Testing Static Equilibrium for Legged Robots

Timothy Bretl, *Member, IEEE*, and Sanjay Lall, *Member, IEEE*

***Abstract*—Consider a legged robot at fixed foot placements. Where can the robot move its center of mass (CM) while remaining in static equilibrium? If the terrain is flat, the CM must lie above the convex hull of the robot's feet. If the terrain is not flat, this often-used approximation can be arbitrarily bad. Instead, the CM must lie above the projection of a nonlinear convex set that is defined by the properties of each foot placement. This paper presents an algorithm to compute the shape of this projection and gives a tight bound on the algorithm's running time. It also presents a method of amortizing the cost of this computation when it is only necessary to test static equilibrium at particular CM positions—that is, when it is only necessary to test the membership of points in the projection of a convex set rather than find its shape.**

***Index Terms*—Computational geometry, frictional contact, legged locomotion, mobile robots, motion planning, static equilibrium.**

I. Introduction

CONSIDER a legged robot at fixed foot placements on rigid terrain. For the robot to achieve *static equilibrium*, it must be able to apply contact forces with its feet on the ground that both compensate for the action of gravity at its center of mass (CM) and avoid causing slip. We would like to know where the robot can move while satisfying this constraint. So, in this paper, we address the following problem: *given fixed foot placements with associated friction models, find the set of CM positions at which contact forces exist that compensate for gravity without causing slip*. As we will see in Section II, this set is always a right cylinder with a noncircular convex cross section. We call this cross section the *support region*. Informally, "compensate for gravity" means that the contact forces and CM position must satisfy linear force and moment balance equations, and "no slip" means that each contact force is restricted to a second-order friction cone. These constraints define a convex but nonlinear set of jointly feasible contact forces and CM positions. The support region is the projection of this set onto the horizontal position of the CM (in a plane perpendicular to gravity). So, to find the support region, we will consider the following abstraction of our physical problem: *compute the projection of a nonlinear convex set onto a 2-D linear subspace*.

We will also discuss how to amortize the cost of this computation when it is only necessary to test static equilibrium at particular CM positions—that is, when it is only necessary to test the membership of points in the projection rather than find its entire shape.

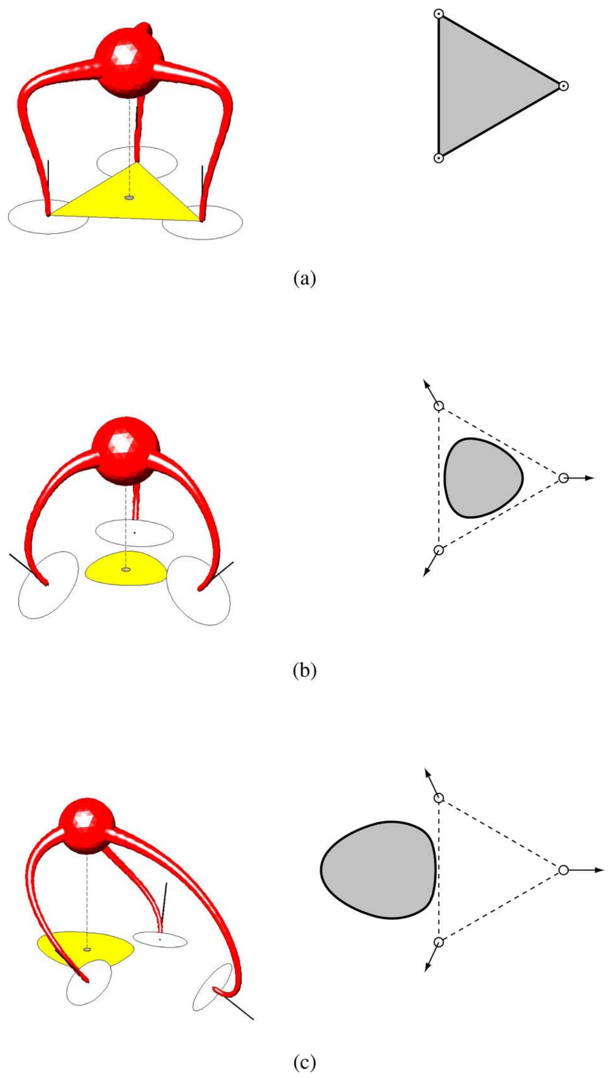


Fig. 1. Robot at three fixed foot placements. Depending on the properties of each foot placement, the support region can be (a) the same, (b) inside, or (c) outside the base of the robot's supports.

A. Why the Support Polygon Is Insufficient

In some cases, computing the support region is easy. For example, on flat terrain with frictional point contacts [see Fig. 1(a)], the support region, in this case, a polygon, is the convex hull of the robot's supporting feet [1]. The right prism having this polygon as cross section is the set of all CM positions at which static equilibrium is possible.

In fact, the terrain does not have to be entirely flat for the "support polygon" to be a good approximation of the support

Manuscript received November 13, 2007; revised April 19, 2008. This paper was recommended for publication by Associate Editor H. R. Choi and Editor H. Arai upon evaluation of the reviewers' comments.

T. Bretl is with the University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: tbretl@illinois.edu).

S. Lall is with Stanford University, Stanford, CA 94305 USA (e-mail: lall@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2001360

region. The two are equivalent if every foot can exert a force parallel to the gravity vector without slipping. With frictionless point contacts, the normal vector at every contact must be vertical (opposite to gravity), but not all contacts need be in the same horizontal plane. With frictional point contacts, every friction cone must contain a vector that is vertical, but the center of each cone (the normal vector at each contact) need not be vertical.

In general, however, the support region is not a polygon [see Fig. 1(b)], and, in fact, may lie completely outside the base of the supports [see Fig. 1(c)]. In particular, notice in Fig. 1 that the locations of each foot placement remain the same. By changing only the direction of each surface normal, we have completely changed the support region. Consequently, the support polygon is a bad approximation of the support region on irregular and steep terrain.

B. Application to Legged Locomotion

The support polygon has been a key concept in legged locomotion, its use vastly documented in the literature. Applied to planning, it can be used to test the feasibility of postures (as in [2]–[4]). Applied to control, it is the basis of the zero moment point (ZMP) as defined in [5] and can be used to construct both static and dynamic stability margins (as in [6] and [7]). And, of course, it has been applied successfully to many fielded legged robots (as in [4], [8], and [9]).

All of this success has come despite the problem described earlier, as well as several other limitations.

1) It only gives necessary conditions for static equilibrium. Additional assumptions must be made (controllability, torque limits) before these conditions are sufficient [10].
2) It tests static equilibrium, but legged robots move dynamically. These robots may not be running or jumping—following a statically stable geometric path at too high a velocity can also cause instability [11].
3) It describes the set of possible CM positions, but usually it is the set of possible robot postures that we care about. Further, the relationship between the CM position and the posture is both nonlinear and nonconvex [12].
4) It tests static equilibrium but not static stability. We would like the robot to be robust to disturbances [6].
5) It indicates when contact forces exist that compensate for gravity, but it does not indicate how to choose these contact forces [13], [14].

Being a generalization of the support polygon to arbitrary foot placements, the support region is subject to the same limitations. *Addressing these limitations is beyond the scope of this paper*. Instead, we accept the same breadth of applicability. In the same ways that the support polygon is useful for legged robots on terrain that is mostly flat, the support region is useful for legged robots on terrain that is irregular and steep.

Recent work considers robots on exactly this type of terrain. For example, robotic rock climbers can traverse featured inclined walls using techniques similar to human climbers [15], and humanoids can use hands to help balance while leaning against a wall or walking up a steep cliff [16]. In these applications, the support region, not the support polygon approximation, plays a key role in both planning and control.

C. Related Work in Legged Locomotion

Past research on computing the support region in 3-D treats the problem, like we do, as one of projecting a high-dimensional convex set onto a 2-D linear subspace. In the work of [17], contacts are assumed frictionless. This assumption reduces the number of constraints on contact forces and CM position and makes these constraints linear. The resulting linear projection problem is solved using computational geometry [18]. In the work of [15] and [19], friction cones are approximated by polyhedra; so, the nonlinear constraints on contact forces are approximated by linear ones. The resulting projection problem is harder than if contacts are assumed frictionless, but is still linear, so is solved in a similar way. In the work of [20], the support region is found exactly using an algebraic approach. This approach, so far, applies only to legged robots making exactly three contacts with the terrain, and is too slow to be used in the context of motion planning and control. However, it does provide insight into the structure of both the equilibrium constraints and the support region. Conceptually, the algorithms we present in this paper are based on our earlier work [19]. The key difference here is that we show provably bounded error, provide a tight bound on running time, and do not require friction cones to be approximated by polyhedra.

D. Related Work in Dexterous Manipulation

Legged locomotion is similar to dexterous manipulation. Just as a legged robot places and removes feet on the ground, a robotic hand places and removes fingers on an object. Of course, for manipulation, the "terrain" (the surface of the object) is rarely flat and horizontal. So, methods for grasp analysis deal with the same nonlinear constraints on contact forces and CM position that arise for legged robots on irregular and steep terrain (see [21] and [22] and the associated references).

However, whereas the support region is a useful concept for legged locomotion, it is not useful for dexterous manipulation. Recall that the support region defines the range of feasible CM positions for a fixed set of contact points. Consequently, it is primarily useful to compute the support region when the CM position can be changed without changing the contact points. During manipulation, it is the object (not the hand) that must be in static equilibrium. If the object is modeled as a rigid body, then it has no unconstrained degrees of freedom when it is grasped. So, for the CM of the object to move, the contact points must move (either locally on the object or globally in space), changing the support region.

Although it is not useful to compute the support region for manipulation, similar types of computations are performed. For example, the work of [23] addresses the problem of finding concurrent grasps of a polyhedral object in 3-D with four point contacts. In a concurrent grasp, the lines of action of all contact forces intersect at a point. It is simpler to find concurrent grasps if the position of this point is eliminated from the equilibrium constraints. By approximating friction cones as polyhedra, as in related work on legged locomotion, this elimination becomes a problem of linear projection (in this case from an 11-D space to an 8-D subspace), for which a new algorithm is proposed [23]. This algorithm does not extend to projection of nonlinear sets. But like ours, it is output sensitive—its running time depends primarily on the complexity of the projection, not of the set to be projected.

Other research in manipulation has motivated our use of nonlinear friction models rather than polyhedral friction cone approximations. In particular, the work of [24] cites four disadvantages of a linear approximation. First, the approximation may be overly conservative. Second, the approximation is nonisotropic, a usual assumption in the Coulomb friction model. Third, the approximation magnifies the affects of small perturbations in model parameters. And fourth, although in theory the approximation can be made arbitrarily precise, in practice, the precision is limited by running time. In [24], problems related to grasp analysis (force closure, feasibility, and optimization) are solved—retaining a nonlinear model—by casting them as convex optimization problems involving linear matrix inequalities. In this paper, we solve a sequence of convex optimization problems to find the support region.

E. Related Work in Computational Geometry and Optimization

Finding the exact projection $\mathcal{Y}$ of a set $\mathcal{X}$ bounded by nonlinear constraints can be done using quantifier elimination. However, existing algorithms, such as cylindrical algebraic decomposition, have complexity at best exponential in problem parameters (such as the number of variables and number of constraints defining $\mathcal{X}$) [25], [26]. Furthermore, there is no guarantee that $\mathcal{X}$ and $\mathcal{Y}$ can be represented in the same way. For example, the projection of a set bounded by second-order cone constraints cannot, in general, be represented using second-order cone constraints (the same is true for linear matrix inequalities). Similarly, although the projection of a semialgebraic set is always semialgebraic, polynomials of arbitrary degree are necessary to represent it.

Finding the exact projection $\mathcal{Y}$ of a set $\mathcal{X}$ bounded by linear constraints is a central problem in computational geometry, and there are many algorithms available (see [27] for sample code). Some algorithms are based on quantifier elimination, specialized for linear constraints (see [28] for a survey). These algorithms construct $\mathcal{Y}$ directly by computing valid inequalities (positive combinations of the linear inequalities defining $\mathcal{X}$), each of which eliminates a variable. Other algorithms are based on methods for finding convex hulls. These algorithms construct $\mathcal{Y}$ indirectly by first enumerating all vertices of $\mathcal{X}$, projecting these trivially onto a low-dimensional subspace (for example, by dropping coordinates), then enumerating all facets of the result [29]–[32]. In fact, due to a correspondence between linear inequalities and vertices (one can be represented as the other in a dual space), algorithms for elimination and enumeration are mathematically equivalent. But, there is still an inherent computational complexity associated with linear projection, since the number of inequalities defining $\mathcal{Y}$ is, in worst case, exponential in the dimensionality of $\mathcal{X}$. Numerical stability is also an issue, and it is difficult to know when linear projection will be easy or hard.

One way to approximate the projection $\mathcal{Y}$ of a set $\mathcal{X}$ bounded by nonlinear constraints is first to find a linear approximation $\mathcal{X}_{\mathrm{linear}}$ of $\mathcal{X}$, and then, to find the exact linear projection $\mathcal{Y}_{\mathrm{linear}}$ of $\mathcal{X}_{\mathrm{linear}}$. For example, friction cones can be approximated by inscribed polyhedra in problems related to static equilibrium [19], [23]. More generally, there are systematic ways of constructing $\mathcal{X}_{\mathrm{linear}}$ so that $\mathcal{Y}_{\mathrm{linear}}$ is a good approximation of $\mathcal{Y}$ [33], and the results of doing so have been analyzed in the context of convex optimization [34]. These studies suggest that linear approximation of $\mathcal{X}$ is effective for problems with many variables and constraints (hundreds of thousands in the case of optimization), which, in practice, can be handled by linear solvers but not by nonlinear ones.

An alternative approach is suggested by *cutting plane algorithms*, which can be used to solve convex feasibility problems. These algorithms maintain a polyhedral approximation to the set of feasible points. At every iteration, a query point is picked inside this polyhedron. Either this query point is feasible, or a hyperplane—a cutting plane—can be found that separates it from the set of feasible points. This hyperplane cuts the polyhedron, making it smaller (more accurate) for the next iteration. In this paper, we use a cutting plane algorithm to construct a polyhedral approximation $\mathcal{Y}_{\mathrm{linear}}$ of the projection $\mathcal{Y}$ directly from the nonlinear set $\mathcal{X}$. Here, we are not interested in finding one feasible point (one feasible CM position), but rather in finding the entire feasible set (the entire support region). As for all cutting plane algorithms, the main problem is choosing a good sequence of query points. Our approach is similar to [35].

F. Our Contributions

This paper presents an algorithm to compute the projection of a nonlinear convex set onto a 2-D linear subspace. Our algorithm maintains inner and outer polygonal approximations to the projection. It iteratively computes points on the boundary of the projection—hence, supporting hyperplanes—by solving a sequence of convex programs. *The key advantage of our algorithm is that it has provably bounded error*. Hence, error can be made arbitrarily small by using a stopping criterion based on the error bound. Our algorithm can be applied to compute the support region to any desired precision for a legged robot making any number of contacts with any type of terrain. It requires only that the frictional contact model involve convex constraints.

We also extend our algorithm to test the membership of points in the projection of a nonlinear convex set without computing the entire shape of the projection. Here, we still maintain inner and outer polygonal approximations to the projection, but iterate only until a query point lies either inside the inner approximation (proving feasibility) or outside the outer approximation (proving infeasibility). We prove a bound on the number of iterations required to achieve a desired precision for each query point. We also show that for some sets of query points, our approach is faster than either testing each query point individually or precomputing the projection. This extension can be applied to test static equilibrium at particular CM positions for a legged

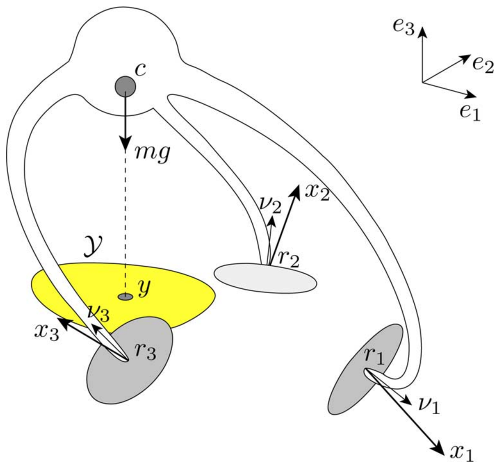


Fig. 2. Three-legged robot in static equilibrium at frictional point contacts.

robot with fixed foot placements. Again, it applies to any number of contacts and any type of terrain, requiring only a frictional contact model involving convex constraints.

II. Mathematical Model of Static Equilibrium

Consider a legged robot walking on uneven terrain in a 3-D workspace (see Fig. 2). Suppose the robot is currently making contact with the terrain at n frictional points $r_1, \ldots, r_n \in \mathbb{R}^3$, each with some outward (unit) normal vector $\nu_i \in \mathbb{R}^3$ and static coefficient of friction μ_i. Let $x_i \in \mathbb{R}^3$ be the reaction force acting on the robot at each point r_i for $i = 1, \ldots, n$. This force can be decomposed into a component $\nu_i^T x_i \nu_i$ normal to the surface of the terrain (in the direction ν_i) and an orthogonal component $(I - \nu_i \nu_i^T)x_i$ tangential to the surface. Let $c \in \mathbb{R}^3$ be the position of the robot's CM, which varies as the robot moves. Assume the robot has mass m and the acceleration due to gravity is $g \in \mathbb{R}^3$. All of these vectors are defined with respect to a global coordinate system with axes $e_1, e_2, e_3 \in \mathbb{R}^3$, where $g = -\|g\|e_3$. Then, we say that the robot is in *static equilibrium* if

$$\sum_{i=1}^{n} x_i + mg = 0 \tag{1}$$

$$\sum_{i=1}^{n} r_i \times x_i + c \times mg = 0 \tag{2}$$

$$\|(I - \nu_i \nu_i^T)x_i\| \leq \mu_i \nu_i^T x_i \qquad \text{for all } i = 1, \ldots, n. \tag{3}$$

These are convex constraints on the reaction forces $x_1, \ldots, x_n$ and the CM position c. Constraints (1) and (2) ensure force and torque balance. Constraint (3) restricts each reaction force to lie in a friction cone of half angle $\phi_i = \tan^{-1} \mu_i$, according to a Coulomb model. Notice that these constraints do not depend on the robot's entire posture. Rather, they depend only on the robot's CM position c and the properties of each foot placement. In fact, denoting the coordinates of c by $c_1, c_2, c_3 \in \mathbb{R}$, we see that

$$c \times mg = m\|g\| \begin{bmatrix} -c_2 \\ c_1 \\ 0 \end{bmatrix}$$

so constraints (1) and (2) depend only on the horizontal position of the CM (in a plane perpendicular to gravity). We define this position as

$$y = Pc \qquad \text{where} \qquad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

For any $z \in \mathbb{R}^3$ with coordinates $z_1, z_2, z_3 \in \mathbb{R}$, we define the skew-symmetric matrix

$$T(z) = \begin{bmatrix} 0 & -z_3 & z_2 \\ z_3 & 0 & -z_1 \\ -z_2 & z_1 & 0 \end{bmatrix}$$

so that $T(z)x = z \times x$ for all $x \in \mathbb{R}^3$. So, (1) and (2) become

$$\sum_{i=1}^{n} x_i + mg = 0 \tag{4}$$

$$\sum_{i=1}^{n} T(r_i)x_i - T(mg)P^T y = 0. \tag{5}$$

We now have the following conditions: $x_1, \ldots, x_n$ and c satisfy (1)–(3) if and only if

$$\begin{aligned} A_1 x + A_2 y &= t \\ \|Bx\| &\leq u^T x \end{aligned} \tag{6}$$

where

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \qquad y = Pc$$

$$A_1 = \begin{bmatrix} I & \ldots & I \\ T(r_1) & \ldots & T(r_N) \end{bmatrix},$$

$$A_2 = \begin{bmatrix} 0 \\ -T(mg)P^T \end{bmatrix}, \qquad t = \begin{bmatrix} -mg \\ 0 \end{bmatrix}$$

$$B = \operatorname{diag}(I - \nu_1\nu_1^T, \ldots, I - \nu_n\nu_n^T), \qquad u = \begin{bmatrix} \mu_1\nu_1 \\ \vdots \\ \mu_n\nu_n \end{bmatrix}.$$

From (6), the set of all feasible reaction forces x and CM positions y at fixed contacts is

$$\mathcal{X} = \{x \in \mathbb{R}^{3n}, y \in \mathbb{R}^2 \mid A_1 x + A_2 y = t, \|Bx\| \leq u^T x\}.$$

Likewise, the *support region* (that is, the set of all feasible CM positions y at fixed contacts) is the projection $\mathcal{Y}$ of $\mathcal{X}$ onto y-space, namely

$$\mathcal{Y} = \{y \in \mathbb{R}^2 \mid \exists x \in \mathbb{R}^{3n} \text{ such that } (x, y) \in \mathcal{X}\}.$$

III. Computing the Support Region

In the previous section, we saw that the support region can be represented as the projection of a nonlinear convex set bounded by linear and second-order cone constraints onto a 2-D linear subspace. In this section, we present an algorithm to compute the shape of this projection.

A. Problem Statement

For some integers $m, n > 0$, given arbitrary $A_1 \in \mathbb{R}^{m\times n}$, $A_2 \in \mathbb{R}^{m\times 2}$, $t \in \mathbb{R}^m$, $B \in \mathbb{R}^{n\times n}$, and $u \in \mathbb{R}^n$, we want to compute the set

$$\mathcal{Y} = \{y \in \mathbb{R}^2 | \exists x \in \mathbb{R}^n \text{ such that } (x, y) \in \mathcal{X}\}$$

where

$$\mathcal{X} = \{x \in \mathbb{R}^n, y \in \mathbb{R}^2 | A_1 x + A_2 y = t, \|Bx\| \leq u^T x\}.$$

The set $\mathcal{X}$ is defined using linear and second-order cone constraints, but, in general, it is not possible to do the same for the projection $\mathcal{Y}$. In fact, although $\mathcal{Y}$ is semialgebraic, representing it exactly requires polynomials of arbitrary degree. So, our goal instead is to approximate $\mathcal{Y}$ by an inscribed polygon

$$\mathcal{Y}_{\text{inner}} = \{y \in \mathbb{R}^2 | a_i^T y \leq b_i \qquad \text{for all } i = 1, \ldots, N\}.$$

Assuming that $\mathcal{Y}$ is bounded and has nonempty interior, then for any $\epsilon > 0$, we will find a_i and b_i for $i = 1, \ldots, N$ such that

$$\mathcal{Y}_{\text{inner}} \subseteq \mathcal{Y} \qquad \text{and} \qquad \text{area}(\mathcal{Y}) - \text{area}(\mathcal{Y}_{\text{inner}}) \leq \epsilon.$$

The parameter ϵ specifies a desired bound on the error of our approximation. By decreasing ϵ, we can approximate $\mathcal{Y}$ to any required precision.

B. Algorithm

We compute $\mathcal{Y}_{\text{inner}}$ using the algorithm Project (see Figs. 3 and 4). This algorithm generates a sequence of points on the boundary of $\mathcal{Y}$ and takes $\mathcal{Y}_{\text{inner}}$ as the convex hull of these points. Each point $v \in \mathbb{R}^2$, extremal in some direction $a \in \mathbb{R}^2$, is found by solving a second-order cone program (SOCP) [36] of the form

$$\begin{aligned} \text{maximize} \quad & a^T z \\ \text{subject to} \quad & A_1 x + A_2 z = t \\ & \|Bx\| \leq u^T x \end{aligned} \tag{7}$$

and taking $v = z_{\text{opt}}$. Every point v also defines a halfspace

$$\{z \in \mathbb{R}^2 | a^T z \leq a^T v\} \supseteq \mathcal{Y}.$$

So, in addition to $\mathcal{Y}_{\text{inner}}$, our algorithm constructs an outer polygonal approximation $\mathcal{Y}_{\text{outer}} \supseteq \mathcal{Y}$ as the intersection of these halfspaces.

The algorithm Project is dual to a cutting plane algorithm. As for all cutting plane algorithms, the main problem is choosing a good sequence of query points; in this case, choosing a good sequence of query directions a in the objective function of each SOCP (7).

We use the difference between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ to guide our choice of a. In particular, we know that any point in $\mathcal{Y}_{\text{inner}}$ is

```
PROJECT($\mathcal{X}, \mathcal{Y}_{\text{inner}}, \mathcal{Y}_{\text{outer}}$)
Given bounded polygons $\mathcal{Y}_{\text{outer}} \supseteq \mathcal{Y}$ and $\mathcal{Y}_{\text{inner}} \subseteq \mathcal{Y}$ with nonempty interior such that every vertex of $\mathcal{Y}_{\text{inner}}$ lies on the boundary of $\mathcal{Y}_{\text{outer}}$.
Repeat while area($\mathcal{Y}_{\text{outer}}$) − area($\mathcal{Y}_{\text{inner}}$) $> \epsilon$:
    • Compute the edges of $\mathcal{Y}_{\text{inner}}$ and denote them
          $\{ z \in \mathcal{Y}_{\text{inner}} \mid a_j^T z = b_j \}$ for all $j = 1, \ldots, N$.
    • Pick the edge cutting off the greatest fraction of $\mathcal{Y}_{\text{outer}}$.
          $i = \arg\max_{j \in \{1,\ldots,N\}} \text{area}(\{ z \in \mathcal{Y}_{\text{outer}} \mid a_j^T z \geq b_j \})$
    • Find the point in $\mathcal{Y}$ farthest outside this edge.
          $v = \arg\max_{z \in \mathcal{Y}} a_i^T z$
      Do this by solving the second-order cone program
          maximize    $a_i^T z$
          subject to  $A_1 x + A_2 z = t$
                      $\|Bx\| \leq u^T x$.
    • Update the outer approximation.
          $\mathcal{Y}_{\text{outer}} = \{ z \in \mathcal{Y}_{\text{outer}} \mid a_i^T z \leq a_i^T v \}$
    • Update the inner approximation.
          $\mathcal{Y}_{\text{inner}} = \text{conv}(\mathcal{Y}_{\text{inner}} \cup \{v\})$
Return $\mathcal{Y}_{\text{inner}}$.
```

Fig. 3. Algorithm to compute the support region.

also in $\mathcal{Y}$ and that any point not in $\mathcal{Y}_{\text{outer}}$ is also not in $\mathcal{Y}$, so our goal is to reduce the set of points

$$\mathcal{Y}_{\text{outer}} - \mathcal{Y}_{\text{inner}} = \{z \in \mathbb{R}^2 | z \notin \mathcal{Y}_{\text{inner}}, z \in \mathcal{Y}_{\text{outer}}\}$$

remaining to be classified. If $\mathcal{Y}_{\text{inner}}$ contains N edges, then this set consists of at most N triangles, each outside and adjacent to exactly one edge a_j of $\mathcal{Y}_{\text{inner}}$ for $j = 1, \ldots, N$. We choose the direction a_i normal to the edge inside the triangle with largest area. So, we choose the direction expected to most reduce the area of the set of points remaining to be classified.

We also use the difference between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ to bound the difference between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}$, and consequently, to bound the approximation error. Since

$$\text{area}(\mathcal{Y}) - \text{area}(\mathcal{Y}_{\text{inner}}) \leq \text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}})$$

then, we iterate until

$$\text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}}) \leq \epsilon.$$

In fact, we can give a tight bound on the number of iterations required to achieve this stopping criterion.

Theorem 1: Suppose $\mathcal{Y}_{\text{inner}}$ begins with η_0 edges and that the initial difference in area between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ is

$$\alpha_0 = \text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}}).$$

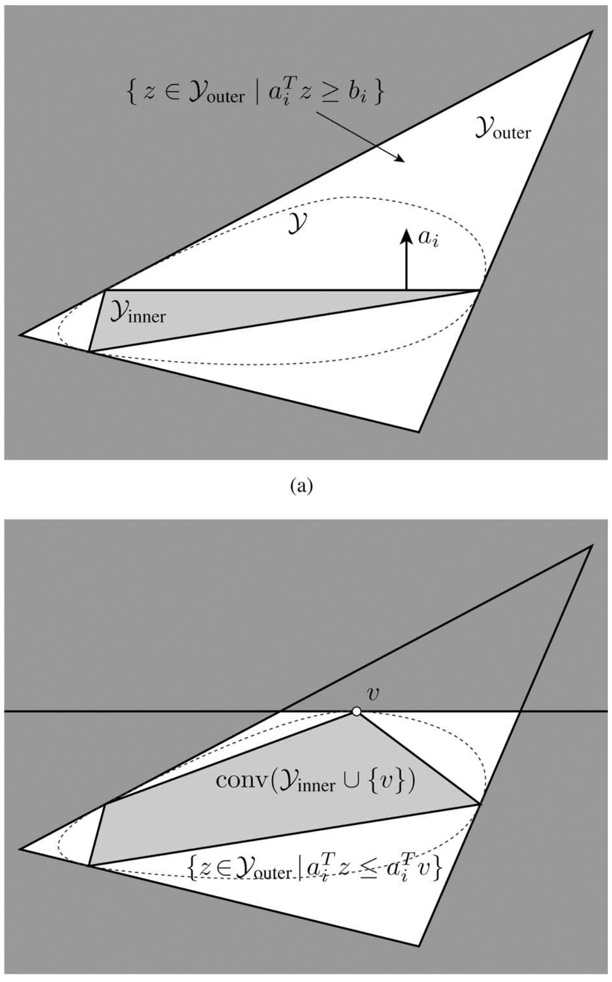

$\{ z \in \mathcal{Y}_{\text{outer}} \mid a_i^T z \geq b_i \}$
$\mathcal{Y}_{\text{outer}}$
$\mathcal{Y}$
a_i
$\mathcal{Y}_{\text{inner}}$
(a)
v
$\text{conv}(\mathcal{Y}_{\text{inner}} \cup \{v\})$
$\{z \in \mathcal{Y}_{\text{outer}} \mid a_i^T z \leq a_i^T v\}$
(b)

Fig. 4. One iteration of the algorithm PROJECT (see Fig. 3).

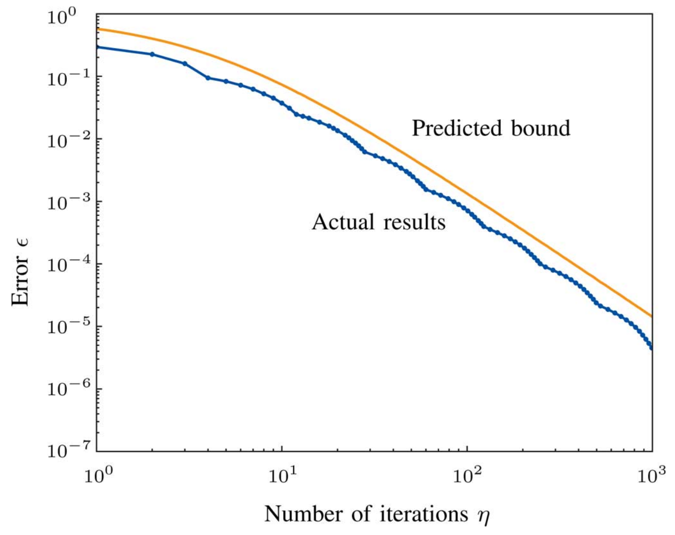


Fig. 5. Number of iterations η required to achieve a given error ϵ with PROJECT for the example shown in Fig. 2.

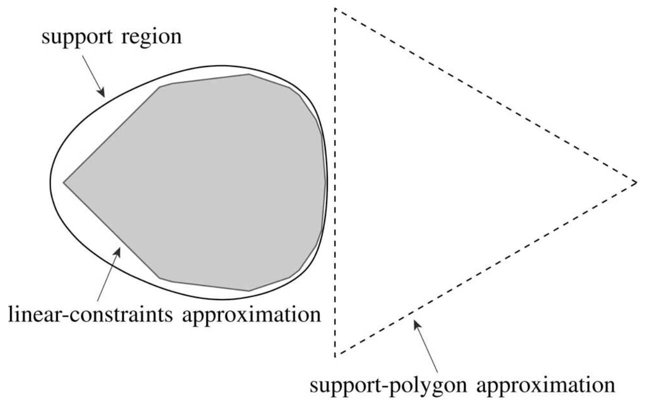

support region
linear-constraints approximation
support-polygon approximation

Fig. 6. Actual support region (computed using PROJECT) as compared to the support-polygon approximation and to the linear-constraints approximation (where friction cones are modeled by six-sided inscribed polyhedra).

Then, for any $\epsilon > 0$, the algorithm PROJECT will terminate after no more than

$$\eta_0 \left(\sqrt{\frac{c\alpha_0}{\epsilon}} - 1 \right)$$

iterations, where $c = 343/243 \approx 1.412$ is a fixed constant.

Proof: See Appendix I. ■

C. Results in Simulation

The algorithm PROJECT computes the projection of a nonlinear convex set onto a 2-D linear subspace. The key advantage of using this algorithm to compute the support region for a legged robot is that Theorem 1 guarantees provably bounded error. As an example, in Fig. 5, we compare the number of iterations η required to achieve a given error bound ϵ for the legged robot shown in Fig. 2. Our results show strong agreement with the bound predicted by Theorem 1.

In contrast, Fig. 6 shows two common approximations to the support region. The first is the support polygon, which is completely incorrect (as described in Section I). The second is obtained by approximating friction cones with six-sided inscribed polyhedra, so that $\mathcal{X}$ is bounded by linear constraints and $\mathcal{Y}$ can be found using exact linear projection. Here, this "linear-constraints" approximation works poorly, resulting in $\epsilon = 0.2$. It is possible to reduce this error by using inscribed polyhedra with more sides, but there is a corresponding increase in computation time and the number of sides required varies from problem to problem.

The actual results in Fig. 5 show an oscillation that is not reflected by the predicted bound. The reason for this oscillation is that, in practice, PROJECT makes batches of cuts to the set $\mathcal{Y}_{\text{outer}} - \mathcal{Y}_{\text{inner}}$ that have similar size. During a batch, the error bound ϵ decreases linearly with the number of iterations η. After a batch is done, the cut size decreases, and so, ϵ decreases linearly at a slower rate. Since a straight line with negative slope and axis crossings $(0, \hat{y})$ and $(\hat{x}, 0)$ on a linear plot is equivalent to a curve with asymptotes at both $x = \hat{x}$ and $y = \hat{y}$ on a log–log plot, then the result of each batch is a curve with asymptotes at progressively lower ϵ and higher η. The oscillation in Fig. 5 is the concatenation of these curves.

In this analysis, we measured the performance of PROJECT by the number of SOCPs it has to solve, since these take up the bulk of the computation time. The rest of the time is spent maintaining vertex and facet representations of both $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ and updating these representations by taking convex hulls, all of which can be done efficiently in 2-D (for details of our software implementation, see our earlier work [19]).

IV. Testing if the Center of Mass is Over the Support Region

In the previous section, we presented an algorithm to compute the projection of a nonlinear convex set onto a 2-D linear subspace. This algorithm can be used to compute the support region for a legged robot. One reason we might be interested in finding the support region is to test static equilibrium quickly at many different CM positions. But, in fact, it is possible to do so without computing the entire support region. In this section, we extend our algorithm to test the membership of points in the projection of a nonlinear convex set without computing its entire shape.

A. Problem Statement

We are given integers $m, n > 0$ and arbitrary $A_1 \in \mathbb{R}^{m\times n}$, $A_2 \in \mathbb{R}^{m\times 2}$, $t \in \mathbb{R}^m$, $B \in \mathbb{R}^{n\times n}$, and $u \in \mathbb{R}^n$ that define

$$\mathcal{X} = \{x \in \mathbb{R}^n, y \in \mathbb{R}^2 \mid A_1x + A_2y = t, \|Bx\| \le u^Tx\}.$$

We want to test the membership of a sequence of query points $y_1, \ldots, y_k \in \mathbb{R}^2$ in the projection

$$\mathcal{Y} = \{y \in \mathbb{R}^2 \mid \exists x \in \mathbb{R}^n \text{ such that } (x, y) \in \mathcal{X}\}.$$

We must answer each query one-by-one, in the order that it is given. The number k of queries is initially unknown. Our goal is to minimize the total expected time required to answer all queries.

B. Algorithm

What makes this problem hard is that we do not know the number of query points k. If k is large, it still makes sense to use the algorithm PROJECT (see Fig. 3 in the previous section) to precompute the ϵ-approximation

$$\mathcal{Y}_{\text{inner}} = \{y \in \mathbb{R}^2 | a_i^T y \le b_i \text{ for all } i = 1, \ldots, N\}$$

where

$$\text{area}(\mathcal{Y}) - \text{area}(\mathcal{Y}_{\text{inner}}) \le \epsilon$$

because then, to test whether each $y_j \in \mathcal{Y}$, we simply verify that $a_i^T y_j \le b_i$ for $i = 1, \ldots, N$. But, if the number of points k is small, and particularly, if the required error bound ϵ is small, then it is wasteful to precompute $\mathcal{Y}_{\text{inner}}$. To do so, we need to solve many SOCPs of the form (7), whereas we only need to solve one SOCP

$$\begin{aligned} &\text{find} && x \\ &\text{subject to} && A_1x = (t - A_2y_j) \\ & && \|Bx\| \le u^Tx \end{aligned} \tag{8}$$

```
TEST-SAMPLE($\mathcal{X}, \mathcal{Y}_{\text{inner}}, \mathcal{Y}_{\text{outer}}, y$)
Given bounded polygons $\mathcal{Y}_{\text{outer}} \supseteq \mathcal{Y}$ and $\mathcal{Y}_{\text{inner}} \subseteq \mathcal{Y}$ with nonempty interior such that every vertex of $\mathcal{Y}_{\text{inner}}$ lies on the boundary of $\mathcal{Y}_{\text{outer}}$.
Repeat while $y \notin \mathcal{Y}_{\text{inner}}$ and $y \in \mathcal{Y}_{\text{outer}}$:
    • Compute the edges of $\mathcal{Y}_{\text{inner}}$ and denote them
          $\{ z \in \mathcal{Y}_{\text{inner}} \mid a_j^T z = b_j \}$ for all $j = 1, \ldots, N$.
    • Pick the (unique) edge separating $\mathcal{Y}_{\text{inner}}$ from $y$.
          $i = \arg\max_{j\in\{1,\ldots,N\}} a_j^T y - b_j$
    • Check the convergence of $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$. If
          $\text{area}(\{ z \in \mathcal{Y}_{\text{outer}} \mid a_i^T z \ge b_i \}) \le \epsilon$
      then declare $y \in \mathcal{Y}_{\text{inner}}$ and exit loop.
    • Otherwise, find the point in $\mathcal{Y}$ farthest outside edge $i$.
          $v = \arg\max_{z\in\mathcal{Y}} a_i^T z$
      Do this by solving the second-order cone program
          maximize     $a_i^T z$
          subject to   $A_1x + A_2z = t$
                       $\|Bx\| \le u^Tx$.
    • Update the outer approximation.
          $\mathcal{Y}_{\text{outer}} = \{ z \in \mathcal{Y}_{\text{outer}} \mid a_i^T z \le a_i^T v \}$
    • Update the inner approximation.
          $\mathcal{Y}_{\text{inner}} = \text{conv}(\mathcal{Y}_{\text{inner}} \cup \{v\})$
If $y \in \mathcal{Y}_{\text{inner}}$ return TRUE, otherwise return FALSE. Also return both $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$.
```

Fig. 7. Algorithm used to test if a CM position is over the support region.

to test if a single point y_j is in $\mathcal{Y}$. Unfortunately, we do not know ahead of time which approach is the best.

To address this problem, we use a method of amortizing computation. In particular, we use the incremental algorithm TEST-SAMPLE (see Figs. 7 and 8) to check if a query point y lies in the projection $\mathcal{Y}$. Our algorithm does not solve a single SOCP to test explicitly if $y \in \mathcal{Y}$. Instead, like PROJECT (see Fig. 3), TEST-SAMPLE constructs inner and outer polygonal approximations $\mathcal{Y}_{\text{inner}} \subseteq \mathcal{Y}$ and $\mathcal{Y}_{\text{outer}} \supseteq \mathcal{Y}$, respectively. But, rather than precompute all of $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$, TEST-SAMPLE computes only enough of them to answer each query. Then, after TEST-SAMPLE terminates, we store $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ to make subsequent evaluations faster. In particular, future queries might be answered without updating $\mathcal{Y}_{\text{inner}}$ or $\mathcal{Y}_{\text{outer}}$ any further.

Each iteration of TEST-SAMPLE solves one SOCP to generate both a point on the boundary of $\mathcal{Y}$ (extremal in some direction a) and a halfspace supporting $\mathcal{Y}$ at this point (normal to a). It constructs $\mathcal{Y}_{\text{inner}}$ as the convex hull of extremal points, and $\mathcal{Y}_{\text{outer}}$ as the intersection of supporting halfspaces. So, as in PROJECT, each iteration shrinks the set of points

$$\{z \in \mathbb{R}^2 \mid z \notin \mathcal{Y}_{\text{inner}}, z \in \mathcal{Y}_{\text{outer}}\}$$

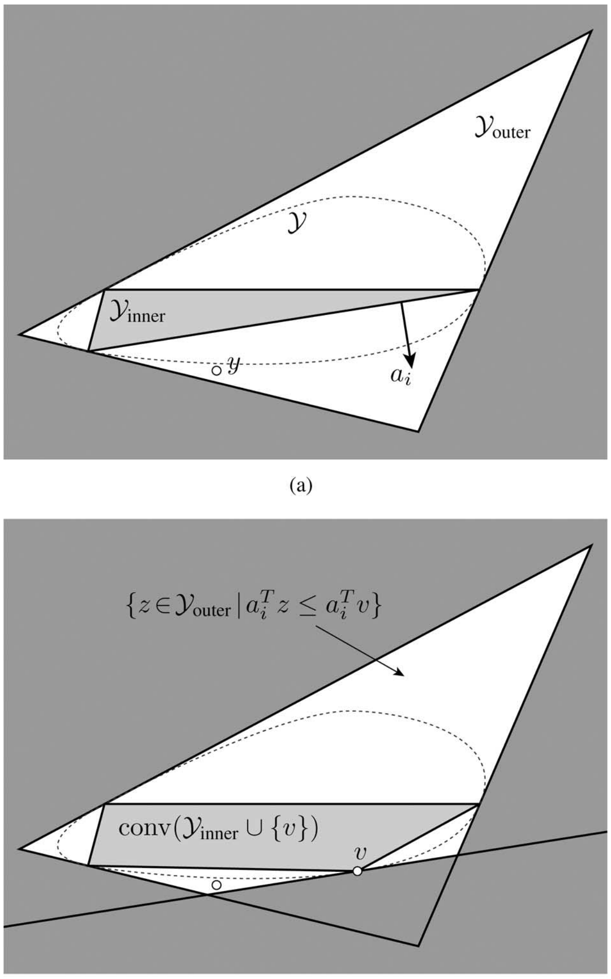

$\mathcal{Y}_{\text{outer}}$
$\mathcal{Y}$
$\mathcal{Y}_{\text{inner}}$
y
a_i
(a)
$\{z \in \mathcal{Y}_{\text{outer}} \,|\, a_i^T z \leq a_i^T v\}$
$\text{conv}(\mathcal{Y}_{\text{inner}} \cup \{v\})$
v
(b)

Fig. 8. One iteration of the algorithm TEST-SAMPLE (see Fig. 7).

remaining to be classified, which consists of at most N triangles if $\mathcal{Y}_{\text{inner}}$ contains N edges. As before, we choose the search direction a_i normal to one of these edges. But, rather than choose the edge adjacent to the largest triangle, we choose the edge adjacent to the triangle that contains y. By construction, this edge is unique—we know that $a_i^T y > b_i$ for some $i \in \{1, \ldots, N\}$ and that $a_j^T y \leq b_j$ for all $j = 1, \ldots, N$ such that $j \neq i$. Hence, each iteration of TEST-SAMPLE localizes y to a smaller region between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$.

Our algorithm loops until either $y \in \mathcal{Y}_{\text{inner}}$ (implying $y \in \mathcal{Y}$) or $y \notin \mathcal{Y}_{\text{outer}}$ (implying $y \notin \mathcal{Y}$). To deal with points lying exactly on the boundary of $\mathcal{Y}$, our algorithm also terminates (declaring $y \in \mathcal{Y}$) if it localizes y to a region smaller than a given error bound ϵ, in particular, if

$$\text{area}(\{z \in \mathcal{Y}_{\text{outer}} \,|\, a_i^T z \geq b_i\}) \leq \epsilon.$$

We can give a bound on the number of iterations required to achieve this stopping criterion.

Theorem 2: Given $y \in \mathbb{R}^2$ where $y \in \mathcal{Y}_{\text{outer}}$ and $y \notin \mathcal{Y}_{\text{inner}}$, let $\{z \in \mathcal{Y}_{\text{inner}} \,|\, a^T z = b\}$ define the edge of $\mathcal{Y}_{\text{inner}}$ for which $a^T y > b$ and suppose that the area outside this edge between

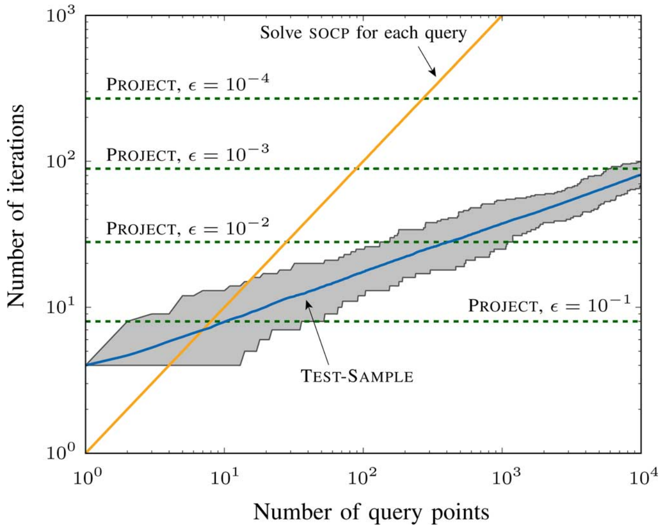


Fig. 9. Number of iterations required to test a variable number of query points for the example shown in Fig. 2. Results for TEST-SAMPLE are averaged over 200 runs; the shaded region shows the range of values.

$\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ is initially

$$\alpha_0 = \text{area}(\{z \in \mathcal{Y}_{\text{outer}} \,|\, a^T z \geq b\}).$$

Then, for any $\epsilon > 0$, the algorithm TEST-SAMPLE will terminate after no more than

$$\frac{\ln(\alpha_0/\epsilon)}{\ln 4}$$

iterations.

Proof: See Appendix II. ■

C. Results in Simulation

The algorithm TEST-SAMPLE tests the membership of a point in the projection of a nonlinear convex set. Theorem 2 guarantees the convergence of this algorithm. In particular, it guarantees that if we are forced to update $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ in order to classify a query point $y \in \mathbb{R}^2$, then this update takes no more than a small number of iterations. However, Theorem 2 does not tell us how often we need to update $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ given a sequence of query points $q_1, \ldots, q_k \in \mathbb{R}^2$. Indeed, the performance of our algorithm depends on the expectation that only a small fraction of query points will force an update.

In practice, this expectation is often satisfied. As an example, consider the case in which query points are sampled uniformly at random in a fixed, closed subset $U \subset \mathbb{R}^2$, where we assume $\mathcal{Y}_{\text{outer}} \subseteq U$. Then, the probability of an update is

$$\frac{\text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}})}{\text{area}(U)}.$$

Every iteration of TEST-SAMPLE improves our approximation of $\mathcal{Y}$ and reduces the difference $\text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}})$, so every time $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ are updated, it is less likely that they will need to be updated in the future.

Fig. 9 shows the number of iterations required to test an increasing number of query points with TEST-SAMPLE, averaged over 200 simulations for the example shown in Fig. 2. The results

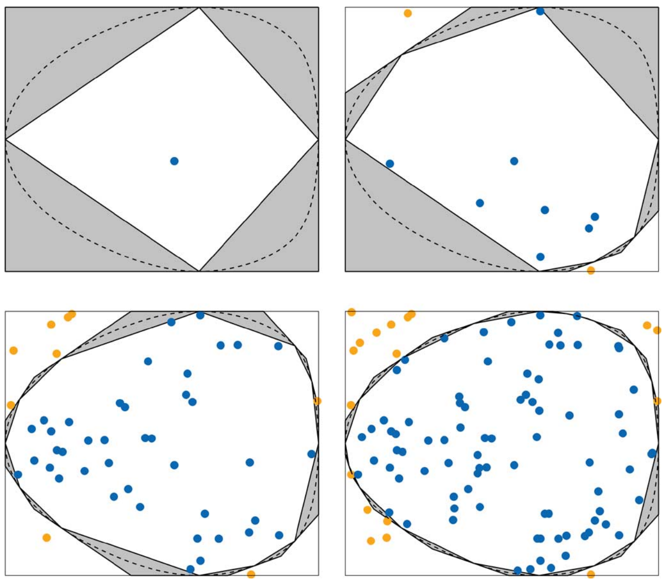


Fig. 10. Application of TEST-SAMPLE to the example shown in Fig. 2 for 1, 10, 50, and 100 query points, respectively, sampled uniformly at random. The set $\mathcal{Y}_{\text{outer}} - \mathcal{Y}_{\text{inner}}$ is shaded.

of one simulation are shown in Fig. 10. We sampled query points uniformly at random and chose U such that the initial probability of an update is $1/2$. For comparison, in Fig. 9, we also show the number of iterations required both if each query point is tested individually [with an SOCP of the form (8)] and if the projection $\mathcal{Y}$ is precomputed (using PROJECT, as in Fig. 3). As expected, individual testing works well for a small number k of queries but poorly for a large number, while precomputation is just the opposite. In contrast, the key advantage of our algorithm TEST-SAMPLE is that it works well over a wide range of k.

As the number of queries k grows very large, precomputation eventually requires fewer iterations than TEST-SAMPLE. However, when this occurs depends on the error bound ϵ specified in PROJECT: we show results for a range of ϵ in Fig. 2. Increasing ϵ decreases the number of iterations, but it also increases the likelihood of misclassifying query points with PROJECT. In contrast, the average number of iterations for TEST-SAMPLE remains the same regardless of ϵ. It is true that the variance in the number of iterations increases as ϵ decreases, but, in practice, we can still choose ϵ as small as we like. In particular, we chose $\epsilon = 10^{-8}$ for TEST-SAMPLE in this example, making the likelihood of misclassifying query points sampled uniformly at random negligibly small.

V. CONCLUSION

In this paper, we presented an algorithm, PROJECT, to compute the projection of a nonlinear convex set onto a 2-D linear subspace, with provably bounded error. Our algorithm can be applied to compute the support region for a legged robot making any number of contacts with any type of terrain, as long as the frictional contact model involves convex constraints.

We also presented a second algorithm, TEST-SAMPLE, to test the membership of points in the projection of a nonlinear convex set without computing its entire shape. We showed that for some sets of query points, TEST-SAMPLE is faster than either testing each query point individually or precomputing the projection. This algorithm can be applied to test static equilibrium for a legged robot at particular CM positions. Although PROJECT can also be used for this purpose, we recommend the use of TEST-SAMPLE since the likelihood of misclassifying query points can be made much lower in practice.

There are several opportunities for future work. For example, it is possible to generalize our approach to compute the projection of a nonlinear convex set onto an n-dimensional linear subspace, where $n > 2$. The proof of convergence follows from a dual interpretation of PROJECT and TEST-SAMPLE, where we take advantage of their similarity to Kelley's cutting plane method [35]. When $n = 3$, this result can be used to compute the robust support region for a legged robot, which (as suggested by [37]) is the set of CM positions at which static equilibrium is possible given any external wrench in a neighborhood of the load due to gravity.

APPENDIX I
PROOF OF THEOREM I

Suppose $\mathcal{Y}_{\text{inner}}$ begins with η_0 edges and that the initial difference in area between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ is

$$\alpha_0 = \text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}}).$$

We will show that the algorithm PROJECT achieves any desired precision $\epsilon > 0$ after no more than

$$\eta_0 \left(\sqrt{\frac{c\alpha_0}{\epsilon}} - 1 \right)$$

iterations, where $c = 343/243 \approx 1.412$ is a fixed constant. Our proof will proceed in three steps. First, we will represent one iteration of PROJECT as a simple operation on a list of positive real numbers (that correspond to areas of triangles). Then, we will bound the sum of these numbers after k iterations. Finally, we will apply this result to bound the number of iterations required by PROJECT to achieve ϵ.

A. *One Iteration of* PROJECT *is a Simple Operation Applied to a List of Triangle Areas*

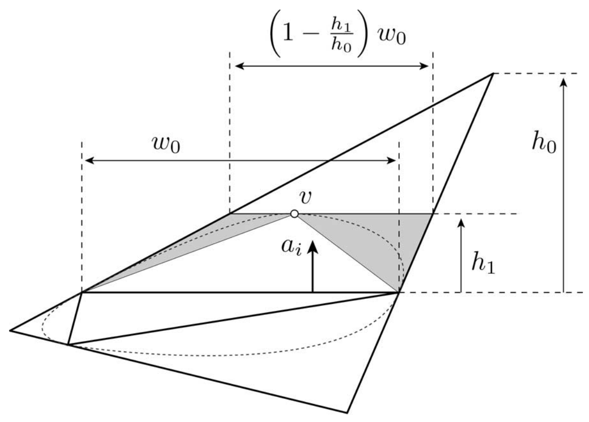
Recall that $\mathcal{Y}_{\text{inner}}$ is a bounded polygon with nonempty interior. So, if $\mathcal{Y}_{\text{inner}}$ contains N edges, then the region between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ consists of N triangles (where we allow these triangles to have zero area). A single iteration of PROJECT replaces the triangle of greatest area with two smaller triangles. In particular, we will show that if the original triangle i has area $p_i \geq 0$, then the two smaller triangles will have total area no greater than $p_i/4$ (see Fig. 11).

Lemma 3: Denote by $\mathcal{Y}^0_{\text{inner}}, \mathcal{Y}^0_{\text{outer}}$ and $\mathcal{Y}^1_{\text{inner}}, \mathcal{Y}^1_{\text{outer}}$ the polygonal approximations to $\mathcal{Y}$ before and after any single iteration of PROJECT, so

$$\mathcal{Y}^1_{\text{outer}} = \{z \in \mathcal{Y}^0_{\text{outer}} \mid a_i^T z \leq a_i^T v\}$$

and

$$\mathcal{Y}^1_{\text{inner}} = \text{conv}(\mathcal{Y}^0_{\text{inner}} \cup \{v\})$$



(a)

Fig. 11. Single iteration of PROJECT reduces the triangle above edge i to the shaded region, the area of which is bounded above for any point v.

for some choice of i and v. Let

$$\beta_0 = \text{area}(\{z \in \mathcal{Y}^0_{\text{outer}} \mid a_i^T z \geq b_i\})$$

and

$$\beta_1 = \text{area}(\{z \in \mathcal{Y}^1_{\text{outer}} \mid a_i^T z \geq b_i \text{ and } z \notin \mathcal{Y}^1_{\text{inner}}\}).$$

Then

$$\frac{\beta_1}{\beta_0} \leq \frac{1}{4}.$$

Proof: Let w_0 be the length of the edge $\{z \in \mathcal{Y}^0_{\text{inner}} \mid a_i^T z = b_i\}$ of $\mathcal{Y}^0_{\text{inner}}$. Let

$$h_0 = \frac{1}{\|a_i\|}\left(\max_{z \in \mathcal{Y}^0_{\text{outer}}} a_i^T z - b_i\right)$$

and

$$h_1 = \frac{1}{\|a_i\|}\left(a_i^T v - b\right).$$

Notice that

$$\beta_0 = \frac{1}{2} w_0 h_0$$

and

$$\begin{aligned}\beta_1 &= \frac{1}{2}\left(w_0 + \left(1 - \frac{h_1}{h_0}\right) w_0\right) h_1 - \frac{1}{2} w_0 h_1 \\ &= \frac{1}{2} w_0 h_1 \left(1 - \frac{h_1}{h_0}\right)\end{aligned}$$

so, we have

$$\frac{\beta_1}{\beta_0} = \frac{h_1}{h_0}\left(1 - \frac{h_1}{h_0}\right).$$

Since

$$\max_{h_1 \in [0, h_0]} \left(\frac{h_1}{h_0}\left(1 - \frac{h_1}{h_0}\right)\right) = \frac{1}{4}$$

then, whatever our choice of v, we must have

$$\frac{\beta_1}{\beta_0} \leq \frac{1}{4}.$$

■

This result allows us to represent one iteration of PROJECT as a simple operation on a list of positive real numbers. In particular, assume $\mathcal{Y}_{\text{inner}}$ contains N edges, so $\mathcal{Y}_{\text{outer}} - \mathcal{Y}_{\text{inner}}$ consists of N triangles that we number $1, \ldots, N$. Define $p \in \mathbb{R}^N$ such that each element $p_i \geq 0$ is equal to the area of triangle i. Then, a single iteration of PROJECT maps the vector p to a new vector $q \in \mathbb{R}^{N+1}$ by replacing the largest element p_i of p with two new elements q_j and q_k of q, such that $q_j + q_k \leq p_i/4$. In the following section, we will consider the (now abstracted) problem of bounding the sum $\mathbf{1}^T q$ after many iterations.

B. Bounding the Sum of Triangle Areas After k Iterations

We first define a convenient sorting function for lists. Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be given by $y = f(x)$ if there exists a permutation P such that $y = Px$ and

$$y_1 \geq y_2 \geq \ldots \geq y_n.$$

Next, given a point $p \in \mathbb{R}^n$, we define a subset $A \subset \mathbb{R}^{n+1}$ as follows. Let $y = f(p)$. Then, a point q is in A if there exist $a, b \geq 0$ such that $a + b \leq y_1/4$ and the vector q is some permutation of the list $(y_2, y_3, \ldots, y_n, a, b)$. Since with every point $p \in \mathbb{R}^n$ we have a set of points $A \subset \mathbb{R}^{n+1}$, this defines a map $g_n : \mathbb{R}^n \to 2^{\mathbb{R}^{n+1}}$ where $g_n(p) = A$. Lemma 3 implies that if p is a list of the n triangle areas between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$, then the set of possible $n+1$ triangle areas after a single iteration of PROJECT is equal to $g_n(p)$.

Finally, for each $k \in \mathbb{Z}$ such that $k \geq 1$, we define the map $\max_k : \mathbb{R}^n \to \mathbb{R}$ by

$$\max_k(x) = (f(x))_k.$$

That is, $\max_k(x)$ is the kth largest element of x where we count multiplicities. For example

$$\max_3(5, 4, 4, 2) = 4.$$

The next three lemmas are key steps in the proof of our main result, Lemma 7, which bounds the sum of triangle areas after k iterations.

Lemma 4: Suppose $p \in \mathbb{R}^n$ and $q \in g_n(p)$ such that $p \geq 0$. Then

$$(1) \quad \sum_{i=1}^{n+1} q_i \leq \sum_{j=1}^{n} p_j - \frac{3}{4}\max_1(p)$$

$$(2) \quad \max_{k+1}(p) \leq \max_k(q) \text{ for all } k = 1, \ldots, n-1.$$

Proof: For part (1), since $q \in g_n(p)$ there exists $a, b \geq 0$ and a permutation matrix P such that

$$q = P\left(\begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} f(p) + \begin{bmatrix} 0 \\ a \\ b \end{bmatrix}\right)$$

where $a + b \leq \max_1(p)/4$ and $I_{n-1} \in \mathbb{R}^{n-1 \times n-1}$ is the identity matrix. So, we have

$$\sum_{i=1}^{n+1} q_i = \mathbf{1}^T \left(\begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} f(p) + \begin{bmatrix} 0 \\ a \\ b \end{bmatrix}\right)$$

$$\begin{aligned}
&= [\,0 \quad \mathbf{1}^T\,]\, f(p) + a + b \\
&= \sum_{i=1}^{n} p_i - \max\nolimits_1(p) + a + b \\
&\leq \sum_{i=1}^{n} p_i - \max\nolimits_1(p) + \max\nolimits_1(p)/4 \\
&= \sum_{i=1}^{n} p_i - \frac{3}{4}\max\nolimits_1(p)
\end{aligned}$$

as desired. For part (2), we have

$$\begin{aligned}
\max\nolimits_k(q) &= \max\nolimits_k\left(\begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} f(p) + \begin{bmatrix} 0 \\ a \\ b \end{bmatrix}\right) \\
&\geq \max\nolimits_k\left(\begin{bmatrix} 0 & I_{n-1} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} f(p)\right) \\
&= \max\nolimits_{k+1}(p)
\end{aligned}$$

as desired. ■

Lemma 5: Let $n \in \mathbb{Z}$ with $n > 0$ and let $k \in \{1, \ldots, n\}$. Also, let $p(n) \in \mathbb{R}^n$ such that $p(n) \geq 0$ and recursively select

$$p(i) \in g_{i-1}(p(i-1))$$

for all $i = n+1, \ldots, k$. Then

$$\sum_{i=1}^{n+k} p_i(n+k) \leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\sum_{i=1}^{k} \max\nolimits_i(p(n))$$

and in particular

$$\sum_{i=1}^{2n} p_i(2n) \leq \frac{1}{4}\sum_{i=1}^{n} p_i(n).$$

Proof: The proof proceeds by induction; we give the first few steps and omit the details. From Lemma 4(1), we have

$$\sum_{i=1}^{n+1} p_i(n+1) \leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\max\nolimits_1(p(n)).$$

Similarly, applying Lemma 4(1) to $p(n+1)$, we have

$$\begin{aligned}
\sum_{i=1}^{n+2} p_i(n+2) &\leq \sum_{i=1}^{n+1} p_i(n+1) - \frac{3}{4}\max\nolimits_1(p(n+1)) \\
&\leq \sum_{i=1}^{n+1} p_i(n+1) - \frac{3}{4}\max\nolimits_2(p(n))
\end{aligned}$$

from Lemma 4(2). Combining these results, we find

$$\sum_{i=1}^{n+2} p_i(n+2) \leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\left(\max\nolimits_1(p(n)) + \max\nolimits_2(p(n))\right).$$

Proceeding in the usual way by induction, we have

$$\sum_{i=1}^{n+k} p_i(n+k) \leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\sum_{i=1}^{k} \max\nolimits_i(p(n))$$

as desired. In particular, we find that

$$\begin{aligned}
\sum_{i=1}^{2n} p_i(2n) &\leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\sum_{i=1}^{n} \max\nolimits_i(p(n)) \\
&= \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\sum_{i=1}^{n} p_i(n) \\
&= \frac{1}{4}\sum_{i=1}^{n} p_i(n).
\end{aligned}$$

■

Lemma 6: Let $n \in \mathbb{Z}$ with $n > 0$ and let $k \in \{1, \ldots, n\}$. Also, let $p \in \mathbb{R}^n$ such that $p \geq 0$. Then

$$\sum_{i=1}^{k} \max\nolimits_i(p) \geq \frac{k}{n}\sum_{i=1}^{n} p_i.$$

Proof: By definition, we must have

$$\max\nolimits_{k+1}(p) \geq \frac{1}{n-k}\sum_{i=k+1}^{n} \max\nolimits_i(p).$$

Also, since $\max_i(p) \geq \max_{k+1}(p)$ for all $i \in \{1, \ldots, k\}$, then

$$\begin{aligned}
\sum_{i=1}^{k} \max\nolimits_i(p) &\geq \frac{k}{n-k}\sum_{i=k+1}^{n} \max\nolimits_i(p) \\
&= \frac{k}{n-k}\left(\sum_{i=1}^{n} p_i - \sum_{i=1}^{k} \max\nolimits_i(p)\right)
\end{aligned}$$

which implies that

$$\left(\frac{n-k}{k} + 1\right)\sum_{i=1}^{k} \max\nolimits_i(p) \geq \sum_{i=1}^{n} p_i$$

and hence

$$\sum_{i=1}^{k} \max\nolimits_i(p) \geq \frac{k}{n}\sum_{i=1}^{n} p_i$$

as desired. ■

Now, we are ready to prove our main result.

Lemma 7: Let $n, k \in \mathbb{Z}$ with $n, k > 0$. Also, let $p(n) \in \mathbb{R}^n$ such that $p(n) \geq 0$ and recursively select

$$p(i) \in g_{i-1}(p(i-1))$$

for all $i = n+1, \ldots, k$. Then

$$\sum_{i=1}^{n+k} p_i(n+k) \leq c\left(\frac{n}{n+k}\right)^2 \sum_{i=1}^{n} p_i(n)$$

where $c = 343/243 \approx 1.412$ is a fixed constant.

Proof: First, we will show that this result holds for $k \in \{0, \ldots, n\}$. From Lemma 5, we have

$$\sum_{i=1}^{n+k} p_i(n+k) \leq \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\sum_{i=1}^{k} \max\nolimits_i(p(n))$$

so from Lemma 6

$$\sum_{i=1}^{n+k} p_i(n+k) \le \sum_{i=1}^{n} p_i(n) - \frac{3}{4}\frac{k}{n}\sum_{i=1}^{n} p_i(n) = \left(1-\frac{3}{4}\frac{k}{n}\right)\sum_{i=1}^{n} p_i(n).$$

As a consequence

$$\begin{aligned}\sum_{i=1}^{n+k} p_i(n+k)\left(\left(\frac{n}{n+k}\right)^2\sum_{i=1}^{n} p_i(n)\right)^{-1} &\le \left(1-\frac{3}{4}\frac{k}{n}\right)\sum_{i=1}^{n} p_i(n)\left(\left(\frac{n}{n+k}\right)^2\sum_{i=1}^{n} p_i(n)\right)^{-1}\\ &= \left(1-\frac{3}{4}\frac{k}{n}\right)\left(\frac{n+k}{n}\right)^2\\ &\le \frac{343}{243}\end{aligned}$$

where equality holds when

$$p_j(n) = \frac{1}{n}\sum_{i=1}^{n} p_i(n) \qquad \text{for all } j = 1,\ldots,n$$

and when $k = 5n/9$, so we have our result. Now, we consider the case when $k > n$. Define

$$l = \left\lfloor \log_2 \frac{n+k}{n} \right\rfloor \qquad \text{and} \qquad m = n+k-2^l n$$

so, in fact $l, m \in \mathbb{Z}$ such that

$$n+k = 2^l n + m \qquad \text{and} \qquad m \in \{0,\ldots,2^l n\}.$$

Then, combining our previous result with Lemma 5, we have

$$\begin{aligned}\sum_{i=1}^{n+k} p_i(n+k) &= \sum_{i=1}^{2^l n+m} p_i(2^l n+m)\\ &\le c\left(\frac{2^l n}{2^l n+m}\right)^2 \sum_{i=1}^{2^l n} p_i(2^l n)\\ &\le c\left(\frac{2^l n}{2^l n+m}\right)^2 \frac{1}{4^l}\sum_{i=1}^{n} p_i(n)\\ &= c\left(\frac{n}{2^l n+m}\right)^2 \sum_{i=1}^{n} p_i(n)\\ &= c\left(\frac{n}{n+k}\right)^2 \sum_{i=1}^{n} p_i(n)\end{aligned}$$

as desired. ■

C. *Bounding the Number of Iterations Required by* PROJECT

Suppose $\mathcal{Y}_{\text{inner}}$ begins with η_0 edges and that the initial difference in area between $\mathcal{Y}_{\text{inner}}$ and $\mathcal{Y}_{\text{outer}}$ is

$$\alpha_0 = \text{area}(\mathcal{Y}_{\text{outer}}) - \text{area}(\mathcal{Y}_{\text{inner}}).$$

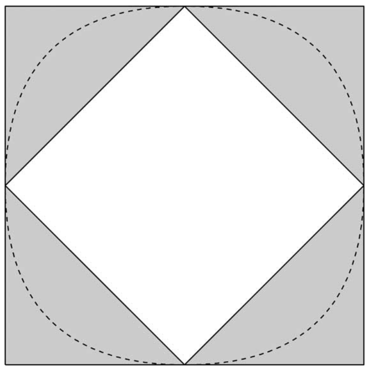


Fig. 12. Example for which the bound of Theorem 1 holds with equality. The set $\mathcal{Y}$ is dotted and the initial approximation $\mathcal{Y}_{\text{outer}} - \mathcal{Y}_{\text{inner}}$ is shaded.

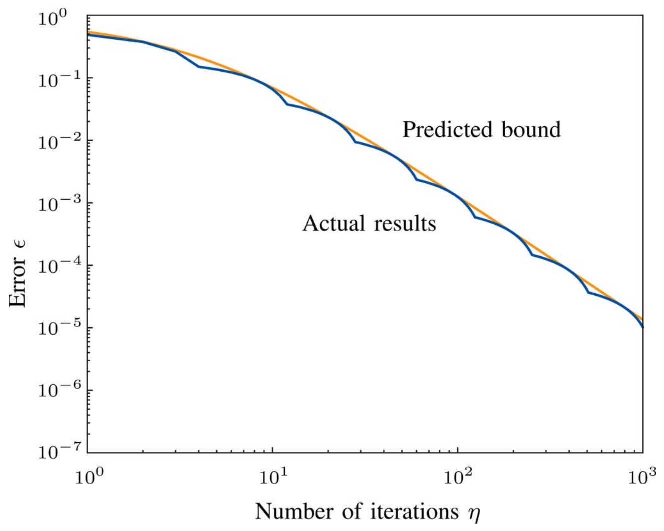


Fig. 13. Number of iterations η required to achieve a given error ϵ with PROJECT for the example shown in Fig. 12. The bound is tight.

Let α_k be the difference in area after k iterations. Then, Lemma 7 tells us that

$$\alpha_k \le c\left(\frac{\eta_0}{\eta_0+k}\right)^2 \alpha_0$$

or equivalently

$$k \le \eta_0\left(\sqrt{\frac{c\alpha_0}{\alpha_k}} - 1\right).$$

So, to achieve $\alpha_k \le \epsilon$ for any desired precision ϵ requires no more than

$$\eta_0\left(\sqrt{\frac{c\alpha_0}{\epsilon}} - 1\right)$$

iterations (where $c = 343/243 \approx 1.412$ is a fixed constant), proving Theorem 1. This bound is tight—in particular, equality holds for the example shown in Figs. 12 and 13, constructed so that every iteration of PROJECT replaces a single triangle of area p_i with two smaller ones, each of area exactly $p_i/8$.

Appendix II

Proof of Theorem 2

Suppose that $\{z \in \mathcal{Y}_{\mathrm{inner}} \mid a^T z = b\}$ defines an edge of $\mathcal{Y}_{\mathrm{inner}}$ such that $a^T y > b$ so that the area outside this edge between $\mathcal{Y}_{\mathrm{inner}}$ and $\mathcal{Y}_{\mathrm{outer}}$ is initially

$$\alpha_0 = \mathrm{area}(\{z \in \mathcal{Y}_{\mathrm{outer}} \mid a^T z \geq b\}).$$

Then, for any desired precision $\epsilon > 0$, we will show that the algorithm TEST-SAMPLE terminates after no more than

$$\frac{\ln(\alpha_0/\epsilon)}{\ln 4}$$

iterations. First, recall that $\mathcal{Y}_{\mathrm{inner}}$ is a bounded polygon with nonempty interior. So, if $\mathcal{Y}_{\mathrm{inner}}$ contains N edges, then the region between $\mathcal{Y}_{\mathrm{inner}}$ and $\mathcal{Y}_{\mathrm{outer}}$ consists of at most N triangles, one of which will contain the query point y. From Theorem 3 (Appendix I), we know that a single iteration of TEST-SAMPLE reduces the area of this triangle by at least 75%. Hence, the triangle containing y after k iterations has area

$$\alpha_k \leq \alpha_0 \left(\frac{1}{4}\right)^k.$$

Since TEST-SAMPLE terminates either when $y \in \mathcal{Y}_{\mathrm{inner}}$ or $y \notin \mathcal{Y}_{\mathrm{outer}}$ are satisfied or when $\alpha_k \leq \epsilon$, then the algorithm must terminate after a number of iterations

$$k \leq \frac{\ln(\alpha_0/\epsilon)}{\ln 4}.$$

References

[1] R. B. McGhee and A. Frank, "On the stability properties of quadruped creeping gaits," *Math. Biosci.*, vol. 3, no. 1–2, pp. 331–351, 1968.

[2] R. B. McGhee and G. I. Iswandhi, "Adaptive locomotion of a multilegged robot over rough terrain," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-9, no. 4, pp. 176–182, Apr. 1979.

[3] D. Wettergreen, "Robotic walking in natural terrain," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, 1995.

[4] J. J. Kuffner, Jr., S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Auton. Robots*, vol. 12, pp. 105–118, 2002.

[5] M. Vukobratovic, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," *IEEE Trans. Biomed. Eng.*, vol. 17, no. 1, pp. 25–36, Jan. 1970.

[6] E. Garcia, J. Estremera, and P. G. de Santos, "A comparative study of stability margins for walking machines," *Robotica*, vol. 20, pp. 595–606, 2002.

[7] M. B. Popovic, A. Goswami, and H. Herr, "Ground reference points in legged locomotion: Definitions, biological trajectories and control implications," *Int. J. Rob. Res.*, vol. 24, no. 12, pp. 1013–1032, 2005.

[8] S.-M. Song and K. J. Waldron, *Machines That Walk: The Adaptive Suspension Vehicle*. Cambridge, MA: MIT Press, 1989.

[9] E. Krotkov and R. Simmons, "Perception, planning, and control for autonomous walking with the ambler planetary rover," *Int. J. Rob. Res.*, vol. 15, pp. 155–180, 1996.

[10] P. G. de Santos, J. Estremera, E. Garcia, and M. Armada, "Including joint torques and power consumption in the stability margin of walking robots," *Auton. Robots*, vol. 18, no. 1, pp. 43–57, Jan. 2005.

[11] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *Int. J. Robot. Res.*, vol. 22, no. 3–4, pp. 187–202, 2003.

[12] Y. Yokokohji, S. Nomoto, and T. Yoshikawa, "Static evaluation of humanoid robot postures constrained to the surrounding environment through their limbs," in *Proc. IEEE Int. Conf. Rob. Aut.*, 2002, pp. 1856–1863.

[13] D. W. Hong and R. J. Cipra, "Visualization of the contact force solution space for multi-limbed robots," *J. Mech. Design*, vol. 128, no. 1, pp. 295–302, 2006.

[14] D. W. Hong and R. J. Cipra, "Optimal contact force distribution for multi-limbed robots," *J. Mech. Design*, vol. 128, no. 3, pp. 566–573, 2006.

[15] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *Int. J. Rob. Res.*, vol. 25, no. 4, pp. 317–342, 2006.

[16] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe, "Using motion primitives in probabilistic sample-based planning for humanoid robots," presented at the Workshop Algorithmic Found. Robot., New York, NY, 2006.

[17] R. Mason, E. Rimon, and J. Burdick, "Stable poses of 3-dimensional objects," in *Proc. IEEE Int. Conf. Rob. Aut.*, Albuquerque, NM, 1997, pp. 391–398.

[18] H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (ser. Monographs in Theoretical Computer Science). vol. 10, New York, NY: Springer-Verlag, 1987.

[19] T. Bretl and S. Lall, "A fast and adaptive test of static equilibrium for legged robots," in *Proc. IEEE Int. Conf. Rob. Aut.*, Orlando, FL, 2006, pp. 1109–1116.

[20] Y. Or and E. Rimon, "Geometric characterization and experimental validation of frictional 3-contact equilibrium stances in three-dimensions," in *Proc. IEEE Int. Conf. Rob. Aut.*, Rome, Italy, 2007, pp. 193–198.

[21] M. Mason and K. Salisbury, *Robot Hands and the Mechanics of Manipulation*. Cambridge, MA: MIT Press, 1985.

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

[23] J. Ponce, S. Sullivan, A. Sudsang, J.-D. Boissonnat, and J.-P. Merlet, "On computing four-finger equilibrium and force-closure grasps of polyhedral objects," *Int. J. Rob. Res.*, vol. 16, no. 1, pp. 11–35, 1997.

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

[25] G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition*. vol. 33, Berlin, Germany: Springer-Verlag, 1975, pp. 134–183.

[26] B. F. Caviness and J. R. Johnson, Eds., *Quantifier Elimination and Cylindrical Algebraic Decomposition* (ser. Texts and Monographs in Symbolic Computation). Vienna, Austri: Springer-Verlag, 1998.

[27] K. Fukuda, "cdd, cddplus, and cddlib homepage," McGill Univ., Montreal, QC, Canada, 2002, http://www.www.cs.mcgill.ca/∼fukuda/software/cdd_home/cdd.html.

[28] G. B. Dantzig and B. C. Eaves, "Fourier–Motzkin elimination and its dual," *J. Combinatorial Theory*, vol. 14, pp. 288–297, 1973.

[29] K. Fukuda and A. Prodon, "Double description method revisited," in *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, vol. 1120, pp. 91–111, 1996.

[30] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996.

[31] D. Bremner, K. Fukuda, and A. Marzetta, "Primal-dual methods for vertex and facet enumeration," in *Proc. ACM Sympos. Comput. Geom.*, Nice, France, 1997, pp. 49–56.

[32] M. Joswig, "Beneath-and-beyond revisited," in *Algebra, Geometry, and Software Systems*, M. Joswig and N. Takayama, Eds. New York: Springer-Verlag, 2003, pp. 1–21.

[33] A. Ben-Tal and A. Nemirovski, "On polyhedral approximations of the second-order cone," *Math. Operations Res.*, vol. 26, no. 2, pp. 193–205, May 2001.

[34] J. Lasserre, "Semidefinite programming vs. LP relaxations for polynomial programming," *Math. Operations Res.*, vol. 27, no. 2, pp. 347–360, May 2002.

[35] J. E. Kelley, Jr., "The cutting-plane method for solving convex programs," *J. Society Ind. Applied Math.*, vol. 8, no. 4, pp. 703–712, Dec. 1960.

[36] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra Appl.*, vol. 284, pp. 193–228, 1998, [special Issue on Linear Algebra in Control, Signals, and Image Processing].

[37] Y. Or and E. Rimon, "Computation and graphical characterization of robust multiple-contact postures in two-dimensional gravitational environments," *Int. J. Rob. Res.*, vol. 25, no. 11, pp. 1071–1086, Nov. 2006.

Timothy Bretl (S'02–M'05) received the B.S. degree in engineering and the B.A. degree in mathematics from Swarthmore College, Swarthmore, PA, both in 1999, the M.S. degree in 2000 and the Ph.D. degree in 2005 in aeronautics and astronautics from Stanford University, Stanford, CA.

During 2005–2006, he was a Postdoctoral Fellow in the Department of Computer Science, Stanford University. Since 2006, he has been with the University of Illinois at Urbana-Champaign, where he is currently an Assistant Professor of Aerospace Engineering. His current research interests include motion analysis, planning, and control of robotic systems.

Sanjay Lall (S'92–M'96) received the B.A. degree in mathematics and the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K., in 1995.

He was earlier a North Atlantic Treaty Organization (NATO) Research Fellow at the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. Until 2000, he was a Research Fellow at the Department of Control and Dynamical Systems, California Institute of Technology. He is currently an Associate Professor of Aeronautics and Astronautics and a Vance D. and Arlene C. Coffman Faculty Scholar at Stanford University, Stanford, CA. His current research interests include the development of advanced engineering methodologies for the design of control systems that occur in a wide variety of aerospace, mechanical, electrical, and chemical systems.

Prof. Lall was the recipient of the George S. Axelby Outstanding Paper Award by the IEEE Control Systems Society in 2007, the National Science Foundation (NSF) Career Award in 2007, and the Graduate Service Recognition Award from Stanford University in 2005.