On Computing Four-Finger Equilibrium and Force-Closure Grasps of Polyhedral Objects

Jean Ponce, Steve Sullivan and Attawith Sudsang
Beckman Institute
University of Illinois
Urbana, IL 61801, USA

Jean-Daniel Boissonnat and Jean-Pierre Merlet
INRIA Sophia-Antipolis
2004 Route des Lucioles
06565 Valbonne Cedex, France

Submitted to the International Journal of Robotics Research, February 1995.

Abstract: This paper addresses the problem of computing stable grasps of three-dimensional polyhedral objects. We consider the case of a hand equipped with four hard fingers and assume point contact with friction. We prove new necessary and sufficient conditions for equilibrium and force closure, and present a geometric characterization of all possible types of four-finger equilibrium grasps. We then focus on concurrent grasps, for which the lines of action of the four contact forces all intersect in a point. In this case, the equilibrium conditions are linear in the unknown grasp parameters, which reduces the problem of computing the stable grasp regions in configuration space to the problem of constructing the eight-dimensional projection of an eleven-dimensional polytope. We present two projection methods: the first one uses a simple Gaussian elimination approach, while the second one relies on a novel output-sensitive contour-tracking algorithm. Finally, we use linear optimization within the valid configuration space regions to compute the maximal object regions where fingers can be positioned independently while ensuring force closure. We have implemented the proposed approach and present several examples.

1 Introduction

When a hand holds an object at rest, the forces and moments exerted by the fingers should balance each other so as not to disturb the position of this object. We will say that such a grasp achieves *equilibrium*. For the hand to hold the object securely, it should also be capable of preventing any motion due to external forces and torques. We will say that such a grasp achieves *force closure*. This paper addresses the problem of characterizing and computing four-finger equilibrium and force-closure grasps of polyhedral objects. Its main contributions are in two areas:

- *We give a new geometric characterization of equilibrium and force closure.* Assuming hard-finger contact and Coulomb friction, we show in the first part of the paper that *non-marginal* equilibrium grasps are in fact force-closure (Proposition 1). We then use line geometry to completely characterize all possible types of four-finger equilibrium grasps (Proposition 2) and prove a simple sufficient condition for equilibrium and force closure (Proposition 3).
- *We present an efficient algorithm for computing concurrent grasps.* In the second part of the paper we focus on *concurrent* grasps, a class of equilibrium grasps for which the lines of action of

the contact forces all intersect in a point. In this case, the equilibrium condition of Proposition 3 is linear in the grasp parameters, which reduces the problem of computing the stable grasp regions in configuration space to the problem of constructing the eight-dimensional projection of an eleven-dimensional polytope. We present two projection methods: the first one uses a simple Gaussian elimination approach, while the second one relies on a novel output-sensitive contour-tracking algorithm. Finally, we use linear optimization within the valid configuration space regions to compute the maximal object regions where fingers can be positioned independently while ensuring force closure.

1.1 Related Work

We briefly review the literature on force closure grasp analysis and synthesis and examine its relation with our work (see also [36, 42] and [38, Chapter 5] for recent surveys). The notion of force closure was introduced by Reulaux at the end of the nineteenth century in his study of the properties of mechanisms [49], and it has been used in the context of robotic grasp analysis since Salisbury's PhD work [53]. Most of the research on force-closure grasping has been concerned with the following problems:

• *Deciding how many fingers are required for force closure.* In the frictionless case, Reulaux [49], Somov [55] and, much later, Lakshminarayana [24] have shown that four (resp. seven) fingers are *necessary* to achieve force closure of a 2D (resp. 3D) object. In turn, Mishra, Schwartz, and Sharir [35] have shown that six (resp. twelve) fingers are always *sufficient* for objects without rotational symmetries, and Markenscoff, Ni, and Papadimitriou [28] have tightened this result by showing that under very general conditions, four (resp. seven) fingers are sufficient to achieve a force-closure grasp of a 2D (resp. 3D) object without rotational symmetries. They have also shown that when Coulomb friction is taken into account, three fingers are sufficient in the 2D case, and four are sufficient in the 3D case. This result and the recent availability of three- and four-finger hands such as the Salisbury hand [53] and the Utah/MIT Dextrous Hand [17] are practical motivations for our study of four-finger grasping under Coulomb friction.

• *Testing for force closure.* Given a set of n primitive contact wrenches (i.e., of vectors combining the forces and moments exerted by the fingers, see next section), Salisbury and Roth [53, Chapter 5], [54] have shown that a necessary and sufficient condition for force closure is that a strictly positive linear combination of the primitive wrenches is zero *and* the primitive wrenches span the whole wrench space (alternative proofs can be found in [39] and [57] for example). Mishra, Schwartz, and Sharir [35] have also shown that equilibrium is achieved when the origin of wrench space lies in the convex hull of the primitive wrenches, and force closure is achieved when the origin lies in the interior of the convex hull. These conditions are essentially binary in nature: a grasp is, or is not, force closure. This has motivated Kirkpatrick, Mishra, and Yap [23] and later Ferrari and Canny [11] to develop *quantitative* tests for force closure using the radius of a maximal ball centered at the origin and included in the convex hull of the primitive wrenches as a measure of goodness of the grasp. An alternative quality measure, proposed by Trinkle [57], is the maximum minimum value of the primitive wrench intensities achieving equilibrium.

The conditions for force closure mentioned so far hold for arbitrary numbers of fingers. In specific cases (two, three, or four fingers) it is possible to characterize the geometric arrangement of the contact forces that achieve equilibrium or force closure: in particular, it is easy to show that two forces are in equilibrium when they oppose each other and share the same line of action, and that three forces are in equilibrium when they add to zero and their lines of action intersect at a point. Ji and Roth [18, Chapter 4], [19] have used this fact to give several conditions on the

finger positions and surface normals that guarantee that the contact wrenches can resist any pure force, any pure moment, and any combination of force and moment. In the planar, two-finger case, Nguyen [39, Corollaries 3 and 4] has proven that non-marginal equilibrium implies force closure, and remarked that the geometric characterization of two-finger equilibrium given above also provided a sufficient condition for force closure. Ponce and Faverjon [45] have generalized this approach to the three-finger case, and we will use this approach once more in the four-finger case. Characterizing equilibrium geometrically is more difficult in this case, but it can be done using classical results from line geometry [2, 8, 33, 37].

• *Planning force-closure grasps.* Algorithms for grasp synthesis may be aimed at computing optimal grasp forces given fixed finger positions, at computing at least one (maybe optimal) force-closure grasp finger configuration, or at computing whole regions of the grasp configuration space that yield force closure. For given finger positions, Kerr and Roth [21, Chapter 5] [22] have proposed a numerical constrained-search algorithm for optimizing the equilibrium forces applied by the fingers, and Ji and Roth [18, Chapter 4] [20] have given an analytical method for minimizing the dependence of the forces achieving equilibrium on the friction coefficient. Mishra, Schwartz, and Sharir [35] have proposed linear-time algorithms for computing *at least one* finger configuration achieving force closure for frictionless polyhedral objects. Markenscoff and Papadimitriou have shown how to choose force-closure grasps of polygonal objects that minimize the worst-case forces that may have to be applied [29], and Mirtich and Canny [34] have recently proposed algorithms for computing optimal grasps of polyhedra according to the criterion of [11] (of course, many other optimality criteria could also be used, including the grasp efficiency measures mentioned earlier [23, 57] or the functional criteria proposed by Li and Sastry [27]).

In each of these works, the grasp-planning algorithm outputs a single grasp for a given set of contact faces. In [39], Nguyen has proposed instead a geometric method for computing *maximal independent* two-finger grasps of polygons, i.e., segments of the polygonal boundary where the two fingers can be positioned independently while maintaining force closure, requiring as little positional accuracy from the robot as possible, and Pollard and Lozano-Pérez [43] have used a direct generalization of this approach to plan three-finger grasps of polyhedral objects as part of a whole manipulation system, including obstacle avoidance, and feasibility and reachability tests. Faverjon, Ponce, and Stam [10, 46] and Chen and Burdick [6] have also generalized Nguyen's approach to two-finger grasping of curved objects by using algebraic cell decomposition and global optimization methods. In [45], Ponce and Faverjon have proposed a totally different computational approach to three-finger grasp planning, relying on variable elimination (or equivalently, polytope projection [12, 16, 25, 26]) to characterize the regions of the grasp configuration space that yield force closure, and using linear optimization within these regions to compute maximal independent grasps. We generalize this approach to three-dimensional four-finger grasping in this paper. From an algorithmic viewpoint, the main difference with the two-dimensional three-finger case is that the dimension of the configuration space is much higher (eleven instead of five), which has prompted us to replace the variant of Fourier's projection algorithm [12] used in [44, 45] by novel and much more powerful algorithms for projecting a polytope from a high-dimensional space onto a lower-dimensional sub-space.

The rest of the presentation is organized as follows. Section 2 discusses the notions of force closure and equilibrium, and gives several necessary and/or sufficient conditions for equilibrium and force closure. Section 3 describes our grasp-planning algorithm. Results are presented in Section 4. Finally, future research directions are briefly discussed in Section 5. All proofs are relegated to the appendix. A preliminary version of this paper appeared in [47].

2 Geometric Conditions for Equilibrium and Force Closure

In this section we start with some elementary notions of screw theory, then formally define force closure and equilibrium and clarify the relationship between these two concepts. We also give several necessary and/or sufficient conditions for equilibrium and force closure, and present a complete characterization of all types of four-finger equilibrium grasps.

The whole discussion is based on the notion of *wrench* from screw theory [2]. Wrenches are six-dimensional vectors combining forces and moments, and are most conveniently studied using a line- rather than a point-based geometry. Accordingly, our characterization of equilibrium grasps is based on the classification of certain varieties of lines in Grassmann geometry [8].

2.1 Screws, Twists and Wrenches

We recall some elementary notions of screw theory. The following is largely based on Roth's excellent introduction [52]. See [2, 4, 14, 31, 40, 41] for more details.

A *screw* is a straight line with a pitch. The pitch is a linear magnitude that can be thought of as the rectilinear distance through which a nut attached to an ordinary screw is translated parallel to the screw axis while the nut is rotated through a unit angle [2]. Screws provide a unified representation for displacements and forces: from Chasles' theorem, any displacement of a rigid body can be described by a single rotation about a unique axis, combined with a unique translation parallel to this axis. The rotation axis is called the screw axis, and the ratio of the linear translation to the rotation angle is the pitch of the screw. The displacement is referred to as a *twist* about a screw. Its magnitude is the angular rotation about the screw axis. Infinitesimal displacements and rigid body motions can also be described by twists. From Poinsot's theorem, any system of forces and moments applied to a rigid body can be uniquely replaced by a single force and a couple, such that the force is parallel to the axis of the couple. In turn, these can be represented by a unique screw axis, a moment about this axis, and a force along it. The pitch of the screw is the ratio of the moment to the force. This combination of force and couple is called a *wrench* acting on a screw. The magnitude of the wrench is the magnitude of the associated force.

Algebraically, a screw can be represented by a sextuple of *screw coordinates*:

$$\mathbf{s} = (\mathbf{u}, \mathbf{x} \times \mathbf{u} + p\mathbf{u}),$$

where $\mathbf{u}$ is a non-zero vector parallel to the screw axis, $\mathbf{x}$ denotes the coordinate vector of an arbitrary point on the axis, and p is the screw pitch. Alternatively, we can write the screw coordinates as $\mathbf{s} = (\mathbf{u}, \mathbf{v})$, where $\mathbf{u}, \mathbf{v}$ are three-dimensional vectors. Screw coordinates are homogeneous and a screw does not have a meaningful magnitude; in other words, screws form a five-dimensional projective space. However, screw coordinates can also be used to represent twists and wrenches, which are truly six-dimensional entities. In this case the magnitude of the screw coordinate vector is the magnitude of the associated twist or wrench.

We are now in a position to define force closure, but before closing this section, let us make one more remark that will be the key to our characterization of equilibrium and force-closure grasps: the wrench associated with a pure force (with no torque component) has a zero pitch; in other words, its screw coordinates are those of a line.

2.2 Force Closure and Equilibrium

We consider *positive grips* [35, 36] constructed as non-negative linear combinations of primitive wrenches (this amounts to assuming non-sticky fingers), and associate with a system of n primitive

wrenches $\mathbf{w}_1, \ldots, \mathbf{w}_n$ the wrench set

$$\mathcal{W} = \{\sum_{i=1}^{n} \xi_i \mathbf{w}_i : \xi_i \geq 0 \text{ for } i = 1, \ldots, n\}.$$

Definition 1 *A system of n wrenches $\mathbf{w}_1, \ldots, \mathbf{w}_n$ is said to achieve force closure when the corresponding wrench set $\mathcal{W}$ is equal to $\mathbb{R}^6$.*

Intuitively, a system of wrenches achieves force closure when any external load can be balanced by a non-negative combination of the primitive wrenches. Force closure is sometimes called *force/torque closure* [35, 36]. A related notion is *form closure* (also called *complete restraint*) [24, 49, 53]: a system of wrenches acting on some object is said to achieve form closure when it prevents all motions (including infinitesimal ones) of this object.[1] Force and form closure are dual of each other, in the same sense as wrenches and infinitesimal twists are dual notions [52] and, as noted in [36, 39] for example, force-closure grasps are form-closure and vice versa. Let us note that there is unfortunately no general agreement on terminology in the grasping literature (see [34, 57] for discussions of this problem): for example, Reulaux [49], Salisbury [53], Ji [18], Markenscoff et al. [28] and Trinkle [57] use the expression form closure for what we call force closure, and reserve the expression force closure for grasps that can only balance certain external loads. Our definitions match the ones used by Mishra et al. [35], Nguyen [39], and Murray et al. [38].

A somewhat weaker notion is equilibrium, defined below.

Definition 2 *A system of n wrenches $\mathbf{w}_1, \ldots, \mathbf{w}_n$ is said to achieve equilibrium when the convex hull of the points $\mathbf{w}_1, \ldots, \mathbf{w}_n$ in $\mathbb{R}^6$ contains the origin.*

In other words, a given system of wrenches achieves equilibrium when the equation

$$\sum_{i=1}^{n} \xi_i \mathbf{w}_i = 0 \tag{1}$$

admits a non-trivial, non-negative solution.

Mishra, Schwartz, and Sharir [35] have shown that a necessary and sufficient condition for a system of wrenches to achieve force closure is that the origin of $\mathbb{R}^6$ lies in the *interior* of the convex hull of the primitive wrenches. In particular, force closure implies equilibrium but there are wrench systems that achieve equilibrium but not force closure.[2]

2.3 Grasps and Friction

From now on we restrict our attention to systems of wrenches generated by hard fingers and assume Coulomb friction. While soft fingers can exert both pure forces and pure torques, a hard finger can only exert a pure force. The wrench associated with a hard finger located at a point $\mathbf{x}$ and exerting a force $\mathbf{f}$ is the *zero-pitch* wrench $\mathbf{w} = (\mathbf{f}, \mathbf{x} \times \mathbf{f})^T$ (here "$\times$" denotes the operator associating to two vectors the determinant of their coordinates, and $\mathbf{x} \times \mathbf{f}$ is the moment of the force $\mathbf{f}$ with respect to

[1] It should be noted that certain grasps which are not form-closure actually immobilize the grasped object: for example three frictionless fingers positioned at the centers of the edges of an equilateral triangle cannot prevent an infinitesimal rotation of the triangle about its center of mass, yet prevent any finite motion. See [7, 34, 50, 51] and Section 5 for discussions of this phenomenon.

[2] These systems of wrenches are called *strong force closure* systems by Trinkle [57]. Of course, as noted before, his notion of force closure is different from ours.

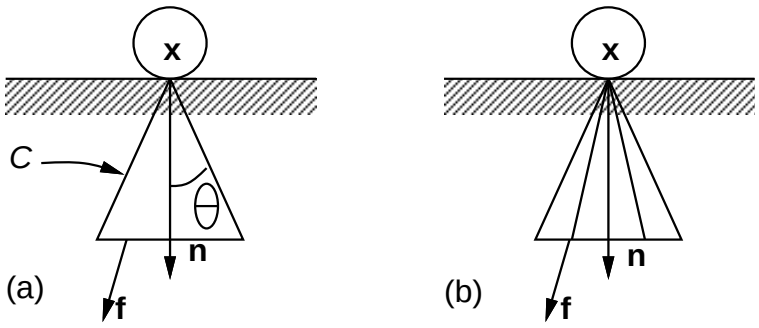


Figure 1: Coulomb friction: the friction cone C of (a) is replaced in (b) by a pyramidal cone.

the origin). Under Coulomb friction, $\mathbf{f}$ is constrained to lie in a friction cone C centered about the internal surface normal at $\mathbf{x}$ with half-angle θ (Figure 1(a)). The tangent of the angle θ is called the friction coefficient.

A force $\mathbf{f}$ in the friction cone is a non-negative combination of primitive unit forces bounding the cone; in other words it belongs to the wrench set associated with this *infinite* set of wrenches. In the sequel, we will approximate the friction cone by an m-sided pyramid (typical values of m are three or four), and the contact forces will be in the wrench set associated with the *finite* set of primitive wrenches corresponding to edges of this pyramid (Figure 1(b)).

A d-finger grasp is defined geometrically by the position $\mathbf{x}_i$ $(i = 1, .., d)$ of the fingers on the boundary of the object. We can associate with each grasp the wrench system formed by the primitive wrench systems corresponding to each finger. This allows us to extend the notions of force closure and equilibrium to grasps.

Definition 3 *A d-finger grasp is said to achieve force closure (resp. equilibrium) when the corresponding system of primitive wrenches achieves force closure (resp. equilibrium).*

Intuitively, a grasp achieves force closure when any external load can be balanced by the wrenches associated with forces lying in the friction cones at the fingertips, while a grasp achieves equilibrium when there exist forces in the friction cones, not all of them being zero, such that the associated wrenches add to zero.[3]

As noted before, force-closure grasps always achieve equilibrium but equilibrium grasps are not always force-closure. We now introduce a sub-class of equilibrium grasps that will be guaranteed to achieve force closure.

Definition 4 *A d-finger grasp is said to achieve non-marginal equilibrium when there exists a set of forces in the* open *friction cones at the fingertips such that the sum of the associated wrenches is zero.*

In other words, a d-finger grasp achieves non-marginal equilibrium when the equation (1) associated with the corresponding system of wrenches admits a strictly positive solution.

Proposition 1 *In the presence of friction, a sufficient condition for three-dimensional, d-finger force closure with $d \geq 3$ is non-marginal equilibrium.*

[3]Of course, real robot hands can only exert bounded forces, so ideal force-closure grasps are not physically achievable. We will come back to this problem in Section 5. See also [23, 11, 34] for discussions of related issues.

This proposition is a generalization of Nguyen's similar result in the two-finger case [39, Corollary 4], and its constructive proof is given in the appendix (see [45] for the two-dimensional three-finger case). It should be noted that Proposition 1 is *not* a trivial corollary of the result by Mishra, Schwartz, and Sharir [35] stated earlier: given *arbitrary* primitive wrenches, (1) may admit a strictly positive solution while the interior of the convex hull of the primitive wrenches is empty. For example, if the primitive wrenches all lie in some hyper-plane, their convex hull will also lie in this hyper-plane, which has an empty interior in $\mathbb{R}^6$. Proposition 1 shows that the *particular* systems of wrenches associated with $d \geq 3$ friction cones yield wrench sets with non-empty interiors.

A grasp achieving equilibrium with non-zero forces for a given friction coefficient trivially achieves non-marginal equilibrium for any strictly greater friction coefficient. This is an extremely important point. In particular, the linear conditions for equilibrium under friction presented in Section 2.6 will allow us in Section 3 to derive an efficient algorithm for planning equilibrium *and thus force-closure* grasps.

2.4 Line Geometry

As noted earlier, a zero-pitch wrench $\mathbf{w} = (\mathbf{f}, \mathbf{x} \times \mathbf{f})$ represents a pure force $\mathbf{f}$ applied at a point $\mathbf{x}$ and its moment $\mathbf{x} \times \mathbf{f}$ with respect to the origin; its screw coordinates also represent a straight line, the line of action of the force $\mathbf{f}$ passing through $\mathbf{x}$. In this case, the six screw coordinates $(w_1, .., w_6)$ of $\mathbf{w}$ are called the Plücker coordinates of the line (note that screw and Plücker coordinates only coincide for zero-pitch screws). Zero-pitch screws, or equivalently straight lines, form a four-dimensional variety in the five-dimensional projective space of all screws: more precisely, they form a quadric surface, called the Grassmannian, defined by $w_1 w_4 + w_2 w_5 + w_3 w_6 = 0$.

Equilibrium implies that the (screws associated with the) lines of action of the forces are linearly dependent. Grassmann geometry characterizes the varieties of various dimensions formed by sets of dependent lines [8, 33], and it can be used to characterize equilibrium geometrically. For example, a classical result is that three linearly-dependent lines form a flat pencil. In other words, a necessary condition for three-finger equilibrium in three dimensions is that the lines of action of the three forces be coplanar and intersect in a point. This condition is similar to those presented in [18, 47], and it can be used to synthesize three-finger force-closure grasps of polyhedral objects. Here, we concentrate on the four-finger case, as illustrated by the following proposition, which is another classical result from Grassmann geometry [8, 14, 33].

Theorem 1 *A set of four linearly-dependent lines either lie in a single plane, intersect in a single point, form two flat pencils having a line in common but lying in different planes, or form a regulus (Figure 2).*

The lines in a regulus lie on a doubly-ruled hyperboloid of one sheet (Figure 2(d)) which is not necessarily a surface of revolution. A regulus can be defined as the set of lines intersecting a fixed set of three skew lines (Figure 2(e)). Note that Theorem 1 is attributed by Ball [2, pp. 186-187, p. 511] to Möbius [37, p. 177] in the following form: if four forces are in equilibrium they must be generators of the same hyperboloid. (The hyperboloids associated with four coplanar or intersecting lines and two pencils lying in different planes and having one line in common are degenerate.)

From now on we restrict our attention to non-planar grasps, i.e., to sets of four contact forces whose lines of action do not all lie in the same plane. Algorithms for computing planar three-finger grasps can be found in [18, 20, 43].

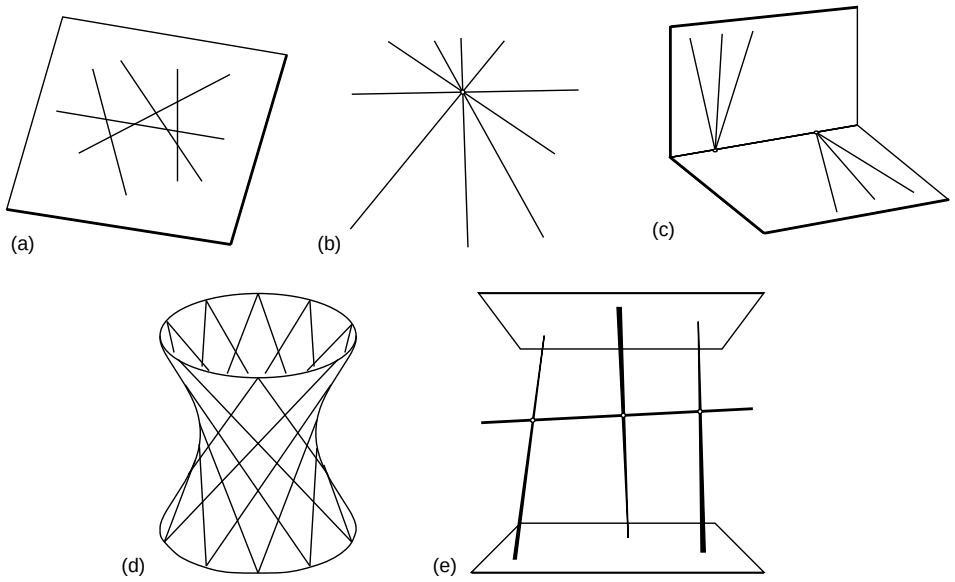


Figure 2: Configurations of linearly-dependent lines: (a) coplanar lines, (b) concurrent lines, (c) two flat pencils of lines having a line in common, (d) a regulus. As shown in (e), each line in the regulus intersects three skew lines. (After [13].)

2.5 A Necessary and Sufficient Condition for Equilibrium

Equilibrium implies the linear dependence of lines belonging to the double-sided friction cones. When is the converse true? Clearly, not every grasp whose contact wrenches are linearly dependent achieves equilibrium: think for example of four forces exerted on the four top sides of a pyramid, all pushing toward the same half-space, with lines of action intersecting in some point. These lines are linearly dependent, but do not achieve equilibrium.

Another interesting question is: can the three types of linearly-dependent non-coplanar lines yield equilibrium grasps? The answer is yes, as demonstrated graphically by Figure 3. If we choose four contact forces adding to zero and group these forces into pairs whose directions lie in two planes, we see that the three types of grasps form a hierarchy characterized by how the lines of action of the forces do or do not intersect in these planes: in Figure 3(a), the four lines intersect in a point that lies in both planes, while in Figure 3(b) the two pairs of lines have been pulled apart, each pair of lines intersecting on the line formed by the intersection of the two planes. Finally, in the regulus case, the lines in each pair do not intersect anymore; they have been pulled away from their original plane and now lie parallel to this plane at a common distance from it (Figure 3(c)). All three of these grasps achieve equilibrium: since the sum of the forces is zero, it is sufficient to show that the resulting moment is zero for some particular choice of origin. This is obvious in the case of Figure 3(a), using the intersection point of the forces as origin. A simple calculation shows that choosing as origin the mid-point of the segment where the two planes intersect also yields a zero moment for the examples of Figure 3(b)-(c). Intuitively, both pairs of forces have a zero moment with respect to that point in the case of Figure 3(b), while the two pairs have opposite moments in the case of Figure 3(c).

The main result of this section is a necessary and sufficient condition for forces whose lines of action are linearly dependent to achieve equilibrium in the case where the contact points are not all coplanar (Proposition 2). We will need the following definition.

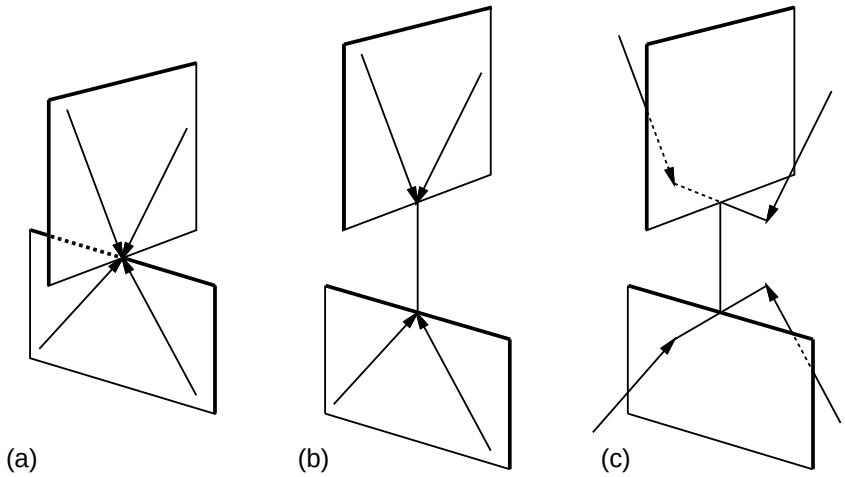


Figure 3: Examples of grasps achieving equilibrium: (a) the contact forces intersect in one point; (b) they form two non-coplanar pencils; (c) they form a regulus. The contact faces are not shown.

Definition 5 *We say that a set of vectors positively span $\mathbb{R}^n$ when any vector in $\mathbb{R}^n$ can be written as a positive combination of these vectors.*

Using this terminology, a grasp achieves force closure when the set of wrenches that can be exerted by the fingers positively span $\mathbb{R}^6$. Here we concentrate on conditions that quadruples of vectors in $\mathbb{R}^3$ must satisfy to positively span that space. The following lemma gives two such conditions.

Lemma 1 *Given four vectors in $\mathbb{R}^3$, the following statements are equivalent:*

(1) the vectors positively span $\mathbb{R}^3$;

(2) no three of the vectors are coplanar, and the zero vector is a strictly positive combination of the four vectors;

(3) no three of the vectors are coplanar, and the direction opposite to each vector lies in the interior of the trihedron formed by the other three.

Note that the trihedron formed by three vectors is the set of positive combinations of these vectors. The proof of the lemma is immediate, and it is omitted for the sake of conciseness. The lemma itself is important because it plays a major role in the proof of Proposition 2 (below) and in the statement of Proposition 3 (Section 2.6), both of which are keys to our grasp-planning approach.

Proposition 2 *A necessary and sufficient condition for four non-coplanar points to form an equilibrium grasp with four non-zero contact forces is that*

(P1) there exist four lines in the corresponding double-sided friction cones that either intersect in a single point, form two flat pencils having a line in common but lying in different planes, or form a regulus, and

(P2) the vectors parallel to these lines and lying in the internal friction cones at the contact points positively span $\mathbb{R}^3$.

The proof of this proposition is given in the appendix.

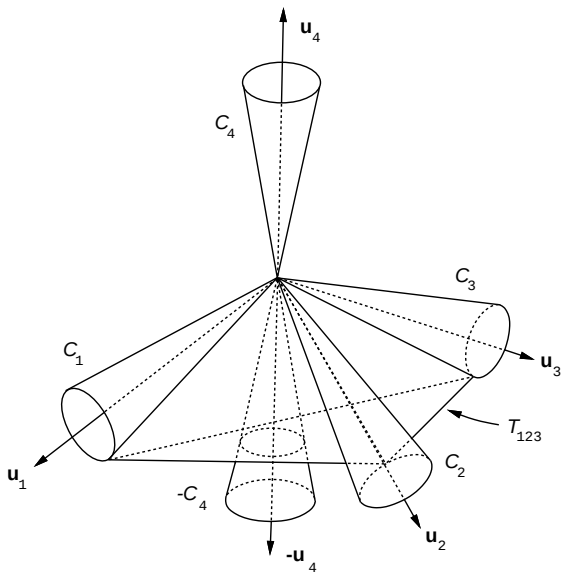


Figure 4: Four vectors θ-positively spanning $\mathbb{R}^3$. T_{123} is the intersection of the trihedra formed by all triples of vectors belonging to C_1, C_2, and C_3.

2.6 A Sufficient Condition for Equilibrium

We want conditions for equilibrium that are linear in the unknown grasp parameters (the finger positions and the contact forces) because this will allow us to use linear programming as a basis for grasp planning (see Section 3). The second condition (P2) of Proposition 2 is definitely non-linear: it can be written as

$$\sum_{i=1}^{4} \alpha_i \mathbf{f}_i = 0, \quad \text{with } \alpha_i > 0 \quad \text{for } i = 1, \ldots, 4,$$

which is a bilinear constraint on the unknown coefficients α_i and contact forces $\mathbf{f}_i$.

In this section, we give a sufficient condition for equilibrium, using a condition on the surface normals which ensures that (P2) is satisfied. Since, for a given choice of four contact faces, the surface normals are fixed, this replaces the non-linear condition (P2) by a simple test on these normals. We first need a definition. As before, θ denotes the friction angle.

Definition 6 *We say that four vectors θ-positively span $\mathbb{R}^3$ when, for any triple $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ of these vectors, the cones C_1, C_2, C_3 of half-angle θ centered on $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ lie in the interior of the same half-space, and the cone $-C_4$ of half-angle θ centered on the direction opposite to the fourth vector $\mathbf{u}_4$ lies in the interior of the intersection of the trihedra formed by all triples of vectors belonging to C_1, C_2, and C_3 (Figure 4).*

Clearly, any vector in $-C_4$ lies in the interior of the trihedron formed by any vectors in C_1, C_2, and C_3, and the following proposition is an immediate corollary of Lemma 1 and Proposition 2.

Proposition 3 *A sufficient condition for four non-coplanar points to form an equilibrium grasp with four non-zero contact forces is that*

(P1) there exist four lines in the corresponding double-sided friction cones that either intersect in a single point, form two flat pencils having a line in common but lying in different planes, or form a regulus, and

(P3) the surface normals at the four contact points θ-positively span $\mathbb{R}^3$.

It should be noted that results similar to Proposition 3 hold in the three-finger case [19, 18, 44, 45]. The main advantage of Proposition 3 over Proposition 2 is that it replaces the condition (P2) –which depends on the actual contact forces' directions– by condition (P3) –which depends on the normals to the grasped faces only. This breaks down the computation of equilibrium grasps into two steps: first select faces whose normals satisfy (P3), then compute the grasp configurations satisfying (P1). In the case of grasps whose contact forces intersect in a single point (*concurrent* grasps), it will be shown in Section 3 that (P1) can be decomposed into sixteen elementary conditions (the common point may lie in the internal or the external friction cone at each contact point), each of them being a conjunction of linear constraints, and this will allow us to use linear programming as a basis for grasp planning.

Proposition 3 does not yield obvious linear conditions for equilibrium in the case of forces lying in two flat pencils or in a regulus. Thus we will focus on concurrent grasps in the rest of this presentation. We will briefly come back to the general case in Section 5.

3 An Efficient Algorithm for Computing Concurrent Grasps

In this section, we restrict our attention to concurrent grasps formed by forces whose lines of action intersect in some point. We assume that the normals to the grasped faces θ-positively span $\mathbb{R}^3$ (this can be tested ahead of time), and present an algorithm for computing the maximal regions of the grasp configuration space that yield force-closure grasps.

We start in Section 3.1 by deriving the linear constraints that define stable grasp regions in an eleven-dimensional space (two parameters per finger plus three extra parameters defining the intersection of the contact forces). We then present in Section 3.2 two efficient algorithms for eliminating the three extra parameters and projecting the stable regions onto the actual eight-dimensional grasp configuration space. We finally discuss in Section 3.3 a simple method, based on linear programming, for computing *maximal independent contact regions* (a notion introduced by Nguyen in the two-dimensional case [39]).

Our overall approach is a generalization of the algorithm proposed in [44, 45] for the two-dimensional, three-finger case to the three-dimensional, four-finger case. The main difference is that the dimension of the configuration space is much higher in the latter case (eleven instead of five). This has prompted us to replace the variant of Fourier's projection algorithm [12] used in [44, 45] by novel and much more powerful algorithms for projecting a polytope from a high-dimensional space onto a lower-dimensional sub-space. These are detailed in Section 3.2, which forms the core of our algorithm presentation.

3.1 Linear Constraints

We assume that the faces of the grasped polyhedron are convex. (This is not a major restriction since non-convex faces could be triangulated into convex ones. Note that we do *not* assume that the polyhedron itself is convex.) Consider four faces F_1, F_2, F_3, F_4 of the polyhedron. Each face F_i can be defined parametrically by $\mathbf{x}_i = \mathbf{x}_{0i} + a_i\mathbf{u}_i + b_i\mathbf{v}_i$, where $(\mathbf{u}_i, \mathbf{v}_i)$ is a vector basis of F_i's plane (Figure 5). If F_i is bounded by n_i edges, the parameters a_i, b_i must also satisfy n_i linear constraints $f_{ij}(a_i, b_i) \leq 0$, with $j = 1, .., n_i$, expressing the fact that $\mathbf{x}_i$ must lie within the face F_i.

Let us represent the internal friction cone C_i associated with the face F_i by an m-sided pyramid, whose faces have internal normals $\mathbf{n}_{ij}$, with $j = 1, .., m$. Writing that a point $\mathbf{x}_0 = (x_0, y_0, z_0)$

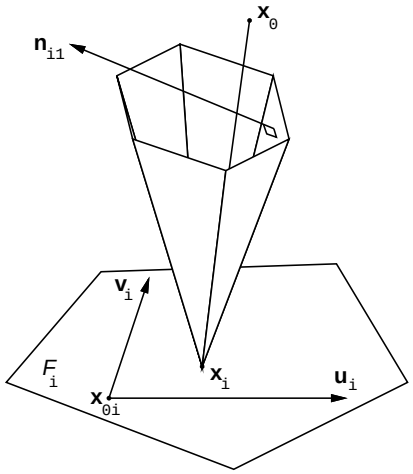


Figure 5: Representation of a pyramidal friction cone.

belongs to C_i yields the following constraints:

$$\begin{cases} (\mathbf{x}_0 - \mathbf{x}_{0i} - a_i \mathbf{u}_i - b_i \mathbf{v}_i) \cdot \mathbf{n}_{ij} \geq 0, & j = 1, \ldots, m, \\ f_{ij}(a_i, b_i) \leq 0, & j = 1, \ldots, n_i. \end{cases} \tag{2}$$

The $4m + \sum_{i=1}^{4} n_i$ constraints associated with the four friction cones define a polytope in $\mathbb{R}^{11}$. The equilibrium grasps satisfying the hypotheses of Proposition 3 can be found by considering in turn all possible combinations of internal and external friction cones at each contact point. The equations are the same as before except that some of the inequalities defining the friction cones in (2) will be reversed. The set of solutions is the union of the polytopes corresponding to the different combinations.

It is important to realize that linear programming can readily be used to assess whether a set of linear inequalities such as (2) admits a solution and to find representative grasp configurations optimizing some linear merit function. This is true even though (2) includes the x_0, y_0, z_0 unknowns besides the variables of interest, a_i, b_i.

However, it is convenient to characterize the equilibrium regions in the eight-dimensional grasp configuration space of the parameters a_i, b_i, independently of the variables x_0, y_0, z_0. This amounts to eliminating these three variables among the constraints (2) or, equivalently, to constructing the projection of the polytope defined by (2) in a eleven-dimensional space onto the eight-dimensional configuration space of the grasp. The projected polytope will itself be defined by a new system of linear constraints (2') in the variables a_i, b_i only. It should be noted that these new constraints yield a conservative test for equilibrium, hence force closure. More importantly, we will see in Section 4 that, while it is theoretically possible to find independent grasp regions without eliminating x_0, y_0 and z_0, this is much less efficient than working directly with the eight-dimensional equilibrium regions.

3.2 Projecting Polytopes

We now attack the problem of eliminating the coordinates of $\mathbf{x}_0$ among the force-closure constraints. This is equivalent to projecting an eleven-dimensional polytope onto an eight-dimensional sub-space. We solve this problem with general algorithms for projecting a d-dimensional polytope onto some

$(d-k)$-dimensional sub-space. Both d and k are assumed to be fixed in the complexity analysis of these algorithms.

Consider a polytope P, defined in the Euclidean space E of dimension d as the intersection of n half-spaces

$$H_i = \{\mathbf{x} : \mathbf{A}_i\mathbf{x} + b_i \leq 0\},\ i = 1, \ldots, n. \tag{3}$$

Here $\mathbf{A}_i = (A_{i1}, .., A_{id})$ is a $1 \times d$ real matrix, $\mathbf{x} = (x_1, \ldots, x_d)^T$ is the coordinate vector of a point in E, and b_i is a real number. The $(d-1)$-faces (i.e., the faces of dimension $d-1$, or facets) f_i of P lie in the hyper-planes bounding the half-spaces H_i, and, for $j = 1, .., d$, the $(d-j)$-faces of P lie in the intersection of j of these hyper-planes.

We address the problem of constructing the orthogonal projection Q of P onto a $(d-k)$-dimensional sub-space F of E. This is a classical problem in geometry with applications in constraint-based languages and spatial reasoning for example [1, 15]. A solution to this problem can be traced back to Fourier [12], but Fourier's method has a time complexity of $O(n^{2^k})$, and its output may contain many redundant hyper-planes [16]. Several other approaches are possible: for example, one can construct the vertices of the polytope P by computing the convex hull D of its dual, projecting (trivially) the vertices of D onto F, then constructing Q as the dual of the convex hull of the projected vertices. The cost of the computation is dominated by the construction of D, which takes time $O(n^{\lfloor \frac{d}{2} \rfloor})$ [5, 3]. Other algorithms based on convex hull and extreme point computation have also been proposed by Lassez and Lassez [26, 25].

While each approach is effective within certain domains (e.g. projection through many dimensions or with dense or redundant polytopes) there has yet to emerge an algorithm which performs well for all types of inputs. In this section we present two projection algorithms which have proven to be particularly suitable to our application, where high-dimensional polytopes are projected through a few dimensions. Our first algorithm is a variation of Fourier's algorithm which achieves an $O(n^{k+2})$ complexity via Gaussian elimination. (This algorithm was originally introduced in [45] but a simpler algorithm, which is a variant of Fourier's approach, was implemented in that paper.) The second one is a novel output-sensitive algorithm which tracks the edges of the apparent contour of the polytope being projected. Its time complexity is $O(tn)$ time complexity, where t is the size of the projection. We have implemented both algorithms, and an empirical comparison of their performances is given in Section 4.

From now on, we assume without loss of generality that $E = \mathbb{R}^d$ and F is defined by $x_i = 0$ for $1 \leq i \leq k$. Before detailing the two algorithms, let us define a few terms (Figure 6). We will say that a hyper-plane is *vertical* when it contains the directions $x_1, \ldots, x_k$. Vertical hyper-planes that contain a face of P are called *support planes*. The corresponding faces form the *apparent contour* of P, whose projection, called the *silhouette*, is the boundary of Q. Half-spaces bounded by a support plane and containing P are called *support half-spaces*. Together, the support half-spaces define a cylindrical polytope C that projects onto Q.

3.2.1 An Approach Based on Gaussian Elimination

We first present a two-step algorithm based on Gaussian elimination [45]: we first eliminate the variables $x_1, \ldots, x_k$ among the n inequalities defining P, which yields a set of candidate support planes and half-spaces; we then discard the redundant half-spaces through linear programming and construct Q by (trivially) projecting the remaining ones onto F.

Geometrically, the first step amounts to identifying all potential $(d-k-1)$-faces of the apparent contour, which in turn determines the support planes and half-spaces. (The contour may contain

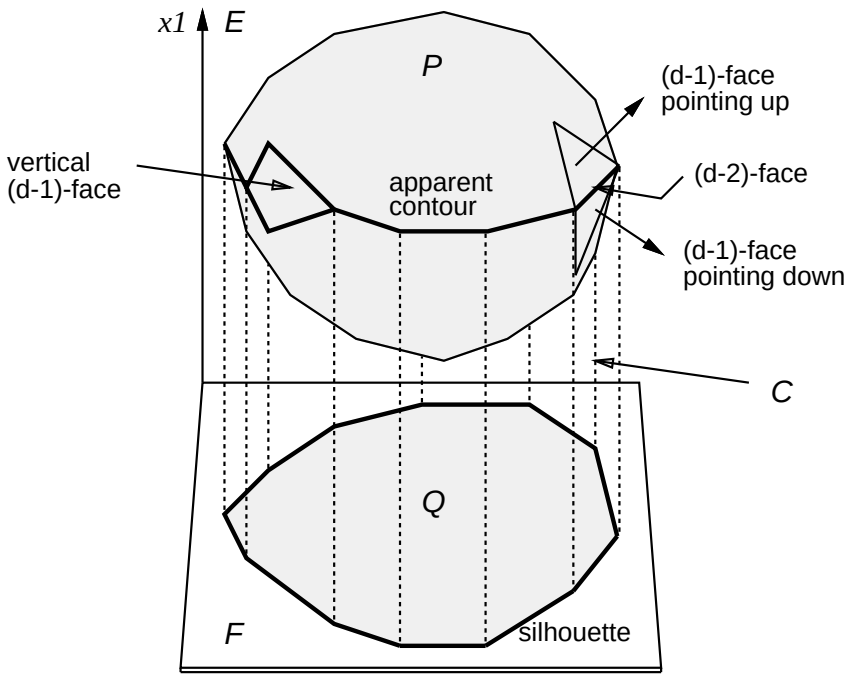


Figure 6: The geometry of the projection algorithm in the case $d = 3$, $k = 1$.

higher-dimensional faces, such as the vertical $(d-1)$-face in Figure 6, but the $(d-k-1)$-faces are sufficient to determine the support planes.)

Step 1. In principle, any $(k+1)$-tuple of $(d-1)$-faces may yield a $(d-k-1)$-face of the apparent contour. The inequalities defining the corresponding half-spaces can be written as

$$\begin{cases} A_{i_1,1}x_1 + \ldots + A_{i_1,k}x_k + A_{i_1,k+1}x_{k+1} + \ldots + A_{i_1,d}x_d + b_{i_1} \leq 0, \\ \ldots \\ A_{i_k,1}x_1 + \ldots + A_{i_k,k}x_k + A_{i_k,k+1}x_{k+1} + \ldots + A_{i_k,d}x_d + b_{i_k} \leq 0, \\ A_{i_{k+1},1}x_1 + \ldots + A_{i_{k+1},k}x_k + A_{i_{k+1},k+1}x_{k+1} + \ldots + A_{i_{k+1},d}x_d + b_{i_{k+1}} \leq 0. \end{cases}$$

We eliminate the variables $x_1, \ldots, x_k$ one by one in k Gaussian elimination steps. It is very important to remark that, because we deal with inequalities instead of equalities, the multiplicative coefficients used during Gaussian elimination must all be positive. In other words, we can only eliminate the variable x_1 if there exists some l in $\{1, \ldots, k+1\}$ such that, for all $m \neq l$ in $\{1, \ldots, k+1\}$, we have $A_{i_l,1}A_{i_m,1} \leq 0$ (intuitively, the normal to one of the hyper-planes under consideration must face "up" while the others face "down", see Figure 7). Eliminating x_1 yields a new system of k inequalities in $x_2, \ldots, x_k$. After k such steps, checking each time that one of the coefficients of the leading variable has a sign different from all the other ones, we obtain a single linear inequality

$$A^*_{k+1}x_{k+1} + \ldots + A^*_d x_d + b^* \leq 0 \tag{4}$$

in $x_{k+1}, \ldots, x_d$ only. This inequality defines the desired half-space.

Step 2. Some of the half-spaces found in the previous step actually are support half-spaces, but since we do not know the adjacency relationships between the hyper-planes bounding P, some other may be redundant (the hyper-planes bounding them actually lie outside the polytope, see Figure 8). We compute the signed distance between the original polytope and each candidate hyper-plane

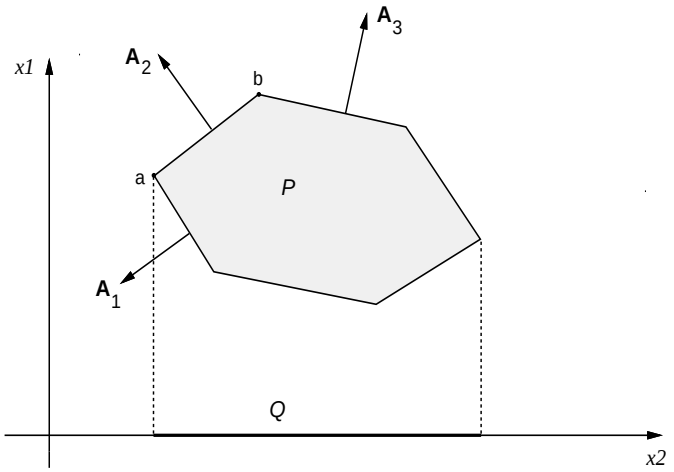


Figure 7: Sign constraints in the case $d = 2, k = 1$ (the $(d-2)$-faces are vertices): the signs of the x_1 coordinates of the normals $\mathbf{A}_1$ and $\mathbf{A}_2$ are different: the $(d-2)$-face a obtained by eliminating x_1 between the equations of the corresponding hyper-planes belongs to the apparent contour. The signs of the x_1 coordinates of the normals $\mathbf{A}_2$ and $\mathbf{A}_3$ are the same: the $(d-2)$-face b obtained by eliminating x_1 between the equations of the corresponding hyper-planes projects inside the silhouette.

by maximizing (4) under the original constraints (3) (this is a linear program). We then reject the hyper-planes lying at a strictly negative distance from the polytope.

What is the cost of this algorithm? We must consider $\binom{n}{k+1}$ $(k+1)$-tuples of $(d-1)$-faces, so for a fixed k we have to consider a total of $O(n^{k+1})$ tuples. For a fixed dimension, the cost of the best linear programming algorithms proposed so far is linear in the number of constraints [32], so rejecting redundant faces can be done in $O(n^{k+2})$ time.

Note that this algorithm is similar to Fourier's method [12]: in the latter algorithm, one variable is first eliminated among all possible pairs of inequalities; a second variable is then eliminated among all pairs of new inequalities, etc.., until k variables have been eliminated. In contrast, our algorithm directly eliminates k variables among all possible sets of $k+1$ inequalities. This simple modification allows us to improve the cost of projection from $O(n^{2^k})$ to $O(n^{k+2})$. It should also be noted that the projection method implemented in [10, 45] is essentially Fourier's algorithm.

3.2.2 A Contour-Tracking Approach

We now propose a novel algorithm that computes the projection Q of P onto F in an output-sensitive way. We restrict the discussion to the case $k > 1$. The case $k = 1$ can be easily solved using linear programming. We will assume that the hyper-planes $\mathbf{A}_i\mathbf{x} + b_i = 0$, $i = 1, \ldots, n$, are in general position, i.e. no $(d+1)$-tuples of hyper-planes have a common intersection and no two vertices of P project onto the same vertex of Q. This ensures that the cone formed by the faces of P incident to each vertex contains a bounded number of faces and that the size of the set of edges that belong to the apparent contour is linear in the number of faces of Q. General techniques such as the simulation of simplicity of [9] can be used to make this hypothesis valid. We will also assume that P is bounded (which is clearly the case in our grasping application). Methods for dealing with unbounded polytopes are described in [25].

The algorithm is again divided into two steps: we first use linear programming to find an initial point on the apparent contour; we then track the edges contained in the contour by shooting rays from the visited vertices, and record for each edge the support plane containing it as well as the

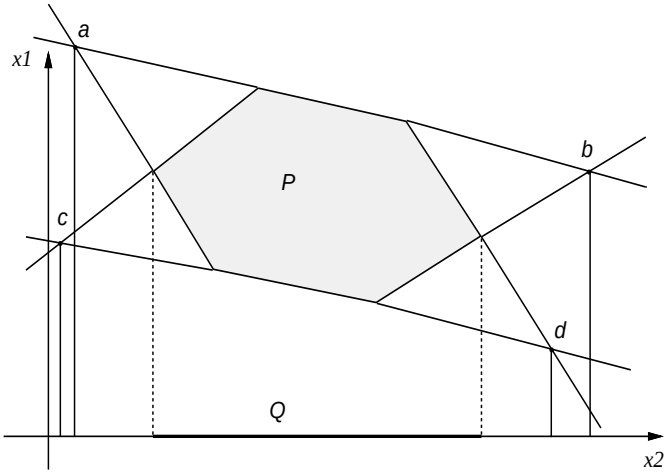


Figure 8: Potential $(d-2)$-faces a, b, c, d actually project outside the silhouette. In this example, $d = 2$, $k = 1$, and the $(d-2)$-faces are vertices.

associated support half-space.

Step 1. Any vertex of P that is extremal in some arbitrary direction orthogonal to the projection directions is guaranteed to lie on the apparent contour. Since P is bounded, such a vertex exists and can be found through linear programming (we extremize the corresponding coordinate under the constraints defining P). Once the vertex is found, we put it in a stack S, along with the hyper-planes passing through it.

Step 2. We track the edges of the apparent contour by repeating the following steps until S is empty:

(a) Take a vertex v out of S and put it in a dictionary D_v of already considered vertices. Construct the support half-spaces whose boundary contains v and the rays contained in the cone formed by the hyper-planes bounding P and incident to v.

(b) Each ray r is considered in turn. If there exists a support plane containing both v and r, then

(i) store all such planes and associated support half-spaces in a dictionary D_h (if they do not already belong to D_h);

(ii) by intersecting r with the constraints defining P, compute the vertex w distinct from v of the edge contained in the ray. If w has not been already considered (it does not belong to D_v), add w (along with the incident hyper-planes) to S.

Let us show that the algorithm has an $O(tn)$ time complexity, where t is the size of Q: As noted before, a linear program defined by n constraints in a d-dimensional space can be solved in $O(n)$ time for a fixed d [32]. It follows that finding the initial vertex takes $O(n)$ time; finding the incident hyper-planes amounts to finding the constraints that are zero at the vertex and also takes $O(n)$ time, so the overall cost of Step 1 is $O(n)$.

Step 2(a) only requires constant time: from our general position assumption, each vertex is incident to exactly d rays. Any $d-1$ hyper-planes potentially define a ray. There are d such groups of $d-1$ hyper-planes and they can be selected in constant time among the hyper-planes incident to the vertex and stored with it in the dictionary. Each potential ray r may or may not actually bound the cone C. We can take advantage of the fact that the ray r touches the cone at a vertex v to devise a simple test: we compute two ray points lying on either side of v by intersecting r with

two hyper-planes; the ray belongs to the cone C if and only if one of the two points satisfies the constraints defining C. This test also requires constant time.

At most $\binom{d-1}{k+1}$ support planes may contain a given ray and they can be found in constant time. A ray contains an edge of the apparent contour if and only if it belongs to one of these planes. Conversely, a potential support plane actually supports the apparent contour if and only if it contains at least one contour ray. All these tests take constant time. Step 2(b) requires shooting each ray against all the hyper-planes bounding P, which takes $O(n)$ time. Note that the hyper-planes incident to the new vertex are found at no extra cost during ray shooting. It follows that the combined cost of Steps 2(a) and 2(b) is dominated by the $O(n)$ cost of Step 2(b). Since the loop in Step 2 is executed $O(t)$ times, the overall cost of this step is $O(tn)$.

In fact, this result can be further improved by preprocessing the hyper-planes H_i so as to answer the ray-shooting queries in sub-linear time, using a recent result by Matoušek and Schwarzkopf [30]. The details of the improved algorithm and its complexity analysis have been relegated to Appendix B. The main result is the following.

Proposition 4 *The projection of a polytope from a Euclidean space E of dimension d onto a $(d-k)$-dimensional sub-space F can be computed in time and space*

$$T = O\left(n^{\gamma+\varepsilon} t^{\gamma+\varepsilon}\right)$$

where ε is any positive constant, t is the size of Q, and

$$\gamma = \frac{1}{1 + \frac{1}{\lfloor \frac{d}{2} \rfloor}}.$$

In our case, $d = 11$ so $\gamma = 5/6$.

3.3 Finding Independent Contact Regions

We now return to the problem of computing the grasps. After eliminating the variables x_0, y_0, z_0, we obtain a set of constraints (2') defining the polytope representing all equilibrium grasps for each quadruple of faces. Because of the uncertainty in robotics systems, we would like to minimize the sensitivity of a grasp to positioning errors. A way of achieving this is to seek quadruples of *independent contact regions* (an idea introduced by Nguyen in the two-dimensional case [39]). These regions are such that for any quadruple of contact points chosen in them, the corresponding grasp achieves equilibrium. In the grasp configuration space, these regions are represented by parallelepipeds with sides aligned with coordinate axes and contained in the polytope of equilibrium grasps.

We define the *maximal* independent contact regions as those maximizing a criterion depending on their size and location. A reasonable criterion is to maximize the minimum of the lengths of the parallelepiped edges. Like in the three-finger planar case [10, 45], we have observed empirically that there is not, in general, a unique solution to this problem, and that, for sufficiently large faces, the size of the contact regions depends only on the size of the friction cones. In this case, there is an infinite set of maximal parallelepipeds, and we add a secondary criterion in order to select a unique solution: we try to center as well as possible the center of mass of the object in the tetrahedron formed by the contact points. This enables us to decrease the effect of gravitational and inertial forces during the motion of the robot.

We now show how to map the problem of finding the maximal independent contact regions into a linear programming problem. Recall that the position of finger number i on face F_i is defined by

two parameters a_i and b_i ($i = 1, .., 4$). A parallelepiped in the grasp configuration space can thus be defined by four rectangles $R_i = [a_i^-, a_i^+] \times [b_i^-, b_i^+]$, $i = 1, .., 4$, which correspond to its projection on the parameter space of each face. Because of convexity, we only need to verify that the 256 vertices of the parallelepiped are contained in the set of force-closure grasps –i.e., satisfy (2')– in order to guarantee that the entire parallelepiped is also contained in it.

Let u be the minimum of the lengths of the corresponding intervals, we add to the existing set of linear constraints the following ones:

$$\begin{cases} u \geq 0, \\ u \leq a_i^+ - a_i^-, \quad i = 1, \ldots, 4, \\ u \leq b_i^+ - b_i^-, \end{cases} \tag{5}$$

which express the fact that u is positive and smaller than the length of each interval. Maximizing the minimum length criterion thus reduces to maximizing u under the constraints (2') –written 256 times, once for each vertex of the parallelepiped– and (5).

The second criterion can also be expressed linearly by introducing an additional variable v measuring the L^∞ distance between the center of mass $\mathbf{g}_p = (x_p, y_p, z_p)$ of the grasped object and the center of mass $\mathbf{g}_c = (x_c, y_c, z_c)$ of the contacts corresponding to the centers of the rectangles R_i. By definition, $v = \max(|x_c - x_p|, |y_c - y_p|, |z_c - z_p|)$, which yields the following constraints:

$$\begin{cases} v \geq x_p - x_c, \\ v \geq x_c - x_p, \\ v \geq y_p - y_c, \\ v \geq y_c - y_p, \\ v \geq z_p - z_c, \\ v \geq z_c - z_p. \end{cases} \tag{6}$$

Since $\mathbf{g}_p$ is fixed and x_c, y_c, z_c depend linearly on the unknowns $a_i^-, a_i^+, b_i^-, b_i^+$, these constraints on v are themselves linear in all the unknowns. Minimizing the distance criterion amounts to maximizing $-v$.

In summary, finding the maximal independent contact regions amounts to solving a linear program: the constraints are (2') –again, written once for each one of the 256 vertices of the parallelepiped–, (5), and (6), and the objective function to maximize is a weighted combination $w_u u - w_v v$ of the two above criteria (the weights w_u and w_v are set a priori by the user, typically to 0.5 and 0.5). There are 18 variables, namely the main variables $a_i^-, a_i^+, b_i^-, b_i^+$ ($i = 1, .., 4$) and the auxiliary variables u and v. We rank the maximal independent contact regions found for different quadruples of faces by using the output value of the simplex (i.e., the weighted sum of the criteria after optimization). For each quadruple of faces, we choose the centers of the maximal independent contact regions as representative grasps.

It should be noted that the independent contact regions can also be found without projecting the original polytope. In this case, we use the original inequalities (2) instead of those defining the projection. However, we must also add 3×256 variables corresponding to the values of $\mathbf{x}_0$ at each vertex of the parallelepiped. The corresponding optimization involves many more variables than the original one, and is correspondingly much more expensive. As shown in Section 4, this is empirically confirmed by our experiments.

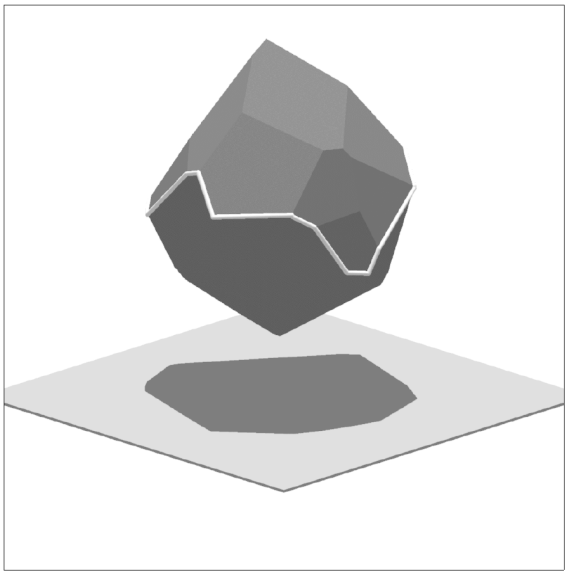


Figure 9: A three-dimensional polytope and its two-dimensional projection. The apparent contour and the projection have been computed using the contour-tracking technique of Section 3.2. Of course, the Gaussian elimination algorithm outputs the same projection.

4 Implementation and Results

In this section we describe our implementation and present some results. The implementation has been entirely written in C, using the simplex routine from *Numerical Recipes in C* [48] for linear programming. All timings reported in this section have been measured on a SUN SPARCstation 10.

4.1 Projection of Polytopes

Our implementation of the projection algorithms is relatively straightforward, and it can be used to project arbitrary polytopes from $\mathbb{R}^d$ onto $\mathbb{R}^{d-k}$ (Figure 9). We use hash tables to store vertices, edges, and planes, and associate with each of these geometric objects the corresponding numerical information (e.g., the position of a vertex) as well as a binary word recording the hyper-planes used to generate it. For a polytope defined by n constraints, we use a n-bit word whose i^{th} bit is set if and only if the corresponding object belongs to the hyper-plane numbered i. This allows us to use bit operations to determine very quickly if some object has already been considered, without having to worry about confusions due to numerical imprecision.

Below, we briefly describe the performance of our two projection methods in the grasp planning context. The corresponding polytopes are *not* in general position: in particular, the edges of the apparent contour often lie in more than ten of the hyper-planes bounding the original polytope. This causes the apparition of degenerate redundant hyper-planes in the projection, and we had to refine our two projection algorithms to eliminate those.

4.1.1 The Gaussian Elimination Algorithm

The Gaussian elimination algorithm finds the hyper-planes bounding the projection by eliminating three variables among all sets of four constraints. Even for a simple problem, this approach is quite

expensive: for example the polytope associated with four triangular faces and the corresponding four-sided friction cones is bounded by only 28 constraints, but $\binom{28}{4} = 20475$ combinations have to be checked. We take advantage of the fact that the hyper-planes which constrain the contact points to lie in each face are vertical, and add those to the projection from the outset. Furthermore, since combining any non-vertical planes with a vertical one produces the same vertical plane, we need only consider combinations of non-vertical planes. For the case above, these are just the planes corresponding to friction-cone constraints, with a total of $\binom{16}{4} = 1820$ combinations to be checked. Beyond the computational savings, this means that the cost of projection is almost exclusively determined by the number of planes used to approximate the friction cones and not by the complexity of the faces being grasped. Step 2 of the original Gaussian elimination algorithm is easily modified to eliminate degenerate support planes that only touch the original polytope along a lower-dimensional face: we maximize (4) under the original constraints (3) minus the constraint under consideration, and reject hyper-planes such that the distance found is negative *or zero*.

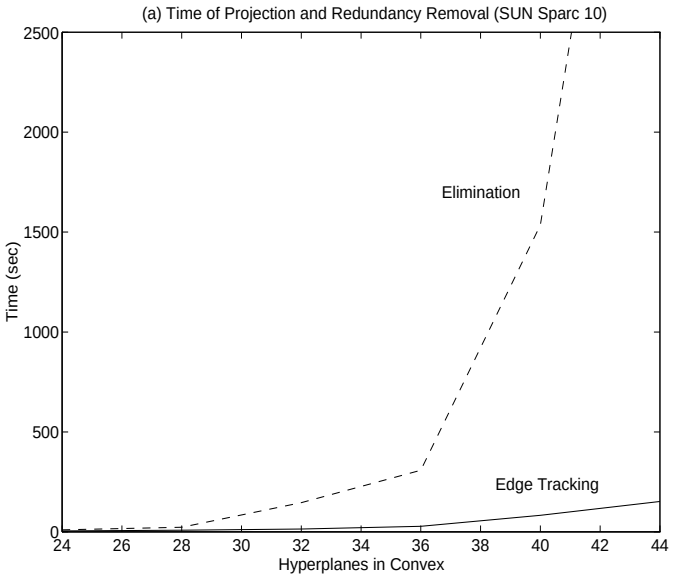
4.1.2 Tracking the Apparent Contour

We have implemented the contour-tracking algorithm as follows: at each successive vertex, we wish to determine which incident edges belong to the apparent contour. Given a vertex v, let us denote by H_v the set of hyper-planes incident to v, and by p_v the size of H_v. We first find a set of vertical hyper-planes containing v, then identify those edges which lie in at least one of the vertical planes. For each of the $\binom{p_v}{4}$ combinations of four hyper-planes in H_v, we perform Gaussian elimination to find the support planes containing v. Then, since edges are defined by $d-1$ hyper-planes, we check each subset of ten hyper-planes in H_v to see if it contains any of the combinations of four which successfully generated a support plane. For those sets which do contain a combination, we calculate the direction of the edge and check whether it intersects the polytope in another vertex. If so, the edge/vertex information is stored for further exploration (and to prevent duplication), and the support planes generated by the combinations associated with the edge are output. Though this procedure may seem circuitous, it can be implemented quite efficiently using our binary word representation to determine which objects have already been examined and whether a combination is included in a particular vertex or edge. Another advantage of this algorithm is that the support hyper-planes are generated as part of the exploration process. There is no need for a post-processing step to build the support planes from the projected vertices and edges.

For generic polytopes, the contour-tracking algorithm does not yield any redundant vertical hyper-plane. However, in the context of grasp planning, there may be vertical hyper-planes touching the original polytope along a lower-dimensional face. We take advantage of the vertex information to eliminate these redundant hyper-planes more efficiently than through linear programming: we project the vertices, then check each support plane to see how many dimensions are spanned by the vertices it contains. Any hyper-plane whose vertices span less than seven dimensions cannot yield a facet of the silhouette polytope, and thus can be rejected as redundant.

4.1.3 Results

To illustrate the relative performances of the Gaussian elimination and edge-tracking methods in the context of grasp planning, we have tested them on the polytope associated with the problem of grasping a tetrahedron with m-sided friction cones. Figure 10(a) shows the run times for values of m ranging from 3 to 8, corresponding to polytopes defined by 24 to 44 constraints. As might have been expected, the contour tracking method outperforms the Gaussian elimination method. The tables of Figure 10(b)-(c) give a quantitative comparison of the two algorithms. As shown by



(b) Cost of Projection for a Typical Grasp Polytope Gaussian Elimination Method						
Planes in Cone	Planes in Polytope	Combinations Checked	Output Size	Final Size	Projection Time	Total Time
3	24	495	84	84	1	10
4	28	1820	173	149	3	29
5	32	4845	291	214	5	67
6	36	10626	468	310	11	291
7	40	20475	829	492	35	1535
8	44	35960	1356	748	69	5229

(c) Cost of Projection for a Typical Grasp Polytope Contour-Tracking Method								
Planes in Cone	Polytope Size	Combinations Checked	Vertices Found	Edges Found	Output Size	Final Size	Projection Time	Total Time
3	24	432	196	1078	84	84	4	5
4	28	1227	368	2112	171	149	7	8
5	32	2355	650	3850	291	214	13	14
6	36	4088	1044	6336	464	310	26	28
7	40	7651	2058	12691	826	492	78	83
8	44	13106	3648	22784	1352	748	147	152

Figure 10: (a) Qualitative comparison of the run times for projecting the grasp polytope of a tetrahedron and removing redundant hyper-planes (averaged over all possible grasps). The dashed line indicates the Gaussian elimination method, and the solid line indicates the edge-tracking algorithm. (b) This table gives the number of hyper-planes involved at each step of the projection process, as well as the time (in seconds) for the Gaussian elimination method. The object grasped is a tetrahedron with m-sided friction cones, with m varying from 3 to 8. (c) Table of results for the contour-tracking projection method. The data is the same as before.

Figure 10(b), the elimination of redundant planes is by far the most expensive part of the Gaussian elimination method.

4.2 Finding Maximal Grasp Regions

As noted earlier, we could in principle find the maximal grasp regions without the projection step. In practice, the corresponding linear program involves more than 750 variables and thousands of constraints, and it simply never ran to completion –we interrupted it after a few hours– most likely due to the very large number of constraints and variables involved. In contrast, the linear program used to compute maximal independent regions from the projected polytope only involves 18 variables, and as shown below runs in seconds.

As discussed earlier, finding the maximal independent grasp regions amounts to inscribing an 8D box into the silhouette polytope, the shortest edge of which should be as long as possible. The box itself may be defined as a set of intervals for the face coordinates $[a_i^-, a_i^+]$, $[b_i^-, b_i^+]$, $i = 1, .., 4$. Because of convexity, the entire box will be contained in the polytope if each of the 256 vertices are valid grasps; our new polytope will consist of versions of the silhouette constraints for each of the 256 combinations of interval endpoints. However, since most of the constraints do not involve all of the face variables, we do not end up with 256 times the number of constraints in the silhouette. Table 1 shows how the size of the polytope varies for the tetrahedron with different friction cone approximations. Finding the maximal box thus amounts to solving a linear program with 18 variables, and for the tetrahedron problem with four-sided friction cones, this takes approximately 30 seconds on a SPARCstation 10.

Size of Grasp-Region Polytope				
Planes in Friction Cone	Planes in Convex	Silhouette Size	Final Convex Size	Time to Solve (sec)
3	24	84	2224	35
4	28	149	2224	30
5	32	214	12192	1265
6	36	311	11632	1050

Table 1: Sizes of a typical polytope used to find maximal grasp regions.

It is clear that the size of the final polytope, and thus the cost of solving the problem, increases quickly with the number of planes in the friction cone approximation. Aligning the sides of the friction cones with the coordinate axes of the grasped object's faces yields constraints which do not involve all variables, and this has the advantage of minimizing the total number of constraints bounding the grasp polytope. This alignment procedure is particularly efficient for friction cones with an even number of sides, as demonstrated by Table 1: the final polytope associated with four-sided friction cones has the same size as the polytope associated with three-sided cones, and the polyope associated with six-sided cones is actually smaller than the polytope associated with five-sided ones.

To illustrate the output of our algorithms, Figure 11 shows the five grasps found for a tetrahedron. It should be noted that the focus point of the forces belongs to the intersection of all *internal* cones in only one of these grasps (the central one of the top row). For each other grasp, the focus point belongs to one *external* cone and three internal ones. Of course, the forces themselves lie in the internal cones. Figure 12 shows some of the grasps found for an 18-face polyhedron.

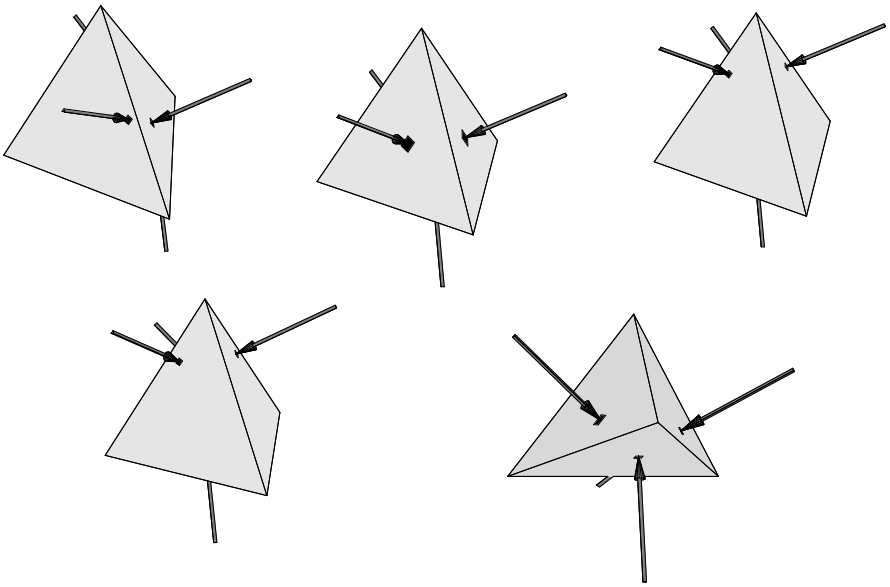


Figure 11: The five grasps found for the tetrahedron.

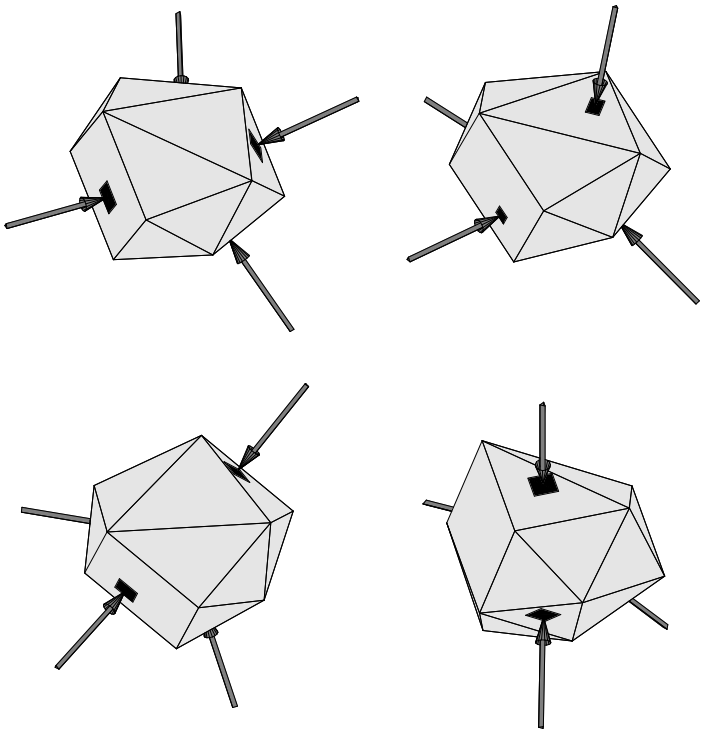


Figure 12: Four typical grasps of a more complicated object (18-sided polyhedron).

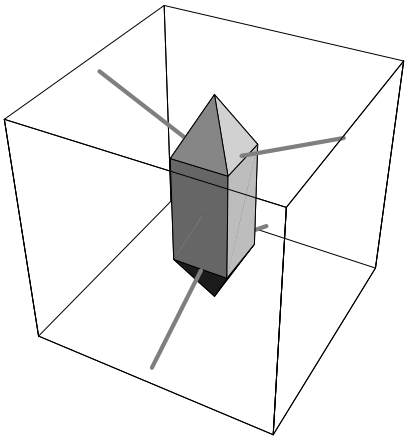


Figure 13: A grasp formed by forces lying in two flat pencils having a line in common but lying in different planes. This grasp was computed using the methods described in [56].

5 Discussion and Future Work

We have presented a geometric characterization of four-finger equilibrium and force-closure grasps, given algorithms for computing maximal concurrent grasps of polyhedral objects, and demonstrated efficient implementations of these algorithms. Let us conclude by discussing some of the issues raised by our work and by sketching some future research directions.

Even though Propositions 2 and 3 hold for all types of non-planar four-finger grasps, we have restricted our attention in the second half of the paper to forces whose lines of action all pass through some point. It is clearly important to generalize our approach to the other types of four-finger grasps. Grasps involving forces that lie in two flat pencils having a line in common may prove particularly important in practice, for example for grasping an elongated object with two cooperating robots equipped with simple two-finger grippers. We have recently developed a method using the intersection of inverted friction cones with the contact faces to compute this type of grasps the direction of the line common to the two pencils is constrained to lie in a prescribed cylinder [56], and Figure 13 shows an example. This is only a first step: as mentioned in Section 2, the equations that characterize equilibrium for these more general grasps are normally non-linear, and new methods will have to be developed.

So far, we have computed grasps that were optimal according to a criterion based on size of the independent grasp regions and how well the center of mass of the object is centered among the contact points. It would also be interesting to compute grasps that are optimal according to some functional consideration [11, 23, 27, 29, 34]: for example, given a fixed set of contact points and some bound on the magnitude of the contact forces, Ferarri and Canny [11] propose to compute a maximal ball centered at the origin and contained in the convex formed by the contact wrenches; clearly, the contact forces can generate any wrench contained in this ball, and its radius provides a measure of the grasp's efficiency. Computing a grasp configuration which is optimal according to this criterion is more challenging; it is a non-linear problem since the moment of the forces depends bilinearly on the finger positions and the force directions. We have recently implemented an iterative approach to this non-linear optimization problem [56] (see [34] for a different method).

In this paper we have restricted our attention to point contact with friction. In the frictionless case, it is known that four fingers in the plane and seven fingers in the three-dimensional case are necessary and, under very general assumptions, sufficient to achieve force- or form-closure

[24, 28, 35]. However, as noted in Section 2, certain grasps which do not achieve force- or form-closure actually immobilize the grasped object. Czyzowicz, Stojmenovic and Urrutia have recently shown that three fingers in the plane and four fingers in the three-dimensional case are sufficient to immobilize a polyhedron, with finger arrangements corresponding to concurrent grasps [7]. New techniques developed by Rimon and Burdick [50, 51] may allow us to decide whether the other two types of equilibrium grasps studied in this paper can also be used to immobilize an object in the absence of friction. More generally, it would be interesting to evaluate the quality of frictionless grasps achieved by a number of fingers between four and seven, with applications to fixture planning.

At the other end of the spectrum, we also plan to explore the case of soft fingers that can exert pure torques in addition to pure forces, a more difficult setting in which general screw theory still applies, but Grassmann geometry does not.

Finally, we plan to investigate other applications [25] of the contour-tracking projection algorithm. Taking full advantage of its potential efficiency will require using a linear-time routine for linear programming [32] as well as implementing the preprocessing step of the projection algorithm.

Appendix A: Proofs of Propositions 1 and 2

We first prove Proposition 1. See [28] for a related argument.

Proposition 1 *In the presence of friction, a sufficient condition for three-dimensional, d-finger force closure with $d \geq 3$ is non-marginal equilibrium.*

Proof : We first prove the three-finger case, then prove the d-finger case, with $d > 3$, as a simple extension. We assume that at least three of the contact points are not collinear.

The key remark is that, if a force $\mathbf{f}_i$ lies in an open friction cone, then there exists some $\varepsilon_i > 0$ such that, for any vector $\mathbf{v}_i$ in the open ball of radius ε_i, $\mathbf{f}_i + \mathbf{v}_i$ also lies in the friction cone. Equivalently, for any vector $\mathbf{v}_i$, there exists $\alpha_i = |\mathbf{v}_i|/\varepsilon_i > 0$ such that, for any $\alpha \geq \alpha_i$, $\alpha\mathbf{f}_i + \mathbf{v}_i$ lies in the friction cone. Given some external load on the grasped object, this will allow us to replace the equilibrium forces $\mathbf{f}_i$ by new forces $\mathbf{f}'_i$ that lie in the friction cones and balance this load.

Suppose that we exert a (not necessarily zero-pitch) external wrench $\mathbf{w} = (\mathbf{f}, \mathbf{m})$ on the object. We want to find a set of forces $\mathbf{f}'_1, \mathbf{f}'_2, \mathbf{f}'_3$ lying in the friction cones and balancing this wrench, i.e.,

$$\begin{cases} \mathbf{f} + \sum_{i=1}^{3} \mathbf{f}'_i = 0, \\ \mathbf{m} + \sum_{i=1}^{3} \mathbf{x}_i \times \mathbf{f}'_i = 0. \end{cases}$$

Since the three points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are not collinear, we can choose the origin outside of the plane formed by these points. It follows that the vectors $\mathbf{x}_i$ are linearly independent and the vectors $\mathbf{x}_i \times \mathbf{x}_{i+1}$ are also independent. Let us denote by $(\lambda_1, \lambda_2, \lambda_3)$ the coordinates of the vector $-\mathbf{x}_1 \times \mathbf{f} + \mathbf{m}$ in the coordinate system $(\mathbf{x}_3 \times \mathbf{x}_1, \mathbf{x}_1 \times \mathbf{x}_2, \mathbf{x}_2 \times \mathbf{x}_3)$.

Consider the vectors $\mathbf{v}_1 = \lambda_1\mathbf{x}_1 - \lambda_2\mathbf{x}_2 - \mathbf{f}$, $\mathbf{v}_2 = \lambda_2\mathbf{x}_2 - \lambda_3\mathbf{x}_3$, and $\mathbf{v}_3 = \lambda_3\mathbf{x}_3 - \lambda_1\mathbf{x}_1$. Suppose we apply the force $\mathbf{f}'_i = \alpha\mathbf{f}_i + \mathbf{v}_i$ in $\mathbf{x}_i$, for $i = 1, 2, 3$, with $\alpha = \max(\alpha_i)$, and α_i chosen so that $\alpha_i\mathbf{f}_i + \mathbf{v}_i$ lies in the friction cone at that point. Clearly, the resultant of the forces $\mathbf{f}'_i$ is $-\mathbf{f}$, and their total moment is $-\mathbf{m}$, which completes the proof.

The d-finger case ($d > 3$) can be treated similarly: we use the d forces $\mathbf{f}_i$, with $i = 1, .., d$ to achieve equilibrium, and construct additional forces $\mathbf{v}_{ij}$, with $j = 1, 2, 3$ in the three friction cones of three non-collinear contact points $\mathbf{x}_{ij}$ to balance the external force and moment. □

We now prove Proposition 2.

Proposition 2 *A necessary and sufficient condition for four non-coplanar points to form an equilibrium grasp with four non-zero contact forces is that*

(P1) there exist four lines in the corresponding double-sided friction cones that either intersect in a single point, form two flat pencils having a line in common but lying in different planes, or form a regulus, and

(P2) the vectors parallel to these lines and lying in the internal friction cones at the contact points positively span $\mathbb{R}^3$.

Proof : The condition is clearly necessary: since the equilibrium forces are non-zero, their lines of action are well defined, and they lie in the double-sided cones. In addition, since the sum of the corresponding wrenches is zero, the equilibrium forces are linearly dependent, but since the contact points are not coplanar, these forces are not coplanar either, and they satisfy (P1). Since the forces lie in the internal friction cones and add up to a zero resultant force, they positively span $\mathbb{R}^3$ and satisfy (P2).

The condition is also sufficient. Let us assume the existence of lines Δ_i $(i = 1, .., 4)$ satisfying (P1) and (P2). Since these lines satisfy (P1), some linear combination of the associated wrenches is zero. In particular, if we denote by $\mathbf{x}_i$ $(i = 1, .., 4)$ the contact points and by $\mathbf{u}_i$ $(i = 1, .., 4)$ the vectors satisfying (P2), there exist coefficients ξ_i such that $\sum_{i=1}^{4} \xi_i \mathbf{u}_i = 0$. Because the vectors $\mathbf{u}_i$ positively span $\mathbb{R}^3$, no three of them can be coplanar, and this implies that none of the coefficients ξ_i is zero. We can therefore set $\xi_1 = 1$ without loss of generality. Since the vectors $\mathbf{u}_i$ positively span $\mathbb{R}^3$, we can also write $\mathbf{u}_1 = -\sum_{i=2}^{4} \xi'_i \mathbf{u}_i$, where the coefficients ξ'_i are strictly positive. In turn, this implies that $\sum_{i=2}^{4} (\xi_i - \xi'_i)\mathbf{u}_i = 0$, hence $\xi_i = \xi'_i$ for $i = 1, .., 4$, so the coefficients ξ_i are strictly positive, and the grasp formed by the four contact points exerting the non-zero forces $\xi_i \mathbf{u}_i$ achieves equilibrium. □

Appendix B: Refined Contour Tracking Algorithm

The contour tracking algorithm proposed in Section 3.2.2 has a time complexity of $O(tn)$, where n is the number of constraints defining the input polytope P, and t is the size of its projection Q. This time complexity can be improved by preprocessing the hyperplanes H_i so as to answer the ray shooting queries in sublinear time.

We use the following recent result by Matoušek and Schwarzkopf [30].

Theorem 2 ([30]) *Let m be a parameter such that $n \leq m \leq n^{\lfloor \frac{d}{2} \rfloor}$ and let ϵ be any positive constant. One can preprocess a polytope defined as the intersection of n half spaces of E in $O(m^{1+\epsilon})$ (deterministic) time and using $O(m^{1+\epsilon})$ space so that, given a query line, the intersection points between the line and the polytope can be computed in time $O\left(\frac{n}{m^{1/\lfloor \frac{d}{2} \rfloor}} (\log n)^{2d+1}\right)$.*

The idea of our algorithm is to start with little memory and do ray shootings until the cumulated time of all the queries performed so far exceeds the preprocessing time and the memory storage. At that point, we can allow more memory. Thus we restart a new preprocessing with a bigger value of m. And repeat that process (each such process will be called a step) until all the faces have been constructed.

Let $T(i)$ be the time spent in step i, m_i be the parameter chosen at that step for the preprocessing and t_i be the number of edges constructed at that step. We will take $t_i = n^{1+\alpha i}$ for some $\alpha > 0$. We have

$$T(i) = O(m_i^{1+\epsilon}) + O\left(\frac{nt_i(\log n)^{2d+1}}{m_i^{1/\lfloor\frac{d}{2}\rfloor}}\right).$$

We take m_i so as to balance the cost of the preprocessing and the cost of the queries performed at step i, i.e.,

$$m_i = \left(nt_i(\log n)^{2d+1}\right)^{\frac{1}{\epsilon+1/\gamma}},$$

where

$$\gamma = \frac{1}{1+\frac{1}{\lfloor\frac{d}{2}\rfloor}}.$$

In order to satisfy the hypotheses of Theorem 2, it is necessary that $m_i \leq n^{\lfloor\frac{d}{2}\rfloor}$. We first suppose that this is the case for all i.

It follows from the expression of $T(i)$ that

$$\begin{aligned} T(i) &= O\left(\left(nt_i(\log n)^{2d+1}\right)^{\gamma\frac{1+\epsilon}{1+\gamma\epsilon}}\right) \\ &= O\left(\left(nt_i(\log n)^{2d+1}\right)^{\gamma(1+\epsilon)}\right). \end{aligned}$$

Using the fact that $t_i = n^{1+\alpha i}$ for some $\alpha > 0$, and summing over the h steps needed to compute all the simplices, we obtain the overall computing time T of the algorithm:

$$T = O\left(\left(n^2(\log n)^{2d+1}\right)^{\gamma(1+\epsilon)} \sum_{i=1}^{h} n^{\alpha\gamma(1+\epsilon)i}\right).$$

The number of steps h is given by

$$\sum_{i=1}^{h} t_i = t = n^{1+\alpha}\frac{n^{h\alpha}-1}{n^{\alpha}-1}, \tag{7}$$

which implies $n^{h\alpha} = O(t/n)$, and since $t = O(n^{\lfloor\frac{d}{2}\rfloor})$, $h = O(1)$.

It follows that

$$T = O\left(n^{\gamma(1+\epsilon)}t^{\gamma(1+\epsilon)}(\log n)^{2d+1}\right).$$

The algorithm above runs as described provided that all m_i, $i = 1, \ldots, h$ remain smaller than $n^{\lfloor\frac{d}{2}\rfloor}$, that is

$$\begin{aligned} m_h &\leq n^{\lfloor\frac{d}{2}\rfloor}, \\ \left(nt_h(\log n)^{2d+1}\right)^{\frac{1}{\epsilon+1/\gamma}} &\leq n^{\lfloor\frac{d}{2}\rfloor}, \\ nt_h &\leq n^{\lfloor\frac{d}{2}\rfloor(\epsilon+1+\frac{1}{\lfloor\frac{d}{2}\rfloor})}, \\ t_h &\leq n^{\lfloor\frac{d}{2}\rfloor(1+\epsilon)}. \end{aligned}$$

Furthermore it follows from (7) that $t \leq \frac{t_h}{1-n^{-\alpha}} \leq 2t_h$ for big enough values of n.

If, for some i, $t_i > n^{\lfloor \frac{d}{2} \rfloor (1+\epsilon)}$, we use any worst-case optimal algorithm [3, 5].

Using the fact that for any large enough values of n, and for any positive numbers k, l, we have $(\log n)^k < n^l$, we finally obtain our main result.

Proposition 4 *The projection of a polytope from a Euclidean space E of dimension d onto a $(d-k)$-dimensional sub-space F can be computed in time and space*

$$T = O\left(n^{\gamma+\varepsilon} t^{\gamma+\varepsilon}\right)$$

where ε is any positive constant, t is the size of Q, and

$$\gamma = \frac{1}{1 + \frac{1}{\lfloor \frac{d}{2} \rfloor}}.$$

Acknowledgment: We wish to thank Dan Halperin, Seth Hutchinson, Catherine and Jean-Louis Lassez, Alison Noble, Elon Rimon, Rajeev Sharma and Ilan Shimshoni for useful discussions and comments. Part of this work was conducted while J. Ponce was visiting INRIA and later Caltech as a Beckman associate with the Center for Advanced Study of UIUC. We gratefully acknowledge support from these institutions and from the Beckman Institute for Advanced Science and Technology of UIUC. S. Sullivan was supported in part by NASA grant NAG 1-613. A. Sudsang was supported in part by a fellowship from the Ananda Mahidol Foundation.

References

[1] *Journal of Symbolic Computation*, 5, 1988. Special issue on Algorithms in Real Algebraic Geometry.

[2] R.S. Ball. *A treatise on the theory of screws.* Cambridge University Press, 1900.

[3] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[4] O. Bottema and B. Roth. *Theoretical Kinematics.* North Holland Publishing Co., 1979.

[5] B. Chazelle. An optimal convex hull algorithm for point sets in any fixed dimension. Technical Report CS-TR-336-91, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, 1991.

[6] I.M. Chen and J.W. Burdick. Finding antipodal point grasps on irregularly shaped objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 2278–2283, Nice, France, June 1992.

[7] J. Czyzowicz, I. Stojmenovic, and J. Urrutia. Immobilizing a polytope. volume 519 of *Lecture Notes in Computer Sciences*, pages 214–227. Springer-Verlag, 1991.

[8] A. Dandurand. The rigidity of compound spatial grid. *Structural Topology*, 10, 1984.

[9] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9:66–104, 1990.

[10] B. Faverjon and J. Ponce. On computing two-finger force-closure grasps of curved 2D objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 424–429, Sacramento, CA, April 1991.

[11] C. Ferrari and J.F. Canny. Planning optimal grasps. In *IEEE Int. Conf. on Robotics and Automation*, pages 2290–2295, Nice, France, June 1992.

[12] J.B.J. Fourier. Reported in: Analyse des travaux de l'académie royale des sciences pendant l'année 1824. In *Partie mathématique, Histoire de l'Académie Royale des Sciences de l'Institut de France*, volume 7. 1827. Partial English translation in: D.A. Kohler, Translation of a Report by Fourier on his work on Linear Inequalities, *Opsearch* 10 (1973) 38-42.

[13] D. Hilbert and S. Cohn-Vossen. *Geometry and the Imagination.* Chelsea, New York, 1952.

[14] K.H. Hunt. *Kinematic Geometry of Mechanisms.* Clarendon, Oxford, 1978.

[15] T. Huynh, L. Joskowicz, C. Lassez, and J-L. Lassez. Reasoning about linear constraints using parametric queries. In *Foundations of Software Technology and Theoretical Computer Science*, volume 472 of *Lecture Notes in Computer Sciences.* Springer-Verlag, 1990.

[16] T. Huynh, C. Lassez, and J-L. Lassez. Practical issues on the projection of polyhedral sets. *Annals of Mathematics and Artificial Intelligence*, 6, 1992.

[17] S.C. Jacobsen, J.E. Wood, D.F. Knutti, and K.B. Biggers. The Utah-MIT Dextrous Hand: Work in progress. *International Journal of Robotics Research*, 3(4):21–50, 1984.

[18] Z. Ji. *Dexterous hands: optimizing grasp by design and planning.* PhD thesis, Stanford University, Dept. of Mechanical Engineering, 1987.

[19] Z. Ji and B. Roth. Contact force in grasping and kinematic constraints. In *Seventh IFToMM World Congress*, pages 1219–1222, Sevilla, Spain, 1987.

[20] Z. Ji and B. Roth. Direct computation of grasping force for three-finger tip-prehension grasps. *Journal of Mechanics, Transmissions, and Automation in Design*, 110:405–413, December 1988.

[21] J.R Kerr. *An Analysis of multi-fingered hands.* PhD thesis, Stanford University, Stanford, CA, 1984.

[22] J.R. Kerr and B. Roth. Analysis of multi-fingered hands. *International Journal of Robotics Research*, 4(4), Winter 1986.

[23] D.G. Kirkpatrick, B. Mishra, and C.K. Yap. Quantitative Steinitz's theorems with applications to multifingered grasping. In 20^{th} *ACM Symp. on Theory of Computing*, pages 341–351, Baltimore, MD, May 1990.

[24] K. Lakshminarayana. Mechanics of form closure. Technical Report 78-DET-32, ASME, 1978.

[25] C. Lassez and J-L. Lassez. Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm. In D. Kapur B. Donald and J. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence*, pages 103–122. Academic Press, 1992.

[26] J-L. Lassez. Querying constraints. In *ACM conference on Principles of Database Systems*, Nashville, 1990.

[27] Z. Li and S. Sastry. Task-oriented optimal grasping by multifingered robot hands. In *IEEE Int. Conf. on Robotics and Automation*, pages 389–394, 1987.

[28] X. Markenscoff, L. Ni, and C.H. Papadimitriou. The geometry of grasping. *International Journal of Robotics Research*, 9(1):61–74, February 1990.

[29] X. Markenscoff and C.H. Papadimitriou. Optimum grip of a polygon. *International Journal of Robotics Research*, 8(2):17–29, April 1989.

[30] J. Matoušek and O. Schwarzkopf. Linear optimization queries. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 16–25, 1992.

[31] J.M. McCarthy. *An Introduction to Theoretical Kinematics.* MIT Press, 1990.

[32] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. Assoc. Comput. Mach.*, 31:114–127, 1984.

[33] J.P. Merlet. Singular configurations of parallel manipulators and grassmann geometry. In J.D. Boissonnat and J.P. Laumont, editors, *Geometry and Robotics*, volume 391 of *Lecture Notes in Computer Science*, pages 194–212. Springer-Verlag, 1988.

[34] B. Mirtich and J.F. Canny. Optimum force-closure grasps. Technical Report ESRC 93-11/RAMP 93-5, Robotics, Automation, and Manufacturing Program, University of California at Berkeley, July 1993.

[35] B. Mishra, J.T. Schwartz, and M. Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica, Special Issue: Robotics*, 2(4):541–558, November 1987.

[36] B. Mishra and N. Silver. Some discussion of static gripping and its stability. *IEEE Systems, Man, and Cybernetics*, 19(4):783–796, 1989.

[37] A.F. Möbius. *Lehrbuch der Statik.* Leipzig, 1837.

[38] R.M. Murray, Z. Li, and S.S. Sastry. *A mathematical introduction to robotic manipulation.* CRC Press, 1994.

[39] V-D. Nguyen. Constructing force-closure grasps. *International Journal of Robotics Research*, 7(3):3–16, June 1988.

[40] M.S. Ohwovoriole. *An extension of screw theory and its application to the automation of industrial assemblies.* PhD thesis, Stanford University, Stanford, CA, 1980.

[41] M.S. Ohwovoriole. An extension of screw theory. *Journal of Mechanical Design*, 103:725–735, 1981.

[42] J. Pertin-Troccaz. Grasping: a state of the art. In O. Khatib, J. Craig, and T. Lozano-Pérez, editors, *The Robotics Review 1.* MIT Press, 1989.

[43] N.S. Pollard and T. Lozano-Pérez. Grasp stability and feasibility for an arm with an articulated hand. In *IEEE Int. Conf. on Robotics and Automation*, pages 1581–1585, Cincinatti, OH, 1990.

[44] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. In *International Conference on Advanced Robotics*, pages 1018–1023, Pisa, Italy, June 1991.

[45] J. Ponce and B. Faverjon. On computing three-finger force-closure grasps of polygonal objects. *IEEE Transactions on Robotics and Automation*, 1995. In press.

[46] J. Ponce, D. Stam, and B. Faverjon. On computing force-closure grasps of curved two-dimensional objects. *International Journal of Robotics Research*, 12(3):263–273, June 1993.

[47] J. Ponce, S. Sullivan, J-D. Boissonnat, and J-P. Merlet. On characterizing and computing three- and four-finger force-closure grasps of polyhedral objects. In *IEEE Int. Conf. on Robotics and Automation*, pages 821–827, Atlanta, Georgia, May 1993.

[48] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[49] F. Reulaux. *The kinematics of machinery.* MacMillan, NY, 1876. Reprint, Dover, NY, 1963.

[50] E. Rimon and J. W. Burdick. Mobility of bodies in contact—i: A new 2^{nd} order mobility index for multiple-finger grasps. In *IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, May 1994.

[51] E. Rimon and J. W. Burdick. Mobility of bodies in contact—ii: How forces are generated by curvature effects. In *IEEE Int. Conf. on Robotics and Automation*, San Diego, CA, May 1994.

[52] B. Roth. Screws, motors, and wrenches that cannot be bought in a hardware store. In *Proc. International Symposium on Robotics Research*, pages 679–693. MIT Press, 1984.

[53] J.K. Salisbury. *Kinematic and force analysis of articulated hands.* PhD thesis, Stanford University, Stanford, CA, 1982.

[54] J.K. Salisbury and B. Roth. Kinematic and force analysis of articulated hands. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 105:33–41, 1982.

[55] P. Somov. Über Gebiete von Schraubengeschwindigkeiten eines starren Korpers bie verschiedener Zahl von Stutzflachen. *Zeitricht für Mathematik und Physik*, 45:245–306, 1900.

[56] A. Sudsang. On computing multi-finger force-closure and optimal grasps of two- and three-dimensional objects. Master's thesis, Department of Computer Science, University of Illinois, 1994.

[57] J.C. Trinkle. On the stability and instantaneous velocity of grasped frictionless objects. *IEEE Transactions on Robotics and Automation*, 8(5):560–572, October 1992.