A New Distance Algorithm and Its Application to General Force-Closure Test

Yu Zheng and Chee-Meng Chew, *Member*, *IEEE*
Control & Mechatronics Laboratory, National University of Singapore
Block EA, #04-06, 9 Engineering Drive 1, Singapore 117576
yuzheng001@gmail.com, mpeccm@nus.edu.sg

***Abstract*—This paper presents an algorithm for computing the distance between a point and a convex cone in *n*-dimensional space. The convex cone is specified by the set of all nonnegative combinations of points of a given set. If the given set is finite, the algorithm converges in a finite number of iterations. The iterative computation speeds up with the help of the derived recursive formulas and effective choice of initial and stopping conditions. The function of this algorithm is demonstrated by its application to force-closure test, which is a fundamental problem arising in research of several mechanisms. Numerical examples show that force-closure can be verified very quickly by this means.**

***Keywords*—cable-driven robots, convex cone, distance, fixtures, force-closure, multifingered grasps**

I. Introduction

Distance functions can be used not only in 3-D object space to detect the collision between objects [1] but also in high-dimensional space to deal with many scientific problems, such as in 6-D wrench space to solve problems of multifingered grasping [12, 13]. Thus a lot of efforts have been made on the methods for computing the traditional Euclidean distance [2–9] and the definitions of several pseudo distances [10–13]. However, the previous work [1–13] focuses on the distance functions between compact (closed and bounded) convex sets and does not cover the unbounded sets, such as the convex cone, which also arises often in scientific problems. In this paper, therefore, we put forward an algorithm for computing the distance between a point and a convex cone in *n*-dimensional space and apply it to solve the force-closure test in 6-D wrench space to verify its usefulness.

The rest of this paper is arranged as follows. Section II summarizes the previous work on distance and force-closure test. Sections III and IV address our distance algorithm and its application to force-closure test, respectively. Numerical examples are given in Section V. Section IV contains some concluding remarks and future work.

II. Summary of Related Work

A. Distance Functions

Various distance functions were proposed in the past two decades for evaluating separation or penetration of sets. They can be categorized into the translational [1–9] and the pseudo [10–13] distances.

The translational distance between two compact convex sets is defined as the amount of the smallest relative translation in the L_2 metric to bring them into contact (neither separation nor penetration). For separated sets, Gilbert, *et al*. [2, 3] found a sequence of simplices in their Minkowski difference, whose distances to the origin converge to the result. Lin and Canny [4], Mirtich [5] calculated the distance between two polyhedra by finding their closest features. Cameron [6, 7] proposed a recursive procedure for evaluating the support functions in the GJK algorithm [2], so that the computational time is greatly reduced. On the basis of Cameron's work [6, 7], Ong and Gilbert [8] further modified the GJK algorithm [2] to enhance its efficiency in incremental distance computation. The distance between penetrated polyhedra was discussed in [9].

Because of the difficulty in computing the translational distance between penetrated non-polyhedral sets, the pseudo distance functions were suggested as an alternative [10–13]. The primary drawback of the pseudo distances [10–13] is that their values are usually not exactly equal to the real distance [1–9]. Moreover, relying on the optimization techniques, their computations are not as efficient as the GJK [2] and LC [4] algorithms. Hence the distance we focus on in this paper is the translational distance.

Although the GJK algorithm [2, 3] deals with the distance between only bounded sets and cannot be applied directly to the case of a convex cone with a point, its idea is quite helpful for developing such an algorithm to fill this void. Inspired by [2, 3], we find a sequence of polyhedral cones in the convex cone, whose distances to the single point converge to the distance to be determined. Every polyhedral cone is generated by at most *n* linearly independent points, which indicate its side edges. The recursive formulas are derived for computing the sequence of distances between the point and the polyhedral cones, so that the convergence of the algorithm is pretty fast.

B. Force-Closure Test

Force-closure test is a fundamental topic in the research of several mechanisms, such as grasping [13–22] and fixturing [21, 23] mechanisms, and cable-driven robots [24]. Till now, a number of good test methods have been proposed [13–22].

From the analysis in the wrench space, Salisbury and Roth [14] indicated that a grasp is force-closure if and only if the primitive wrenches positively span the entire wrench space, which is equivalent to that the origin of the wrench space lies strictly inside the convex hull of the primitive contact wrenches [15]. By this condition, the test methods based on linear [16] or

nonlinear programming [17] were proposed. Besides the wrench space, force-closure can also be analyzed in the contact force space [18, 19] or the dual spaces of the wrench and contact force spaces [20–22].

The force-closure condition presented in [14] is satisfied if and only if seven points, which can positively span the whole wrench space, are contained in the convex cone generated by the primitive wrenches. Then the proposed distance algorithm can be directly applied to the force-closure test. Numerical examples show that its efficiency is very high, even higher than the most efficient method so far [16]. Moreover, the test using the distance algorithm possesses the merits of other methods, such as being applicable to all the contact types and not requiring the linearization of the friction models [17].

III. A Fast Procedure for Computing the Distance Between a Point and a Convex Cone

A. Distance Formulation

Let A be a compact set in $\mathbb{R}^n$ having nonzero points and $\operatorname{co} A$ the convex cone generated by A, which is defined as the set of all nonnegative combinations of points of A:

$$\operatorname{co} A = \left\{ c_1 \boldsymbol{a}_1 + c_2 \boldsymbol{a}_2 \in \mathbb{R}^n \;\middle|\; c_1, c_2 \geq 0,\ \boldsymbol{a}_1, \boldsymbol{a}_2 \in A \right\} \quad (1)$$

Let $\boldsymbol{b}$ be a nonzero point in $\mathbb{R}^n$. Then the distance between $\boldsymbol{b}$ and $\operatorname{co} A$ is defined by the closest point of $\operatorname{co} A$ to $\boldsymbol{b}$:

$$d(\boldsymbol{b}, \operatorname{co} A) = \min_{\boldsymbol{a} \in \operatorname{co} A} \| \boldsymbol{a} - \boldsymbol{b} \| \quad (2)$$

where $\| \bullet \|$ denotes the Euclidean norm. Let $p(\boldsymbol{b}, \operatorname{co} A)$ denote the closest point of $\operatorname{co} A$ to $\boldsymbol{b}$ such that $d(\boldsymbol{b}, \operatorname{co} A) = \| p(\boldsymbol{b}, \operatorname{co} A) - \boldsymbol{b} \|$. Because of the convexity of $\operatorname{co} A$, the choice of $p(\boldsymbol{b}, \operatorname{co} A)$ is unique.

The support function h_A of A, $h_A : \mathbb{R}^n \to \mathbb{R}$, is defined by

$$h_A(\boldsymbol{r}) = \max_{\boldsymbol{a} \in A} \boldsymbol{r}^T \boldsymbol{a} \quad (3)$$

where $\boldsymbol{r} \in \mathbb{R}^n$. The support mapping, $s_A : \mathbb{R}^n \to A$, finds a point $s_A(\boldsymbol{r}) \in A$ such that $h_A(\boldsymbol{r}) = \boldsymbol{r}^T s_A(\boldsymbol{r})$. Note that the choice of $s_A(\boldsymbol{r})$ may be not unique. Several nice properties of h_A and s_A are reported in [3]. Herein we pick out two and list them below, which will be used in this paper.

Theorem 1: Let $A, A_1, \ldots, A_m$ be nonempty compact sets in $\mathbb{R}^n$ and $\boldsymbol{G}$ a real $n' \times n$ matrix. Then the following are true:

(a) $h_{A_1 \cup A_2 \cup \cdots \cup A_m}(\boldsymbol{r}) = h_{A_j}(\boldsymbol{r})$, $s_{A_1 \cup A_2 \cup \cdots \cup A_m}(\boldsymbol{r}) = s_{A_j}(\boldsymbol{r})$, where $j = j(\boldsymbol{r})$ satisfies $h_{A_j}(\boldsymbol{r}) = \max_{i=1,2,\ldots,m} h_{A_i}(\boldsymbol{r})$.

(b) $h_{\boldsymbol{G}(A)}(\boldsymbol{r}) = h_A(\boldsymbol{G}^T \boldsymbol{r})$, $s_{\boldsymbol{G}(A)}(\boldsymbol{r}) = \boldsymbol{G} s_A(\boldsymbol{G}^T \boldsymbol{r})$.

B. Iterative Computation

Let $A_k = \{ \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_l \}$ be a subset of l ($1 \leq l \leq n$) linearly independent points of A and $\hat{A}_k$ a subset of A_k such that $p(\boldsymbol{b}, \operatorname{co} A_k) \in \operatorname{co} \hat{A}_k$. Then $\hat{A}_k$, $p(\boldsymbol{b}, \operatorname{co} A_k)$, and $d(\boldsymbol{b}, \operatorname{co} A_k)$ can be determined as follows. Let

$$\boldsymbol{A}_k = [\, \boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \cdots \quad \boldsymbol{a}_l \,] \quad (4)$$

$$\boldsymbol{c}_k = (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{b} \quad (5)$$

$$\boldsymbol{p}_k = \boldsymbol{A}_k \boldsymbol{c}_k \quad (6)$$

$$\boldsymbol{r}_k = \boldsymbol{b} - \boldsymbol{p}_k . \quad (7)$$

The point $\boldsymbol{p}_k$ is the projection of $\boldsymbol{b}$ onto the subspace spanned by A_k. If the components of $\boldsymbol{c}_k$ are all nonnegative, then $\hat{A}_k = A_k$; otherwise $\hat{A}_k$ is a proper subset of A_k. Let $A_k' = \{ \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{l'} \}$ be a subset of l' ($1 \leq l' \leq l-1$) points of A_k. Then $\hat{A}_k = A_k'$ if and only if $\boldsymbol{c}_k$ and $\boldsymbol{r}_k$ computed by (4)–(7) again with respect to A_k' satisfy:

(a) The components of $\boldsymbol{c}_k$ are all nonnegative;

(b) $\boldsymbol{r}_k^T \boldsymbol{a} < 0$ for all $\boldsymbol{a} \in A_k \setminus A_k'$.

Therefore, $p(\boldsymbol{b}, \operatorname{co} A_k) = p(\boldsymbol{b}, \operatorname{co} \hat{A}_k) = \boldsymbol{p}_k$ and $d(\boldsymbol{b}, \operatorname{co} A_k) = d(\boldsymbol{b}, \operatorname{co} \hat{A}_k) = \| \boldsymbol{r}_k \|$.

Theorem 2: Assume $\boldsymbol{r}_k \neq \boldsymbol{0}$. Let $A_{k+1} = \hat{A}_k \cup \{ s_A(\boldsymbol{r}_k) \}$. If $h_A(\boldsymbol{r}_k) > 0$, then $d(\boldsymbol{b}, \operatorname{co} A_{k+1}) < d(\boldsymbol{b}, \operatorname{co} A_k)$.

From the above discussion, we attain an iterative procedure, which generates several sequences, namely $\boldsymbol{r}_k$, $h_A(\boldsymbol{r}_k)$, $d(\boldsymbol{b}, \operatorname{co} A_k)$, and $p(\boldsymbol{b}, \operatorname{co} A_k)$. Before the first iteration, in the absence of any information about $\boldsymbol{r}_k$, an effective choice for its initial value $\boldsymbol{r}_0$ is $\boldsymbol{b}$, which is equivalent to projecting $\boldsymbol{b}$ onto the origin $\boldsymbol{0}$ and taking $\boldsymbol{p}_0$ to be $\boldsymbol{0}$. Now we shall clarify the convergences of these sequences.

Theorem 3: If $h_A(\boldsymbol{r}_0) > 0$, then $\boldsymbol{r}_k$, $h_A(\boldsymbol{r}_k)$, $d(\boldsymbol{b}, \operatorname{co} A_k)$, and $p(\boldsymbol{b}, \operatorname{co} A_k)$ converge to $\boldsymbol{b} - p(\boldsymbol{b}, \operatorname{co} A)$, 0, $d(\boldsymbol{b}, \operatorname{co} A)$, and $p(\boldsymbol{b}, \operatorname{co} A)$, respectively; otherwise, $d(\boldsymbol{b}, \operatorname{co} A) = \| \boldsymbol{b} \|$ and $p(\boldsymbol{b}, \operatorname{co} A) = \boldsymbol{0}$. Furthermore, if A is a finite set, then the convergence is finite.

Distance Algorithm: Given a nonzero point $\boldsymbol{b}$ and a compact set A having nonzero points, compute the distance between $\boldsymbol{b}$ and the convex cone $\operatorname{co} A$ generated by A. Let ε be the termination tolerance on $h_A(\boldsymbol{r}_k)$.

Step 1: Set $\boldsymbol{r}_0 = \boldsymbol{b}$, $A_0 = \varnothing$, $\hat{A}_0 = \varnothing$, $d(\boldsymbol{b}, \operatorname{co} A_0) = \| \boldsymbol{b} \|$, $p(\boldsymbol{b}, \operatorname{co} A_0) = \boldsymbol{0}$, and $k = 0$.

Step 2: If $h_A(\boldsymbol{r}_k) \leq \varepsilon$, then set $d(\boldsymbol{b}, \operatorname{co} A) = d(\boldsymbol{b}, \operatorname{co} A_k)$ and $p(\boldsymbol{b}, \operatorname{co} A) = p(\boldsymbol{b}, \operatorname{co} A_k)$, and terminate this procedure.

Step 3: Determine A_{k+1} and $\boldsymbol{r}_{k+1}$. Set $k = k+1$ and return to Step 2.

C. Recursive Formulas

In this procedure, Step 3 needs to compute $\boldsymbol{r}_{k+1}$ after setting $A_{k+1} = \hat{A}_k \cup \{ s_A(\boldsymbol{r}_k) \}$. From the argument at the beginning of Section III-B, we first try to compute $\boldsymbol{r}_{k+1}$ by (4)–(7) with $\boldsymbol{A}_{k+1} = [\, \boldsymbol{A}_k \quad s_A(\boldsymbol{r}_k) \,]$. Let $\boldsymbol{a}_{l+1} = s_A(\boldsymbol{r}_k)$. Then we have

$$\boldsymbol{A}_{k+1}^T \boldsymbol{A}_{k+1} = \begin{bmatrix} \boldsymbol{A}_k^T \boldsymbol{A}_k & \boldsymbol{A}_k^T \boldsymbol{a}_{l+1} \\ \boldsymbol{a}_{l+1}^T \boldsymbol{A}_k & \boldsymbol{a}_{l+1}^T \boldsymbol{a}_{l+1} \end{bmatrix} \quad (8)$$

$$\boldsymbol{A}_{k+1}^T \boldsymbol{b} = \begin{bmatrix} \boldsymbol{A}_k^T \boldsymbol{b} \\ \boldsymbol{a}_{l+1}^T \boldsymbol{b} \end{bmatrix} \tag{9}$$

$$(\boldsymbol{A}_{k+1}^T \boldsymbol{A}_{k+1})^{-1} = \frac{1}{\lambda} \begin{bmatrix} \lambda(\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} + (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{a}_{l+1} \boldsymbol{a}_{l+1}^T \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} & -(\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{a}_{l+1} \\ -\boldsymbol{a}_{l+1}^T \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} & 1 \end{bmatrix} \tag{10}$$

$$\boldsymbol{c}_{k+1} = \frac{1}{\lambda} \begin{bmatrix} \lambda \boldsymbol{c}_k + (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{a}_{l+1} \boldsymbol{a}_{l+1}^T \boldsymbol{p}_k \\ \boldsymbol{a}_{l+1}^T \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{b} - \boldsymbol{a}_{l+1}^T \boldsymbol{b} \end{bmatrix} \tag{11}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k + \frac{1}{\lambda} \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{a}_{l+1} \boldsymbol{a}_{l+1}^T \boldsymbol{r}_k + \frac{1}{\lambda} \left(\boldsymbol{a}_{l+1}^T \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{b} - \boldsymbol{a}_{l+1}^T \boldsymbol{b} \right) \boldsymbol{a}_{l+1} \tag{12}$$

where $\lambda = \boldsymbol{a}_{l+1}^T \boldsymbol{a}_{l+1} - \boldsymbol{a}_{l+1}^T \boldsymbol{A}_k (\boldsymbol{A}_k^T \boldsymbol{A}_k)^{-1} \boldsymbol{A}_k^T \boldsymbol{a}_{l+1}$. It turns out that $\lambda \neq 0$ since $\boldsymbol{a}_{l+1}$ is not in the range of $\boldsymbol{A}_k$.

If $\boldsymbol{c}_{k+1}$ computed by this means has negative components, then we need to recalculate $\boldsymbol{r}_{k+1}$ by (5)–(7) with $\boldsymbol{A}'_{k+1}$, which comprises l' columns of $\boldsymbol{A}_{k+1}$. Since $\boldsymbol{A}_{k+1}$ has at most n columns, we may check every selection of columns in the order of decreasing l'; that is, progressively removing a column of $\boldsymbol{A}_{k+1}$ and substituting the resulting matrix for $\boldsymbol{A}_k$ in (5)–(7) until $\boldsymbol{c}_{k+1}$ and $\boldsymbol{r}_{k+1}$ satisfy the conditions (a) and (b) given in Section III-B. In general, we need to compute $\boldsymbol{A}_i^T \boldsymbol{A}_i$ and $(\boldsymbol{A}_i^T \boldsymbol{A}_i)^{-1}$ when $\boldsymbol{A}$, $\boldsymbol{A}^T \boldsymbol{A}$, and $(\boldsymbol{A}^T \boldsymbol{A})^{-1}$ are given, where $\boldsymbol{A}$ is full column rank and $\boldsymbol{A}_i$ is the matrix consisting of all the columns of $\boldsymbol{A}$ excluding the i-th column $\boldsymbol{a}_i$.

Let $\boldsymbol{B}_{11}$ be the matrix consisting of all the entries of $\boldsymbol{A}^T \boldsymbol{A}$ except the i-th row and the i-th column, $\boldsymbol{B}_{12}$ the i-th column of $\boldsymbol{A}^T \boldsymbol{A}$ except the (i,i)-th entry, $\boldsymbol{B}_{21}$ the i-th row of $\boldsymbol{A}^T \boldsymbol{A}$ except the (i,i)-th entry, B_{22} the (i,i)-th entry of $\boldsymbol{A}^T \boldsymbol{A}$. Let $\boldsymbol{C}_{11}$, $\boldsymbol{C}_{12}$, $\boldsymbol{C}_{21}$, and C_{22} be the matrices constructed from $(\boldsymbol{A}^T \boldsymbol{A})^{-1}$ in the similar way. It is evident that $\boldsymbol{B}_{11} = \boldsymbol{A}_i^T \boldsymbol{A}_i$ and $B_{22} = \boldsymbol{a}_i^T \boldsymbol{a}_i$. Thus $\boldsymbol{B}_{11}$ is nonsingular and B_{22} is nonzero. Moreover, these matrices satisfy

$$\boldsymbol{B}_{11} \boldsymbol{C}_{11} + \boldsymbol{B}_{12} \boldsymbol{C}_{21} = \boldsymbol{I} \tag{13}$$

$$\boldsymbol{B}_{11} \boldsymbol{C}_{12} + \boldsymbol{B}_{12} C_{22} = [\,0 \quad 0 \quad \cdots \quad 0\,]^T \tag{14}$$

$$\boldsymbol{B}_{21} \boldsymbol{C}_{11} + B_{22} \boldsymbol{C}_{21} = [\,0 \quad 0 \quad \cdots \quad 0\,] \tag{15}$$

$$\boldsymbol{B}_{21} \boldsymbol{C}_{12} + B_{22} C_{22} = 1. \tag{16}$$

It can be proved from (14) and (16) that C_{22} is also nonzero. From (13)–(15) we have

$$\boldsymbol{B}_{11} \left(\frac{1}{B_{22} C_{22}} \boldsymbol{C}_{12} \boldsymbol{B}_{21} \boldsymbol{C}_{11} + \boldsymbol{C}_{11} \right) = \boldsymbol{I}.$$

Therefore we attain

$$(\boldsymbol{A}_i^T \boldsymbol{A}_i)^{-1} = \boldsymbol{B}_{11}^{-1} = \frac{1}{B_{22} C_{22}} \boldsymbol{C}_{12} \boldsymbol{B}_{21} \boldsymbol{C}_{11} + \boldsymbol{C}_{11}. \tag{17}$$

IV. Force-Closure Test

A. Force-Closure Definition

Multifingered grasps and fixtures have the same model of statics, which is described as below. Assume that a 3-D object is grasped or fixtured with m contacts. The contacts can be frictionless point contact (FPC), point contact with friction (PCwF), and/or soft finger contact (SFC). Let $\boldsymbol{n}_i$, $\boldsymbol{o}_i$, and $\boldsymbol{t}_i$ denote the unit inward normal at contact i ($i = 1, 2, \ldots, m$) and two unit tangent vectors such that $\boldsymbol{n}_i = \boldsymbol{o}_i \times \boldsymbol{t}_i$. Then the contact force can be expressed in the coordinate frame $\{\boldsymbol{n}_i, \boldsymbol{o}_i, \boldsymbol{t}_i\}$ by

$$\text{FPC: } \boldsymbol{f}_i = [\,f_{i1}\,]$$

$$\text{PCwF: } \boldsymbol{f}_i = [\,f_{i1} \;\; f_{i2} \;\; f_{i3}\,]^T$$

$$\text{SFC: } \boldsymbol{f}_i = [\,f_{i1} \;\; f_{i2} \;\; f_{i3} \;\; f_{i4}\,]^T$$

where f_{i1} is the normal force component, f_{i2} and f_{i3} are two tangential force components, and f_{i4} is the spin moment about $\boldsymbol{n}_i$. To avoid separation and slip at contact, $\boldsymbol{f}_i$ must conform to the following contact constraints:

$$\text{FPC: } F_i = \{\, \boldsymbol{f}_i \mid f_{i1} \geq 0 \,\} \tag{18}$$

$$\text{PCwF: } F_i = \left\{ \boldsymbol{f}_i \mid f_{i1} \geq 0,\ \sqrt{f_{i2}^2 + f_{i3}^2} \leq \mu_i f_{i1} \right\} \tag{19}$$

$$\text{SFCl: } F_i = \left\{ \boldsymbol{f}_i \mid f_{i1} \geq 0,\ \frac{\sqrt{f_{i2}^2 + f_{i3}^2}}{\mu_i} + \frac{|\,f_{i4}\,|}{\mu_{si}} \leq f_{i1} \right\} \tag{20}$$

$$\text{SFCe: } F_i = \left\{ \boldsymbol{f}_i \mid f_{i1} \geq 0,\ \sqrt{\frac{f_{i2}^2 + f_{i3}^2}{\mu_i^2} + \frac{f_{i4}^2}{\mu_{si}'^2}} \leq f_{i1} \right\} \tag{21}$$

where μ_i is the Coulomb friction coefficient, and μ_{si} and μ'_{si} are the coefficients of spin moment for SFC with linear (SFCl) and elliptic (SFCe) models, respectively. The set F_i defines the set of feasible contact forces under the friction law at each contact, which is a convex cone, known as the friction cone.

The grasp matrix $\boldsymbol{G}_i$ transforms F_i into a convex cone in $\mathbb{R}^6$, which consists of all feasible wrenches contact i, where $\boldsymbol{G}_i$ takes one of the following forms dictated by its contact type:

$$\text{FPC: } \boldsymbol{G}_i = \begin{bmatrix} \boldsymbol{n}_i \\ \boldsymbol{q}_i \times \boldsymbol{n}_i \end{bmatrix}$$

$$\text{PCwF: } \boldsymbol{G}_i = \begin{bmatrix} \boldsymbol{n}_i & \boldsymbol{o}_i & \boldsymbol{t}_i \\ \boldsymbol{q}_i \times \boldsymbol{n}_i & \boldsymbol{q}_i \times \boldsymbol{o}_i & \boldsymbol{q}_i \times \boldsymbol{t}_i \end{bmatrix}$$

$$\text{SFC: } \boldsymbol{G}_i = \begin{bmatrix} \boldsymbol{n}_i & \boldsymbol{o}_i & \boldsymbol{t}_i & \boldsymbol{0} \\ \boldsymbol{q}_i \times \boldsymbol{n}_i & \boldsymbol{q}_i \times \boldsymbol{o}_i & \boldsymbol{q}_i \times \boldsymbol{t}_i & \boldsymbol{n}_i \end{bmatrix}$$

where $\boldsymbol{q}_i$ is the position vector of contact i and $\boldsymbol{0}$ denotes the zero vector. Then $\sum_{i=1}^{m} \boldsymbol{G}_i(F_i) = \boldsymbol{G}(F)$ is a convex cone in $\mathbb{R}^6$, containing all feasible wrenches that the m contacts can generate, where $\boldsymbol{G} = [\boldsymbol{G}_1 \ \ \boldsymbol{G}_2 \ \ \cdots \ \ \boldsymbol{G}_m]$ and $F = \prod_{i=1}^{m} F_i$.

The static model of cable-driven robots is similar to the case of FPC [24]. Then, $\boldsymbol{q}_i$ designates the attaching point of the i-th cable on the robot and $\boldsymbol{n}_i$ is the unit vector along the i-th cable indicating the direction of tension. In general, hence, we use $\boldsymbol{G}$ to depict any of multifingered grasps, fixtures, and cable-driven robots. These robotic systems are often required to achieve the force-closure property, which is defined as below.

Definition 1: A robotic system $\boldsymbol{G}$ is said to be *force-closure* if $\boldsymbol{G}(F) = \mathbb{R}^6$ [14].

B. Force-Closure Test Using the Distance Algorithm

The force-closure property means that the robotic system can generate any wrench in $\mathbb{R}^6$. Let $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$ be seven points in $\mathbb{R}^6$, which can positively span $\mathbb{R}^6$. Then we have

Proposition 1: A robotic system $\boldsymbol{G}$ is force-closure if and only if $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$ are all contained in $\boldsymbol{G}(F)$.

The convex cone F_i can be rewritten as

$$F_i = \mathrm{co}\, U_i \tag{22}$$

where U_i is called the primitive contact force set:

$$\text{FPC: } U_i = \{ \boldsymbol{f}_i \mid f_{i1} = 1 \} \tag{23}$$

$$\text{PCwF: } U_i = \left\{ \boldsymbol{f}_i \mid f_{i1} = 1, \sqrt{f_{i2}^2 + f_{i3}^2} = \mu_i \right\} \tag{24}$$

$$\text{SFCl: } U_i = \left\{ \boldsymbol{f}_i \mid f_{i1} = 1, \frac{\sqrt{f_{i2}^2 + f_{i3}^2}}{\mu_i} + \frac{|f_{i4}|}{\mu_{si}} = 1 \right\} \tag{25}$$

$$\text{SFCe: } U_i = \left\{ \boldsymbol{f}_i \mid f_{i1} = 1, \sqrt{\frac{f_{i2}^2 + f_{i3}^2}{\mu_i^2} + \frac{f_{i4}^2}{\mu_{si}'^2}} = 1 \right\}. \tag{26}$$

From (22) it follows that

$$\boldsymbol{G}_i(F_i) = \boldsymbol{G}_i(\mathrm{co} U_i) = \mathrm{co}(\boldsymbol{G}_i(U_i)) = \mathrm{co} W_i$$

where W_i is called the primitive contact wrench set:

$$W_i = \boldsymbol{G}_i(U_i). \tag{27}$$

Then

$$\boldsymbol{G}(F) = \sum_{i=1}^{m} \mathrm{co} W_i = \mathrm{co}\left(\bigcup_{i=1}^{m} W_i \right) = \mathrm{co} W \tag{28}$$

where

$$W = \bigcup_{i=1}^{m} W_i. \tag{29}$$

From (28), Proposition 1, and the distance defined in Section III, we obtain

Proposition 2: A robotic system $\boldsymbol{G}$ is force-closure if and only if $d(\boldsymbol{w}, \mathrm{co} W) = 0$ for all $\boldsymbol{w} = \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$.

For computing $d(\boldsymbol{w}, \mathrm{co} W)$, we indicate how to calculate $h_W(\boldsymbol{r})$ and $s_W(\boldsymbol{r})$ now. First, from (29) and Theorem 1(a) it follows that

$$h_W(\boldsymbol{r}) = h_{W_j}(\boldsymbol{r}), \ s_W(\boldsymbol{r}) = s_{W_j}(\boldsymbol{r}) \tag{30}$$

where $j = j(\boldsymbol{r})$ satisfies $h_{W_j}(\boldsymbol{r}) = \max_{i=1,2,\ldots,m} h_{W_i}(\boldsymbol{r})$. From (27) and Theorem 1(b) we next attain

$$\begin{aligned} h_{W_i}(\boldsymbol{r}) &= h_{U_i}(\boldsymbol{G}_i^T \boldsymbol{r}) = h_{U_i}(\boldsymbol{u}_i), \\ s_{W_i}(\boldsymbol{r}) &= \boldsymbol{G}_i s_{U_i}(\boldsymbol{G}_i^T \boldsymbol{r}) = \boldsymbol{G}_i s_{U_i}(\boldsymbol{u}_i) \end{aligned} \tag{31}$$

where

$$\boldsymbol{u}_i = \boldsymbol{G}_i^T \boldsymbol{r}. \tag{32}$$

The formulas for computing $h_{U_i}(\boldsymbol{u}_i)$ and $s_{U_i}(\boldsymbol{u}_i)$ are given below (see the Appendix for their derivations):

FPC:

$$h_{U_i}(\boldsymbol{u}_i) = u_{i1}, \ s_{U_i}(\boldsymbol{u}_i) = [1]; \tag{33}$$

PCwF:

$$\begin{aligned} h_{U_i}(\boldsymbol{u}_i) = u_{i1} + h_{T_i}, \ s_{U_i}(\boldsymbol{u}_i) &= \begin{bmatrix} 1 & \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} \end{bmatrix}^T, \\ h_{T_i} &= \mu_i \sqrt{u_{i2}^2 + u_{i3}^2}; \end{aligned} \tag{34}$$

SFCl:

$$h_{U_i}(\boldsymbol{u}_i) = u_{i1} + h_{T_i},$$

$$s_{U_i}(\boldsymbol{u}_i) = \begin{cases} \begin{bmatrix} 1 & \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} & 0 \end{bmatrix}^T, \\ \qquad \text{if } \mu_i \sqrt{u_{i2}^2 + u_{i3}^2} \geq \mu_{si} |u_{i4}| \\ \begin{bmatrix} 1 & 0 & 0 & \dfrac{\mu_{si}^2 u_{i4}}{h_{T_i}} \end{bmatrix}^T, \text{ otherwise,} \end{cases}$$

$$h_{T_i} = \max\left\{ \mu_i \sqrt{u_{i2}^2 + u_{i3}^2}, \mu_{si} |u_{i4}| \right\}; \tag{35}$$

SFCe:

$$h_{U_i}(\boldsymbol{u}_i) = u_{i1} + h_{T_i},$$

$$s_{U_i}(\boldsymbol{u}_i) = \begin{bmatrix} 1 & \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} & \dfrac{\mu_{si}'^2 u_{i4}}{h_{T_i}} \end{bmatrix}^T,$$

$$h_{T_i} = \sqrt{\mu_i^2 (u_{i2}^2 + u_{i3}^2) + \mu_{si}'^2 u_{i4}^2} \tag{36}$$

where u_{i1}, u_{i2}, u_{i3}, and u_{i4} are the components of $\boldsymbol{u}_i$. If $h_{T_i} = 0$ in the above equations, then $s_{U_i}(\boldsymbol{u}_i)$ can be taken to be any point of U_i.

V. Numerical Examples

We implement the proposed algorithm using MATLAB on a notebook with Pentium-M 1.86GHz CPU and 512MB RAM. Its effectiveness and efficiency are checked by the examples of the application, in which $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$ are taken to be the vertices of an origin-centered regular 6-simplex:

$$\boldsymbol{w}_v = \boldsymbol{w}_v^0 - \frac{1}{7}\sum_{v=1}^{7} \boldsymbol{w}_v^0 \quad \text{for } v = 1, 2, \ldots, 7$$

where

$$\boldsymbol{w}_1^0 = [1 \;\; 1 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$$
$$\boldsymbol{w}_2^0 = [-1 \;\; -1 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$$
$$\boldsymbol{w}_3^0 = [-1 \;\; 1 \;\; -1 \;\; 0 \;\; 0 \;\; 0]^T$$
$$\boldsymbol{w}_4^0 = [1 \;\; -1 \;\; -1 \;\; 0 \;\; 0 \;\; 0]^T$$
$$\boldsymbol{w}_5^0 = \begin{bmatrix} 0 & 0 & 0 & \sqrt{5} & 0 & 0 \end{bmatrix}^T$$
$$\boldsymbol{w}_6^0 = \begin{bmatrix} 0 & 0 & 0 & \sqrt{5}/5 & 2\sqrt{30}/5 & 0 \end{bmatrix}^T$$
$$\boldsymbol{w}_7^0 = \begin{bmatrix} 0 & 0 & 0 & \sqrt{5}/5 & 2/\sqrt{30} & \sqrt{42}/3 \end{bmatrix}^T.$$

Fig. 1 depicts a gold ingot gripped by four contacts. In Fig. 1(a), the contacts are PCwF, whose positions together with the normals therein are given by

$$\boldsymbol{q}_1 = \begin{bmatrix} 0 & -2 & 3\sqrt{3}/4 \end{bmatrix}^T, \; \boldsymbol{n}_1 = \begin{bmatrix} 0 & \sqrt{3}/2 & 1/2 \end{bmatrix}^T;$$
$$\boldsymbol{q}_2 = \begin{bmatrix} 3/4 & 0 & \sqrt{3}/2 \end{bmatrix}^T, \; \boldsymbol{n}_2 = \begin{bmatrix} -\sqrt{3}/2 & 0 & 1/2 \end{bmatrix}^T;$$
$$\boldsymbol{q}_3 = \begin{bmatrix} 0 & 2 & 3\sqrt{3}/4 \end{bmatrix}^T, \; \boldsymbol{n}_3 = \begin{bmatrix} 0 & -\sqrt{3}/2 & 1/2 \end{bmatrix}^T;$$
$$\boldsymbol{q}_4 = \begin{bmatrix} -3/4 & 0 & \sqrt{3}/2 \end{bmatrix}^T, \; \boldsymbol{n}_4 = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 \end{bmatrix}^T.$$

By the distance algorithm, we obtain that $d(\boldsymbol{w}, \mathrm{co}W) \neq 0$ for all $\boldsymbol{w} = \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$, as shown in Fig. 2(a). One $d(\boldsymbol{w}, \mathrm{co}W)$ being nonzero implies that the grasp is not force-closure. The total CPU time for running the distance algorithm for $\boldsymbol{w} = \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$ is 5.86 ms.

In Fig. 1(b), two contacts are FPC and the others are SFC. Their positions and normals are given by

$$\boldsymbol{q}_1 = \begin{bmatrix} 0 & -3/2 & \sqrt{3}/2 \end{bmatrix}^T, \; \boldsymbol{n}_1 = \begin{bmatrix} 0 & \sqrt{3}/2 & 1/2 \end{bmatrix}^T;$$
$$\boldsymbol{q}_2 = \begin{bmatrix} 0 & 3/2 & \sqrt{3}/2 \end{bmatrix}^T, \; \boldsymbol{n}_2 = \begin{bmatrix} 0 & -\sqrt{3}/2 & 1/2 \end{bmatrix}^T;$$

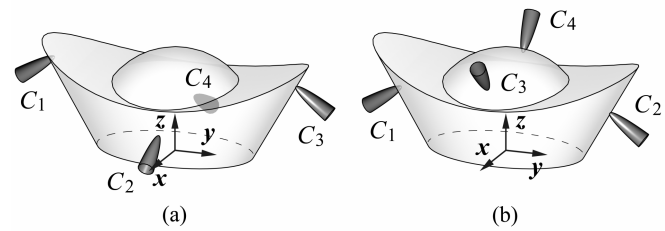

Figure 1. Gold ingot used as money in ancient China. (a) Contacts C_1–C_4 are all PCwF. (b) Contacts C_1, C_2 are SFC, while C_3, C_4 are FPC.

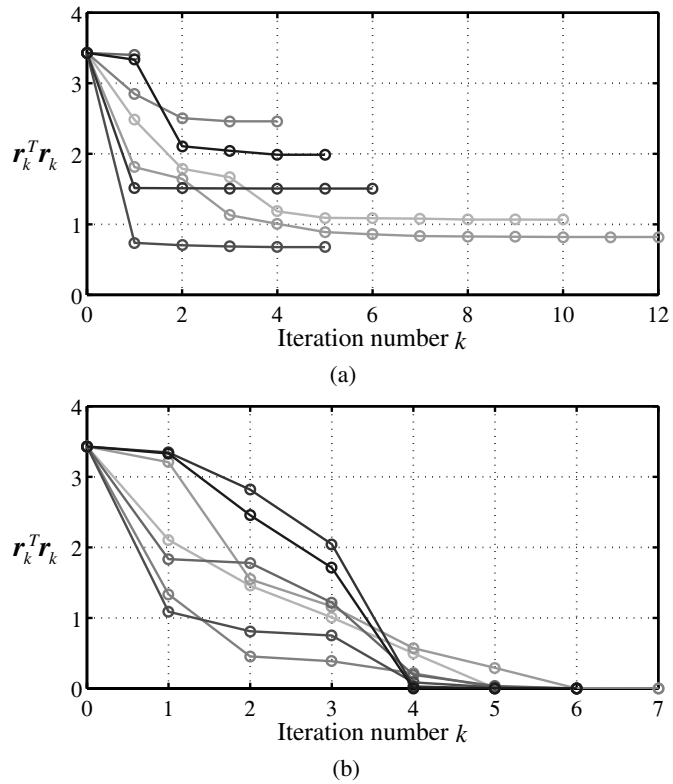

Figure 2. Results of the distance algorithm.

$$\boldsymbol{q}_3 = \begin{bmatrix} 3\sqrt{3}/8 & 0 & 3\sqrt{3}/4 \end{bmatrix}^T, \; \boldsymbol{n}_3 = [-3/5 \;\; 0 \;\; -4/5]^T;$$
$$\boldsymbol{q}_4 = \begin{bmatrix} -3\sqrt{3}/8 & 0 & 3\sqrt{3}/4 \end{bmatrix}^T, \; \boldsymbol{n}_4 = [3/5 \;\; 0 \;\; -4/5]^T.$$

By running the distance algorithm for $\boldsymbol{w} = \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_7$ with the CPU time of 4.95 ms, we attain Fig. 2(b), which indicates that the grasp is force-closure.

VI. Conclusion and Future Work

In this paper we have presented an algorithm for computing the distance between a point and a convex cone in $\mathbb{R}^n$. It turns out a sequence of polyhedral cones in the convex cone such that the distances between the point and the polyhedral cones converge to the distance to be computed. The closest point of the convex cone to the single point is also determined. With the aid of the recursive formulas for calculating the sequence of distances, the algorithm is very efficient. The application to the force-closure test in 6-D wrench space verifies its performance.

Future work can focus on the application of this algorithm to grasp or fixture synthesis and force distribution of multi-contact robotic system. Furthermore, the proposed algorithm is a general mathematical methodology. It may have applications in other scientific fields.

APPENDIX

For FPC, the derivation of $h_{U_i}(\boldsymbol{u}_i)$ and $s_{U_i}(\boldsymbol{u}_i)$ is trivial. For PCwF and SFC, from (23)–(26) they can be written as

$$h_{U_i}(\boldsymbol{u}_i) = u_{i1} + h_{T_i}(\boldsymbol{\tau}_i) = u_{i1} + h_{T_i},$$
$$s_{U_i}(\boldsymbol{u}_i) = \begin{bmatrix} 1 & s_{T_i}(\boldsymbol{\tau}_i)^T \end{bmatrix}^T$$

where $h_{T_i} = h_{T_i}(\boldsymbol{\tau}_i)$, $\boldsymbol{\tau}_i$ is a vector consisting of u_{i2}, u_{i3}, and/or u_{i4}, and T_i has one of the following forms:

$$\text{PCwF: } T_i = \left\{ [f_{i2} \;\; f_{i3}]^T \;\middle|\; \sqrt{f_{i2}^2 + f_{i3}^2} = \mu_i \right\}$$

$$\text{SFCl: } T_i = \left\{ [f_{i2} \;\; f_{i3} \;\; f_{i4}]^T \;\middle|\; \frac{\sqrt{f_{i2}^2 + f_{i3}^2}}{\mu_i} + \frac{|f_{i4}|}{\mu_{si}} = 1 \right\}$$

$$\text{SFCe: } T_i = \left\{ [f_{i2} \;\; f_{i3} \;\; f_{i4}]^T \;\middle|\; \sqrt{\frac{f_{i2}^2 + f_{i3}^2}{\mu_i^2} + \frac{f_{i4}^2}{\mu_{si}'^2}} = 1 \right\}.$$

The set T_i is a circle of $\mathbb{R}^2$ for PCwF, a bicone of $\mathbb{R}^3$ for SFCl, and an ellipsoid of $\mathbb{R}^3$ for SFCe. For PCwF, it is not so difficult to derive

$$h_{T_i}(\boldsymbol{\tau}_i) = \mu_i \sqrt{u_{i2}^2 + u_{i3}^2}, \quad s_{T_i}(\boldsymbol{\tau}_i) = \begin{bmatrix} \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} \end{bmatrix}^T.$$

For SFCl, $h_{T_i}(\boldsymbol{\tau}_i)$ may be attained at or $s_{T_i}(\boldsymbol{\tau}_i)$ may be a point on the circle or one of the two vertices. Then we attain

$$h_{T_i}(\boldsymbol{\tau}_i) = \max\left\{ \mu_i \sqrt{u_{i2}^2 + u_{i3}^2}, \mu_{si} |u_{i4}| \right\},$$

$$s_{T_i}(\boldsymbol{\tau}_i) = \begin{cases} \begin{bmatrix} \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} & 0 \end{bmatrix}^T, & \text{if } h_{T_i} = \mu_i \sqrt{u_{i2}^2 + u_{i3}^2} \\ \begin{bmatrix} 0 & 0 & \dfrac{\mu_{si}^2 u_{i4}}{h_{T_i}} \end{bmatrix}^T, & \text{otherwise.} \end{cases}$$

For SFCe, by algebraic computation, we obtain

$$h_{T_i}(\boldsymbol{\tau}_i) = \sqrt{\mu_i^2 (u_{i2}^2 + u_{i3}^2) + \mu_{si}'^2 u_{i4}^2},$$
$$s_{T_i}(\boldsymbol{\tau}_i) = \begin{bmatrix} \dfrac{\mu_i^2 u_{i2}}{h_{T_i}} & \dfrac{\mu_i^2 u_{i3}}{h_{T_i}} & \dfrac{\mu_{si}'^2 u_{i4}}{h_{T_i}} \end{bmatrix}^T.$$

REFERENCES

[1] E. G. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," IEEE J. Robot. Automat., vol. RA-1, no. 1, pp. 21–30, Feb. 1985.

[2] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," IEEE J. Robot. Automat., vol. 4, no. 2, pp. 193–203, Apr. 1988.

[3] E. G. Gilbert and C. P. Foo, "Computing the distance between general convex objects in three-dimensional space," IEEE Trans. Robot. Automat., vol. 6, no. 1, pp. 53–61, Feb. 1990.

[4] M. Lin and J. Canny, "A fast algorithm for incremental distance calculation," in Proc. IEEE Int. Conf. Robotics and Automation, Sacramento, CA, 1991, pp. 1008–1014.

[5] B. Mirtich, "V-Clip: Fast and robust polyhedral collision detection," ACM Trans. Graphics, vol. 17, no. 3, pp. 177–208, July 1998.

[6] S. A. Cameron, "Enhancing GJK: Computing minimum and penetration distances between convex polyhedra," in Proc. IEEE Int. Conf. Robot. Automat., Albuquerque, NM, 1997, pp. 3112–3117.

[7] ——, "A comparison of two fast algorithms for computing the distance between convex polyhedra," IEEE Trans. Robot. Automat., vol. 13, no. 6, pp. 915–920, Dec. 1997.

[8] C. J. Ong and E. G. Gilbert, "Fast versions of the Gilbert-Johnson-Keerthi distance algorithm: additional results and comparisons," IEEE Trans. Robot. Automat., vol. 17, no. 4, pp. 531–539, Aug. 2001.

[9] D. P. Dobkin, J. Hershberger, D. G. Kirkpatrick, and S. Suri, "Computing the intersection depth of polyhedra," Algorithmica, vol. 9, no. 6, pp. 518-533, June 1993.

[10] Y. L. Xiong and H. Ding, "General criterion and control strategy of collision-free movement for manipulators," Int. J. Robot. Automat., vol. 4, no. 2, pp. 75–80, 1989.

[11] C. J. Ong and E. G. Gilbert, "Growth distance: New measures for object separation and penetration," IEEE Trans. Robot. Automat., vol. 12, no. 6, pp. 888–903, Dec. 1996.

[12] X.-Y. Zhu, H. Ding, and S. K. Tso, "A pseudodistance function and its applications," IEEE Trans. Robot. Automat., vol. 20, no. 2, pp. 344–352, Apr. 2004.

[13] X.-Y. Zhu and J. Wang, "Synthesis of force-closure grasps on 3-D objects based on the Q distance," IEEE Trans. Robot. Automat., vol. 19, no. 4, pp. 669–679, Aug. 2003.

[14] J. K. Salisbury and B. Roth, "Kinematic and force analysis of articulated hands," ASME J. Mech. Transm. Automat. Design, vol. 105, no. 1, pp. 35–41, Mar. 1983.

[15] B. Mishra, J. T. Schwarz, and M. Sharir, "On the existence and synthesis of multifingered positive grips," Algorithmica, vol. 2, no. 4, pp. 541–558, Mar. 1987.

[16] Y.-H. Liu, "Qualitative test and force optimization of 3-D frictional form-closure grasps using linear programming," IEEE Trans. Robot. Automat., vol. 15, no. 1, pp. 163–173, Feb. 1999.

[17] X.-Y. Zhu, H. Ding, and Y. Wang, "A numerical test for the closure properties of 3D grasps," IEEE Trans. Robot. Automat., vol. 20, no. 3, pp. 543–549, June 2004.

[18] R. M. Murray, Z. X. Li, and S. S. Sastry, A Mathematical Introduction to Robotic Manipulation. Boca Raton, FL: CRC Press, 1994.

[19] L. Han, J. C. Trinkle, and Z. X. Li, "Grasp analysis as linear matrix inequality problems," IEEE Trans. Robot. Automat., vol. 16, no. 6, pp. 663–674, Dec. 2000.

[20] A. Bicchi, "On the closure properties of robotic grasping," Int. J. Robot. Res., vol. 14, no. 4, pp. 319–334, Aug. 1995.

[21] Y. L. Xiong, H. Ding, and M. Y. Wang, "Quantitative analysis of inner force distribution and load capacity of grasps and fixtures," ASME J. Manuf. Sci. Eng., vol. 124, no. 2, pp. 444–455, May 2002.

[22] Y. Zheng and W.-H. Qian, "Coping with the grasping uncertainties in force-closure analysis," Int. J. Robot. Res., vol. 24, no. 4, pp. 311–327, Apr. 2005.

[23] M. Y. Wang and D. M. Pelinescu, "Optimizing fixture layout in a point-set domain," IEEE Trans. Robot. Automat., vol. 17, no. 3, pp. 312–323, June 2001.

[24] P. A. Voglewede and I. Ebert-Uphoff, "Application of the antipodal grasp theorem to cable-driven robots," IEEE Trans. Robot. Automat., vol. 21, no. 4, pp. 713–718, Aug. 2005.