New Formulation of Mixed-Integer Conic Programming for Globally Optimal Grasp Planning

Min Liu1,3, Zherong Pan2, Kai Xu1, and Dinesh Manocha3

Abstract—We present a two-level branch-and-bound (BB) algorithm to compute the optimal gripper pose that maximizes a grasp metric in a restricted search space. Our method can take the gripper’s kinematics feasibility into consideration to ensure that a given gripper can reach the set of grasp points without collisions or predict infeasibility with finite-time termination when no pose exists for a given set of grasp points. Our main technical contribution is a novel mixed-integer conic programming (MICP) formulation for the inverse kinematics of the gripper that uses a small number of binary variables and tightened constraints, which can be efficiently solved via a low-level BB algorithm. Our experiments show that optimal gripper poses for various target objects can be computed taking 20-180 minutes of computation on a desktop machine and the computed grasp quality, in terms of the Q1 metric, is better than those generated using sampling-based planners.

I. INTRODUCTION

Grasp planning is a well-studied problem in robotics and there is a large amount of work in grasp metric computation [2] and gripper pose planning [3]. Since the two components are somewhat independent, practitioners can build versatile planning frameworks that allow an arbitrary combination of grasp metrics and gripper pose planners for different applications [4]. A high number of choices have been proposed for grasp metrics [2], and a few gripper pose planners are also known. Some planners such as [1], [5] return sub-optimal solutions, which are sensitive to initial guesses and can return grasps of low qualities. Another planner based on simulated annealing (SA) was proposed in [3], which can compute the optimal solution if an infinite number of samples is allowed.

A promising direction of previous works [6], [7] use branch-and-bound (BB) to compute optimal grasp points that maximize a given grasp metric. Unlike SA, BB returns the optimal solution or predicts infeasibility. However, BB algorithms in [6], [7] only consider the optimality in grasp points, the kinematics feasibility of gripper is either omitted in [7] or considered without optimality guarantee in [6]. SCORES [8] uses a recursive neural network for shape composition, and the representation of shape composition can be used for 3D object grasp planning; Triangle lasso [9] can be used for simultaneous clustering and optimization, and is potential for grasp optimization. To take the gripper’s kinematics into consideration, an inverse kinematics (IK) algorithm is needed to determine whether a given set of grasp points can be reached by the gripper. However, most available inverse kinematics algorithms, such as [10], [11], are not optimal and can miss feasible solutions when one exists. Recently, a complete IK algorithm is presented in [12], which reformulates IK as a mixed-integer conic programming (MICP). However, [12] involves the use of a large number of integer parameters making it slow to solve because the worst-case complexity of MICP is exponential in the number of integer variables.

Main Results: We present a novel, two-level BB algorithm to compute the optimal gripper pose that maximizes a given grasp metric in a restricted search space. Our high-level BB algorithm searches for points on the object’s surface that can potentially be used as contact points. Our low-level BB algorithm searches for collision-free gripper poses that realize the given set of contact points. A set of lazy-evaluation heuristic techniques are used to remove unnecessary searches and reduce the branch factor. We have tested our algorithm on 10 target objects grasped by a 3-finger gripper with 15 DOFs and a Barrett Hand with 10 DOFs. Our experiments show that optimal grasps can be computed within 20-180 minutes on a desktop machine for different grippers. Furthermore, our low-level BB formulation results in a speedup of 100× over [12] in terms of gripper’s kinematics feasibility check. We have also compared our algorithm’s performance with a sample-based grasp planner [3] and observed the following benefits:

- Our method always computes higher quality grasps based on Q1 metric, though we are 6–10× slower.
- Our method can detect infeasibility within finite time, which happens frequently when target objects are large compared with the gripper.

II. RELATED WORK

We review previous works on grasp metric computation, gripper pose planning, and IK algorithms.

Grasp Planning takes the gripper’s kinematics feasibility into consideration, which computes a gripper pose given the set of contact points as end-effector constraints. Some sampling-based planners [3], [4] determine the gripper’s pose first by sampling in the gripper’s configuration space. Varava [13] presented an algorithm that can check whether a geometric body can cage another one or detect infeasibility. However, it is rather difficult for the fingers to exactly lie on
the surface of target objects, so these planners have to close the gripper to have the fingers on the object surface. Other planners, such as [6] and our method, first select contact points, compute the grasp quality, and then solve the IK problem to compute the gripper’s pose.

**Grasp Metrics** measure the quality of a grasp and provide ways to compare different grasps. A dozen different grasp metrics have been proposed and summarized in [2]. Sampling-based planners can be used to optimize all kinds of grasp metrics. However, BB can only be used when a grasp metric is monotonic [6], [7], i.e. the grasp metric value computed for a superset of grasp points must be larger than or equal to that computed for a subset. Fortunately, most grasp metrics have been proposed and summarized in [2].

**IK Algorithms** can be used to check kinematics feasibility. However, these algorithms are sub-optimal or sampling-based. A sub-optimal IK algorithm such as [17], [10], [11] can return false negatives, i.e. reporting infeasibility when a solution exists. On the other hand, a sampling-based IK algorithm such as [11] can always find the solution but assume an infinite amount of samples are used. Recently, a new IK algorithm based on MICP relaxation is proposed in [1], which finds a solution or detects infeasibility in a finite amount of time. However, we found that the formulation of [1] requires too many binary variables, making MICP solve time-consuming for its combinatorial worst-case complexity.

### III. PROBLEM STATEMENT

In this section, we formulate the problem of grasp planning. All the symbols used in this paper are summarized in Table I. As shown in Figure 1, the planner input includes:

- A target object that occupies a volume $\Omega_0 \subset R^3$.
- A set of $P$ potential grasp points: $\{p_1, \ldots, p_\rho\} \in \partial \Omega_0$.
- A gripper represented as an articulated object, i.e. a set of $L$ rigid links. Each link occupies a volume $\Omega_i(\theta) \subset R^3$, where $i = 1, \ldots, L$ and $\theta$ is the set of joint angle and globally transformation parameters. On the gripper, there is a set of $K$ fingertips points: $\{x_{1,\ldots,K}\}(\theta)$ and $K < L$. Without loss of generality, we always assume the first $K$ links are fingertips links so that $x_{1} \in \Omega_i$.
- A grasp metric $Q(X)$ whose input is a set $X$ of grasp points satisfying $\forall x \in X, x \in \partial \Omega_0$. Moreover, we assume that the grasp metric is monotonic, i.e. $A \subseteq B \implies Q(A) \leq Q(B)$.

In this paper, we assume that the first 6 parameters in $\theta$ are extrinsic parameters (3 for rotation and 3 for translation) and the rest are intrinsic parameters, i.e. joint angles. Given these inputs, the planner either predicts that the problem is infeasible or outputs $\theta^*$ satisfying the following conditions:

- **C1:** The gripper does not collide with the target object or have self-collisions. In other words, $\forall i = 1, \ldots, L, \Omega_i(\theta^*) \cap \Omega_0 = \emptyset$ and $\forall 1 \leq i < j \leq L, \Omega_i(\theta^*) \cap \Omega_j(\theta^*) = \emptyset$.
- **C2:** Each fingertip point lies on the object surface, i.e. $x_{1,\ldots,K}(\theta^*) \in \{p_1, \ldots, p_\rho\}$.
- **C3:** For all other $\theta$ satisfying C1 and C2, we have: $Q(\{x_{1,\ldots,K}(\theta)\}) \leq Q(\{x_{1,\ldots,K}(\theta^*)\})$.

Note that previous works [6], [7] ignore C1 and C2 and solve the problem using a one-level BB algorithm. On the other hand, the sampling-based planner [3] solves the full version of this problem by generating samples of $\theta$ and then testing C1, C2, and C3, but SA cannot detect infeasibility. Instead, our two-level BB algorithm takes C1, C2, and C3...
into consideration with finite time termination. Conceptually, we identify a large enough subset $\Theta$ of the entire solution space. If we restrict ourselves by adding a condition, $C4$: \( \theta \in \Theta \), then the optimal solution to the planning problem can be efficiently solved within a finite amount of computational time. We solve for global optimal solutions in a restricted search space because we only sample finite potential grasp points for $C1$ and we use a subset of the solution space for a gripper’s kinematics parameters for $C4$.

IV. TWO-LEVEL BRANCH-AND-BOUND FORMULATION

A. BB Algorithm

BB algorithms can efficiently find globally optimal solutions for non-convex optimization problems [18] in the form of disjoint convex sets, which means that an optimization problem can be decomposed into several sub-cases where each case is convex. A BB algorithm can efficiently prune sub-optimal cases at an early stage and accelerate the computation. To this end, a search tree is constructed, each node of which corresponds to a relaxed convex problem. Starting from the root node, each node is evaluated to either find a solution or to prove infeasibility or sub-optimality. If a node is infeasible or its solution is sub-optimal, all its child nodes must also be infeasible or sub-optimal and they will be excluded from further traversal. Otherwise, the node is branched into two or more child nodes. The key to the success of a BB algorithm is the design of the relaxed convex problem. In our high-level BB, the relaxation is provided by the monotonicity of the grasp metric. In our low-level BB, the relaxation is provided by turning all the integer variables into continuous variables.

B. High-Level BB

Our high-level BB takes a very similar form as [6]. We select $K$ points, $x_{1,\ldots,K}$, from the set of $P$ potential grasp points, $\{p_{1,\ldots,P}\}$, such that the grasp quality $Q(\{x_{1,\ldots,K}\})$ is maximized. To solve this problem, we first build a KD-tree for $\{p_{1,\ldots,P}\}$. As illustrated in Figure 1b, each KD-tree node is uniquely denoted by a subset $X \subset \{p_{1,\ldots,P}\}$. The KD-tree is used both by our high-level and low-level BB. A balanced KD-tree can effectively reduce the length of search path in high-level BB. It can also restrict the search space and accelerate MICP solve in low-level BB.

The BB algorithm builds a search tree and keeping track of the best solution with the largest grasp quality metric found so far, which is defined as $Q_{best}$. Each node on the search tree can be uniquely denoted by BBNode($X^1,\ldots,X^K$), where each $X^i$ is the KD-tree node for the $i$th fingertip point. This $X^i$ is also the set of potential grasp points that $x_i$ can possibly be at. In other words, each BBNode represents a Cartesian product of the $K$ set of potential graph points. At each BBNode, we encounter one of the two cases:

- If $|X^i| = 1$ for all $i$, then the BBNode is a leaf node and we compute tentative grasp quality for this node: $Q(X^1 \cup X^2 \cdots \cup X^K)$. If the tentative grasp quality is larger than $Q_{best}$, then this BBNode is known as an incumbent and $Q_{best}$ is updated.
- If there is an $i$ such that $|X^i| > 1$, then the BBNode is a non-leaf node. In this case, we also compute the tentative grasp quality, $Q_{upper} = Q(X^1 \cup X^2 \cdots \cup X^K)$, for this node. If the tentative grasp quality is smaller than $Q_{best}$, i.e. $Q_{upper} < Q_{best}$, then this BBNode is eliminated for further processing. Otherwise, we branch on all the $X^i$ with $|X^i| > 1$.

It has been shown in [6], [7] that this algorithm will find the optimal $\{x_{1,\ldots,K}\}$ if $Q$ is monotonic. When $Q$ is monotonic, the tentative grasp quality $Q_{upper}$ is an lifting of the grasp quality metric to a superset, which is also an upper bound of the actual grasp quality. Therefore, rejecting BBNode when $Q_{upper} < Q_{best}$ will not miss better solutions. However, our high-level BB does not take the gripper’s kinematics feasibility into consideration. Each BBNode essentially specifies all the possible positions of each fingertip point. If it is impossible for the gripper to reach these positions, then the given BBNode does not contain feasible solutions and should be cut early to avoid the redundant search.

C. Gripper’s Inverse Kinematics

Before we discuss feasibility checks of BBNode, we first propose a novel, MICP-based optimal IK algorithm, which is the cornerstone of our feasibility check algorithm. Compared with [1], which can be applied to solve IK for any articulated robot, our formulation only works for the problem of gripper pose planning but uses much fewer binary variables, leading to significant speedup. As illustrated in Figure 1c, our main idea is that applying a global transformation of the gripper is equivalent to applying a global inverse transformation of the target object while keeping the palm of gripper fixed. However, if we keep the palm of gripper fixed, then the fingers of the gripper become decoupled. Specifically, we assume that each fingertip $x_i(\theta) = x_i(\theta_0)$ such that $\theta = (0, 0, 0, 0, 0, \theta_1, \theta_2, \cdots, \theta_K)$. This assumption holds if we allow the target object to have a global rigid transformation.

Based on this assumption, we can relax the IK problem as MICP. Specifically, we introduce auxiliary variables $R_{\theta}$ for the rotation and translation of the $i$th link. The main constraint to relax is $R_{\theta} \in SO_3$ where $R_{\theta}$ is also a function of $\theta_0$. We relax $R_{\theta}$ using a piecewise linear approximation by introducing the following constraints:

$$R_{\theta} = \sum_{d_1,\ldots,d_{|\theta|}=0}^{N} \lambda_{d_1,\ldots,d_{|\theta|}} R_{\theta}^{d_1,\ldots,d_{|\theta|}}$$

where $SO_2$ is the special ordered set of type 2 [19] and $\lambda_{d_1,\ldots,d_{|\theta|}}$ are continuous-valued auxiliary variables. This piecewise linear approximation restricts the solution space, which corresponds to our last condition $C4$ in Section III. The mixed-integer constraints in Equation 1 require $|\theta| SO_2$
constraints and hence $\theta_i[\log_2 N]$ binary decision variables. Finally, $\theta_i^{d_1,\ldots,d_{|\theta_i|}}$ is defined as:

$$
\theta_i^{d_1,\ldots,d_{|\theta_i|}} = \left( \frac{l_i^j(1 - \frac{d_i}{N}) + u_i^j \frac{d_i}{N}}{t_i^j(1 - \frac{d_i}{N}) + u_i^j \frac{d_i}{N}} \right),
$$

where $l_i^j$ and $u_i^j$ are joint limits. In other words, we build a $|\theta_i|$ dimensional grid with $N$ cells along each dimension. Next, we discretize $R_i(\theta_i)$ on the grid and use mixed-integer constraints to ensure that $R_i$ falls inside one of the $N^{|\theta_i|}$ cells. Note that all the $R_i(\theta_i^{d_1,\ldots,d_{|\theta_i|}})$ are precomputed using forward kinematics and used as coefficients of the linear constraints (Equation 1).

Since the palm of the gripper is fixed, we have to inversely transform the target object. As a result, each potential grasp point $p_i$ can be transformed into $R_i p_j + t$ where $R \in SO_3$. The technique to relax $SO_3$ transform the target object. As a result, each potential grasp point $p_i$ can be transformed into $R_i p_j + t$ where $R \in SO_3$. The technique to relax $SO_3$ transform the target object. In practice, we do not need to use the method of solving $SO_3$ transform the target object. Instead, we use a similar technique to Equation 1. Based on the Rodrigues’ formula $R = \exp(w)$, where $w$ is an arbitrary 3D vector, we introduce MICP constraints:

$$
R = \sum_{d_1,d_2,d_3=1}^N \beta^{d_1,d_2,d_3} \exp\left(-\pi \left(1 - \frac{d_i}{N}\right) + \pi \frac{d_i}{N}\right) - \pi \left(1 - \frac{d_i}{N}\right) + \pi \frac{d_i}{N}\right) - \pi \left(1 - \frac{d_i}{N}\right) + \pi \frac{d_i}{N}\right) - \pi \left(1 - \frac{d_i}{N}\right) + \pi \frac{d_i}{N}\right)
$$

which requires $3[\log_2 N]$ binary decision variables and $\beta^{d_1,d_2,d_3}$ are continuous-valued auxiliary variables. Given these constraints, the requirement that the $i$th fingertip point is at $p_j$ can be formulated as a linear constraint:

$$
R_i x_i + t_i = R_i p_j + t.
$$

In summary, we reduce the IK problem for the gripper to a set of linear constraints, whose feasibility can be efficiently verified using off-the-shelf solvers such as [20]. Putting the two parts together, our formulation needs $(|\theta_i| - 3)[\log_2 N]$ binary decision variables to solve the IK problem.

### D. Low-Level BB

The goal of solving low-level BB is to check whether a BBNode($X^1,\ldots,X^K$) contains a feasible solution in terms of gripper’s kinematics. In Section IV-C, the IK problem is formulated as a MICP. However, solving IK is not enough for feasibility checks of BBNodes because Equation 4 constrains that each $x_i$ can only be at one point, while a BBNode generally allows $x_i$ to be at one of several points in non-leaf cases. In the latter case, we have at least one $|X^i| > 1$ so that $x_i$ can be at any point in the set $\{R_i p_j + t | p_j \in X^i\}$. In order for the feasibility check to be performed using the off-the-shelf MICP solver [20], we have to relax this point-in-set constraint as a linear or conic constraint. A typical relaxation is to constrain that $x_i$ lies in the convex hull of the set. However, this constraint takes the following form which is not convex:

$$
R_i x_i + t_i = \sum_j w_j (R_i p_j + t) \quad w_j \geq 0 \quad \sum w_j = 1.
$$

This is because $w_j$ and $R$ are both variables, leading to a bilinear form. It is possible to relax a bilinear form into MICP by requiring additional binary decision variables. Instead, we propose to construct a minimal bounding sphere for the set $X^i$ denoted as:

$$
X^i \subseteq B(X^i) \equiv \{x \in X^i | R(x^i - c) \leq r(X^i)\},
$$

where $c(X^i)$ is the center of the sphere and $r(X^i)$ is the squared radius. Next, we relax the point-in-set constraint as:

$$
\|R_i x_i + t_i - R_c(X^i) - t\| \leq r(X^i),
$$

which is a quadratic cone and can be handled by [20]. Note that $c(X^i)$ and $r(X^i)$ are constants and can be precomputed for each node of the KD-tree (see Appendix A for details). A minor issue is that Equation 6 is not as tight as Equation 5 in terms of the volume of the constrained space. To alleviate this problem, we notice that if $X^i$ has a parent in KD-tree denoted as $X_p$, then $x_i$ should also satisfy the point-in-set constraint of $X_p$. Therefore, we can backtrack $X_i$ to the root KD-tree node and add all the constraints of Equation 6 along the path, as illustrated in Figure 1e.

![Fig. 2. (a): We illustrate the normal of fingertip points $n(x_i)$ (white arrow) and the inward normal of potential grasp points $n(p_j)$ (black arrow). We allow $n(x_i)$ to lie in a normal cone around $n(p_j)$ (blue) with a threshold $\epsilon$ (dashed line). (b): We illustrate the relaxed normal cone of the two potential grasp points (red) with threshold denoted as $\epsilon(X^i)$.
](image)

### E. Normal Constraints

We can further optimize our formulation by taking the surface normals of the target object into consideration, leading to even tighter constraints. As illustrated in Figure 2a, each potential grasp point $p_j$ can be associated with an inward surface normal denoted by $n(p_j)$. Also, each fingertip point $x_i$ can also be associated with a normal $n(x_i)$. It is intuitive to constrain that $n(x_i)$ should be pointing at a similar direction to $n(p_j)$. In practice, we do not need $n(x_i)$ to align with $n(p_j)$ exactly but allow $n(x_i)$ to lie in a small vicinity. Therefore, if a leaf BBNode($X^1,\ldots,X^K$) is encountered, then we add the following constraint to MICP for each $X^i = \{p\}$:

$$
\|R_i n(x_i) - R_i n(p)\| \leq \epsilon,
$$

where $\epsilon$ is a user-defined threshold. If a non-leaf BBNode is encountered, then we have to add a normal-in-set constraint.
Using a similar technique as Section IV-D, we construct a normal cone denoted as:

\[ \{ n(p) | p \in X^i \} \subseteq C(X^i) \triangleq \{ n \| n - m(X^i) \| ^2 \leq \epsilon(X^i) \}, \]

for each internal KD-tree node during precomputation. Here \( m(X^i) \) is the central direction of the normal cone and \( \epsilon(X^i) \) is the squared radius. We can then add the relaxed normal-in-set constraint for \( X^i \):

\[ \| Rn(x_i) - Rm(X^i) \| ^2 \leq \epsilon'(X^i), \]

where \( \epsilon'(X^i) \) is the squared radius of the normal cone taking the user-defined threshold into consideration, as illustrated in Figure 2b (see Appendix A). Finally, we can further tighten the normal-in-set constraint using a similar technique as Section IV-D. We can backtrace \( X^i \) to the root KD-tree node and add all constraints of Equation 8 along the path.

F. Collision Handling using Lazy-MICP

In addition to checking the gripper’s kinematics feasibility, our low-level BB also ensures that gripper’s links do not collide with each other or with the target object. It has been shown in [21], [1] that collision constraints can be relaxed as MICP. In order to reduce the use of binary decision variables, we propose to add collision constraints in a lazy manner.

Specifically, we assume that the target object \( \Omega_o \) and all gripper links \( \Omega_i \) are convex objects. If \( \Omega_o \) is not convex then we approximate it using a union of convex shapes. We first ignore all collision constraints and solve MICP. Then we detect collisions between \( R\Omega_o + t \) and \( \Omega_i(\theta) \) and record the pair of points with the deepest penetration denoted as \( D \), e.g. using [22]. If we find that \( a \in \Omega_o \) and \( b \in \Omega_i \) are in collision, then we pick a separating direction from a set of possible separating directions \( \{ s_1, \ldots, s_S \} \) and introduce the following constraint as illustrated in Figure 3a and Figure 3b:

\[ s_k^T(Ra + t) + D \leq s_k^T(Rb + t) + (1 - \gamma^o_k)M \forall k \]

\[ \gamma^o_k \geq 0 \quad \sum_{k=1}^S \gamma_k^{o} = 1 \quad \gamma_1^{o}, \ldots, \gamma_S^{o} \in \text{SOS}_1, \quad (9) \]

where SOS1 is the special ordered set of type 1 [19], \( \gamma_k^{o} \) are the auxiliary variables, and \( M \) is the big-M parameter [23]. Similarly, if there is a collision between a pair of points, \( a \in \Omega_i \) and \( b \in \Omega_j \), then we have the following constraint:

\[ s_k^T(Ra + t_j) + D \leq s_k^T(Rb + t_i) + (1 - \gamma_i^{o})M \forall k \]

\[ \gamma_i^{o} \geq 0 \quad \sum_{k=1}^S \gamma_k^{i} = 1 \quad \gamma_1^{i}, \ldots, \gamma_S^{i} \in \text{SOS}_1, \quad (10) \]

After adding collision constraints, the new MICP is solved again with a warm-start and we again perform collision-detection. This is looped until no new collisions are detected or MICP becomes infeasible. Note that if a new collision is detected for a link-link or link-object pair for which collision has been detected in previous loops, then only the first lines of Equation 9 and Equation 10 are needed. In other words, binary variables are needed once for each link-link and link-object pair and the binary variables number is \( \lceil \log_2 S \rceil \).

Note that the collisions between the first K fingertip links and the target object do not need to be detected or handled by MICP. This is because each fingertip link contacts the target object at one point with matched normal when Equation 7 holds with \( \epsilon = 0 \), which is a sufficient condition for two convex objects to be collision-free [24]. In practice, we allow users to set a small, positive \( \epsilon \) to account for inaccuracies in gripper and target object shapes.

V. ALGORITHM ACCELERATION

Our method discussed in Section IV is computationally costly due to the repeated use of the MICP-based IK algorithm, to check the kinematics feasibility of the gripper. In this section, we discuss three techniques to reduce the cost of MICP solving. Our first technique is bottom-up kinematics check, which is based on the following observation:

Lemma 5.1. If the MICP-based IK problem for a BBNode is feasible, then the MICP-based IK problem for its parent is also feasible.

Proof: The IK problem for a BBNode is derived by adding more constraints (in forms of Equation 4,7,6,8) to the IK problem of its parent. (See Algorithm 4 for more details on the construction of a MICP-based IK problem.)

Therefore, we can check the gripper’s kinematics feasibility lazily. Specifically, if a BBNode is a non-leaf node and it has not been checked for gripper’s kinematics feasibility, we skip the check and continue branching. If a BBNode is a leaf node, we solve MICPs to check for gripper’s kinematics feasibility for all the BBNodes on the path between this leaf node and the root BBNode. If any of the MICPs appears to be feasible in this process, all ancestor nodes will also be feasible and their checks can be skipped. Our second technique is warm-started MICP solve. We store the solution of MICP for each BBNode and use this solution as the initial guess for the MICP solves of its children. Our third technique is local optimization. Note that MICPs are convex relaxations of non-convex optimization problems. Non-convex optimization problems are sub-optimal but efficient to solve. Therefore, we propose to solve a non-convex optimization before invoking a call to the MICP solver. If the non-convex optimization appears to be feasible, we skip the MICP solves. We use interior point algorithm [25] as our non-convex optimization solver. The key steps of our algorithm are illustrated in Figure 4 and we summarized our method in Appendix B.

VI. EXPERIMENTS & RESULTS

We perform all the experiments on a single desktop machine with one Intel i7-8750H CPU (6-cores at 2.27Hz). Given a target object, we first sample \( p_1, \ldots, p_P \) on \( \partial \Omega_o \) using parallel Poisson disk sampling [26] and then build a KD-tree...
for the set of $P$ points using [27]. Finally, we solve low-level MICP problems using [20]. To grasp the object, we use a 3-finger axial-symmetric gripper with $|\theta| = 6 + 3 \times 3 = 15$ and $|\theta_1| = 3$. Each finger of the gripper is controlled by one ball joint and one hinge joint. Under this setting, our IK formulation requires $12\log_2 N$ binary decision variables while [1] requires $630\log_2 N$ binary decision variables. The average solving time using our formulation and [1] are compared in Table II, which indicates that our formulation is over $100 \times$ more efficient.

**TABLE II**

| Object | MICP (min) | Total (min) | Percentage (%) |
|--------|------------|-------------|----------------|
| Bottle | 43.766     | 71.337      | 61.351         |
| Table  | 9.614      | 49.659      | 19.360         |
| Chair  | 6.004      | 19.669      | 30.255         |
| Camera | 4.595      | 50.000      | 9.190          |
| Cabinet| 54.182     | 73.394      | 73.823         |

Finally, we compare the performance of our method with a sampling-based method. Figure 8 using both the 3-finger gripper and the 10-DOF Barrett Hand. Being incomplete, a sampling-based method sometimes cannot find solutions, especially when the target object is large compared with the gripper. This is because feasible grasps become rare in the configuration space when object size grows and most samples are not valid. For GraspIt![4] settings, the space search type is Axis-angle, the energy formulation is AUTO_GRASP_QUALITY_ENERGY, the maximum iteration number is 45000, and the planner type is Sim.Ann. As a result, the initial guess of the gripper’s pose is important when using [4]. However, our method can always find a solution when one exists and we do not require users to provide an initial guess. Even when a sampling-based method can find a solution, our solution always has a higher quality in terms of the value of $Q_2$ metric. On the other hand, [4] can find a sub-optimal solution within 10min which is $10 \times$ faster than our method. However, we show in Figure 7a that giving [4] more computational time does not improve the solution and we speculate that the solution has fallen into a local minimum. If only sub-optimal solutions are needed, the user can choose to terminate our algorithm when $Q$ is larger than a threshold. According to the convergence history in Figure 5, our method can usually find feasible solutions after exploring $1 - 5K$ nodes, which also takes several minutes. However, if our algorithm is allowed to explore more nodes, as shown in Figure 7b, it can output multiple grasps for an object by storing all the feasible solutions. This makes our algorithm potentially useful for offline grasp dataset construction and online learning-based grasp systems such as [28], [29].

**VII. CONCLUSION & LIMITATIONS**

We present a two-level BB algorithm to search for the optimal grasp pose in a restricted search space that maximizes a given monotonic grasp metric. The high-level BB selects grasp points that maximize grasp quality, while the low-level BB cut infeasible BBNodes in terms of gripper’s kinematics. Our low-level BB uses a compact MICP formulation that requires a small number of binary variables. Experiments show that our method can plan grasps for complex objects. Our work has several limitations. First, we only plan gripper poses without considering other sources of infeasibility such as environmental objects and robot arms. When robot arms are considered, the decoupled assumption of Section IV-C is violated and we need new techniques for relaxing IK as MICP. Second, although our IK relaxation is more efficient than [1], our method is not an outer-approximation.
In other words, if a gripper pose is feasible using exact IK, it might not be feasible under our relaxed IK constraints. Third, we only plan for precision grasps with fingertipcontacts, while generating power grasps or caging grasps is a good topic for future work. In addition, our formulation incurs a high computational cost for complex object shapes, such as those acquired from scanning real-world objects. Finally, our method does not consider modeling or sensing uncertainty, which is necessary to realize the grasp in a physical platform.

APPENDIX A: BOUNDING SPHERES & CONES

Each KD-tree node $X$, contains a set of potential grasp points $p_j$, each of which has an outward normal direction $n(p_j)$. To generate an efficient MICP-based IK problem, we need to compute a minimal bounding sphere that encloses $p_j$ and a minimal bounding cone that encloses $n(p_j)$. In this section, we present methods to compute these bounding volumes via numerical optimization using the following lemmas:

**Lemma 7.1:** The minimal bounding sphere $B(X)$ can be computed by solving the following conic programming problem:

$$
\arg\min_{c(X)} \quad r(X) \quad \text{s.t.} \quad \|c(X) - p_j\|^2 \leq r(X) \forall p_j \in X,
$$

where $c(X)$ is the center of the bounding sphere and $r(X)$ is the radius of the bounding sphere.

**Proof:** Any valid bounding sphere should contain all the potential grasp points $p_j$, which justifies our constraints. A minimal bounding sphere has the smallest radius, which justifies our objective function.

**Lemma 7.2:** The minimal bounding cone $C(X)$ can be computed by solving the following non-convex programming problem:

$$
\arg\min_{\epsilon(X)} \quad \epsilon(X) \quad \text{s.t.} \quad \|m(X) - n(p_j)\|^2 \leq \epsilon(X) \forall p_j \in X,
$$

where $m(X)$ is the central axis of the normal cone and $\epsilon(X)$ is the radius as defined in Figure 2.

**Proof:** Any valid bounding cone should contain all the potential grasp normals $n(p_j)$, which justifies our constraints. A minimal bounding cone has the smallest radius, which justifies our objective function.

This optimization is non-convex due to the unit length constraint $\|m(X)\| = 1$. To compute the minimal bounding cone, we relax the unit length constraint using MICP via the technique presented in [1]. Finally, we take the user-defined threshold into consideration and compute $\epsilon'(X)$ as follows:

$$
\theta = 2\sin^{-1}\left(\frac{\sqrt{\epsilon(X)}}{2}\right) + 2\sin^{-1}\left(\frac{\sqrt{\epsilon}}{2}\right)
$$

$$
\epsilon'(X) = 2\sin\left(\frac{\min(\theta, \pi)}{2}\right)^2.
$$

APPENDIX B: ALGORITHMS

A summary of algorithms. Given a gripper and a target object, we first perform a precomputation using Algorithm 1. Afterward, we use Algorithm 3 as high-level BB and use Algorithm 4 as low-level BB. The accelerated bottom-up kinematics check is summarized in Algorithm 2, which is used as a middleware between the two levels.
Algorithm 1 Precomputation
1: Sample $p_1, \ldots, p_p$ on $\partial \Omega$, using [26]
2: Construct KD-tree for $p_1, \ldots, p_p$ using [27]
3: for each $X$ in KD-tree do
4: Construct $B(X)$ and $C(X)$ (Appendix A)
5: for each link on the gripper do
6: Construct relaxed IK constraints (Equation 1)
7: Construct relaxed IK constraint for $\Omega_0$ (Equation 3)

Algorithm 2 BottomUpKinematicsCheck(BBNode)
1: $\triangleright$ Check if ancestor BBNode.LowLevel=True
2: CurrentBBNode = BBNode
3: do
4: if CurrentBBNode.LowLevel=False then
5: BBNode.LowLevel=True
6: Return
7: CurrentBBNode = CurrentBBNode.Parent
8: while CurrentBBNode is not the root
9: CurrentBBNode = BBNode $\triangleright$ Bottom-up Kinematics Check
10: ChildFeasible=True
11: do
12: if CurrentBBNode.LowLevel=Unknown then
13: Break $\triangleright$ The BBNode/ancestors have been checked
14: else if ChildFeasible=True then
15: CurrentBBNode.LowLevel=True
16: else
17: CurrentBBNode.LowLevel=
18: LowLevelBB(CurrentBBNode)
19: ChildFeasible = CurrentBBNode.LowLevel
20: CurrentBBNode = CurrentBBNode.Parent
21: while CurrentBBNode is not the root

Algorithm 3 HighLevelBB
1: $\triangleright$ BBNode.LowLevel marks gripper’s feasibility
2: $\triangleright$ BBNode.LowLevel initializes to Unknown
3: Queue $\leftarrow \emptyset$, Queue.insert(BBNode($X_R$, $X_R$))
4: while Queue # $\neq \emptyset$ do
5: BBNode($X^1$, $\ldots$, $X^K$) $\leftarrow$ Queue.pop()
6: if BBNode is leaf then
7: BottomUpKinematicsCheck(BBNode)
8: if BBNode.LowLevel=True then
9: $Q_{curr} \leftarrow (X^1 \cup X^2 \cup \cdots \cup X^K)$ $\triangleright$ Bound
10: if $Q_{curr} \triangleright Q_{best}$ then
11: $Q_{best} \leftarrow Q_{curr}$
12: else if BBNode.LowLevel=True $\triangleright$ Unknown
13: Find $|X^i| \geq 1$ $\triangleright$ Branch
14: Queue.insert(BBNode($\ldots$, $X^i$, $\ldots$))
15: Queue.insert(BBNode($\ldots$, $X^i$, $\ldots$))
16: Return $Q_{best}$

Algorithm 4 LowLevelBB(BBNode($X^1$, $\ldots$, $X^K$))
1: MICP $\leftarrow \emptyset$
2: MICP.add(Equation 1,3)
3: for $i = 1, \ldots, K$ do
4: for each $X$ from $X^i$ to $X_R$ do
5: if $|X^i| = 1$ then
6: MICP.add(Equation 4) for $i$
7: else
8: MICP.add(Equation 6.8) for $X$
9: do
10: Solve MICP using [20] $\triangleright$ Warm-start if possible
11: if MICP.Feasible=False then
12: Return False
13: Detect penetration $D$ among $\Omega$, and $\Omega_1$
14: if $D > 0$ then
15: MICP.add(Equation 9 or Equation 10)
16: while $D > 0$
17: Return True

ACKNOWLEDGMENT
This work was supported in part by the National Key Research and Development Program of China (No.2018AAA0102200), NSFC (61572507, 61532003, 61622212), ARO grant W911NF-18-1-0313, and Intel. Min Liu is supported by the China Scholarship Council.

REFERENCES
[1] H. Dai, A. Majumdar, and R. Tedrake, “Synthesis and optimization of force closure grasps via sequential semidefinite programming,” in Robot. Res., 2018, pp. 285–305.
[2] M. A. Roa and R. Suárez, “Grasp quality measures: review and performance,” Auto. Robot., vol. 38, no. 1, pp. 65–88, 2015.
[3] M. Ciocarlie, C. Goldfeder, and P. Allen, “Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem,” in Robot. Sci. Syst. Manipulation Workshop, 2007, pp. 19–25.
[4] A. T. Miller and P. K. Allen, “Grapshot! a versatile simulator for robotic grasping,” IEEE Robot. Autom. Mag., vol. 11, no. 4, pp. 110–122, 2004.
[5] X. Zhu and J. Wang, “Synthesis of force-closure grasps on 3-d objects based on the q distance,” IEEE. T. Robotic. Autom., vol. 19, no. 4, pp. 669–679, 2003.
[6] K. Hang, J. A. Stork, N. S. Pollard, and D. Kragic, “A framework for optimal grasp contact planning,” IEEE Robot. Autom. Lett., vol. 2, no. 2, pp. 704–711, 2017.
[7] J. D. Schulman, K. Goldberg, and P. Abbeel, “Grasping and fixturing as submodular coverage problems,” in Robot. Res., 2017, pp. 571–583.
[8] C. Zhu, K. Xu, S. Chaudhuri, R. Yi, and H. Zhang, “Scores: Shape composition with recursive substructure priors,” ACM Transactions on Graphics (TOG), vol. 37, no. 6, pp. 1–14, 2018.
[9] Y. Zhao, K. Xu, E. Zhu, X. Liu, X. Zhu, and J. Yin, “Triangle lasso for simultaneous clustering and optimization in graph datasets,” IEEE Transactions on Knowledge and Data Engineering, vol. 31, no. 8, pp. 1610–1632, 2018.
programming for multi-vehicle path planning,” in 2001 European Control Conference, 2001, pp. 2603–2608.

[22] Y. J. Kim, M. C. Lin, and D. Manocha, “Incremental penetration depth estimation between convex polytopes using dual-space expansion,” IEEE Trans. Vis. Comput. Gr., vol. 10, no. 2, pp. 152–163, 2004.

[23] J. Samuel and K. J. T. Pizzo, Iterative methods for nonlinear optimization problems. Prentice Hall, Englewood Cliffs, 1972.

[24] R. T. Rockafellar, Convex analysis, ser. Princeton Mathematical Series. Princeton, N. J.: Princeton University Press, 1970.

[25] R. H. Byrd, J. Nocedal, and R. A. Waltz, “Knitro: An integrated package for nonlinear optimization,” in Large-Scale Nonlinear Optimization. Springer, 2006, pp. 35–59.

[26] L.-Y. Wei, “Parallel poisson disk sampling,” in ACM Trans.Gr., vol. 27, no. 3, 2008, p. 20.

[27] S. Popov, J. Gunther, H.-P. Seidel, and P. Slusallek, “Experiences with streaming construction of sah kd-trees,” in 2006 IEEE Symposium on Interactive Ray Tracing. IEEE, 2006, pp. 89–94.

[28] M. Liu, Z. Pan, K. Xu, K. Ganguly, and D. Manocha, “Generating grasp poses for a high-dof gripper using neural networks,” in Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst., 2019, pp. 1518–1525.

[29] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, “Planning multi-fingered grasps as probabilistic inference in a learned deep network,” in Robot. Res., 2020, pp. 455–472.