{y} - \dot{y}\ddot{x})/(\dot{x}^2 + \dot{y}^2)$ of the DDB, achieving efficient trajectory planning for DDMoMa, where $\eta = 1 \vee -1$ denotes the DDB is moving forward or backward.

However, when the DDB is stationary or is spinning in place, i.e., when $v(t) = 0$, the function atan2 in the formula for yaw will be undefined, and the denominator in the formula for angular velocity will be zero, which is known as the singularity of differential flatness [9]. This singularity causes numerical instabilities that prevent the satisfaction of the DDB's angular velocity and angular acceleration constraints. To alleviate this, existing methods impose workarounds like a

minimum velocity constraints [10] or dense constraint points near singular points [2], which inadvertently reduce planning efficiency. Moreover, such parameterization method is unable to represent spinning-place-maneuvers, where $\dot{x}(t) = \dot{y}(t) = 0$, resulting in undefined yaw and angular velocity.

To address this problem, in this work, we introduce a novel trajectory representation based on arc length $s(t)$ and yaw angle $\theta(t)$, as shown in Fig. 1(b). In which, the position is given by integration of the kinematics equations:

$$x(t) = \int_0^t \dot{s}(\tau)\cos\theta(\tau)d\tau + x_0, \tag{1}$$

$$y(t) = \int_0^t \dot{s}(\tau)\sin\theta(\tau)d\tau + y_0, \tag{2}$$

where $[x_0, y_0]^{\mathrm{T}}$ is the initial position of the DDB. Following the optimality condition for multi-stage control effort minimization problem [7], we use quintic piecewise polynomials with four times continuously differentiable at the segmented points to represent the trajectory of DDMoMa for minimizing jerk [18]. Each piece of trajectory is denoted as:

$$s_j(t) = \boldsymbol{c}_{s_j}^{\mathrm{T}}\gamma(t) \qquad t \in [0, T_j], \tag{3}$$

$$\theta_j(t) = \boldsymbol{c}_{\theta_j}^{\mathrm{T}}\gamma(t) \qquad t \in [0, T_j], \tag{4}$$

$$q_{kj}(t) = \boldsymbol{c}_{q_{kj}}^{\mathrm{T}}\gamma(t) \qquad t \in [0, T_j], \tag{5}$$

where $q_{kj}(t)$ is the trajectory of the manipulator. $k \in \mathbb{Z} \cap [1, \mathrm{N}]$ is the index of the joints, $j \in \mathbb{Z} \cap [1, \mathrm{M}]$ is the index of piecewise polynomial. T_j is the duration of a piece of the trajectory. $c_* \in \mathbb{R}^6, * = \{s_j, \theta_j, q_{kj}\}$ is the coefficient of polynomial. $\gamma(t) = [1, t, t^2, ..., t^5]^{\mathrm{T}}$ is the natural base. In this work, we employ Simpson's rule to calculate the integrals (1) and (2) numerically.

C. Trajectory Optimization

In this paper, we formulate the trajectory optimization problem for DDMoMa as:

$$\min_{\boldsymbol{c},\boldsymbol{e},\boldsymbol{T}} f(\boldsymbol{c},\boldsymbol{e},\boldsymbol{T}) = \int_0^{\sum_{j=1}^{\mathrm{M}} T_j} \boldsymbol{j}(t)^{\mathrm{T}}\boldsymbol{W}\boldsymbol{j}(t)dt + \rho\|\boldsymbol{T}\|_1 \tag{6}$$

$$s.t. \quad \mathbf{FK}(\boldsymbol{s}_f) = \boldsymbol{p}_e, \tag{7}$$

$$\boldsymbol{M}(\boldsymbol{T})\boldsymbol{c} = \boldsymbol{b}(\boldsymbol{P},\boldsymbol{e}), \quad \boldsymbol{T} > \boldsymbol{0}, \tag{8}$$

$$|v_{\max}\omega(t) \pm \omega_{\max}v(t)| \le v_{\max}\omega_{\max}, \tag{9}$$

$$a^2(t) \le a_{\max}^2, \quad \beta^2(t) \le \beta_{\max}^2, \tag{10}$$

$$(\boldsymbol{q}\circ\boldsymbol{q})(t) \le \boldsymbol{q}_{\max 2}, \quad (\dot{\boldsymbol{q}}\circ\dot{\boldsymbol{q}})(t) \le \dot{\boldsymbol{q}}_{\max 2}, \tag{11}$$

$$(\ddot{\boldsymbol{q}}\circ\ddot{\boldsymbol{q}})(t) \le \ddot{\boldsymbol{q}}_{\max 2}, \tag{12}$$

$$\mathbf{SDF}(\mathbf{ColliPts}(\boldsymbol{c},\boldsymbol{T})) \ge \boldsymbol{r}_{thr}, \tag{13}$$

$$\mathbf{SelfColli}(\boldsymbol{c},\boldsymbol{T}) \ge \boldsymbol{0}, \tag{14}$$

where $\boldsymbol{c} = [\boldsymbol{c}_1^{\mathrm{T}}, \boldsymbol{c}_2^{\mathrm{T}}, ..., \boldsymbol{c}_{\mathrm{M}}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6\mathrm{M}\times(\mathrm{N}+2)}$, $\boldsymbol{c}_k = [\boldsymbol{c}_{k1}, \boldsymbol{c}_{k2}, ..., \boldsymbol{c}_{k(\mathrm{N}+2)}] \in \mathbb{R}^{6\times(\mathrm{N}+2)}, k = 1, 2, ..., \mathrm{M}$ are coefficient matrices. For other optimization variables, $\boldsymbol{s}_f = [x_f, y_f, \theta_f, \boldsymbol{q}_f]^{\mathrm{T}}$, $\boldsymbol{T} = [T_1, T_2, .., T_{\mathrm{M}}]^{\mathrm{T}}$ denote the end state of the robot and time durations of the trajectory, respectively, where $\boldsymbol{q}_f = [q_{1f}, q_{2f}, ..., q_{\mathrm{N}f}]$. x_f, y_f are calculated using $\boldsymbol{e} = [s_f, \theta_f, \boldsymbol{q}_f]^{\mathrm{T}}$ and $\boldsymbol{c}, \boldsymbol{T}$. All equations and inequalities are taken element-wise.

In the objective function, $\boldsymbol{j}(t) = [s^{(3)}(t), \theta^{(3)}(t), \boldsymbol{q}^{(3)}(t)]^{\mathrm{T}}$ denotes the jerk of the trajectory. $\boldsymbol{W} \in \mathbb{R}^{(2+\mathrm{N})\times(2+\mathrm{N})}$ is a diagonal positive matrix representing the weights. $\rho > 0$ is a constant to give the trajectory some aggressiveness.

Eq.(7) denotes the end-effector pose constraints for the robot end state. $\mathbf{FK} : SE(2) \times \mathbb{R}^{\mathrm{N}} \mapsto SE(3)$ is the forward kinematics function. Conditions (8) denotes the combinations of the continuity constraints mentioned in last subsection and boundary conditions of the trajectory:

$$[s(0), \dot{s}(0), \ddot{s}(0)] = [0, v_0, a_0], \tag{15}$$

$$[\theta(0), \dot{\theta}(0), \ddot{\theta}(0)] = [\theta_0, \omega_0, \beta_0], \tag{16}$$

$$[\boldsymbol{q}(0), \dot{\boldsymbol{q}}(0), \ddot{\boldsymbol{q}}(0)] = [\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \ddot{\boldsymbol{q}}_0], \tag{17}$$

$$[s(T_f), \dot{s}(T_f), \ddot{s}(T_f)] = [s_f, 0, 0], \tag{18}$$

$$[\theta(T_f), \dot{\theta}(T_f), \ddot{\theta}(T_f)] = [\theta_f, 0, 0], \tag{19}$$

$$[\boldsymbol{q}(T_f), \dot{\boldsymbol{q}}(T_f), \ddot{\boldsymbol{q}}(T_f)] = [\boldsymbol{q}_f, \boldsymbol{0}, \boldsymbol{0}]. \tag{20}$$

$\boldsymbol{P} \in \mathbb{R}^{(\mathrm{N}+2)\times(\mathrm{M}-1)}$ is the segment points matrix. $T_f = \|\boldsymbol{T}\|_1$ is the total duration of the trajectory.

Conditions (9)$\sim$(12) are dynamic feasibility constraints, including coupled linear velocity $v(t) = \dot{s}(t)$ and angular velocity $\omega(t) = \dot{\theta}(t)$ constraints [8], limitation of linear acceleration $a(t) = \dot{v}(t)$, angular acceleration $\beta(t) = \dot{\omega}(t)$, joint angles, joint angular velocities and joint angular accelerations, where $v_{\max}, \omega_{\max}, a_{\max}, \beta_{\max}, \boldsymbol{q}_{\max 2} = \boldsymbol{q}_{\max} \circ \boldsymbol{q}_{\max}, \dot{\boldsymbol{q}}_{\max 2} = \dot{\boldsymbol{q}}_{\max} \circ \dot{\boldsymbol{q}}_{\max}, \ddot{\boldsymbol{q}}_{\max 2} = \ddot{\boldsymbol{q}}_{\max} \circ \ddot{\boldsymbol{q}}_{\max}$ are constants or constant vectors. The operator $\circ : \mathbb{R}^{\mathrm{N}} \times \mathbb{R}^{\mathrm{N}} \mapsto \mathbb{R}^{\mathrm{N}}$ denotes the Hadamard product.

Condition (13) denotes constraints related to obstacle avoidance. In this paper, we construct a Signed Distance Field (SDF) to represent the environment. In SDF, the value at each state of the space is the distance to the edge of the nearest obstacle, which is negative inside the obstacles. As shown by the robot collision model in Fig. 3, we approximate the collision geometry using a set of cylinders and spheres. The function $\mathbf{ColliPts}(\cdot)$ maps the state points to sets of collision detection points, i.e., the centers of the cylinder and the spheres that constitute the collision model shown in Fig.3. The function $\mathbf{SDF}(\cdot)$ maps the set of points to the corresponding SDF values, where the SDF for the DDB is constructed with a 2D grid map. $\boldsymbol{r}_{thr}$ is a constant vector, which includes the radii of the cylinder and spheres. Condition (14) imposes self-collision avoidance constraints on the robot's configuration. Since our collision model consists of cylinder and spheres, the function $\mathbf{SelfColli}(\cdot)$ calculates the distance vector representing the distance between each pair of them, corresponding to each state point of the trajectory.

D. Problem Solving

We address Conditions (8) using technique from existing work [7], converting the optimization variables $\{\boldsymbol{c}, \boldsymbol{e}, \boldsymbol{T}\}$ to $\{\boldsymbol{P}, \boldsymbol{e}, \boldsymbol{\tau}\}$, where for each element T_j in $\boldsymbol{T}$ and τ_j in $\boldsymbol{\tau}$,

$$\tau_j = \mathrm{L}_{c2}(T_j) = \begin{cases} 1 - \sqrt{2T_j^{-1} - 1} & 0 < T_j \le 1 \\ \sqrt{2T_j - 1} - 1 & T_j > 1 \end{cases}. \tag{21}$$

To guarantee that constraint $(\boldsymbol{q} \circ \boldsymbol{q})(t) \leq \boldsymbol{q}_{\max 2}$ is satisfied when $t = T_j, j \in \mathbb{Z} \cap [1, \mathrm{M}]$ and at $\boldsymbol{q}_f$, we use the following inverse sigmoid-like function to convert $\boldsymbol{q}_f$ and each column of the submatrix $\boldsymbol{Q} \in \mathbb{R}^{\mathrm{N} \times (\mathrm{M}-1)}$ of $\boldsymbol{P}$ with respect to the joint angles of the manipulator to $\mathbf{q}_f$ and $\boldsymbol{\mathcal{Q}}_f$, respectively:

$$\mathfrak{q}(q) = \mathrm{L}_{c2} \left(\frac{q_{\max} + q}{q_{\max} - q} \right). \tag{22}$$

To deal with continuous polynomial trajectories, we discretize each piece of the duration T_j as K time stamps $\tilde{t}_l = (l/\mathrm{K}) \cdot T_j, l = 0, 1, ..., K-1$, imposing the constraints on these time stamps. To efficiently solve this optimization problem, we convert all inequality constraints into penalty terms and add them to the objective function as discrete integrals. Specifically, let the objective function be $g(\text{traj})$, and one constraint function be $\mathcal{C}(\text{traj})$, then the new objective function is

$$g(\text{traj}) + \rho_c \sum_{j=1}^{\mathrm{M}} \sum_{l=0}^{K-1} \frac{T_j}{\mathrm{K}} \mathcal{L}(\max\{0, \mathcal{C}[\text{traj}_j(\frac{lT_j}{\mathrm{K}})]\}), \tag{23}$$

where $\mathcal{L}$ denotes a function that smooths the constraint function $\mathcal{C}$ into a C^2 function, traj_j is the j-th polynomial trajectory. ρ_c denotes the penalty weight. This technique has been demonstrated to be practically effective in numerous studies [2], [7], [8], [10].

We employ Powell-Hestenes-Rockafellar Augmented Lagrangian method [24] (PHR-ALM) to handle the equation constraint (7). It combines Lagrange multipliers with a quadratic penalty to enforce constraints while maintaining good convergence properties. In this work, the efficient quasi-Newton optimizer L-BFGS [25] is chosen as the inner loop iteration algorithm for PHR-ALM.

IV. Results

A. Implementation Details

To validate the performance of our pipeline in real-world applications, we deploy it on a DDMoMa, as shown in the top right of Fig. 3. All computations are performed by an onboard computer Intel NUC 11 Phantom Canyon. The software system of the robot primarily consists of three modules, perception, planning, and control, as shown in the green box area in Fig. 3. In the perception module, we utilize FAST-LIO2 [26] for localization, and ROG-Map [27] for maintenance and live-updating of the 3D grid map and SDFs.

In the planning and control module, we use Boost[1] to launch and manage multithreading. We employ an early termination strategy that waits an additional period after the first successful outcome result is returned by any thread. The shortest optimized trajectory from all successful threads is then selected. The selected one will be fed into a controller based on model predictive control to acquire control commands consisting of velocity, angular velocity of the DDB, and joints velocity of the manipulator. In order to enable adaptability to dynamic environment, a replanning procedure is invoked upon detection of obstacles colliding with currently tracking trajectory.

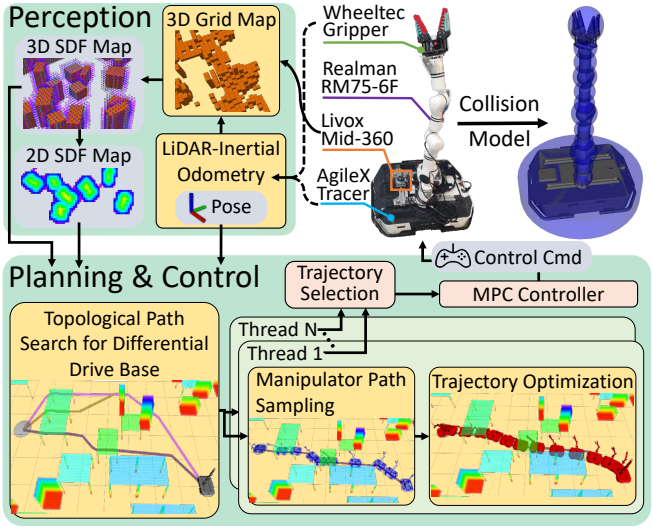

Fig. 3: Hardware settings and software system of the DDMoMa used in real-world experiments.

We also build two different simulation environments based on ROS[2] to carry out comparative experiments in different scenarios. All simulations are run on Ubuntu 20.04 with an Intel i7-12700 CPU.

B. Real-world Experiments

For our real-world experiments, we tasked the DDMoMa with classical pick-and-place operation in three different indoor environments, as shown in Fig. 4. For these experiments, the DDMoMa's task requires it to navigate among obstacles and involves reaching a designated area, grasping an object, delivering it to another area, and finally returning to its initial position.

In each experiment, the robot has no prior knowledge of the environment, apart from the initial position and drop-off location of the object, and thus relies on onboard sensor and real-time replanning to adapt to dynamic environments. As shown in Case 1, at $t = 16.2s$ and $t = 23.0s$, the robot performs two replannings upon discovery of obstacles in its original trajectory. Ultimately, the robot successfully finds a collision-free path and traverses under the beam-like obstacle.

In case 3, we modify the environment in the process of its operation. A beam-like obstacle is placed on top of two cubic obstacles, forming a bridge-like structure that is clearly visible at $t = 40.2s$. The robot exhibited adaptability to dynamic environment by successfully finding a trajectory to avoid newly placed obstacle in real-time by bending its manipulator.

This series of experiments demonstrates the effectiveness and real-time performance of our algorithm. More demonstrations with static scenarios can be found in the attached

[1] https://www.boost.org/

[2] https://www.ros.org/

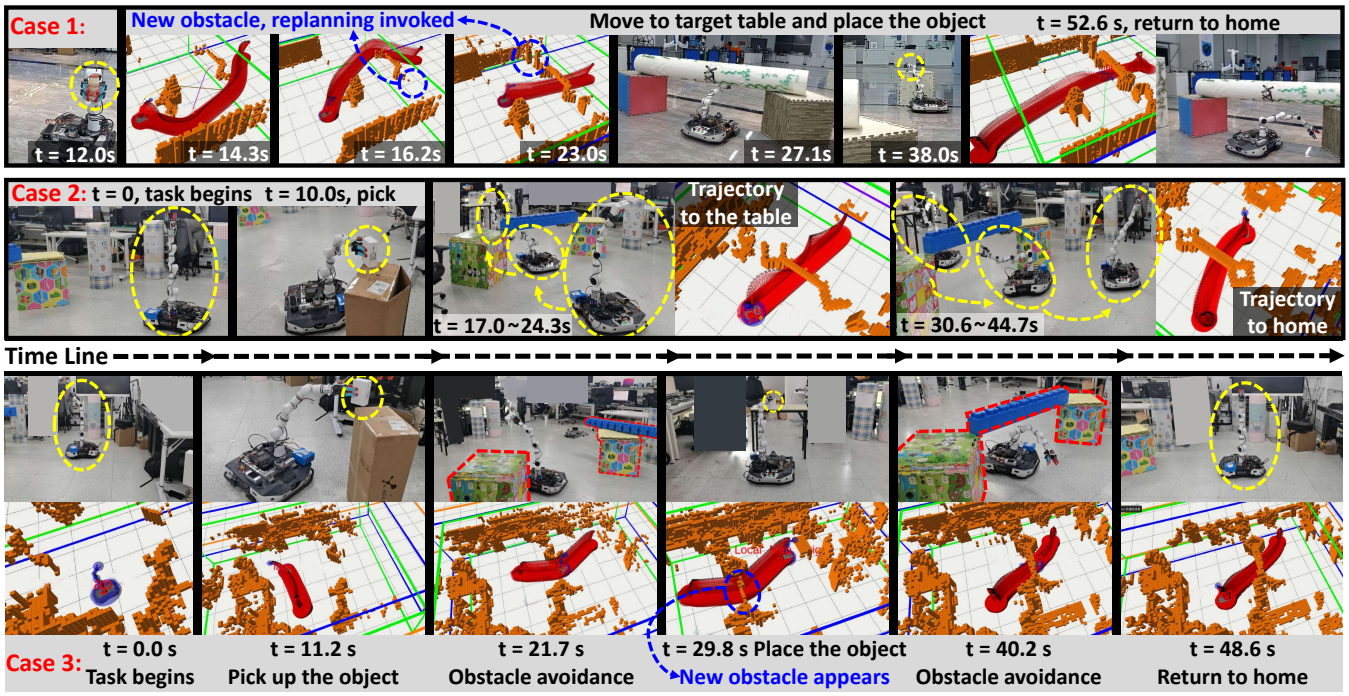

Fig. 4: Real-world experiments. The DDMoMa successfully performs pick-and-place tasks. Snapshots of filmed videos and RViz visualizations are presented in chronological order. In RViz visualizations, **orange** boxes illustrate the 3D occupancy grid map being updated in real-time. The **blue** silhouette represents the current state of the DDMoMa. The series of **red** silhouettes represent the trajectory currently tracked by the DDMoMa.

multimedia. In these cases, we randomly set the target state of the robot to force it to continuously traverse the obstacle area, demonstrating the practicality of the method.

In real-world experiments, the limits of linear velocity, linear acceleration, angular velocity, and angular acceleration of the DDB are set to $v_{\max} = 1.0m/s$, $a_{\max} = 0.8m/s^2$, $\omega_{\max} = 0.9rad/s$, and $\beta_{\max} = 1.0rad/s^2$, respectively. The limits of joint angles, joint angular velocities, and joint angular accelerations of the manipulator are set to $\boldsymbol{q}_{\max} = [3.1, 2.26, 3.1, 2.355, 3.1, 2.23, 6.28]^{\mathrm{T}}$, $\dot{\boldsymbol{q}}_{\max} = \texttt{ConstantVec}(2.35)$, and $\ddot{\boldsymbol{q}}_{\max} = \texttt{ConstantVec}(6.28)$, where the units are rad, rad/s, rad/s^2, respectively. The function $\texttt{ConstantVec} : \mathbb{R} \mapsto \mathbb{R}^7$ constructs a constant vector from a constant. For safety, we set $\boldsymbol{r}_{thr} = [40, 6, 6, 8, 4, 4, 7, 3.5, 3.5, 6, 3.5, 3.5, 8]$, where the unit is centimeter, and the order corresponds to the geometric solids from bottom to top in the collision model in Fig. 3.

C. Simulation Experiments

We employ two different scenes in simulation for ablation study and benchmark comparisons, which are denoted by *Cuboids* and *Tables*, as shown in Fig. 5 (a) and (b), respectively. Both are $20m \times 20m$ walled rooms filled with randomly placed obstacles of random size. For *Cuboids*, there are 80 grounded cuboids and 80 floating cuboids. For *Tables*, there are 80 grounded cuboids and 40 tables with rows and columns of legs. *Tables* is considered a more challenging scene, as cluttered tables with nonconvex structure form many narrow passages and the dense distribution of table legs makes the DDMoMa more susceptible to collision.

TABLE I: Ablation Studies. In each scene, **bold** or underline indicates TopAY achieves the **best** or the second best result.

Scene	Method	S.R. (%)	T.P. (ms)	T.D. (s)
Cuboids	TopAY	94.4	**218.0**	13.2
	TopAY-JPS	88.1	226.3	13.7
	TopAY-SEQ	95.1	783.3	13.0
Tables	TopAY	**92.5**	312.0	15.6
	TopAY-JPS	88.4	295.0	15.6
	TopAY-SEQ	92.2	1363	15.3

1) Ablation Studies: Ablation studies are conducted to demonstrate the efficacy of topological paths search and parallel processing in proposed pipeline. With the starting position fixed at the center of the room, we randomly generate 1,000 goal states and conduct planning with three distinct methods: TopAY, TopAY-JPS, where the topological path search is replaced by Jump Point Search, and TopAY-SEQ, where parallel manipulator path sampling and trajectory optimization is not performed.

We record the success rate (S.R.), average time consuming of planning (T.P.) and average trajectory duration (T.D.), as summarized in Table I. Planning attempts will be considered failed if either the planner cannot find feasible path for DDB and manipulator within limited time or none of the optimization processes successfully return with dynamically feasible and collision-free trajectories. Only cases where all

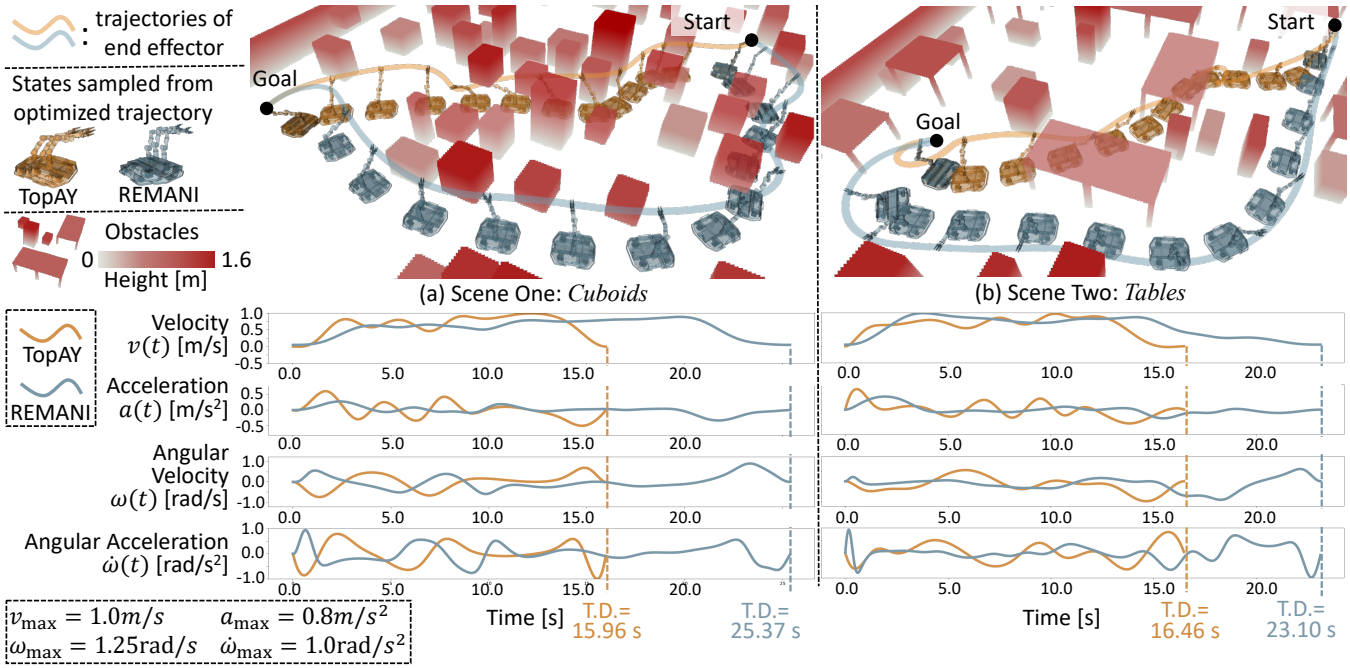

Fig. 5: Motion trajectories of the DDMoMa (upper half) and time plots of the DDB's kinematic variables (lower half).

TABLE II: Algorithm Comparisons

Scene	Scale	Small (3m ∼ 8m)			Medium (8m ∼15m)			Large (15m ∼ 30m)		
	Method	S.R. (%)	T.P. (ms)	T.D. (s)	S.R. (%)	T.P. (ms)	T.D. (s)	S.R. (%)	T.P. (ms)	T.D. (s)
Cuboids	TopAY	**98.0**	**225.4**	**11.4**	**98.2**	**382.6**	**18.1**	**97.0**	**581.1**	**25.5**
	REMANI [2]	88.1	467.6	17.1	81.4	845.5	24.1	71.1	1491	31.8
Tables	TopAY	**87.5**	**225.6**	**12.4**	**74.1**	**417.6**	**20.0**	**61.8**	**794.7**	**28.6**
	REMANI [2]	55.1	800.5	18.0	25.1	1346	26.5	11.4	4442	37.5

planners succeed contibute to T.P. and T.D. .

TopAY achieves higher success rates compared to TopAY-JPS, owing to the enhanced probability of finding feasible subspaces for the manipulator through topological paths search. Similarly, this strategy makes it possible for optimizer to obtain better initial values, thereby reducing the trajectory durations. Although introducing multiple initial values increases computational load, the efficiency gap remains within 6% thanks to the parallelization. TopAY achieves even higher efficiency in *Cuboids* due to the adoption of the early termination strategy in parallel trajectory optimization.

Compared to TopAY-SEQ, the proposed method significantly reduced T.P. through extensive parallelization, achieving approximately a threefold improvement. Meanwhile, the early termination strategy does not result in a noticeable decrease in success rate or optimality, with the gaps remaining within 1% and 2%, respectively.

2) Comparisons with REMANI [2]: We compare ToPAY with the SoTA trajectory optimization method for DDMoMa, REMANI [2]. Using the same scenarios, evaluation metrics and failure criteria as the ablation studies, 1,000 planning tasks are randomly generated for each scene and different distance range of the base. In each case, the parameters of all algorithms are meticulously tuned for the best overall performance. Results are summarized in Table II.

In *Cuboids*, TopAY maintains exceptional S.R. (>95%) in all distance ranges, while REMANI [2] exhibits noticeable deterioration over increasing distance, which is more pronounced in *Tables*. This stems from the greedy initial value acquisition of REMANI [2], which samples manipulator paths based on only one kinematically feasible path for the base. When this strategy fails, it falls back to full state space sampling. As the problem scale increases, the failure probability of the greedy strategy rises, and full state sampling also becomes more difficult, leading to a significant increase in failure rate within finite time.

In terms of planning efficiency, TopAY consumes less than half the time of REMANI [2], and can even achieve a five-fold speedup in *Tables*. We attribute this to the use of topological paths and parallel processing. In obstacle-dense environments, planners are highly susceptible to failures, potentially requiring multiple optimization attempts. REMANI [2] adopts a simple retry-until-success strategy with no mechanism to ensure diversity of initialization. In contrast, TopAY significantly accelerates this process by leveraging parallel optimization with diverse initial values derived from topological paths.

Table II also indicates TopAY's advantage over REMANI

[2] in terms of trajectory duration. We attribute this enhancement to the usage of topological paths and arc length-yaw parameterization. The former facilitates the exploration of a diverse set of initial values, potentially yielding better solutions than methods using single initial value. At a theoretical level, the later addresses the difficulty of enforcing the DDB's angular velocity and angular acceleration constraints induced by singularity of differential flatness. Fig. 5 provides a more intuitive comparison of TopAY and REMANI [2] through the motion trajectories of the robot and the time plots of the DDB's kinematic variables.

V. Conclusion & Limitations

In this paper, we present TopAY, an optimization-based trajectory planner for DDMoMa. The framework combines a hierarchical initial value acquisition method with a polynomial trajectory representation based on arc length–yaw parameterization to address high-dimensional state spaces and nonholonomic constraints. Simulation experiments demonstrate that TopAY achieves higher planning efficiency and success rate compared to SoTA method.

Despite these promising results, several limitations remain to be addressed for broader application. Our pipeline acquires initial value in a decoupled manner, significantly accelerating the procecss. However, this approach sacrifices completeness guarantee. Furthermore, in challenging cases, our method can remain computationally expensive and may fail to find stable and efficient solution within a reasonable time limit.

Neural networks, as a powerful implicit representation, have been shown to be a promising technique in motion planning [9], [16]. In the future, we will try to incorporate them to address above issues to improve robustness. Besides, to further exploit the advantages of the efficiency of our method, adapting it to multi-arm robot systems and different floating bases will also be taken into consideration.

References

[1] C. A. Contreras, A. Rastegarpanah, M. Chiou, and R. Stolkin, "A mini-review on mobile manipulators with variable autonomy," *Frontiers in Robotics and AI*, vol. 12, p. 1540476, 2025.

[2] C. Wu, R. Wang, M. Song, F. Gao, J. Mei, and B. Zhou, "Real-time whole-body motion planning for mobile manipulators using environment-adaptive search and spatial-temporal optimization," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 1369–1375.

[3] Y. Yang, F. Meng, Z. Meng, and C. Yang, "Rampage: Toward whole-body, real-time, and agile motion planning in unknown cluttered environments for mobile manipulators," *IEEE Transactions on Industrial Electronics*, vol. 71, no. 11, pp. 14 492–14 502, 2024.

[4] 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.

[5] 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.

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *ann arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[7] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[8] M. Zhang, N. Chen, H. Wang, J. Qiu, Z. Han, Q. Ren, C. Xu, F. Gao, and Y. Cao, "Universal trajectory optimization framework for differential drive robot class," *IEEE Transactions on Automation Science and Engineering*, 2025.

[9] Z. Han, M. Tian, Z. Gongye, D. Xue, J. Xing, Q. Wang, Y. Gao, J. Wang, C. Xu, and F. Gao, "Hierarchically depicting vehicle trajectory with stability in complex environments," *Science Robotics*, vol. 10, no. 103, p. eads4551, 2025.

[10] M. Zhang, C. Xu, F. Gao, and Y. Cao, "Trajectory optimization for 3d shape-changing robots with differential mobile base," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 104–10 110.

[11] 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.

[12] 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.

[13] 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.

[14] Y. Lu, Y. Ma, D. Hsu, and P. Cai, "Neural randomized planning for whole body robot motion," *arXiv preprint arXiv:2405.11317*, 2024.

[15] 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.

[16] S. Yan, Z. Zhang, M. Han, Z. Wang, Q. Xie, Z. Li, Z. Li, H. Liu, X. Wang, and S.-C. Zhu, "M 2 diffuser: Diffusion-based trajectory optimization for mobile manipulation in 3d scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[17] C. Zheng, Y. Li, Z. Song, Z. Bi, J. Zhou, B. Zhou, and J. Ma, "Local reactive control for mobile manipulators with whole-body safety in complex environments," *IEEE Robotics and Automation Letters*, 2025.

[18] 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 robot motion generation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 8112–8119.

[19] L. Zheng, R. Yang, M. Y. Wang, and J. Ma, "Barrier-enhanced parallel homotopic trajectory optimization for safety-critical autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2024.

[20] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1208–1214.

[21] O. De Groot, L. Ferranti, D. M. Gavrila, and J. Alonso-Mora, "Topology-driven parallel trajectory optimization in dynamic environments," *IEEE Transactions on Robotics*, 2024.

[22] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility-based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[23] 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.

[24] R. T. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM Journal on Control*, vol. 12, no. 2, pp. 268–285, 1974.

[25] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989.

[26] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[27] Y. Ren, Y. Cai, F. Zhu, S. Liang, and F. Zhang, "Rog-map: An efficient robocentric occupancy grid map for large-scene and high-resolution lidar-based motion planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8119–8125.