Hierarchical Motion Planning Framework for Cooperative Transportation of Multiple Mobile Manipulators

Heng Zhang, Haoyi Song, Wenhang Liu, Xinjun Sheng, Zhenhua Xiong and Xiangyang Zhu

Abstract—Multiple mobile manipulators show superiority in the tasks requiring mobility and dexterity compared with a single robot, especially when manipulating/transporting bulky objects. When the object and the manipulators are rigidly connected, closed-chain will form and the motion of the whole system will be restricted onto a lower-dimensional manifold. However, current research on multi-agent motion planning did not fully consider the formation of the whole system, the redundancy of the mobile manipulator and obstacles in the environment, which make the tasks challenging. Therefore, this paper proposes a hierarchical framework to efficiently solve the above challenges, where the centralized layer plans the object’s motion offline and the decentralized layer independently explores the redundancy of each robot in real-time. In addition, closed-chain, obstacle-avoidance and the lower bound of the formation constraints are guaranteed in the centralized layer, which cannot be achieved simultaneously by other planners. Moreover, capability map, which represents the distribution of the formation constraint, is applied to speed up the two layers. Both simulation and experimental results show that the proposed framework outperforms the benchmark planners significantly. The system could bypass or cross obstacles in cluttered environments, and the framework can be applied to different numbers of heterogeneous mobile manipulators.

Index Terms—Multiple mobile manipulators, closed-chain, formation, redundancy, obstacle avoidance, capability map

I. INTRODUCTION

MULTIPLE robots system (MRS) allows simpler, cheaper, modular robotic units to be reorganized into a group based on the task at hand. It can be as effective as a task-specific, larger, monolithic robot, which may be more expensive and has to be rebuilt according to the task [1]. Inspired by nature [2], MRS has evolved into a variety of forms, such as multiple mobile robots [3], multiple manipulators [4], multiple drones [5] and multiple underwater robots [6], and has been applied in various scenarios like construction [7], transportation [6] and rescue [9].

Compared with a single robot, the fundamental challenge of the MRS lies in the cooperation among robots. Cooperation may happen in several aspects, such as knowledge sharing and physical manipulation. In this paper, we focus on the motion planning of the MRS when manipulating bulky objects. In the task shown in Fig. 9, the robots need to cooperate and transport a large object, and first of all plan a path connecting the start and goal while satisfying a set of user-specified constraints.

As a combination of mobile robot and manipulator, mobile manipulator (MM) inherits the mobility from the mobile robot and the dexterity from the manipulator [10]. In the multiple mobile robots system, they are implicitly or explicitly required to maintain specific formation to improve the efficiency of communication and collaboration [11]. In the multiple manipulators system, they are rigidly connected to achieve robust manipulation of the object [12]. Therefore, formation and closed-chain constraints will meet in the multiple mobile manipulators system. Furthermore, compared with the mobile robot and the manipulator, MM is redundant. This implies that the same task at the end effector can be executed in different ways in the configuration space, which gives the possibility of avoiding forbidden regions and optimizing the robot configurations [15].

Closed-chain, formation, redundancy and obstacles are inevitable factors for motion planner of multiple mobile manipulators. Firstly, closed-chain constraint implicitly defines a lower-dimensional manifold in the configuration space, hence the probability that a uniform sample will satisfy this constraint is zero [14]–[16]. Moreover, obstacles in the environment will affect the connectivity of this manifold and bring additional challenges to the problem. Secondly, redundancy means that the motion of the system is under-constrained. Although it provides the chance to achieve extra behaviors, designing meaningful behaviors is nontrivial as they may vary significantly in different applications [17]–[18]. Finally, the influences of the redundancy and closed-chain constraint on the formation constraint are closely coupled. Redundancy is the basis for formation optimization. For example, due to the coordination of the torso and arms, human could transport the object while adjusting their formation. On the other hand, closed-chain constraint and the limited reachable workspace of the manipulator will greatly reduce the space of formation optimization, which is even worse in cluttered environments.

A. Contribution

To simultaneously deal with closed-chain and formation of the whole system, redundancy of the mobile manipulator and obstacles in the environment, this paper proposes a motion planning framework with the following innovations:

- Hierarchical framework: the centralized layer plans the object’s motion, and the formation of the system is optimized...
in the decentralized layer. A unique feature is that closed-chain, obstacle-avoidance and the lower bound of the formation constraints can be quickly checked in the centralized layer, which ensures compatibility with the decentralized layer. Moreover, the decentralized layer is distributed, and the redundancy of each robot can be explored independently in real-time. Therefore, the complex planning problem can be simplified while not violating the constraints.

- Diversified and efficient obstacle-avoidance strategies: on the basis of the closed-chain constraint, the system can either adjust the formation to bypass obstacles or adjust the self-motion of the redundant robots to cross obstacles. As a result, the motion planner exhibits excellent performance in complex environments, which is verified by comparing it with the benchmark planners.
- The framework is unified and can be applied to different numbers of heterogeneous mobile manipulators. Moreover, to the best of our knowledge, this is the first time to achieve six-dimensional object manipulation by multiple mobile manipulators in cluttered real-world environments.

B. Outline of the Paper

The outline of this paper is as follows. In Section II, related works on redundancy, closed-chain and formation constraints of the MRS will be reviewed. Section III will give a formal definition of the motion planning problem in detail. After that, we will introduce the proposed hierarchical framework in Section IV. The feasibility and superiority of the motion planner will be verified by simulation and real-world experiments in Section V. Finally, the paper is concluded in Section VI.

II. RELATED WORKS

The motion planners of multiple mobile manipulators in the literature are mostly inherited from its two subsystems: multiple manipulators and multiple mobile robots. The former focuses on the closed-chain constraint, while the latter emphasizes the formation constraint. In addition, these two classes of algorithms usually do not solve the redundancy explicitly. Therefore, in the following, we will first review the algorithms that deal with the redundancy of a single MM and then extend to related works from two perspectives: closed-chain constraint and formation constraint.

A. Redundancy of the Mobile Manipulator

Redundancy of the MM can be solved at position-level [17] [19] or velocity-level [20] [21]. For the former, redundancy parameters will be designed to restrict the motion of the MM. For the latter, redundancy can be dealt with by task-augmentation and task-priority algorithms. Compared with the task-augmentation algorithm [20], task-priority algorithm [21] is superior when dealing with conflict among different tasks. In this algorithm, the additional task is only satisfied in the null space of the primary task, which gives a higher priority to the primary task. For example, in our previous work [22], the motion of the end effector and the mobile robot were treated separately as the primary task and the additional task. The experimental results showed that the manipulator could grasp the object while the mobile robot was moving.

In the position-based and velocity-based algorithms, redundancy resolution will be transformed into nonlinear optimization problems about redundancy parameters and additional tasks, respectively. To speed up the optimization process and avoid local minimum, capability map (CM) can be queried to determine the seed [23] [24]. CM was initially proposed in [25] [26] and used to handle the base placement of the MM [27] [28]. It encodes the distribution of the user-specified constraints and could guide the iterative direction of the optimization algorithm. The effectiveness and superiority of CM were verified in our previous work in the human-robot collaboration task [24]. However, CM has only been used in a single MM, and how to combine it with the MRS is still an open question. In this paper, we will extend it to multiple mobile manipulators to accelerate the proposed framework.

B. Closed-Chain Constraint

Closed-chain constraint manifold has its dimension lower than that of the configuration space, hence uniform sampling in the configuration space has zero probability of generating a valid state that satisfies this constraint [14]–[16]. One solution to this problem is introducing an allowable tolerance [31]. Therefore, the volume of the satisfying subset grows, and various sampling-based motion planners for the unconstrained problem can be adopted. However, much of the complexity of handling the constraint transfers from the higher-level motion planner to the lower-level motion controller. In the case that multiple robots are rigidly connected, motion deviation caused by constraint relaxation may damage the robots and the transported object.

Another approach to resolve the closed-chain constraint is Projection (PJ). Several algorithms were developed based on it, such as RGD [32] and CBiRRT [33]. It iteratively projects a random sample onto the manifold according to the Jacobian-inverse gradient descent operation. Combining the projection operator with the workspace decomposition technique, [34] realized the motion planning of three mobile manipulators in the simulated environment. However, the constraint Jacobian is not guaranteed to be invertible all the time, and this approach relies heavily on gradient descent operation, which is time-consuming for high-dimensional robots. To improve the computational efficiency, Atlas (AT) [35], which uses piece-wise tangent spaces to locally approximate the constraint manifold, and Tangent Bundle (TB) [36], which is similar to AT but with lazy states checking, were proposed. However, to get the tangent space, the kernel of the constraint Jacobian has to be computed, which requires complex matrix decomposition. Therefore, the computational efficiency will be reduced.

In addition, inverse kinematics can be seen as a special projection operator. Compared with the Jacobian-based projection operator, it is more efficient, especially for robots with the analytical inverse kinematic solvers. For example, RLG, which combines the random sampling technique with inverse kinematics, was developed to deal with the closed-chain constraint [37]. RLG works well for the non-redundant multiple manipulators system. However, it is difficult to integrate
with other constraints simultaneously. Moreover, the kinematic characteristic of the fixed base manipulator is different from the MM, hence RLG is hard to extend to multiple mobile manipulators directly.

C. Formation Constraint

Formation constraint is widely studied in the field of multiple mobile robots, and typical algorithms are the leader-follower approach, behavioral approach and virtual structure approach [11]. In the domain of multiple mobile manipulators, [38] defined a virtual structure named system outlined rectangle (SOR), and the leader-follower approach was applied to plan the motion of the system, where the leader adjusted the width of the SOR according to the environment. Similarly, the convex region was defined in [39], and then the formation control problem was transformed into an optimization problem with respect to the convex region. However, the obstacle-avoidance strategies in [38]–[40] are conservative. As long as the virtual structure (e.g., SOR or convex region) intersects with obstacles, the system is considered in collision. As a result, bypassing becomes the only way to avoid obstacles, no matter how small they are. However, by utilizing the redundancy of the system, crossing obstacles is sometimes a more reasonable choice, which is common in the human-human collaborative tasks.

Obstacle crossing is challenging and rarely studied previously. It not only requires cooperation among different mobile manipulators but also requires coordination between the mobile robot and the manipulator. In our previous work, the traversability of obstacles was studied in the multiple mobile robots system with a deformable sheet [41]. The experimental results showed that the system could intelligently bypass or cross obstacles, which greatly increases the success rate of the motion planner in cluttered environments. Therefore, in this paper, we will extend this idea from multiple mobile robots to multiple mobile manipulators.

In summary, the coupling among redundancy, closed-chain, formation and obstacle-avoidance makes the motion planning of multiple mobile manipulators one of the most challenging problems in robotics. Although outstanding works have been done from a single perspective, e.g., task-priority algorithm [21] for redundancy, projection-based frameworks [32] [33] [35] [36] for the closed-chain constraint and virtual structure-based approach [39] for the formation constraint, they are unable to solve the above challenges simultaneously. As a result, the system cannot fully take its advantage brought by multi-robot and redundancy, and is prone to failure in cluttered environments.

III. PROBLEM DEFINITION

In this section, we derive the kinematic model of the system and define some mathematical notations for subsequent discussion. Moreover, a formal definition of the motion planning problem along with the analysis is presented afterward.

A. Modeling of the Multiple Mobile Manipulators System

Consider multiple mobile manipulators (M\textsuperscript{MM}) manipulating an object in Fig. 1. Let \( O_wX_wY_wZ_w, O_{obj}X_{obj}Y_{obj}Z_{obj}, O_{b,i}X_{b,i}Y_{b,i}Z_{b,i} \) and \( O_{c,i}X_{c,i}Y_{c,i}Z_{c,i} \) be the frames of the world, the object, the mobile base and the end effector of the \( i \)-th MM, where \( i = 1, 2, ..., n \) and \( n \) is the number of the robots in the system.

To simplify the model of the holonomic mobile robot, it is equivalent to an open chain [22] [42]. Therefore, the configuration of the \( i \)-th mobile robot can be expressed as \( \mathbf{q}_{b,i} \in \mathbb{R}^{n_{b,i}} \), where \( n_{b,i} \) is the degrees of freedom (DOF) of the mobile robot. In most cases \( n_{b,i} = 3 \) as the holonomic mobile robot can move and rotate in all directions on the ground. The configuration of the \( i \)-th manipulator is \( \mathbf{q}_{a,i} \in \mathbb{R}^{n_{a,i}} \), where \( n_{a,i} \) represents its DOF. Therefore, the configuration space of the \( i \)-th MM can be formally defined as \( \mathcal{C}^{i}_{MM} \subseteq \mathbb{R}^{n_{a,i}+n_{b,i}} \), which is the set of configurations of the \( i \)-th MM \( \mathbf{q}_i = (\mathbf{q}_{a,i}^T, \mathbf{q}_{b,i}^T)^T \).

To represent the homogenous transformation of frame \( i \) with respect to frame \( j \), we define \( \mathcal{X}_i^j \in SE(3) \) and its minimum representation \( \mathbf{t}_i = (p_i^T, \alpha_i^T)^T \in \mathbb{R}^6 \), in which \( p_i^T \) denotes the relative position, and \( \alpha_i^T \) denotes a minimum description of orientation. Therefore, the configuration space of the object is defined as \( \mathcal{C}_{obj} \subseteq \mathbb{R}^5 \), which is the set of the object’s poses with respect to the world frame \( \mathbf{t}_{obj}^{w} \).

Let the configuration space of the system \( \mathcal{C}_{MM} = \mathcal{C}^1_{MM} \times \cdots \times \mathcal{C}^n_{MM} \times \mathcal{C}_{obj} \subseteq \mathbb{R}^{\sum_{i=1}^{n} (n_{a,i}+n_{b,i})+6} \) be the set of \( \mathbf{c} = (\mathbf{q}_1^T, \ldots, \mathbf{q}_n^T, \mathbf{t}_{obj}^w, \mathbf{T}_i^j)^T \). The forward kinematics of the \( i \)-th MM at position-level and velocity-level are defined as Eq. (1) and Eq. (2), respectively.

\[
\mathbf{t}_{w} = f_k(\mathbf{q}_i) \tag{1}
\]

\[
\dot{\mathbf{t}}_{w} = \frac{\partial f_k(\mathbf{q}_i)}{\partial \mathbf{q}_i} \mathbf{q}_i = J_i(\mathbf{q}_i)\dot{\mathbf{q}}_i \tag{2}
\]

where \( f_k(\cdot) \) is the forward kinematic operator, and \( J_i(\mathbf{q}_i) \in \mathbb{R}^{6 \times (n_{a,i}+n_{b,i})} \) is the analytical Jacobian matrix of the \( i \)-th MM. When a six-DOF manipulator mounting on a mobile robot, the column of \( J_i(\mathbf{q}_i) \) is larger than that of the row, leading to the redundancy of the MM.
B. Closed-Chain Constraint

Given a random sample \( c = (q_1^T, ..., q_n^T, t^w_{obj})^T \), we define the vector of the end effector’s poses as \( E = (t_{e,1}^w, ..., t_{e,i}^w, ..., t_{e,n}^w)^T \in \mathbb{R}^{6n} \) where \( t_{e,i}^w = f_k(q_i) \).

The vector of the grasping poses on the object is denoted as \( G = (t_{g,1}^w, ..., t_{g,i}^w, ..., t_{g,n}^w)^T \in \mathbb{R}^{6n} \), in which \( t_{g,i}^w \) represents the \( i \)th grasping pose with respect to the world frame. For convenience, a projection \( \pi : C_{MM} \rightarrow C_{obj} \) is defined so that \( \pi(c) = t_{obj}^w \). Assuming the manipulated object is rigid, \( G \) can be easily derived by the geometric transformation \( g : C_{obj} \rightarrow \mathbb{R}^{6n} \) in Eq. (3).

\[
G = g(\pi(c)) \tag{3}
\]

When a rigid grasp exists between \( t_{e,i}^w \) and \( t_{g,i}^w \), for all \( i = 1, 2, ..., n \), the closed-chain constraint (C3) forms, and it can be formally described by \( f_{C3} : C_{MM} \rightarrow \mathbb{R}^{6n} \) in Eq. (4).

\[
f_{C3}(c) = E - G = 0 \tag{4}
\]

Therefore, the set of configurations that satisfy the closed-chain constraint is denoted as \( C_{C3} \subseteq C_{MM} \) in Eq. (5).

\[
C_{C3} = \{ c | c \in C_{MM}, f_{C3}(c) = 0 \} \tag{5}
\]

C. Formation Constraint

The fundamental difference between the closed-chain constraint (C3) and the formation constraint (FC) is that C3 is “hard” but FC is “soft”, in which “hard” constraints have to be satisfied exactly everywhere, and “soft” constraints are usually described by cost functions and optimized as much as possible. Therefore, FC can be transformed into an optimization problem and formally defined as follow.

\[
\max_c f_{FC}(c) \quad \text{s.t.} \quad c \in C_{C3} \cap C_{free}
\]

where \( C_{free} \) represents the set of collision-free configurations with appropriate dimensions. \( f_{FC} : C_{C3} \rightarrow \mathbb{R} \) is the formation metric of the system. Its design philosophy and expression will be illustrated in Section IV-C in detail.

D. Problem Definition

In addition to the notations above, we define a mapping \( \Pi : C_{obj} \rightarrow C_{C3} \) so that given a pose of the object \( t_{obj}^w \), \( \Pi(t_{obj}^w) = \{ c | c \in C_{C3} \cap C_{free}, \pi(c) = t_{obj}^w \} \). It is the set of configurations that satisfy collision and closed-chain constraints while the object’s pose is invariant. Therefore, the motion planning of multiple redundant mobile manipulators under closed-chain, formation and obstacle-avoidance constraints is defined as follows.

**Problem Definition:** given a start configuration \( c_{start} \in C_{C3} \cap C_{free} \) and a target pose of the object \( t_{obj}^w \), find a path \( \tau : [0, 1] \rightarrow C_{C3} \cap C_{free} \) such that 1) \( \tau(0) = c_{start} \); 2) \( \tau(1) \in \Pi(t_{obj}^w) \); and 3) the formation metric \( f_{FC} \) is optimized throughout \( \tau \).

E. Problem Analysis

The key to the closed-chain constraint is to design efficient samplers since uniform sampling can not work directly. As discussed above, PJ, AT and TB are centralized frameworks and move a random state \( c \in C_{MM} \) to a new state \( c_{new} \in C_{C3} \) iteratively, which is computationally expensive. Unlike the projection-based frameworks, our motion planner will not calculate \( q_i \), explicitly when dealing with the closed-chain constraint. On the contrary, we define the allowed sampling region for the mobile robot and query the capability map of the manipulator to quickly check \( \pi(c) \) is extensible or not.

Ideally, it is necessary to optimize the formation on each sample and generate a globally optimal path about \( f_{FC} \). However, this is hard to achieve both technically and theoretically. On the one hand, most samples are not on the final path connecting \( \tau(0) \) and \( \tau(1) \), hence optimizing \( f_{FC} \) on these samples are unnecessary and time-consuming. On the other hand, the convergence to optimality is not guaranteed for the existing optimal sampling-based motion planners like RRT* when optimizing over something other than the path length.

Therefore, we use a hierarchical way to resolve the complex motion planning problem. The centralized layer plans the motion of the object, and the formation optimization is delayed in the decentralized layer. Unlike the decoupled framework in [44], the “hard” closed-chain constraint \( f_{C3} \) and the lower bound of the “soft” constraint \( f_{FC} \) are guaranteed in our centralized layer to avoid conflict with the decentralized layer. In this way, the complex motion planning problem of multiple mobile manipulators is simplified while not decreasing the success rate and violating the constraints.

IV. HIERARCHICAL MOTION PLANNING FRAMEWORK

A. Overview of the Hierarchical Framework

Overview of the hierarchical motion planning framework is shown in Fig. 2. The centralized layer receives the motion planning request from the user and plans the motion of the object. Collision constraint, closed-chain constraint and the lower bound of the formation constraint are checked for each sample to ensure the object’s motion is executable by the physical system. Moreover, collision map and capability map are queried to accelerate the checking process. The decentralized layer makes full use of the computing resource on each MM and can be executed in a distributed way in real-time. It consists of two components: the formation optimizer and the task-priority motion controller. Taking the motion of the object as input, the former optimizes the formation and dexterous configuration of each robot as much as possible. To track the desired motion on the real redundant robot, task-priority motion controller is designed to send the joint motion command to the lower-level hardware. The details of the framework are explained as follows. Its feasibility and superiority will be verified in Section [V].
optimizes the formation and dexterous configuration of each mobile manipulator. The task-priority motion controller runs the feedback algorithm and sends distributed way in real-time, consists of the formation optimizer and the task-priority motion controller. The former takes the object’s motion as input and 

Collision constraint, closed-chain constraint

Fig. 2. Overview of the hierarchical motion planner. The centralized layer receives the motion planning request and computes the motion of the object.

Alg. 1: Object Motion Planner

1 Function plan(t_start, t_goal, t_obj)
2 if validChecking(t_goal) is false then
3 return fail;
4 end
5 T_s.init(t_start);
6 T_g.init(t_goal);
7 for t ← 0 to t_obj do
8 t_rand ← sampleValidObjConfig(T_s);
9 T_s.add(t_rand);
10 if connect(T_s, T_g) is success then
11 return extractPath(T_s, T_g);
12 end
13 swap(T_s, T_g);
14 end
15 return fail;

Function sampleValidObjConfig(T_s)

16 for j ← 1 to maxSample do
17 t_rand ← sampleAndExtend(T_s);
18 if validChecking(t_rand) is true then
19 return t_rand;
20 end
21 end
22 return none;

Function validChecking(t_rand)

23 if t_rand is in collision then
24 return false;
25 end
26 // compute G according to Eq. (3)
27 G ← getGraspingPoses(t_rand);
28 for each t_g,i in G do
29 // defined in Alg. 3
30 if sampleInASR(t_g,i) is none then
31 return false;
32 end
33 end
34 return true;

B. Centralized Layer

The sampler is designed as a primitive in the centralized layer. Therefore, working with different sampling-based algorithms, such as PRM [45] and EST [46], becomes a trivial task. We will introduce the sampler in the structure of RRTConnect algorithm [47] in the following.

For brevity of notations, we omit the superscript and subscript of t^w_rob in this part. For example, t_{start}, t_{goal} and t_{rand} represent the start, the goal and random pose of the object with respect to the world frame, separately. Some key functions in Alg. 1 are explained as follows.

- plan(t_{start}, t_{goal}, t_{obj}) generates the path of the object when the planning request and allowed time are specified. Before planning, the validity of t_{goal} is first checked, and two trees are initialized. T_s starts from t_{start} and T_g starts from t_{goal}. Then random valid pose t_{rand} is sampled and added to the tree T_s. In each iteration, the two trees try to connect with each other by connect() and extract the valid path by extractPath(). On the real robot, the path should be post-processed and time-parameterized to get the smooth object’s motion. If the two trees are unable to connect, they are swapped and continue to grow in the next iteration until the path is found or the allowed time t_{obj} is exceeded.

- sampleValidObjConfig(T_s) is the sampler of the centralized layer. It generates valid object’s poses that are connectable to T_s. sampleAndExtend() is a standard operation in the sampling-based algorithm. It first generates a randomized pose by uniform or gaussian samplers and then searches the nearest pose in T_s and extends to t_{rand} by the given step size. If t_{rand} is valid, it will be returned.

- validChecking(t_{rand}) checks whether t_{rand} can be extended to a composite configuration c_{rand} ∈ C_{cs} ∩ C_{free} so that π(c_{rand}) = t_{rand}. Given the pose of the object, the grasping poses vector G is calculated based on Eq. [3]. For each pose t_g,i in G, the allowed sampling region is sampled to check whether the closed-chain constraint and the lower bound of the formation constraint can be
satisfied for each $MM$. When these conditions are met for all $i = 1, 2, ..., n$, $t_{\text{rand}}$ is seen as a valid sample. The detail of `sampleInASR()` is introduced in the following.

Fig. 3. (a) Although the pose of the object is collision-free, it exceeds the workspace of the robots and the closed-chain constraint is unable to be satisfied. (b) Although a configuration can be found to satisfy the closed-chain constraint, one of the robots must be in an awkward configuration due to the obstacle. Therefore, these samples should be discarded as they are unable to be executed or lead to poor and unstable formations in the decentralized layer.

1) Allowed Sampling Region: When planning the motion of the object, in addition to meeting the “hard” closed-chain constraint $f_{FC}(e)$, it is necessary to ensure the lower bound of the “soft” formation constraint $f_{FC}(e)$. Otherwise, the centralized layer may conflict with the decentralized layer. For example, although the object’s pose is collision-free in Fig. 3(a), it exceeds the workspace of the robots, and the closed-chain constraint is unable to be satisfied. Similarly, for the object’s pose in Fig. 3(b), although a configuration can be found to satisfy the closed-chain constraint, one of the robots must be in an awkward configuration due to obstacles. Therefore, the poses of the object in these cases should be abandoned as they are inexecutable or lead to poor and unstable formations in the decentralized layer.

To check the validity of $t_{g,i}^w$, we define the allowed sampling region (ASR) for each mobile manipulator in Eq. (7).

$$\text{ASR}(t_{g,i}^w, \text{thres}) = \{q_{b,i} \mid \exists q_{a,i}, \forall q_k(q_i) = t_{g,i}^w, \quad \text{and } f_{FC,i}(q_i) \geq \text{thres and } q_i \in C_{\text{free}}\}$$

where $f_{FC,i}(q_i)$ is the formation metric of the $i$th $MM$, and $\text{thres}$ is the allowed threshold of this metric. $f_{FC,i}(q_i)$ is a component of $f_{FC}(e)$, and their expressions are defined in Eq. (12) and Eq. (13). Therefore, we can check whether existing $q_{b,i} \in \text{ASR}(t_{g,i}^w, \text{thres})$ to see the validity of $t_{g,i}^w$.

ASR changes with $t_{g,i}^w$ and $\text{thres}$. Due to the complexity of multiple mobile manipulators, it is usually obtained by sampling. According to the definition of ASR, given a random sample $q_{b,i}, q_{a,i}$ has to be computed based on the inverse kinematics of the manipulator, and the following conditions should be checked. We name this method $IKCL$, which means the Inverse Kinematics-based Centralized Layer.

- **closed-chain constraint**: $f_k(q_i) = t_{g,i}^w$
- **lower bound of the formation constraint**: $f_{FC,i}(q_i) \geq \text{thres}$
- **collision constraint**: $q_i \in C_{\text{free}}$

Although computing $q_{a,i}$ and checking these conditions for one sample is fast, this module will be invoked frequently in `sampleInASR()`, hence reducing the time cost of it will improve the performance of Alg. 1 significantly. In the following, a novel tool named capability map (CM) will be developed to speed up this module. Querying CM can avoid computing $q_{a,i}$ and $f_{FC,i}(q_i)$ while satisfying $f_k(q_i) = t_{g,i}^w$ and $q_i \in C_{\text{free}}$ automatically. We name this method CMCL, which means the Capability Map-based Centralized Layer.

2) Capability Map: CM was initially proposed to solve the base placement when the mobile robot and the manipulator move asynchronously [27]-[30]. It was improved to guide the coordinated motion planning of the $MM$ in our previous work [23]-[24]. CM stores the configurations and/or the poses of the end effector along with the corresponding index. It is built in advance and then queried to reduce the online computational burden. Besides, using offline data structure to accelerate the online computing process can be seen in other robotic fields. For example, Sucan et al. generated a precomputed database of the constraint-satisfying states, and the planning algorithms can sample in this database to get valid states quickly [48]. Due to its high efficiency, it has been merged into MoveIt [49], which is a famous open-source motion planning software.

![Table 1](image1.png)

The relationship among $O_w$, $X_w$, $Y_w$, $Z_w$, $O_{b,i}$, $X_{b,i}$, $Y_{b,i}$, $Z_{b,i}$ and $O_{e,i}$, $X_{e,i}$, $Y_{e,i}$, $Z_{e,i}$ can be expressed as Eq. (8).

$$X_{e,i}^{-1} = (X_{w}^{-1})^{-1}X_{e,i}$$

In the mobile manipulation task, although $X_{e,i}^{-1}$ and $X_{w}^{-1}$ may change significantly when the mobile robot moving around, $X_{e,i}^{-1}$ must be within the reachable workspace of the manipulator. Therefore, the distribution of $X_{e,i}^{-1}$ and the corresponding collision/formation metric can be treated as prior information and stored in advance to construct $CM$. 

![Diagram](image2.png)
CM can be constructed by configuration space or cartesian space sampling. The former is faster than the latter as the inverse kinematics of the manipulator should be called for each sample in the cartesian space. However, configuration space sampling will lead to nonuniform end effector’s poses, which is not good for the querying process. Moreover, CM is only constructed offline once, hence the build time is acceptable in most cases. Therefore, cartesian space sampling is preferred in this work. A comprehensive comparison between them is available in [50].

In addition, various helpful information about \( X_{e,i}^{b,i} \) and \( q_{a,i} \) can be stored in CM. For example, the dexterity of the manipulator was saved in [26], [27] also stored statically and indicate the validity of the MM, so they will be saved in CM.

Alg. 2 shows the pseudo-code to construct and query CM, and the key functions are explained as follows.

- **constructCM(discreteRes, thres)**: the workspace is first discretized according to the given resolution. For each \( t_{g,i}^{b,i} \), the inverse kinematics of the manipulator is invoked, and the formation metric \( f_{FC,i} \) is computed based on Eq. (12). If \( q_i \) is not in self-collision and \( f_{FC,i} \) is larger than the given threshold \( thres \), \( t_{g,i}^{b,i} \) and \( f_{FC,i} \) are added to CM. Poses with low \( f_{FC,i} \) are discarded as they waste storage space and will lead to poor formations in the querying process. CM with different \( thres \) is shown in Fig. [4](a). Colors represent the mean \( f_{FC,i} \) of all 6D poses inside the 3D voxel.

- **queryCM(tw,i)**: the discrete CM is an approximation of the continuous workspace, so the requested pose may not be in CM exactly. Therefore, we round the requested pose based on **discreteRes** to get \( t_{round} \). For example, 0.51 will be rounded to 0.5 when the resolution is 0.1. When \( t_{round} \) is within the reachable workspace of the manipulator and the formation metric is larger than the threshold, \( f_{FC,i} \) stored at \( t_{round} \) is returned.

After constructing CM and the querying process is ready, let us take a closer look at how to check the validity of \( q_{b,i} \) based on it. In \( ASR(t_{g,i}^{w}, thres) \), \( t_{g,i}^{w} \) is constant and the same as \( X_{e,i}^{w} \) when the closed-chain constraint is satisfied. Given a sample \( q_{b,i}, X_{e,i}^{b,i} \) can be calculated according to Eq. (8). As the formation metric and self-collision information about the MM have been computed and stored in advance, they can be retracted from CM and indicate the validity of \( q_{b,i} \). Therefore, the inverse kinematic solver of the manipulator and the computational process of \( f_{FC,i} \) are avoided.

The pseudo-code of **sampleInASR(tw,i, bounds)** is shown in Alg. 3. **bounds** represents the lower and upper limits that the mobile robot configuration could be. It is useful when the heuristic information about the ASR is known. In **sampleInASR()** of Alg. [4] **bounds** is set to its default value, which is the whole movable space of the mobile robot. Its value in the decentralized layer will be explained later in Alg. [4] **sampleMobileRobot(bounds)** generates random mobile robot states within **bounds**, and then the collision status is checked according to the 2D collision map of the environment. **collisionMap** will be updated continuously in another thread, hence the uncertainty of the environment can be captured and colliding with obstacles is avoided. After getting \( t_{b,i}^{w} \), we invoke **queryCM()** to retract the formation metric stored at it. Finally, the pair of the valid sample \( q_{b,i} \) and the corresponding \( f_{FC,i} \) are returned.

**ASR** can be built by calling **sampleInASR()** repeatedly. Fig. [4](b) shows the **ASR** with different \( thres \). The origin and direction of the arrow represent the position and orientation of the mobile robot, respectively. The color indicates the formation metric when the mobile robot locates at the arrow. As can be seen, when \( thres \) is too low, there are lots of poses in **ASR**, and most of them will lead to poor formations in the decentralized layer. When \( thres \) is too high, there are few poses in **ASR**, and we have to spend much time in **sampleInASR()**. Therefore, \( thres \) should be moderate to balance the quality of the formation and the computational time of the centralized layer.

**Alg. 3: sampleInASR(tw,i, bounds)**

1. for \( j \leftarrow 1 \) to maxSample do
2. \( q_{b,i} \leftarrow sampleMobileRobot(bounds); \)
3. if collision(qb,i, collisionMap) is true then
4. continue
5. end
6. \( t_{b,i}^{w} \leftarrow getTransform(q_{b,i}, tw,i); \)
7. \( f_{FC,i} \leftarrow queryCM(t_{b,i}^{w}); \)
8. if \( f_{FC,i} \) is not none then
9. return \( [q_{b,i}, f_{FC,i}]; \)
10. end
11 end
12 return none;

C. Decentralized Layer

As shown in Fig. [2] the decentralized layer consists of the formation optimizer and the task-priority motion controller. It can be deployed in a distributed way in real-time. Therefore, each robot could sense the uncertainty of the environment and adjust its motion timely.

1) **Formation Optimizer**: The characteristic of formation control naturally leads to the question of what variables to sense and what variables to control to achieve the desired formation [11]. In previous works [38]–[40], the inter-robot variables, such as the relative position and orientation among different robots, are sensed and controlled. In these works, the system is usually treated as a rigid body and loses the ability to cross obstacles. On the contrary, the information between the robot and the object is easier to sense when rigid grasp exists. Therefore, we measure the dexterity of each robot and control the geometric variables relative to the object. In this way, the formation metric of the complex system is decoupled, and the redundancy of each **MM** can be explored independently.

Different metrics were proposed to measure the dexterity of the open-chain robot [51], and the most popular is the
The formation metric of the system can be defined as the maximum manipulability developed by Yoshikawa [52].

$$\omega(q) = \sqrt{\text{det}(J_i(q_i)J_i(q_i)^T)}$$  \hspace{1cm} (9)$$

where $J_i(q_i)$ is the analytical Jacobian matrix of the $i$th MM. According to [51], $\omega(q_i)$ is an unbounded index and only indicates the relative distance to singularity. Therefore, it is normalized to obtain an absolute metric.

$$\mu(q_i) = \frac{\omega(q_i)}{\max(\omega(q_i))}$$  \hspace{1cm} (10)$$

where $\max(\omega(q_i))$ is the maximum manipulability of the $i$th MM, hence $\mu(q_i)$ is limited within $[0, 1]$.

To represent the relative information between the robots and the object, geometric variables are defined in Fig. 4. $O_{e,i}$ and $O_{obj}$ are the center of the $i$th end effector and the object, respectively. Their projections on the $X_{b,i}O_{b,i}Y_{b,i}$ plane are represented as $P_{Ob,i}$ and $P_{Oobj}$, respectively. $\theta_{eo,i}$ is the angle between the $i$th end effector and the object, and $\theta_{eb,i}$ represents the angle between the $i$th end effector and the base. An ideal formation will be formed when $\theta_{eo,i}$ and $\theta_{eb,i}$ are equal to zero for all $i = 1, 2, ..., n$.

As can be seen, when the formation constraint is relaxed (low $\text{thres}$), there are few poses in $\text{CM}$ and $\text{ASR}$, and we have to spend much time to sample a valid one. Therefore, $\text{thres}$ should be moderate to balance the quality of the formation and the computational time.

Fig. 5. Geometric variables represent the relative information between the robots and the object. $P_{O_{b,i}}$ and $P_{O_{obj}}$ are the projection of the $i$th end effector $O_{e,i}$ and the object $O_{obj}$ on the $X_{b,i}O_{b,i}Y_{b,i}$ plane, respectively. $\theta_{eo,i}$ is the angle between the end effector and the object, and $\theta_{eb,i}$ is the angle between the end effector and the base. An ideal formation will be formed when $\theta_{eo,i}$ and $\theta_{eb,i}$ are equal to zero for all $i = 1, 2, ..., n$. Therefore, we define $r(q_i)$ to measure this relationship between the $i$th MM and the object.

$$r(q_i) = f_r(\theta_{eb,i}) \times f_r(\theta_{eo,i})$$  \hspace{1cm} (11)$$

where $f_r(.)$ is a continuous monotonic decreasing function. It reaches the maximum of 1 when the independent variable is zero and gradually decreases to 0 as the independent variable increases. Therefore, $r(q_i)$ is limited within $[0, 1]$.

Combining $r(q_i)$ with the dexterity of the robot, the formation metric of the $i$th MM can be defined as follow.

$$f_{FC,i}(q_i) = \mu(q_i) \times r(q_i)$$  \hspace{1cm} (12)$$

It is a trade-off between $\mu(q_i)$ and $r(q_i)$. As can be seen, no inter-robot information is required in the definition of $f_{FC,i}(q_i)$, hence it can be optimized in a distributed way.

The formation metric of the system can be defined as the minimum $f_{FC,i}(q_i)$ of all robots. When $f_{FC,i}(q_i)$ reaches its
maximum for all robots, the system forms an ideal formation.

\[ f_{FC}(\epsilon) = \min \{f_{FC,i}(q_i) | 1 \leq i \leq n \} \tag{13} \]

### Alg. 4: Formation Optimizer

1. **Function** optimize\(_{(w, \epsilon)}(q_{ref}, v_{max}, t_{opt}, \epsilon)\)
2. \( \text{bounds} \leftarrow \text{getBounds}(q_{ref}, v_{max}, t_{opt}) \);
3. for \( t \leftarrow 0 \) to \( t_{opt} \) do
4. \( (q_{b,i}, f_{FC,i}) \leftarrow \text{sampleInASR}(t_{w,i}, \text{bounds}) \);
5. \( q_{seed} \leftarrow \text{update}(q_{b,i}, f_{FC,i}) \);
6. end
7. \( q_{b,i} \leftarrow \text{iterate}(q_{seed}, (1-\epsilon)t_{opt}) \);
8. return \( q_{b,i} \);

The formation optimizer can be executed offline or online. Generally speaking, online algorithm is superior to the offline version when dealing with the uncertainty of the environment. However, it relies on the efficient method to solve the constraint. As can be seen from Eq. (12), complex nonlinear relationship exists between \( f_{FC,i} \) and \( q_i \). Therefore, the convergence rate may be slow, and the notorious local minimum may affect the optimization process.

To make the optimization process faster, once again, we query the precomputed CM to determine the seed. In this way, the optimizer could start with a sub-optimal formation and quickly converge to the optimal solution. The pseudo-code of the formation optimizer is shown in Alg. 4.

Given the grasping pose \( t_{w,g,i} \), Alg. [4] generates the mobile robot’s configuration \( q_{b,i} \) with the highest \( f_{FC,i} \). \( q_{ref} \) is the reference configuration of the mobile robot and can be set to the current state or the state in the last iteration. \( v_{max} \) is the maximum speed of the mobile robot. \( t_{opt} \) is the maximum time for the optimizer, and \( \epsilon \in [0,1] \) is the time ratio allowed for searching the seed. \( \text{bounds} \) represents the area that the mobile robot’s motion of the mobile robot will violate the maximum velocity bounds or the end effector pose constraint \( t_{w,i} \) when \( q_{b,i} \) exceeds \( \text{bounds} \).

Taking advantage of the precomputed CM, we sample in ASR within the allowed time \( t_{opt} \) to get the seed, and then the optimizer starts from \( q_{seed} \) and iterates to the optimal solution within the remaining time \((1-\epsilon)t_{opt}\). The Nelder-Mead simplex algorithm [53] is chosen as the solver in `iterate()`. It has no requirement on the differentiability of \( f_{FC,i} \) and shows good performance in practice.

2) **Task-priority Motion Controller:** The centralized layer plans the motion of the object and the end effector \( t_{w,e,i} \in \mathbb{R}^6 \). The formation optimizer in the decentralized layer generates the optimal motion of the mobile robot \( t_{w,b,i} \in \mathbb{R}^{n_{b,i}} \). To send joint motion command to the lower level hardware, closed-loop motion controller should be designed.

Historically, several algorithms were developed to compute the joint motion command of the redundant robot, such as pseudo-inverse Jacobian-based, Jacobian transpose-based, task-augmentation-based and task-priority-based algorithms [20] [21]. Task-priority-based algorithm outperforms others when dealing with conflict among different tasks and is pruned to the mobile manipulator in this paper.

In this algorithm, the motion of the end effector and the mobile robot are treated as the primary task and the additional task, respectively. The formulation of the task-priority motion controller is shown in Eq (14) [13] [22].

\[ \dot{q}_i = J_{i}^T \omega_{e,i} + [J_{b,i}(I - J_{i}^T J_{i})]^{-1} (\dot{t}_{w,b,i} - J_{b,i} J_{i}^T \dot{t}_{w,e,i}) \tag{14} \]

where \( J_i \) is the identity matrix with dimension \((n_{b,i} + n_{a,i}) \times (n_{b,i} + n_{a,i})\). \( J_{b,i} \) is the matrix related to the additional task and derived as follows.

\[ J_{b,i} = \frac{\partial \dot{t}_{w,b,i}}{\partial q_{b,i}} = \left( \frac{\partial \dot{t}_{w,b,i}}{\partial q_{i}} \frac{\partial q_{i}}{\partial a_{i}} \right) = (I_{n_{b,i} \times n_{b,i}} \cdot O_{n_{a,i} \times n_{a,i}}) \]

In Eq (14), the additional task is only satisfied in the null space of the primary task. When they are in conflict, the primary task will be satisfied as a higher priority. More detail about this algorithm is available in [13] [21] [22].

The closed-loop version of Eq (14) is shown as follows.

\[ \dot{q}_i = J_{i}^T \omega_{e,i} + [J_{b,i}(I - J_{i}^T J_{i})]^{-1} (\omega_{e,i} - J_{b,i} J_{i}^T \omega_{e,i}) \tag{15} \]

where \( \omega_{e,i} = K_{e,i} (\alpha_i \dot{t}_{w,b,i} - \dot{t}_{w,e,i}) \) is the tracking error between the desired and current poses of the end effector.

\[ \omega_{b,i} = K_{b,i} (\alpha_i \dot{t}_{w,b,i} - \dot{t}_{w,b,i}) \]

and the relationship among \( \dot{t}_{obj}, t_{g,w} \) and \( q_i \) are tested. To demonstrate the superiority of the proposed hierarchical motion planner, numerous simulations and real-world experiments are conducted in Section V-E and Section V-F respectively.

Moreover, different numbers of heterogeneous robots are considered in this paper. As shown in Fig. 1, heterogeneous manipulators, including UR [4], ZU [7] and Jaco [8], and heterogeneous mobile robots, including the quadruped robot [2] and the omnidirectional wheeled mobile robot, will be tested. All the codes are implemented in the architecture of ROS (Robot Operating System) [54] based on C++. CM is saved in the point cloud format [55]. The computer is Intel i7-8700 (3.2GHz) with 32GB memory. The attached video shows the result of simulations and real-world experiments [2].

In the real-world experiments, all robots are equipped with wireless transmission modules in the same local network for

\[ 1 \text{https://www.universal-robots.com} \]
\[ 2 \text{https://www.jaka.com} \]
\[ 3 \text{https://www.kinovarobotics.com} \]
\[ 4 \text{https://unitree.com} \]
\[ 5 \text{https://youtu.be/c2Lxm15fY-c} \]
efficient communication. Leader-follower approach is adopted for the system, where the leader plans the global path of the object and each follower is equipped with a 6D force-torque sensor and a visual sensor on the end effector to detect the motion relative to the leader. Moreover, each robot is equipped with a laser scan to sense the static and dynamic obstacles in the environment.

B. Redundancy of the Mobile Manipulator

Fig. 6. Redundancy of the mobile manipulator. The mobile robot moves on the ground while the end effector keeps still.

The redundancy of the MM is shown in Fig. 6. As can be seen, when the end effector keeps still, there is extra DoF for the robot to move on the ground. This is called the self-motion of the redundant robot. It represents the null space of the Jacobian matrix $J_i(q_i)$ and corresponds to the second term on the right side of Eq. (14). Redundancy is the basis for formation optimization, but it also brings challenges to the motion planning problem.

C. Performance of the Capability Map

In this part, we evaluate the time cost when checking the validity of ASR by querying CM and compare the results with other inverse kinematic solvers on the heterogeneous robots. The mean time of the checking process for $1 \times 10^3$ samples is shown in Table I.

| Robot           | KDL [56] | TRAC-IK [57] | IKFast [58] | CM (ours) |
|-----------------|----------|--------------|-------------|----------|
| UR5-based MM (ms) | 4813.4   | 471.3       | 441.7       | 0.26     |
| ZU7-based MM (ms)    | 9406.8   | 751.4      | 440.5       | 0.26     |
| Jace-based MM (ms)   | 7843.1   | 3937.0     | 409.7       | 0.27     |

*KDL* (Kinematics and Dynamics Library) provides the Jacobian-based inverse kinematic solver and is a component of Orocos [56]. TRAC-IK [57] is built on top of KDL and also combines the sequential quadratic nonlinear optimization approach to handle joint limits. IKFast is a powerful inverse kinematics solver that can analytically solve different manipulators’ kinematic equations [58]. From Table I, we can see that querying CM is faster than invoking the inverse kinematic solvers. Moreover, CM querying time is stable, which is a good property when planning the motion of heterogeneous robots.

D. Task-Priority Motion Controller and Multiple Robots System Model

The task-priority motion controller and the relationship among $t_{obj}^w$, $t_{obj}^w$ and $q_i$ are tested in this part. The system consists of three robots and one object. The orientation (roll-pitch-yaw) of the object is required to move along the sinusoidal trajectory while the center (x-y-z) keeps stationary. For each object’s pose $t_{obj}^w$, the grasping poses vector $G$ will be calculated according to Eq. (14). Given the grasping pose $t_{g,j}^w$, the formation optimizer plans the optimal mobile robot’s pose $t_{b,i}^w$. Finally, the task-priority motion controller receives $t_{g,j}^w$ and $t_{b,i}^w$, and calculates the desired joint velocity $\dot{q}_i$ based on Eq. (15).
The snapshots during the task are shown in Fig. 9 and the motion of the system is available in the attached video. Fig. 9(a) shows the desired and actual trajectories of the object in roll direction, and the trajectories in pitch and yaw directions are similar to that in roll direction and not shown here. The desired and actual trajectories of the mobile robot and the end effector of MM1 are shown in Fig. 9(b)-(j), respectively. The trajectories of MM2 and MM3 are similar to that of MM1 and not shown here. As can be seen from Fig. 9, the mobile robot and the end effector are able to track the desired trajectories simultaneously. Meanwhile, multiple robots could cooperate with each other and manipulate the object precisely.

E. Performance of the Hierarchical Motion Planner

1) Simulation Scene: As obstacles or the number of robots increase, the simulation scenes are shown in Fig. 9(a)-(f). The size of the scenes is 15m × 25m. The start and goal states are displayed on the most left and most right, respectively. Snapshots during the tasks are also shown in Fig. 9. The pictorial motion of the tasks is available in the attached video. In these tasks, the start and goal states are assumed to satisfy the closed-chain and collision constraints, and valid paths connecting the start and goal states exist theoretically. Moreover, the height of some obstacles is lower than the maximum allowed height of the object, which means that obstacles may be crossed by the system.

2) Simulation 1: In this simulation, we compare the proposed hierarchical framework with the centralized frameworks, such as Projection (PJ) [32] [33], Atlas (AT) [35] and Tangent Bundle (TB) [36]. In addition, we also compare the performance of IKCL with CMCL to see the advantage of CM. As discussed in Section II-B, PJ computes the Jacobian of the system and projects a random sample c onto the manifold based on Newton procedure. AT and TB are built on top of PJ with some modifications. For the fairness of the comparison, in IKCL, the inverse kinematic solver is chosen as KDL, which is based on the Jacobian of the single MM.

In the tasks shown in Fig. 9(a)-(f), the start state \( t_{\text{start}} \) and goal state \( t_{\text{goal}} \) of the object are specified by the user. However, PJ, AT and TB are centralized frameworks. Not only the state of the object but also the state of the robots should be specified. Therefore, a random goal state \( c_{\text{goal}} \) will be selected for each scene so that \( c_{\text{goal}} \in C_{\text{free}} \cap C_{\text{free}} \) and \( \pi(c_{\text{goal}}) = t_{\text{goal}} \). Moreover, formation constraint can not be resolved directly in PJ, AT and TB, hence only closed-chain and collision constraints are considered in these centralized frameworks. PJ, AT and TB on the multiple mobile manipu-
### TABLE II

| Task | Method | Total | RRT Connect | BKPIECE | EST | STRIDE | PRM |
|------|--------|-------|-------------|---------|-----|--------|-----|
|      |        | Success/Total | Success/Total | Success/Total | Success/Total | Success/Total | Success/Total |
|      |        | Time(s) | Time(s) | Time(s) | Time(s) | Time(s) | Time(s) |
|      |        |         |         |         |         |         |         |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 9/10  | 7.53±0.39 | 2/10 | 11.87±3.80 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 1.78±1.11 | 7/10 | 6.59±6.28  |
| Fig. 2(a) | | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 21.73±1.53 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 1.42±0.66 | 10/10 | 20.19±0.16 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 5.73±0.39 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 2.29±0.43 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 1.10±0.09 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 12.85±2.17 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 1.56±0.71 |
| Fig. 2(b) | | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 0/10 | 3.80±2.72 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 7.51±1.10 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 0/10   | 20.25±10.19 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 8.95±1.07 | 2/10 | 11.39±3.60 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 5.39±2.30 | 10/10 | 31.11±0.24 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 8.95±0.76 | 2/10 | 21.31±1.55 |
|      |        | 0/10   | 0/10   | 0/10   | 0/10   | 10/10 | 8.95±0.76 | 2/10 | 21.31±1.55 |

The performances of different frameworks are shown in Table II. The terms and settings are explained as follows. The maximum allowed planning time \( t_{obj} \) in Alg. 1 is set to 20s, 20s, 30s, 30s, 50s and 50s for the tasks in Fig. 2(a)-(f), respectively. Each framework is combined with six random planning algorithms, namely RRT [59], RRTConnect [47], BKPIECE [60], EST [46], STRIDE [61] and PRM [45]. Each combination runs 10 times to get the statistical results. “Success/Total” columns represent the successful and total simulations conducted. “Time(s)” columns represent the mean time and the standard deviation of the successful simulations. When all simulations fail, “Time(s)” columns are set to “—”. PRM constructs a roadmap of the entire environment that can be used for multiple queries. It runs out of the given time to build the roadmap and then searches a valid path. Due to the post-process of the motion planner, the time cost of PRM will be slightly over the maximum allowed time (see the last column in Table II). The other five planning algorithms construct trees that can be used for a single query. They terminate as long as finding a valid path. BKPIECE relies heavily on the projection evaluator to guide the exploration of the continuous space. However, designing an efficient projection evaluator for the high-dimensional constrained space is nontrivial, hence it shows the worst performance in the simulations. From Fig. 2 and Table II, we can also get the following results.

Overall, the performance of PJ, AT and TB increases sequentially on these tasks, which is similar to the results reported in the literature. They work well when the scenes are simple, like the task in Fig. 2(a). When obstacles increase, their performance degrades significantly. Moreover, we also found that PJ-like frameworks are sensitive to the hyper-parameters when debugging the code, such as the constraint tolerance in PJ and the maximum radius of the chart validity region in AT and TB, which puts extra burdens on the developers.

From Table II, we know that IKCL outperforms PJ-like frameworks significantly. This owes to the proposed framework as we plan the motion of the object and the robots hierarchically while PJ-like frameworks plan the motion of the system simultaneously. Comparing the results of IKCL with CMCL, we can see that CM can further improve the performance of the proposed motion planner. The combination of these features makes our motion planner outperform the benchmark significantly. In addition, RRTConnect (or RRT) shows the best performance when working with our framework, hence it is chosen as the default algorithm in the following experiments.

3) Simulation 2: In this simulation, we compare the proposed hierarchical framework with the decoupled framework. When the closed-chain constraint and the lower bound of the formation constraint are not guaranteed in Fig. 2 or ASR is not checked in Alg. 1 (line 28-33), the centralized layer and the decentralized layer become fully decoupled, resulting in a framework similar to [44]. The performance of the decoupled framework in the tasks in Fig. 2(a)-(f) is shown in Table III.

When the scenes and the tasks are simple, the centralized layer and the decentralized layer are always compatible, so checking the closed-chain constraint and the lower bound of the formation constraint can be avoided. Therefore, the decoupled framework performs well in Fig. 2(a)-(b). However, obstacles will complicate the motion planning problem and
TABLE III
PERFORMANCE OF THE DECOUPLED FRAMEWORK [44]

| Task | RRT [59] | RRTConnect [47] | BKPIECE [60] | EST [42] | STRIDE [81] | PRM [15] |
|------|----------|----------------|-------------|---------|-------------|---------|
|      | Success/Total | Time(s) | Success/Total | Time(s) | Success/Total | Time(s) | Success/Total | Time(s) | Success/Total | Time(s) |
| Fig. 9(a) | 10/10 | 0.06±0.01 | Fig. 9(b) | 10/10 | 0.05±0.02 | Fig. 9(c) | 10/10 | 0.77±0.35 | Fig. 9(d) | 10/10 | 0.08±0.04 |
| Fig. 9(e) | 10/10 | 0.05±0.01 | Fig. 9(f) | 10/10 | 0.53±0.20 | Fig. 9(g) | 10/10 | 0.07±0.03 | Fig. 9(h) | 10/10 | 0.09±0.03 |
| Fig. 9(i) | 10/10 | 0.07±0.04 | Fig. 9(j) | 10/10 | 0.07±0.02 | Fig. 9(k) | 10/10 | 0.07±0.03 | Fig. 9(l) | 10/10 | 20.01±0.00 |

Fig. 10. Robot path planned by the decoupled framework [44]. Although the motion of the object is collision-free, [44] is unable to find a valid path for the robots that satisfies the collision, closed-chain and formation constraints simultaneously.

make the two layers in conflict. Therefore, in Fig. 9(c)-(f), the performance of the decoupled framework decreases dramatically. A failure case is shown in Fig. 10. As can be seen, although the motion of the object is collision-free, [44] cannot find a valid path for the robots that satisfies the collision, closed-chain and formation constraints simultaneously.

4) Simulation 3: In this simulation, we compare the proposed hierarchical framework with algorithms based on the virtual structure. In [39], the object and the robots are encircled by polygons on the ground. As long as the virtual structure intersects with obstacles, the system is considered in collision. Therefore, the obstacle-avoidance strategy is conservative. Take the task in Fig. 9(b) as an example. Our motion planner is able to find a shorter path that crosses obstacles. However, the path in Fig. 9(b) becomes infeasible for [39], and they can only find a longer path that bypasses obstacles, which is shown in Fig. 11. Crossing obstacles is an essential ability for the multiple robots system and rarely mentioned in the previous works. It changes the topological structure of the configuration space and will increase the success rate of the motion planner in cluttered environments.

F. Real-World Experiments

In this part, we extend the proposed motion planner to real-world multiple mobile manipulators. \( v_{\text{max}}, t_{\text{opt}} \) and \( \epsilon \) in Alg. 4 are set to 0.1m/s, 40ms and 0.5, respectively. \( K_{b,i} \) and \( K_{e,i} \) in the task-priority motion controller are set to 2 \( \times I_{3 \times 3} \) and 2 \( \times I_{6 \times 6} \), respectively. \( f_r(.) \) in Eq. 11 is chosen as the sixth-order polynomial due to its continuity and simplicity. \( \text{thres} \) is set to 0.4 to couple the centralized and decentralized layers.

1) Experiment 1: In this experiment, we conduct the cooperative transportation task by two robots in real-world environment. Obstacles include a pedestrian and two slopes with varying angles and heights. The system should transport the object to the destination while avoiding static and dynamic obstacles. The decentralized layer will optimize the formation of the system in real-time and computes the desired joint motion command based on the task-priority motion controller. Snapshots during the task are shown in Fig. 12. The trajectories of the object, and \( \mu(q) \) and \( r(q) \) of \( \text{MM1} \) and \( \text{MM2} \) are shown in Fig. 13. The motion process is explained as follows.

- At \( t=0s \), the system is far away from obstacles. Both \( \text{MM1} \)
From $t=5s$ to $t=10s$, $MM1$ meets the pedestrian twice. As a redundant robot, it is able to avoid the dynamic obstacle while not affecting the transportation task. As a punishment, $\mu(q)$ of $MM1$ is sacrificed (Fig. 13(c)). However, when the obstacle moves away, $\mu(q)$ can be recovered to a high value.

- From $t=28s$ to $t=35s$, the system adjusts $z$ and roll of the object to suit the height and direction of slope1. Thanks to the real-time formation optimizer, $\mu(q)$ and $r(q)$ of the two robots remain at a high value during this period.

- From $t=49s$ to $t=55s$, the height of the $MM1$’s and $MM2$’s end effector decreases and increases, respectively, which means that the system adjusts pitch of the object to suit the height and direction of slope2. During this period, $MM2$ gradually approaches the boundary of the workspace in $z$ direction, hence $\mu(q)$ decreases quickly (Fig. 13(d)).

- From $t=75s$ to $t=80s$, the system has crossed all obstacles and then lowers the object to finish the autonomous cooperative transportation task. Finally, the height of the $MM2$’s end effector decreases, and $\mu(q)$ returns to a high value.

2) Experiment 2: In this experiment, we conduct the cooperative transportation task by three robots in real-world environment. Obstacles include a pedestrian and a cuboid with a certain height. Other settings are similar to the former experiment. Snapshots during the task are shown in Fig. 14 and the pictorial motion is available in the attached video.

From this and the former experiments, we can see that multiple robots are able to cooperate with each other to realize the transportation task in cluttered environments. Moreover, each robot could plan the optimal joint configuration and track the desired trajectories precisely.

G. Discussions

As can be seen from Fig. 2, the proposed hierarchical framework is partially coupled and partially decoupled. Coupling occurs between different layers, which means that we guarantee the closed-chain constraint $f_{c\odot}(c)$ and the lower bound of the formation constraint $f_{FC}(c)$ in the centralized layer, and then optimize $f_{FC}(c)$ on the basis of obeying $f_{c\odot}(c)$. Decoupling arises among different robots, which means that the decentralized layer is fully distributed, and the redundancy of each robot can be explored independently in real-time.

As can be seen from Fig. 2, the proposed hierarchical framework is partially coupled and partially decoupled. Coupling occurs between different layers, which means that we guarantee the closed-chain constraint $f_{c\odot}(c)$ and the lower bound of the formation constraint $f_{FC}(c)$ in the centralized layer, and then optimize $f_{FC}(c)$ on the basis of obeying $f_{c\odot}(c)$. Decoupling arises among different robots, which means that the decentralized layer is fully distributed, and the redundancy of each robot can be explored independently in real-time.

Compared with the centralized frameworks, which use the numerical projection operator to generate valid samples and plan the motion of the object and the robots simultaneously, the hierarchical feature can simplify the motion planning problem of the high-dimensional system, which is verified in Section V-E2. Compared with the decoupled frameworks, which plan the motion of the object and the robots separately, coupling the two layers can avoid conflict between them and enhance the performance of the motion planner significantly, which is verified in Section V-E3. Compared with the virtual structure-based algorithms, the proposed motion planner exhibits diversified and efficient obstacle-avoidance skills, such as bypassing and crossing, which is also verified in Section V-E4. The features of different frameworks when dealing with the closed-chain, formation, redundancy and obstacle-avoidance constraints are summarized in Table IV.

Another advantage of the proposed motion planner is $CM$, and its effects are twofold. On the one hand, it accelerates the performance of the centralized layer, which is verified by comparing IKCL with CMCL. On the other hand, it helps the optimization algorithm determine the seed, which speeds up the decentralized layer and effectively avoids local minimum. $CM$ is essential when the motion planning time is a key index. However, the disadvantage of $CM$ is that extra storage space is needed, and it has to be loaded from the local hard disk when the program starts. In the future, $CM$ can be saved in the cloud server and treated as an infrastructure to help boost the motion planner.

The proposed framework can be applied to different numbers of heterogeneous mobile manipulators. To the best of our knowledge, such complex applications have not been seen...
TABLE IV
FEATURES OF DIFFERENT MOTION PLANNING FRAMEWORKS WHEN DEALING WITH DIFFERENT CONSTRAINTS

| Method                     | Closed-Chain | Formation | Redundancy | Obstacle-Avoidance | Other Key Features |
|----------------------------|--------------|-----------|------------|-------------------|-------------------|
| Centralized Framework      | ✓            | ✗         | ✗          | ✗                 | numerical projection, time-consuming |
| Decoupled Framework        | ✓            | ✗         | ✗          | ✗                 | conflict between different layers |
| Virtual Structure-based    | ✓            | ✓         | ✓          | ✓                 | only planar object’s motion |
| Hierarchical Framework (ours) | ✓          | ✓         | ✓          | ✓                 | time-efficient; compatibility between different layers; spatial object’s motion |

before in the robotic community. However, there is still a long way to go to deal with multiple heterogeneous robots. For example, the speed, payload, sensing capability, kinematic and dynamic characteristics of each robot may be different, hence the planner should allocate proper motion for each robot. In addition, some robots may crash when working, and the motion planner should monitor the status of each robot and prepare the emergency measure for such cases.

Another interesting ability for multiple mobile manipulators is avoiding dynamic obstacles. A simple case is shown in the experiment. However, dynamic obstacles may be complex. For example, they may be fast and lock the motion of all robots simultaneously. In such cases, what strategy to take and how to balance the motion of different robots are still open questions.

VI. CONCLUSIONS

In this paper, we proposed a hierarchical framework to deal with the motion planning of multiple redundant mobile manipulators. The motion of the object is planned in the centralized layer offline. In the decentralized layer, the redundancy of each robot is explored independently in real-time to achieve the desired formation and dexterous configuration. Closed-chain, obstacle-avoidance and the lower bound of the formation constraints are checked in the centralized layer to ensure the object’s motion is executable in the decentralized layer. The hierarchical framework simplifies the complex motion planning problem while not violating the constraints, and its superiority is verified in the experiments.

In addition, a novel tool named CM is introduced to boost the performance of the motion planner. The validity of the object’s pose in the centralized layer can be checked quickly by looking up CM. The seed of the optimization algorithm in the decentralized can also be determined by querying it. The proposed framework outperforms the golden benchmark motion planners significantly and can be applied to different numbers of heterogeneous mobile manipulators, which are also verified in the simulated and real-world experiments.

APPENDIX

A. Projection Operation for the Centralized Frameworks

Projection is the basis for the centralized frameworks \([32] \, [33] \, [35] \, [36]\) in Section V-E1. Alg. 5 shows the iterative procedure that uses the pseudo-inverse Jacobian matrix of the closed-chain constraint function \(f_{c3}(c)\).

Given a random sample \(c\), Alg. 5 computes the constrained error according to Eq. (4) and compares it with the maximum allowed error \(\varepsilon\). When the error is larger than \(\varepsilon\), the pseudo-inverse of the constrained Jacobian matrix \(J_{c3}^+(c)\) will be calculated to project the random sample by the gradient descent operation.

B. Constrained Jacobian Matrix

In this part, we derive the constrained Jacobian matrix \(J_{c3}(c) \in \mathbb{R}^{6n \times \sum_{i=1}^{n} (n_{a,i} + n_{b,i}) + 6}\) of the system.

\[
J_{c3}(c) = \frac{\partial f_{c3}(c)}{\partial c} = \frac{\partial E - \partial G}{\partial c} \tag{16}
\]

According to the definition of \(E\) and \(G\) in Section III-B, \(E\) has no relationship with \(r_{obj}\), and \(G\) has no relationship with \(q_i\). Therefore, Eq. (16) can be simplified to Eq. (17).

\[
\frac{\partial E - \partial G}{\partial c} = \left( \frac{\partial E}{\partial q_1}, \ldots, \frac{\partial E}{\partial q_{a,i}}, \frac{\partial E}{\partial q_{a,i}^m}, \frac{\partial G}{\partial t_{obj}^m} \right) \tag{17}
\]

where \(\frac{\partial E}{\partial q_i}\) and \(\frac{\partial G}{\partial t_{obj}^m}\) are defined in Eq. (18) and Eq. (19).

\[
\frac{\partial E}{\partial q_i} = \left( \frac{\partial f_1(q_i)}{\partial q_i}, \ldots, \frac{\partial f_k(q_i)}{\partial q_i}, \ldots, \frac{\partial f_k(q_n)}{\partial q_i} \right)^T \tag{18}
\]

\[
\frac{\partial G}{\partial t_{obj}^m} = \left( \frac{\partial g_{w,1}}{\partial t_{obj}^m}, \ldots, \frac{\partial g_{w,1}}{\partial t_{obj}^m}, \ldots, \frac{\partial g_{w,n}}{\partial t_{obj}^m} \right)^T \tag{19}
\]

where \(\frac{\partial f_k(q_i)}{\partial q_i}\) is given by Eq. (20).

\[
\frac{\partial f_k(q_i)}{\partial q_i} = \begin{cases} J_i(q_i), & \text{if } i = j \\ O_{6 \times (n_{a,i} + n_{b,i})}, & \text{if } i \neq j \end{cases} \tag{20}
\]

where \(J_i(q_i)\) is the analytical Jacobian matrix of the \(i\)th MM. Suppose the constant homogeneous transformation of frame \(O_{g,i}X_{g,i}Y_{g,i}Z_{g,i}\) relative to frame \(O_{obj}X_{obj}Y_{obj}Z_{obj}\) is \(X_{g,i}^{obj} = \begin{pmatrix} R_{g,i}^{obj} & p_{g,i}^{obj} \\ O_{1 \times 3} & 1 \end{pmatrix} \).
Suppose the velocity of a frame is \( \mathbf{\xi} = (v^T, \omega^T)^T \), in which \( v \) and \( \omega \) represent the linear and angular velocity of the frame. According to [63], Eq. (21) holds between \( \mathbf{\xi}^{w_{g,i}} \) and \( \mathbf{\xi}^{w_{obj}} \):

\[
\mathbf{\xi}^{w_{g,i}} = \begin{pmatrix} I_{3 \times 3} & -S(\mathbf{p}_{w_{g,i}}^i) \end{pmatrix} \mathbf{O}_{3 \times 3}^{w_{obj}} \tag{21}
\]

where \( S(\cdot) \) is the skew-symmetric matrix operator. In addition, we use a minimum description to represent the orientation in \( t \), hence Eq. (22) holds between \( t \) and \( \mathbf{\xi} \):

\[
\mathbf{\xi} = \begin{pmatrix} I_{3 \times 3} & \mathbf{O}_{3 \times 3}^{w} \end{pmatrix} \mathbf{B}(\alpha) \mathbf{t} \tag{22}
\]

When roll-pitch-yaw angle is used to represent the orientation, we have \( \alpha = (\phi, \psi, \theta)^T \) and \( \mathbf{B}(\alpha) = \begin{pmatrix} c_{\phi}c_{\psi} - s_{\phi}s_{\psi} & c_{\phi}\theta & s_{\phi}\theta \\ c_{\psi} & c_{\psi}\theta & s_{\psi}\theta \\ -s_{\phi} & s_{\phi}\theta & c_{\phi} \end{pmatrix} \), where \( s \) and \( c \) represent sine and cosine operator, respectively. Combining Eq. (21) and Eq. (22), \( \frac{\partial t_{g,i}^{w}}{\partial \mathbf{\xi}_{obj}} \) can be derived in Eq. (23):

\[
\frac{\partial t_{g,i}^{w}}{\partial \mathbf{\xi}_{obj}} = \mathbf{W}_i = \begin{pmatrix} I_{3 \times 3} & -S(\mathbf{p}_{w_{g,i}}^{w})\mathbf{B}(\mathbf{\alpha}_{w_{obj}}^{w}) \end{pmatrix} \mathbf{O}_{3 \times 3}^{w} \mathbf{B}(\mathbf{\alpha}_{obj}) \tag{23}
\]

Combining Eq. (16)-(20) and Eq. (23), \( \mathbf{J}_{g\cdot i}(c) \) is finally given in Eq. (24):

\[
\mathbf{J}_{g\cdot i}(c) = \begin{pmatrix} J_1(q_1) & \ldots & \mathbf{O} & -\mathbf{W}_0 \\
\vdots & \ddots & \vdots & \vdots \\
\mathbf{O} & \ldots & J_n(q_n) & -\mathbf{W}_n \end{pmatrix} \tag{24}
\]

REFERENCES

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield et al., “The grand challenges of science robotics,” *Science robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[2] J. K. Parrish and L. Edelstein-Keshet, “Complexity, pattern, and evolutionary trade-offs in animal aggregation,” *Science*, vol. 284, no. 5411, pp. 99–101, 1999.

[3] L. E. Parker, D. Rus, and G. S. Sukhatme, “Multiple mobile robot systems,” in *Springer Handbook of Robotics*. Springer, 2016, pp. 1335–1384.

[4] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham, “Can robots assemble an ikea chair?”, *Science Robotics*, vol. 3, no. 17, p. eaat6385, 2018.

[5] S.-J. Chiang, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, “A survey on aerial swarm robotics,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[6] A. Branch, M. M. Flexas, B. Claas, A. F. Thompson, Y. Zhang, E. B. Clark, S. Chien, D. M. Fratantoni, J. C. Kinsey, B. Hobson et al., “Front delineation and tracking with multiple underwater vehicles,” *Journal of Field Robotics*, vol. 36, no. 3, pp. G68–S86, 2019.

[7] K. H. Petersen, N. Napp, R. Stuart-Smith, D. Rus, and M. Kovac, “A review of collective robotic construction,” *Science Robotics*, vol. 4, no. 28, p. eaau8479, 2019.

[8] D. Koug, O. Kermorgant, I. Fantoni, and L. Belouaer, “Cooperative multi-robot object transportation system based on hierarchical quadratic programming,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6406–6417, 2021.

[9] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, “Search and rescue under the forest canopy using multiple uavs,” *The International Journal of Robotics Research*, vol. 39, no. 10-11, pp. 1201–1221, 2020.

[10] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal, “Robots in human environments: Basic autonomous capabilities,” *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 684–696, 1999.

[11] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” *Automatica*, vol. 53, pp. 424–440, 2015.
[35] L. Juillet and J. M. Porta, “Path planning with loop closure constraints using an atlas-based rrt,” in *Robotics Research*. Springer, 2017, pp. 345–362.

[36] 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.

[37] M. Gharbi, J. Cortés, and T. Simeon, “A sampling-based path planner for dual-arm manipulation,” in *2008 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 2008, pp. 383–388.

[38] Q. Tang, Y. Zhang, F. Yu, and J. Zhang, “An obstacle avoidance approach based on system outlined rectangle for cooperative transportation of multiple mobile manipulators,” in *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*. IEEE, 2018, pp. 533–538.

[39] J. Alonso-Mora, S. Baker, and D. Rus, “Multi-robot formation control and object transport in dynamic environments via constrained optimization,” *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[40] J. Jiao, Z. Cao, N. Gu, S. Nahavandi, Y. Yang, and M. Tan, “Transportation by multiple mobile manipulators in unknown environments with obstacles,” *IEEE Systems Journal*, vol. 11, no. 4, pp. 2894–2904, 2015.

[41] J. Hu, W. Liu, H. Zhang, J. Yi, and Z. Xiong, “Multi-robot object transport motion planning with a deformable sheet,” *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9350–9357, 2022.

[42] J. Cheong, “A consistent kinematic modeling method for mobile manipulators,” in *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, 2010, pp. 805–810.

[43] 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.

[44] T. Hekmatfar, E. Masehian, and S. J. Mousavi, “Cooperative object transportation by multiple mobile manipulators through a hierarchical planning architecture,” in *2014 Second RSJ/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 2014, pp. 503–508.

[45] 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.

[46] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in *Proceedings of International Conference on Robotics and Automation*, vol. 3. IEEE, 1997, pp. 2719–2726.

[47] 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.

[48] I. A. Şucan and S. Chitta, “Motion planning with constraints using configuration space approximations,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1904–1910.

[49] D. Coleman, I. Sucan, S. Chitta, and N. Correll, “Reducing the barrier to entry of complex robotic software: a moveit! case study,” *arXiv preprint arXiv:1404.3785*, 2014.

[50] O. Porges, T. Stouraitis, C. Borst, and M. A. Roa, “Reachability and capability analysis for manipulation tasks,” in *ROBOT2013: First Iberian robotics conference*. Springer, 2014, pp. 703–718.

[51] S. Patel and T. Sobh, “Manipulator performance measures-a comprehensive literature survey,” *Journal of Intelligent & Robotic Systems*, vol. 77, no. 3, pp. 547–570, 2015.

[52] T. Yoshikawa, “Manipulability of robotic mechanisms,” *The international journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[53] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[54] Stanford Artificial Intelligence Laboratory et al., “Robotic operating system.” [Online]. Available: https://www.ros.org

[55] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[56] R. Smits, “KDL: Kinematics and Dynamics Library.” [http://www.orocos.org/kdl]

[57] P. Beeson and B. Ames, “Trac-ik: An open-source library for improved solving of generic inverse kinematics,” in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015.

[58] R. Diankov, “Automated construction of robotic manipulation programs,” 2010.