Abstract—Closed kinematic chains are created whenever multiple robot arms concurrently manipulate a single object. The closed-chain constraint, when coupled with robot joint limits, dramatically changes the connectivity of the configuration space. We propose a regrasping move, termed “IK-switch,” which allows efficiently bridging components of the configuration space that are otherwise mutually disconnected. This move, combined with several other developments, such as a method to stabilize the manipulated object using the environment, a new tree structure, and a compliant control scheme, enables us to address complex closed-chain manipulation tasks, such as flipping a chair frame, which is otherwise impossible to realize using existing multi-arm planning methods.

Index Terms—Dual arm manipulation, manipulation planning, motion and path planning.

I. INTRODUCTION

Bimanual or, more generally, multi-arm robotic systems are necessary to manipulate large and heavy objects. It is however much more challenging to plan and control multi-arm motions than single-arm motions, because of the closed-chain kinematic constraint. The closed-chain constraint affects multi-arm motions at different levels.

First, at the “local” level, the feasible configurations of a closed-chain system are restricted to a sub-manifolds of a lower dimension than the configuration space. Thus, connecting nearby configurations by a valid path is non-trivial, and requires projection or differential IK techniques [1]–[3].

Second, when using sampling-based motion planners such as PRM [4] or RRT [5], one needs to generate a large number of evenly distributed feasible configurations (which will be connected to each other through local motions). As the set of feasible configurations is of lower dimension, its volume is zero, which again requires non-trivial modifications to the sampling method. We call this the “(connected) component” level.

We identify in this letter another level, termed the “global” level, which encompasses different connected components that are mutually disconnected. Indeed, the closed-chain constraint, when coupled with robot joint limits, dramatically changes the connectivity of the set of feasible configurations. Fig. 1(a) illustrates this point: because of the closed-chain constraint and the joint limits of the small green arm, the big blue arm cannot switch from the upper configuration to the lower configuration.

With the “local” and “component” levels being relatively well understood [1], [3], [6]–[8], it is the “global” level that constitutes a major hurdle when deploying multi-arm systems to address practical tasks, such as flipping a chair frame as shown in Fig. 1(b) and (c).

Contributions and Organization of the Letter

Tasks involving large or heavy objects can be impractical for single-arm systems because of payload and/or gripper strength limitations. Our goal here is to develop a planning and control framework for multi-arm systems to carry out such tasks, such as flipping a chair frame as in Fig. 1(b, c). For this, we introduce the following contributions:

1) a move, termed “IK-switch”, which is a regrasping move that allows connecting different components that are otherwise mutually disconnected. We argue that such “IK-switch” moves help address the “global” problem discussed previously;
2) a method to use the environment to help stabilize large manipulated objects during “IK-switch” moves;
3) a tree structure adapted to the “IK-switch” move, which allows accelerating the planning;
4) the integration of the above contributions into a planning and control framework that can tackle complex manipulations. In particular, we implement a compliant control scheme that allows executing closed-chain motions under model uncertainty. We showcase the framework on a difficult manipulation task: flipping a chair frame using two robot arms. Planning time is less than 20 seconds and execution is smooth, as shown in the accompanying video. We have not seen in the literature a demonstration of planning and execution of a bimanual task of such a level of complexity.

Fig. 1. The closed-chain constraint, coupled with robot joint limits, changes the connectivity of the configuration space. (a): Suppose that the small green arm cannot bend backward, then the big blue arm cannot switch from the upper configuration to the lower configuration. (b,c): There is no continuous motions to flip a chair frame between the start configuration (b) and the goal configuration (c).
The letter is organized as follows. In Section II, we discuss related works in closed-chain motion planning and manipulation planning using regrasping. In Section III, we analyze in detail how the closed-chain constraint changes the connectivity of the configuration space and formulate the planning problem. In Section IV, we present the core technical contributions of this letter that allow efficiently addressing complex closed-chain manipulation tasks. In Section V, we describe the simulations and hardware experiments (which include the challenging task of flipping of a chair frame using two robot arms) to validate the proposed framework. Finally, Section VI draws a conclusion of our proposed approach.

II. RELATED WORKS

A. Motion Planning With the Closed-Chain Constraint

Direct sampling in configuration space has zero probability of generating a random configuration which satisfies closed-chain constraints. This is due to the fact that the constraint manifold has its dimension lower than that of the ambient space [1]. To generate a random closed-chain configuration, the authors of [6] proposed to break the closed-chain into several (open) sub-chains. A configuration of one sub-chain can be directly sampled and the configurations of other sub-chains computed so as to close the kinematic loop. This method was further refined in [9]. Random Gradient Descent was used in [1] to move a randomly sampled configuration toward a constraint manifold. In more recent work, [7] and [8] sample configurations on a tangent space of the constraint manifold, and [3] used the Newton-Raphson method for projection on the constraint manifold to obtain valid configurations and paths. However, while all these planners might be able to find a path (if it exists) within a single component, they lack the ability to address the problem at a “global” level.

B. Regrasping

Although regrasping itself is merely a robot breaking and re-initiating contacts (grasps) with an object, how to do regrasping in such a way that facilitates manipulation of the object into its desired goal transformation is not trivial. Several tools, including Grasp-Placement Table [10], Regrasp Graph [11], high-level Grasp-Placement Graph [12], have been devised to help reason over a large number of possible combinations of grasps and placements such that the planner can choose only a few combinations that would sufficiently bring the system toward the goal.

Previous work considering regrasps used such moves mainly for the purpose of changing grasps, either one robot changing from one grasp to another or changing from one robot grasping to another robot grasping [13], [14]. In this work, we utilize regrasping moves not necessarily to change grasps. Instead, we use them to establish bridges between different disconnected components, which is essentially useful in planning.

C. Bimanual Manipulation Planning

A pioneering work in this direction was published in [15]. In the letter, the authors presented three manipulation planning algorithms for two-arm robotic systems. The first two algorithms employed exhaustive search over discretized configuration space and therefore could only solve some simplified planar bimanual manipulation planning problems. The third algorithm adapted the randomized potential field technique [16] to work with a closed-chain system. It used regrasping as a way to escape once trapped in a local minimum in the potential field. Although the work itself is interesting and the authors also provided some basic understanding and characterization of the problem, they totally disregarded joint limits in the planning and the planners could only cope with very limited ranges of problems.

In [17], the authors presented a dual-arm motion planner which was able to plan motions crossing different closed-chain-induced manifolds via singular configurations. Although motions generated by this planner will not require regrasping, the method itself relies on the full knowledge of IK classes characterization and the ability to sample directly singular configurations, both of which may not usually be available in practice.

More recent work addressing bimanual manipulation planning exist. They, however, either do not consider any closed-chain motions [13], [14], [18], or use heuristic search over a discretized configuration space [19] which is only capable of planning simple motions.

However, they either consider using two arms only for increasing workspace and therefore not considering any closed-chain motions [13], [14], or use heuristic search over a discretized configuration space [19] which is only capable of planning simple motions.

III. PROBLEM FORMULATION AND ANALYSIS

In this Section, we define some mathematical notations for subsequent discussions and present a formal formulation of the problem. Moreover, we analyze the problem by dividing it into different cases that might be encountered in planning, and present the “IK-switch” move to address them.

A. Closed-Chain Constraint

Consider a system consisting of \( k \) robots and a movable object. Let \( C^i_{\text{robot}} \subseteq \mathbb{R}^{n_i} \) be the configuration space of the \( i^{\text{th}} \) robot, where \( n_i \) is its number of degrees-of-freedom (DOF), and \( C_{\text{obj}} \subseteq SE(3) \) the configuration space of the object. The composite configuration space of the system is described as \( C_{\text{composite}} = C^1_{\text{robot}} \times C^2_{\text{robot}} \times \ldots \times C^k_{\text{robot}} \times C_{\text{obj}} \). A composite configuration \( c \in C_{\text{composite}} \) can then be written as \( c = (q_1, q_2, \ldots, q_k, T_{\text{obj}}) \), where \( q_i \in C^i_{\text{robot}} \) is the configuration of the \( i^{\text{th}} \) robot and \( T_{\text{obj}} \in SE(3) \) is the homogeneous transformation of the object.

When all the robots are grasping the object with their end-effectors, the system forms closed kinematic chains. In this case, the composite configuration \( c \) implicitly determines \( G \), the set of grasping poses of the robots. In other words, it defines the relative transformations from the object to the end-effector of each robot. This constraint can be described in the form \( F_G(c) = 0 \), where \( 0 \) is a zero vector of appropriate dimension. Let \( C_{\text{cc}} \subseteq C_{\text{composite}} \) be defined as

\[
C_{\text{cc}} = \{ c \mid c \in C_{\text{composite}}, F_G(c) = 0 \}.
\]

Excluding singularities, \( C_{\text{cc}} \) is a set of manifolds of a lower dimension lying in \( C_{\text{composite}} \) [1], [20].

B. Essentially Mutually Disconnected (EMD) Components

**Definition 1:** Given a feasible configuration \( c \), we define the (connected) component \( S(c) \) as the set of all feasible configurations which can be reached from \( c \) by continuous and feasible
paths (i.e., paths that are collision-free and respect the closed-chain constraint and robot joint limits). Two components, \( S(c_1) \) and \( S(c_2) \), are essentially mutually disconnected (EMD) if they are indeed disconnected or if, in practice, one cannot find any connection between the two within a reasonable amount of time.

Note that we use the term "essentially" in the above definition as it is very difficult, in an actual problem instance, to provide a rigorous certificate that two components are indeed disconnected. Consider the system in Fig. 1(a). One can clearly see that the components containing respectively the upper configuration and the lower configuration are disconnected. Yet a certificate of this disconnectedness would involve complex trigonometry formulae. Such certificates are even more difficult, if not impossible, to obtain in high-DOF systems such as in Fig. 1(b) and (c). One can however say that the configurations in Fig. 1(b) and (c) are essentially mutually disconnected after running state-of-the-art planners – without regrasping – for hours without finding any solution.

In order to bridge EMD components, our planning algorithm plans not only the component-level closed-chain motions, but also regrasping moves that help the system "jump" across different EMD components. This significantly enlarges the size of the solution space.

C. Problem Formulation

In addition to \( C_{\text{cc}} \) which satisfies the closure constraint, we denote by \( C_{\text{free}} \subseteq C_{\text{composite}} \) the set containing all collision-free composite configurations. Moreover, define \( \pi : C_{\text{composite}} \rightarrow SE(3) \) as a projection from a composite configuration space to \( C_{\text{obj}} \) such that for \( c = (q_1, q_2, \ldots, q_k, T_{\text{obj}}) \), \( \pi(c) = T_{\text{obj}} \). In a general closed-chain motion planning problem for a multi-arm system, typically one is given a start composite configuration which imposes a closure constraint on the system, and a goal configuration of the object; with regards to the goal configuration, grasping pose of each robot is pre-determined by the start composite configuration, while the specific configuration is unknown. In addition, in cases when a multi-arm robot system is required, it is certain that the object’s contact stability is critical. By using the notations presented above, such a problem can be stated as follows.

Problem 1: Given a start composite configuration \( c_{\text{start}} \in C_{\text{cc}} \cap C_{\text{free}} \) and a goal object configuration \( T_{\text{obj}} \), find a path \( P : [0, 1] \rightarrow C_{\text{cc}} \cap C_{\text{free}} \) such that

\[
P(0) = c_{\text{start}},
\]

\[
\pi(P(1)) = T_{\text{obj}},
\]

and the system maintains contact stability throughout \( P \).

D. Problem Analysis

For convenience, given an EMD component \( S \), we define a projected space \( \Pi(S) \) as \( \Pi(S) = \{ \pi(c) \mid c \in S \} \).

Consider Problem 1. When given the goal object transformation \( T_{\text{goal}} \), there exists multiple \( e_{\text{goal}} \) since different inverse kinematic (IK) solutions exists for a certain set of end-effector transforms. Let \( e_{\text{goal}} = \{ e_1^{\text{goal}}, e_2^{\text{goal}}, \ldots, e_m^{\text{goal}} \} \) denote the collection of all \( m \) possible goal composite configurations. With regards to the relation between \( e_{\text{goal}} \) and \( \pi(e_{\text{goal}}) \), two possible cases exist as follows.

\[\text{Case 1: } \exists e_{\text{goal}} \in e_{\text{goal}} \text{ such that } e_{\text{goal}} \in S(c_{\text{start}}).\]

This means that there exists a feasible path \( P \) from \( c_{\text{start}} \) to some \( e_{\text{goal}} \) with no regrasping. However, there might exists other \( e_{\text{goal}} \in e_{\text{goal}} \) which do not lie in \( S(c_{\text{start}}) \). In the searching efficiency of a bi-directional planner, such as a BiRRT [5], is desired, we need to manually select a goal configuration \( e_{\text{goal}} \in e_{\text{goal}} \) which is also in \( S(c_{\text{start}}) \). However, to the best of our knowledge, there is currently no effective method for such selection. It is likely that for \( e_{\text{goal}} \) selected based on certain heuristics, \( e_{\text{goal}} \notin S(c_{\text{start}}) \). This would fall into sub-cases discussed in Case 2.

One possible approach to avoid selecting goal configurations is to extend the idea of BiSpace planning [21] to such closed-chain systems. However, this approach is limited to the condition that an ideal \( e_{\text{goal}} \) exists in \( S(c_{\text{start}}) \), and cannot handle the cases presented below.

\[\text{Case 2: } \forall e_{\text{goal}} \in e_{\text{goal}}, e_{\text{goal}} \notin S(c_{\text{start}}).\]

In this case, all possible goal configurations are essentially mutually disconnected from \( c_{\text{start}} \). Therefore, regrasping is necessary for the system to traverse different \( S \) to reach a goal configuration. Considering choosing \( e_{\text{goal}} \in e_{\text{goal}} \) at random. There are two sub-cases as follows.

\[\text{Case 2.1: } \Pi(S(c_{\text{start}})) \cap \Pi(S(e_{\text{goal}})) \neq \emptyset.\]

Since the intersection is not empty, there exists a path in \( \Pi(S(c_{\text{start}})) \cup \Pi(S(e_{\text{goal}})) \) for the object to move from \( \pi(c_{\text{start}}) \) to \( \pi(e_{\text{goal}}) \). The required regrasping action can be done once the object configuration is in the intersection, with the kinematic chain jumping from \( S(c_{\text{start}}) \) to \( S(e_{\text{goal}}) \).

\[\text{Case 2.2: } \Pi(S(c_{\text{start}})) \cap \Pi(S(e_{\text{goal}})) = \emptyset.\]

No path exists in \( \Pi(S(c_{\text{start}})) \cup \Pi(S(e_{\text{goal}})) \) to bring the object from \( \pi(c_{\text{start}}) \) to \( \pi(e_{\text{goal}}) \).

Let \( S_{\text{inter}} = S(c_{\text{start}}) \) and \( S_{\text{inter}}^{i+1} = S(e_{\text{goal}}) \). The problem is solvable if and only if there exists \( p \leq 1 \) intermediate EMD components, denoted by \( S_{\text{inter}}, S_{\text{inter}}^2, \ldots, S_{\text{inter}}^p \), such that

\[
\Pi(S_{\text{inter}}^i) \cap \Pi(S_{\text{inter}}^{i+1}) \neq \emptyset \quad \forall i \in \{0, 1, \ldots, p\}. \quad (2)
\]

With the aid of theses intermediate EMD components, an object path can be found in \( \bigcup_{i=0}^{p-1} \Pi(S_{\text{inter}}) \). The intersections between projections of these components provide shared regions to bridge themselves together and in turn, connects \( S(c_{\text{start}}) \) and \( S(e_{\text{goal}}) \).

The discussion above summarizes possible scenarios that may be encountered in a planning problem. Note that in order to maintain contact stability, our planner will try to find feasible placement configurations for the object to seek support from the environment whenever a regrasping is necessary. Consider the case where a regrasping is needed to connect \( S(c_1) \) and \( S(c_2) \), the regrasping should be performed at a composite configuration \( e \) such that \( \pi(e) \) is a valid placement configuration, where regrasping trajectory can be found and the object remains in static equilibrium throughout the regrasping process.

E. IK-Switch

We now discuss in detail the “IK-switch” move that allows bridging different EMD components, see Fig. 2. For a given 6D end-effector pose (translation and rotation), a robot arm with 6 revolute joints has up to 16 IK solutions. Different IK solutions can belong to different EMD components, see Fig. 1(a). Therefore, “jumping” between different IK solutions corresponding to the same end-effector pose can bridge different EMD components.
InterpolateSE3Path

ComputePath

is the maximum number of iterations allowed for tree extension. The planner also takes as its input a parameter \( N_{max} \), which sets the maximum number of regrasping attempts.

Parameter \( E \) is a description of the workspace environment, which will be used for planning IK-switch moves later. Some key functions in planning for global path are explained below.

1) SampleSE3Config samples a transformation matrix in \( SE(3) \). We separately sample orientation and translation parts. The orientation is uniformly sampled from Special Orthogonal Group \( SO(3) \) [23] while the translation is uniformly sampled from a user-defined range.
2) NearestNeighbor searches over all vertices in the given tree and returns the one with a transformation closest to \( T_{rand} \). We use the distance metric which is a combination of Euclidean distance (for translation part) and the minimal geodesic distance in \( SO(3) \) (for rotation part; see [24] for more details). To limit total number of regrasping attempts in the final path, this function also takes \( R_{max} \) as an input and ignores any vertices having \( \text{RegraspCount} \) greater than \( R_{max} \).
3) InterpolateSE3Path generates an \( SE(3) \) trajectory connecting the given transformations. The procedure used here is similar to the one in [25].
4) ComputePath generates a composite path required for the motion of the closed-chain system. The detail implementation is listed in Algorithm 2. It discretizes the input object path into a series of transformations, according to a pre-defined time step, and stores them in a list \( L_T \). It iterates through \( L_T \) and for each \( T_{obj} \), it calls ComputeCompositeConfig to compute corresponding composite configurations. At each time instant, we use a differential IK algorithm [2], [26] to generate a new IK solution for each robot.

We use a differential IK solver to ensure that each newly generated composite configuration remains in the same \( S \) as the previous ones. When the solver fails, we use IsNearBoundary to check if the failure is because some robots reach their configuration space boundaries. If this is the case, GetRegraspConfig will compute the most flexible composite configuration \( c_{regrasp} \). In particular, it computes all IK solutions (via OpenRAVE IKFast [27]) for the robot \( index \) to grasp the object at \( \pi(c_{goal}) \). Then in SelectMostFlexibleIK, we use a scoring function as a heuristic to choose the best solution (the one with highest score). Given the lower and upper joint limits of the robots, \( q_1 \) and \( q_u \), the score for a configuration \( q \) is \( (q_u - q) \times (q - q_1) \). Then a composite path \( P_{composite} \) can

\[ \begin{align*}
\text{Algorithm 1.} & \quad \text{Algorithm 2.} \\
\text{Stage 1: Planning Global Path} & \quad \text{Stage 2: Planning Composite Path}
\end{align*} \]

IV. PATH PLANNING

Based on the previous analysis, we propose here a planner that can address complex multi-arm manipulation tasks. Our planner is derived from the classical BiRRT structure and comprises two planning stages. In the first stage, it plans a global path for the closed-chain system, which includes

- segments storing closed-chain motions without breaking the chain and
- the vertices connecting these segments, including necessary IK-switch regrasping requests.

In the second stage, it completes the global path by planning IK-switch moves at tree vertices where regrasping is needed. Delaying IK-switch planning to the second stage helps improve efficiency of the planner sharply since most of regrasping requests will not be in the final path connecting \( c_{start} \) and \( c_{goal} \). Note that planning for IK-switch at these connecting vertices might fail when no feasible regrasping moves can be found. In this case, we implement an efficient data structure (see Section IV-C) that re-organizes all vertices stored in itself to retain the space information obtained previously before the planner returns to the first stage and re-plans a new global path.

A. Stage 1: Planning Global Path

Given a planning query determined by \( c_{start} \) and \( T_{goal} \), we first pick one composite configuration \( c_{goal} \) such that \( \pi(c_{goal}) = T_{goal} \), either randomly or according to certain heuristics, such as picking the one closest to \( c_{start} \). In a multi-robot system described previously, the movable object is the nexus linking all the robots. Thus, we take a decomposed approach to plan closed-chain motions: the planner plans a rigid body motion of the object in \( SE(3) \) first, then enforces the robots to follow the object’s path. The flow of the global path planner is summarized in Algorithm 1.

This global path planner grows two trees rooted at \( c_{start} \) and \( c_{goal} \). \( N_{max} \) is the maximum number of iterations allowed for tree extension. The planner also takes as its input a parameter \( R_{max} \), which sets the maximum number of regrasping attempts. Parameter \( E \) is a description of the workspace environment, which will be used for planning IK-switch moves later. Some key functions in planning for global path are explained below.

1) SampleSE3Config samples a transformation matrix in \( SE(3) \). We separately sample orientation and translation parts. The orientation is uniformly sampled from Special Orthogonal Group \( SO(3) \) [23] while the translation is uniformly sampled from a user-defined range.
2) NearestNeighbor searches over all vertices in the given tree and returns the one with a transformation closest to \( T_{rand} \). We use the distance metric which is a combination of Euclidean distance (for translation part) and the minimal geodesic distance in \( SO(3) \) (for rotation part; see [24] for more details). To limit total number of regrasping attempts in the final path, this function also takes \( R_{max} \) as an input and ignores any vertices having \( \text{RegraspCount} \) greater than \( R_{max} \).
3) InterpolateSE3Path generates an \( SE(3) \) trajectory connecting the given transformations. The procedure used here is similar to the one in [25].
4) ComputePath generates a composite path required for the motion of the closed-chain system. The detail implementation is listed in Algorithm 2. It discretizes the input object path into a series of transformations, according to a pre-defined time step, and stores them in a list \( L_T \). It iterates through \( L_T \) and for each \( T_{obj} \), it calls ComputeCompositeConfig to compute corresponding composite configurations. At each time instant, we use a differential IK algorithm [2], [26] to generate a new IK solution for each robot.

We use a differential IK solver to ensure that each newly generated composite configuration remains in the same \( S \) as the previous ones. When the solver fails, we use IsNearBoundary to check if the failure is because some robots reach their configuration space boundaries. If this is the case, GetRegraspConfig will compute the most flexible composite configuration \( c_{regrasp} \). In particular, it computes all IK solutions (via OpenRAVE IKFast [27]) for the robot \( index \) to grasp the object at \( \pi(c) \). Then in SelectMostFlexibleIK, we use a scoring function as a heuristic to choose the best solution (the one with highest score). Given the lower and upper joint limits of the robots, \( q_1 \) and \( q_u \), the score for a configuration \( q \) is \( (q_u - q) \times (q - q_1) \). Then a composite path \( P_{composite} \) can

\[ \begin{align*}
\text{Algorithm 1.} & \quad \text{Algorithm 2.} \\
\text{Stage 1: Planning Global Path} & \quad \text{Stage 2: Planning Composite Path}
\end{align*} \]
be generated from all feasible configurations in $L_c$ together with a IK-switch request.

5) Connect: When the planner attempts to connect the backward tree $T_b$ to a given vertex in $T_f$, it computes an allowed number of regrasping first for NearestNeighbor to select a vertex from $T_b$. Similar to Extend, it uses the differential IK solver to compute a composite path $P_{connect}$ from $V_{new}$ to $V_{new}$. If discrepancy exists between the last configuration in $P_{connect}$ and the one stored in $V_{new}$, we add one more regrasping request to $P_{connect}$ if the limit is not exceeded.

After $P_{connect}$ is returned from Connect, a global path is considered found, then the planner enters the second planning stage to plan for IK-switch moves.

B. Stage 2: Planning IK-Switch Moves

1) Path Planning: The planner calls PlanIKSwitch to generate paths for all the IK-switch requests in the global path computed in the first stage. Algorithm 3 presents the detailed flow. Key functions are discussed below.

ComputePath(Cont, $P_{filter}$, $P_{composite}$, $P_{regconnect}$)
1. Let $L_c$ be the discretized path.
2. $L_c$ ← EmptyList()
3. $c_{appr}$ ← Cont
4. for each $T_{obj}$ in $L_c$ do
5. if $c_{appr}$ is not None then
6. if $c_{appr}$ is not None then
7. $L_c$.append ($c_{appr}$)
8. if $c_{appr}$ is None then
9. $c_{appr}$ ← CompositePath($c_{appr}$, $T_{obj}$)
10. if $c_{appr}$ is None then
11. $c_{appr}$ ← CompositePath($c_{appr}$)
12. if $c_{appr}$ is None then
13. $c_{appr}$ ← TRAPPED
14. else
15. GetRegraspConfig($c_{appr}$, $T_{obj}$)
16. if $L_c$ is empty then
17. $q_i$ ← SelectNextFlexibleIK($L_c$
18. $r_{composite}$ ← UpdateCompositeConfig($c_{appr}$, $q_i$, $T_{obj}$)
19. return $r_{composite}$

Algorithm 3: IK-Switch Planner.

PlanIKSwitch($T_f$, $T_b$, $V_{fail}$, $V_{new}$)
1. for each $V$ in GlobalPathVertices($T_f$, $T_b$) do
2. if $V$ is Not New and Not $V_{hasRegrasp}$ then
3. $success$ ← False
4. for $i ← 1$ to $N_{max}$ do
5. $T_{plan}$ ← SamplePlacementConfig($V_{new}$, $T_f$, $T_b$)
6. if $T_{plan}$ is None then
7. $V_{fail}$ ← $V$
8. return False
9. $\{status, p_{appr}, P_{regrasp}\}$
10. ComputePath($V_{plan}$, $V_{plan}$, $V_{child}$)
11. if $status$ is failure then
12. $P_{regrasp} ← PlanRegraspPath($p_{appr}$, $P_{regrasp}$)
13. if $P_{regrasp}$ is None then
14. continue
15. $V_{hasRegrasp} ← True$
16. $V_{addRegrasp} ← PlanRegraspPath($p_{appr}$, $P_{regrasp}$)
17. $success ← True$
18. break
19. if $success$ then
20. $V_{fail} ← V$
21. return False
22. return True
placement configuration $T_{\text{place}}$ and checks if $T_{\text{place}}$ is feasible (i.e., is reachable by the robots and is in static equilibrium, as discussed below). If $T_{\text{place}}$ is feasible, it is returned. Otherwise, it continues until some maximum number of iterations is reached.

When performing re-grasping at $T_{\text{place}}$, there can be at least one robot holding the object at any time instant. Therefore, $T_{\text{place}}$ does not necessarily have to be stable. Instead, it only needs to stay in static equilibrium with the help of contact forces provided by grasping robots. Considering this, we can explore all types of contact with the convex hull of the object and the supporting surface (e.g., a floor or a table): face-face, edge-face, and vertex-face. For example, if we consider a contact of type edge-face, we proceed by finding the edge $e$ of the convex hull of the object at $T_{\text{obj}}$ closest to the supporting surface $S$. Then it adds small rotation to the object such that $e$ is parallel to $S$. Then the object is translated until it touches the surface. The resulting transformation is $T_{\text{place}}$. In our implementation, we compute several such transformations by adding small translational perturbations, parallel to $S$, to the original $T_{\text{place}}$. Then we check each one of them for static equilibrium.

To check static equilibrium of a placement configuration $T_{\text{place}}$ given that some robots are grasping the object, we re-formulate Newton-Euler equations as a matrix equation

$$
\begin{align*}
\begin{bmatrix}
-mg & -m(p_{\text{COM}} \times g) \\
0 & \mathbf{I}
\end{bmatrix}
\begin{bmatrix}
\mathbf{p}_1 \\
\mathbf{p}_2 \\
\vdots \\
\mathbf{p}_k
\end{bmatrix}
= \\
\begin{bmatrix}
I_3 & I_3 & \cdots & I_3 \\
[p_1] & [p_2] & \cdots & [p_k]
\end{bmatrix}
\begin{bmatrix}
\mathbf{f}_{\text{all}}
\end{bmatrix},
\end{align*}
$$

where $m$ is the mass of the object, $g$ the acceleration due to gravity, $p_{\text{COM}}$ the position of the COM of the object at $T_{\text{place}}$, $p_i$ the position of the $i^{th}$ contact point, $f_i$ the force exerted on the object at $p_i$, $k$ the total number of contact points, $w_{\text{GT}}$ is the gravito-inertial wrench [28], $f_{\text{all}} = [f_1^T \ f_2^T \ \cdots \ f_k^T]^T$, and the operator $[\cdot]$ maps a vector in $\mathbb{R}^n$ to a $3 \times 3$ skew-symmetric matrix. Note that we can take into account surface contacts by considering forces at the vertices of the contact area [28].

All the constraints related to contact forces (linearized friction-cone constraints and the max grip force constraint) can also be written as a vector inequality constraint $U f_{\text{all}} \preceq b$. Then the configuration in consideration is in static equilibrium if there exists $f_{\text{all}}$ satisfying 3 and all constraints. This feasibility problem can be solved via a linear programming solver or other available methods (c.f. [28], [29]).

C. Handling Failures in IK-Switch Planning

After a global path is found, the IK-switch planner starts planning IK-switch moves along the global path, from $V_{\text{goal}}$ in the forward tree $T_f$ to $V_{\text{goal}}$ in the backward tree $T_b$. In cases where a failure is encountered in this planning stage, the planner will return back to the first stage to find a new global path. In such cases, the failed vertex $V_{\text{fail}}$, of which the requested IK-switch cannot be found is useless; furthermore, all the child vertices in the subtree rooted at $V_{\text{fail}}$ become disconnected to $T_f$. In order to handle such failure properly so that all the information stored in these vertices can be retained for future use in regenerating a global path, we adopt a new variation of the BiRRT structure. In particular, our tree re-organizes itself in case of failure.

It abandons only the edge connecting $V_{\text{fail}}$ and its parent vertex, and keeps the edge connecting the two trees. This way, the disconnected subtree becomes a subtree of $T_b$. These two new $T_f$ and $T_b$ will then be used for re-planning the global path. A blacklisted region is set within a certain radius from $\pi(V_{\text{fail}}; c)$; no IK-switch request will be allowed in this region in subsequent planning. If all IK-switch requests are solved in $T_f$, the IK-switch planner starts from $V_{\text{goal}}$ and deals with $T_b$ similarly. This data structure helps improve our planner’s efficiency substantially, since with all the vertices retained, the planner only needs to find a path to bypass $V_{\text{fail}}$ so as to find a new global path. This idea would also be applicable to all other similar planning strategies containing multiple planning stages.

D. Properties of the Planner

The complexity of the planning problem is dependent on multiple properties of its configuration space, including dimension, expansiveness [30] and e-goodness [31]. With regards to the dimension, the complexity depends exponentially on the dimension of the configuration space of the robot system [32]. Increasing the system’s DOF will therefore exponentially increase the complexity, hence the planning time.

At the local and connected-component levels, our planner relies on variants of the classical RRT algorithm. It therefore inherits, at these levels, many of RRT’s key properties, such as spatial bias towards unexplored regions. Our current implementation does not guarantee probabilistic completeness as we sample configurations in the object space while growing trees in the composite configuration space; however, other complete variants can be used at a cost of increase in computational time. At the global level, to obtain completeness guarantees might require considering grasps classes and enumerating all possible IK solutions during switches.

We introduced a simple heuristic in Section IV-A for selecting an IK solution for regrasping. Selecting a “bad” IK (one that does not come with enough space for the transfer motion) might result in more frequent IK-switch moves, which in turn will adversely affect the planning speed as well as the quality of the planned trajectory.

V. Experiments

In this Section, we illustrate the effectiveness of our planner by solving two difficult bimanual tasks. The planner was implemented in Python. The open-source code is available at https://github.com/quangounet/bimanual.

We used OpenRave [27] environment as a test-bed. All simulations were run on a desktop computer with a 4.0 GHz Intel® Core™ CPU.

In addition, we present a compliant control strategy for closed-chain motion execution, together with hardware demonstration, in Section V-C.

A. Experimental Setup

Our bimanual robotic platform consists of two 6-DOF industrial manipulators Denso V5060. Each manipulator is equipped with a Robotiq 2-Finger 85 Gripper, with a gripping force ranging from 30 to 100 N. One ATI Gamma Force-Torque (F/T) sensor is attached between the end of each manipulator and its gripper.
\[ \mathbf{q}_e - \mathbf{q}_f \] 

of the follower 

\[ \mathbf{k}_f = \mathbf{E} \]

as \( \mathbf{q}_i \) is ran-

\[ \mathbf{T} = \mathbf{J} \]

is

\[ \mathbf{x}_f \]

from the F/T sensor

\[ \mathbf{f}_T \]

of the follower is a summation of the 

\[ \mathbf{e}_T \]

is

\[ x \]

\[ \mathbf{f}_r \]

and the real

\[ x_f \]

1.042 m, in order to maximize each robot's manipulability

\[ \mathbf{q}_i \]

in Fig. 4).

\[ \mathbf{t} \]

in order to maximize each robot's manipulability and the bimanual system's reachability, following procedures presented in a related work [33].

\[ \mathbf{q}_c \]

and the bimanual system's reachability, following procedures presented in a related work [33].

\[ \mathbf{J} \]

\[ \mathbf{k}_c \]

from start to placement configuration,. (d)-(h): IK-switch moves. (f)-(h): Closed-chain motion from placement to goal configuration.

\[ \mathbf{k}_\Delta \]

in Fig. 4). In particular, we read the feedback \( \mathbf{f}_r \) from the F/T sensor attached to the follower’s wrist and produce a perturbation \( \mathbf{x}_f \), given by \( \mathbf{x}_f[k] = \mathbf{k}_\Delta \mathbf{f}_r[k] + \mathbf{k}_c \mathbf{f}_s[k] \), where \( \mathbf{f}_r = \mathbf{f}_i - \mathbf{f}_r \) is the difference between the ideal contact force \( \mathbf{f}_i \) and the real force \( \mathbf{f}_r \), and \( \mathbf{f}_s[k] = (\mathbf{f}_s[k] - \mathbf{f}_s[k-1]) / \Delta t \). We then compute the required perturbation in \( \mathbf{C}_{\text{robot}} \) as \( \mathbf{q}_f[k] = \mathbf{J}^{-1} \mathbf{x}_f[k] \) where \( \mathbf{J} \) is the follower’s Jacobian matrix. The compliance margin \( \mathbf{q}_c[k] \) is then given by

\[ \mathbf{q}_c[0] = \mathbf{0}, \mathbf{q}_c[k] = \mathbf{q}_c[k-1] + \mathbf{q}_f[k] \]  

(4)

And finally, the target joint value at time instant \( k \) of the follower is set to be \( \mathbf{q}_f[k] = \mathbf{q}_i[k] + \mathbf{q}_c[k] \).

This compliant control approach is able to substantially reduce the stress introduced by modeling errors and the robot’s hardware imprecision, resulting in successful executions of the planned composite trajectory. We implemented the trajectory planned for Task 2 (as explained in Section V-B) on our hardware system with this compliant control strategy. The robots in our bimanual setup were able to execute the trajectory smoothly and successfully completed the desired task. Snapshots of the system performing the task are shown in Fig. 4. The complete demonstration, together with the planned trajectory for both task 1 and 2 in simulation, can be found in the accompanying video.

\[ \begin{array}{c|c|c|c|c|c} 
\hline
\# regrasp limit & \# failures & global planning & regrasp planning & total \\
\hline
\text{Task 1} & 1 & 4.7 & 0.36 & 2.43 & 2.79 \\
\text{Task 2} & 3 & 0.02 & 3.66 & 12.15 & 15.81 \\
\hline
\end{array} \]

Fig. 4. Snapshots of the bimanual hardware system completing Task 2. (a, b, d, e, h, i, k, l): Closed-chain motion. (c, f, g, j): IK-switch moves.

C. Trajectory Execution and Control

A composite path generated by our planner comprises both open-chain and closed-chain motions. In open-chain motions where each robot performs IK-switch independently, the robot has no interaction with the surroundings and thus can be controlled freely via position control. For closed-chain motions, however, the closed-chain constraint needs to be satisfied at every time instant and each robot interacts with others through the object they are grasping. Using pure position control, small discrepancies between the simulation models and the real environment as well as robot precision errors can cause serious damage to the manipulated object. Therefore, we introduce compliance into our control method by using position-based force control [33], [34].

The robots were controlled in a leader-follower fashion. The leader robot was solely position controlled while the follower robot executed motion with compliance added. In a discrete-time form with a system sampling time of \( \Delta t \), at each time instant \( k \), the target joint value \( \mathbf{q}_f[k] \) of the follower is a summation of the theoretical value \( \mathbf{q}_i[k] \) (given by the planner) and a compliance margin \( \mathbf{q}_c[k] \).

The start and the goal configuration are verified to be mutually disconnected, as sending them to a local closed-chain planner yields no solution in half an hour.

To ensure the quality of the composite trajectory, the allowed number of regrasping in Task 1 and Task 2 are set as 1 and 3, respectively. We ran our planner 50 times for each task. The total planning time, its decomposition in seconds and number of failures encountered in the second planning stage were averaged and reported in Table I.

In both cases, the planner is able to find a feasible composite path comprising a series of closed-chain motions and IK-switch moves within a reasonable amount of time. It can be seen that the IK-switch regrasping planning occupies a crucial part in the total planning time, since the second stage involves planning of multiple regrasping trajectories as well as static equilibrium checkings. Snapshots of the bimanual system executing planned trajectories (after being smoothened) to complete Task 1 are shown in Fig. 3. For Task 2, it is implemented on the real hardware system as explained in Section V-C below (see snapshots in Fig. 4).

TABLE I

AVERAGE PLANNING TIME FOR EACH TASK

| Task   | # regrasp limit | # failures | global planning | regrasp planning | total |
|--------|-----------------|------------|-----------------|------------------|-------|
| Task 1 | 1               | 4.7        | 0.36            | 2.43             | 2.79  |
| Task 2 | 3               | 0.02       | 3.66            | 12.15            | 15.81 |

Fig. 3. Snapshots of robots completing Task 1. (a)-(c): Closed-chain motion from start to placement configuration,. (d)-(h): IK-switch moves. (f)-(h): Closed-chain motion from placement to goal configuration.
VI. Conclusion

This letter presents a path planner for multi-arm systems manipulating a large/heavy object. Such a system is constrained by closed-chain constraints and planning without breaking the kinematic chain may be practically ineffective or even impossible. We proposed a planning algorithm which effectively deals with such issues. The algorithm utilizes regrasplings to bridge essentially mutually disconnected space components together and hence allows us to solve queries that are otherwise considered as no-solution.

The planner plans a global path first. Then it plans regrasping moves termed as “IK-switch” to complete the global path. When planning for IK-switch, we resort to the environment in vicinity to provide support for the object to maintain contact stability. We also presented an efficient data structure which reorganizes itself to reduce information loss when certain vertex has to be discarded from a planning tree. Finally, we presented a compliant control method for closed-chain trajectory execution. We illustrated effectiveness of our planner via two difficult bi-manual manipulation tasks. With the control method proposed, we also successfully executed closed-chain trajectories on real hardware.

Our method still contains a number of limitations. First of all, the planner uses a single pre-determined grasping pose throughout the planning process. This has the advantage of not requiring the prior construction of grasp classes [12]. However, the possibility of choosing other grasping poses after regrasping would definitely increase the flexibility of the planner. Secondly, when exploring possible placement configurations in current demonstrations, we assume a planar workspace, while the ability to handle more complex environment is necessary for practical problems. Our future work will address these issues so that the planner can deal with more complex tasks.

REFERENCES

[1] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, “Randomized path planning for linkages with closed kinematic chains,” IEEE Trans. Robot. Autom., vol. 17, no. 6, pp. 951–958, Dec. 2001.

[2] Q.-C. Pham and O. Stasse, “Time-optimal path parameterization for redundantly actuated robots: A numerical integration approach,” IEEE/ASME Trans. Mechatronics, vol. 20, no. 6, pp. 3257–3263, Dec. 2015.

[3] J. Mirabel et al., “HPP: A new software for constrained motion planning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2016, pp. 383–389.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE Trans. Robotic Autom., vol. 12, no. 4, pp. 566–580, Aug. 1996.

[5] J. J. Kuffner and S. M. LaValle, “RRT-connect: An efficient approach to single-query path planning,” in Proc. IEEE Int. Conf. Robot. Autom., 2000, vol. 2, pp. 995–1001.

[6] L. Han and N. M. Amato, “A kinematics-based probabilistic roadmap method for closed chain systems,” in Proc. Int. Workshop Algorithmic Found. Robotic, 2000, pp. 233–246.

[7] L. Jaillet and J. M. Porta, “Path planning under kinematic constraints by rapidly exploring manifolds,” IEEE Trans. Robotic., vol. 29, no. 1, pp. 105–117, Feb. 2013.

[8] B. Kim, T. T. Um, C. Suh, and F. Park, “Tangent bundle RRT: A randomized algorithm for constrained motion planning,” Robotica, vol. 34, no. 1, pp. 202–225, 2016.

[9] J. Cortés, T. Siméon, and J.-P. Laumond, “A random loop generator for planning the motions of closed kinematic chains using PRM methods,” in Proc. IEEE Int. Conf. Robot. Autom., 2002, pp. 2141–2146.

[10] P. Tournassoud, T. Lozano-P´erez, and E. Mazer, “Regrasping,” in Proc. IEEE Int. Conf. Robot. Autom., 1987, vol. 4, pp. 1924–1928.

[11] W. Wan, M. M. Mason, R. Fukui, and Y. Kuniyoshi, “Improving regrasp algorithms to analyze the utility of work surfaces in a workcell,” in Proc. IEEE Int. Conf. Robot. Autom., 2015, pp. 4326–4333.

[12] P. Lertkultanon and Q.-C. Pham, “A single-query manipulation planner,” IEEE Robot. Autom. Lett., vol. 1, no. 1, pp. 198–205, 2015.

[13] J.-P. Saut, M. Gharbi, J. Cortés, D. Sidober, and T. Siméon, “Planning pick-and-place tasks with two-hand regrasping,” in Proc. IEEE/RSJ Int. Conf. Robot. Autom., 2008, pp. 4528–4533.

[14] K. Harada, T. Tsuji, and J.-P. Laumond, “A manipulation motion planner for dual-arm industrial manipulators,” in Proc. IEEE Int. Conf. Robot. Autom., 2014, pp. 928–934.

[15] Y. Koga and J.-C. Latombe, “Experiments in dual-arm manipulation planning,” in Proc. IEEE Int. Conf. Robot. Autom., 1992, pp. 2238–2245.

[16] J. Barraquand and J.-C. Latombe, “Robot motion planning: A distributed representation approach,” Int. J. Robot. Res., vol. 10, no. 6, pp. 628–649, 1991.

[17] M. Gharbi, J. Cortés, and T. Siméon, “A sampling-based path planner for dual-arm manipulation,” in Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics, 2008, pp. 383–388.

[18] G. Lee, T. Lozano-P´erez, and L. P. Kaebbling, “Hierarchical planning for multi-contact non-prehensile manipulation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2015, pp. 264–271.

[19] B. Cohen, S. Chitta, and M. Likhachev, “Single- and dual-arm motion planning with heuristic search,” Int. J. Robot. Res., vol. 33, no. 2, pp. 305–320, 2013.

[20] J. W. Burdick, “On the inverse kinematics of redundant manipulators: Characterization of the self-motion manifolds,” in Advanced Robotics: 1989. Berlin, Germany: Springer-Verlag, 1989, ch. 3.

[21] R. Diankov, N. Ratliff, D. Ferguson, S. Srinivasa, and J. Kuffner, “Bispace planning: Concurrent multi-space exploration,” in Proc. Robot., Sci. Syst. IV, 2008, vol. 63, pp. 2–3.

[22] F. Thomas, “The self-motion manifolds of the N-bar mechanism,” in Computational Kinematics, J. Angeles, G. Hommel, and P. Kovács, Eds. New York, NY, USA: Springer-Verlag, 1993, ch. 28, pp. 95–106.

[23] J. J. Kuffner, “Effective sampling and distance metrics for 3D rigid body path planning,” in Proc. IEEE Int. Conf. Robot. Autom., 2004, vol. 4, pp. 3993–3998.

[24] F. C. Park and B. Ravan, “Smooth invariant interpolation of rotations,” ACM Trans. Grap., vol. 16, no. 3, pp. 277–295, 1997.

[25] H. Nguyen and Q.-C. Pham, “Time-optimal path parameterization of rigid-body motions: Applications to spacecraft reorientation,” J. Guid. Control, Dyn., vol. 39, no. 7, pp. 1–5, 2016.

[26] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, “Differential kinematics and statics” in Robotics: Modelling, Planning and Control (ser. Advanced Textbooks in Control and Signal Processing), 1st ed. London, U.K.: Springer-Verlag, 2009, pp. 105–160.

[27] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon Univ., Robot. Inst., Pittsburgh, PA, USA, Aug. 2010.

[28] S. Caron, Q.-C. Pham, and Y. Nakamura, “Leveraging cone double description for multi-contact stability of humanoids with applications to dynamics,” in Proc. Robot. Sci. Syst., 2015, p. 4.

[29] A. D. Prete, S. Tonneau, and N. Mansard, “Fast algorithms to test robust static equilibrium for legged robots,” in Proc. IEEE Int. Conf. Robot. Autom., 2016, pp. 1601–1607.

[30] D. Hsu, J.-C. Latombe, and R. Motwani, “Path planning in expansive configuration spaces,” in Proc. IEEE Int. Conf. Robot. Autom., 1997, vol. 3, pp. 2719–2726.

[31] L. Kavraki, J.-C. Latombe, R. Motwani, and P. Raghavan, “Randomized query processing in robot motion planning,” J. Comput. Syst. Sci., vol. 57, no. 1, pp. 50–60, 1998.

[32] S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[33] F. Sui-´rez-Ruiz and Q.-C. Pham, “A framework for fine robotic assembly,” in Proc. IEEE Int. Conf. Robot. Autom., 2016, pp. 421–426.

[34] H. Seraji, “Adaptive admittance control: An approach to explicit force control in compliant motion,” in Proc. IEEE Int. Conf. Robot. Autom., 1994, pp. 2705–2712.