3D robust stability polyhedron in multi-contact

Hervé Audren, Abderrahmane Kheddar

▶ To cite this version:

Hervé Audren, Abderrahmane Kheddar. 3D robust stability polyhedron in multi-contact. 2017. lirmm-01477362v1

HAL Id: lirmm-01477362
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01477362v1

Preprint submitted on 27 Feb 2017 (v1), last revised 12 Feb 2018 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

3D robust stability polyhedron in multi-contact

Hervé Audren and Abderrahmane Kheddar

***Abstract*—We propose algorithms to compute the 3D robust stability region in multi-contact. It is well known that the stability region is a product of convex cones and hence is a convex polyhedron. Our stability region extends existing recursive 2D static stability approaches to 3D by accounting for possible center of mass accelerations. We provide algorithms that construct the region of robust stability in a systematic way. We compare our algorithms and discuss possible computation of intermediary shapes using morphing. Finally, we provide an example of usage in generating robust static postures that can serve the purpose of multi-contact planning.**

***Index Terms*—Multi-contact, 3D Robust Static Stability.**

I. INTRODUCTION

LEGGED robots have been substantially improved in the last years in terms of hardware and control. For legged robots, to hold *stable* postures in multi-contact is critical to avoid falling under perturbations. Thus a criteria is necessary to enforce the stability of the equilibrium for a given contact stance. Ideally, this criteria will be in the form of a convex function to be used in fast optimization programs.

The stability criteria are tightly linked to the dynamic equations, that are known to be non-linear [17]. They define a relationship between joint torque, acceleration and contact forces through inertial parameters. Even when the whole-body dynamics is reduced to its center-of-mass [38], [2], the resulting equations exhibit the angular momentum of the body as a cross-product between the contact forces and the CoM position; an operation that is neither linear nor convex.

Our idea is to compute a hull $\mathcal{P}$ for the CoM, such that $\forall$ CoM $\in \mathcal{P}$, the stability is guaranteed to be robust. We do so for all possible CoM accelerations that lie within a given convex set $\mathcal{G}$. In fact, $\forall$ CoM $\in \mathcal{P}$, there exist a set of contact forces that can generate any acceleration in $\mathcal{G}$. In other words, we compute the intersection of $\mathcal{G}$ with the set of all possible motions. We show that this intersection results in a convex volume that we project in the CoM space. Hence, by defining $\mathcal{G}$, the CoM acceleration and its position are decoupled.

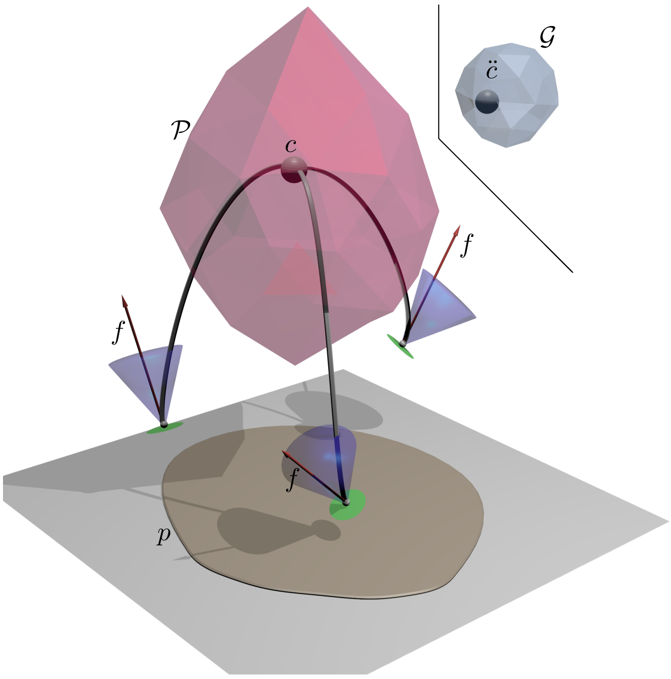

Fig. 1. Illustration of the problem annotated with the variables. We propose algorithms to compute the robust static stability $\mathcal{P}$ for the CoM c, knowing the space $\mathcal{G}$ of the CoM's accelerations $\ddot{c}$. The forces are all noted f, the set to which they belong. Contact surfaces are represented by green disks with the contact points and their normals, represented by the blue friction cones. p is the static stability polygon as computed in [9].

In previous work, some restrictions have been set to obtain such a decoupling. One of the most stringent is to set the CoM acceleration to zero, resulting in the static stability criterion [30]. Static stability (i.e. equilibrium) is used most notably in multi-contact posture generation [16], [7], [11].

Static stability can be a function of the gravity orientation. Indeed [29], [32] present a way to find all the gravity orientations (equivalently, all orientations of the assembly base) that satisfy static stability of assemblies: as it is a convex region defined by inequalities. They apply early vertex enumeration techniques —that we also use, to compute the acceptable region. Yet, the objects were fixed and not actuated.

The best known approach considers a virtual point that accounts for both the CoM position and its acceleration: the zero moment point (ZMP). The original ZMP criterion [43] is the extension of the *convex hull criterion* and has been widely used in biped locomotion on flat grounds assuming high friction, e.g. [26]. The ZMP is better defined in [21], and its commonalities to the Center of Pressure (CoP) highlighted in [41]. The CoP is the local version of the ZMP criterion which allows it to be used in multi-contact but entails control-

---

Manuscript received February XX, 2017; revised Xxxxx XX, 20XX; accepted Xxxxxx XX, 20XX. Date of publication Xxxxxxxx X, 20XX; date of current version Xxxxxxxxx X, 20XX. This paper was recommended for publication by Associate Editor X. Xxxxxxxx and Editor X. XXXX upon evaluation of the reviewers comments.

This work was partially supported by the EU H2020 COMANOID project.

H. Audren and A. Kheddar are with the CNRS-UM2 LIRMM Interactive Digital Humans group, France, and the CNRS-AIST Joint Robotics Laboratory (JRL), UMI3218/RL, Japan

This paper has supplementary video downloadable material available at http://ieeexplore.ieee.org.

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

Digital Object Identifier 00.0000/TRO.201X.0000000

0000–0000/00$00.00 © 201X IEEE

ling independently the CoP at each contact area. Yet, the CoP is not defined when the normal force applied on the contact is nil, and is not easily extensible to bilateral contacts. Even with these drawbacks, CoPs was successfully applied in multi-contact control schemes such as in [42], [24], [44], [34] when combined with a CoM regulation policy for stabilization.

There are other multi-contact criteria. The resultant wrench of the contact forces must remain in a polyhedral convex cone. Pioneered by [40], it formed the basis of the work in [23] ostensibly titled “Adios ZMP”. It is applicable to multi-contact motion while remaining linear and global. This criterion is more clearly and properly established and applied to multi-contact motion in [13], [14]. In the latter works, a new ZMP-like criterion (a pseudo-ZMP) that has to remain in the two-dimensional projection of the convex polyhedral wrench cone is proposed. It is applicable to multi-contact motion and takes into account friction. Note that, similarly to the ZMP and the CoP, the pseudo-ZMP support area depends on the instantaneous CoM position. Instead, our approach imposes constraints on the resultant acceleration and finds a linear, global corresponding constraint on the CoM. It is an extension of our previous work in [1]: instead of computing regular static stability regions, we add explicit stability margins.

Computing wrench cones relies on the double description method [19] to perform projections. We observed that when the number of contacts increases slightly, the computation times increases drastically. Alternatively, *incremental projection* [27], computes an arbitrary close polyhedral approximation of convex projections without the need of having the full-dimensional polytope. As the static support area is a convex shape, this technique computes fast and is used in stability checks [9] and in multi-contact control [1]. Moreover, this technique does not require friction cones discretization. A very recent approach [37] suggests a quasi-analytic formulation of this region but it cannot handle arbitrary arrangements of contacts, and requires friction cones discretization in practice.

Constraining the CoM to remain in p is a global, CoM based, and linear criterion. Unfortunately, static stability (equilibrium) criterion does not imply dynamic stability [20], [1]. Indeed tracking a statically stable trajectory with changes in acceleration may induce falling. Thus, static stability is marginal, i.e. not robust to changes in the total acceleration.

We propose algorithms to compute a *robust* stability region $\mathcal{P}$ for a given $\mathcal{G}$. Our approach has commonalities with [5], devised to check the robustness of a known trajectory and used linearized friction cones. [13] noted that such a region exists, but it was only sampled. In [36] a vertical cross-section of this region was computed using a line-sweep algorithm, not unlike that of [6] but of course not directly transposable to 3D.

We thus show the following:

- $\mathcal{P}$ is a three dimensional convex shape (section III);
- It can be efficiently approximated by convex polyhedrons (section IV). Also, if $\mathcal{G}$ is limited to a convex polytope, it is the intersection of non-right prisms;
- Our construction algorithm can be modified to quickly test the equilibrium of many points (section VI)
- Changes in acceleration can be well approximated by morphing polyhedrons (section VII).
- We exemplify our stability criteria with a robust posture generation problem (section VIII).

We recall the recursive projection algorithm [9] that computes the static stability polygon p, represented in Fig. 1; this is because our work follows a similar methodology.

II. COMPUTATION OF THE STATIC STABILITY POLYGON

A. Recursive projection algorithm

The algorithm in [9] consists in building an inner and outer approximation of the true polygon, p, by solving a sequence of second-order cone programs:

$$\begin{aligned} \max_{c,f} \quad & d^T c \\ \text{s.t.} \quad & A_1 f + A_2(g)c = T(g) \\ & \|Bf\| \leq u(\mu)^T f \end{aligned} \tag{1}$$

where (see also Fig. 1),

- f is the set of contact forces;
- c is the CoM position;
- d is the given search direction;
- A_1 is the contact matrix that sums all forces and moments;
- A_2 represents the gravity wrench at position c;
- B is the friction cone matrix;
- u is the matrix representing the contact normals and the friction coefficient μ;
- T is the gravity wrench in function of the gravity g and the robot mass m;
- $\|.\|$ is the euclidian norm.

The solution c^* is an extremal CoM position in the direction d, and is thus added to the inner approximation. Conversely, the half-plane defined by $\left\{c \in \mathbb{R}^2 | d^T c < d^T c^*\right\}$ is a facet of the outer approximations. All points within the inner approximation are thus stable, those outside of the outer approximation are unstable, whereas the stability is undetermined for the points in-between the inner and outer approximations.

The main difficulty of this class of algorithms is the choice of the search direction d. In statics, the undetermined region between the inner and outer approximation is entirely composed of triangles. The next search direction is chosen to be perpendicular to the edge of the inner approximation that belongs to the triangle of maximum area and pointing outwards.

B. Problems associated with bilateral contacts

If we have opposite normals, there is no boundary on the forces applied along the pair of contacts line, and the problem is infeasible. Thus we limit the search region [1]:

- A bounding sphere on the CoM position: we add a cone constraint $\|c\| \leq r_{\max}$, which can be set from the kinematics limits of the robot;
- A limit on the forces: for each force, no single component should be greater (in absolute value) than the weight of the robot. As this is much of a heuristic, we rather set a limit on the actuator torque generated by the forces at each contact point.

III. ROBUST STATIC STABILITY

The previously generated shape p gathers all the statically stable CoM positions. For a given set of contacts, we would like to know where the CoM can be, under a given set of feasible accelerations[1], that keeps the forces within their friction cones. This is what we call the robust stability region, illustrated by $\mathcal{P}$ in the Fig. 1.

A. Problem formulation

Definition 1 (Robust static equilibrium)**.** A CoM position c is in robust static equilibrium with respect to a given residual radius r iff:

$$\forall\, \tilde{g}_i \text{ such that } \|\tilde{g}_i\| < r, \exists\, f_i \text{ such that}$$

$$A_1 f_i + A_2(g + \tilde{g}_i)c = T(g + \tilde{g}_i) \tag{2}$$

$$\|Bf_i\| \leq u^T f_i \tag{3}$$

Where all notations are the same as Equation 1. Recall that A_2 represents the cross product with the gravity. In the non-robust case, its last line is nil as g is aligned with the vertical axis. Hence, the vertical component of c does not contribute anything, and eq. (1) defines a two-dimensional shape. In the robust case, $g + \tilde{g}_i$ will almost always *not* be aligned with the vertical axis and eq. (2) defines a three dimensional shape. Moreover, this three-dimensional volume is an infinite intersection of convex shapes –one for each couple (f_i, g_i), therefore it is a convex shape.

In [5], $\mathcal{P}$ is a given point, of a predefined CoM trajectory, at which they compute $\mathcal{G}$. The latter is a sphere, of unknown radius r, centered on the acceleration that is obtained for a given contact forces. They provide an algorithm to compute r based on gravito-inertial wrench projections and use it as a robustness measure for the given trajectory.

In [13] robust static stability was envisioned as the set of CoM positions $\mathcal{P}$, where a set of lateral accelerations $\mathcal{G}$, centered around g, could be generated by contact forces. Here, $\mathcal{P}$ is only sampled, whereas our method systematically computes $\mathcal{P}$ and allows us to fast test robust equilibrium of any points (see section VI).

Another approach in [39] proposes a set of algorithms, including a faster version of [9], to deal with postural robustness. However, they considered robustness to contact force errors. That is to say, for a given uncertainty of the force, they seek if it can be balanced by the others. They thus missed a point of the utmost importance: in the robust case, the region describing all possible CoM positions is no longer a 2-dimensional polygon, but a 3-dimensional polyhedron as described previously.

In Definition (1), we state the problem $\forall \tilde{g}_i$. In order to compute $\mathcal{P}$, it is necessary to reduce it to a tractable form. We do so using two different approaches. The first approach writes a more conservative constraint on eq. (2) and eq. (3). The second approach discretizes $\mathcal{G}$.

The goal of both approaches is to obtain a finite set of convex constraints. Those constraints can then be used in a convex optimization problem. Combining this problem with the recursive projection algorithm [27], [9] allows to compute the corresponding convex shape. Furthermore, if those constraints are linear they can also be used in a direct projection algorithm as in [13].

B. Formulating a stricter constraint

In this section, we derive a new inequality (depending on r but not individual $\tilde{g}_i$), that will induce eq. (3). This new constraint has the same form as eq. (1). Let us consider the solution for the static problem:

$$A_1 f + A_2(g)c = T(g) \tag{4}$$

$$\|Bf\| \leq u^T f \tag{5}$$

Proposition 1. *If* $\exists\, (l, s) \in \mathbb{R}^2$ *such that* $\forall\, f_i$*:*

$$u^T f_i \geq u^T f + l \tag{6}$$

$$\|Bf_i\| \leq \|Bf\| + s \tag{7}$$

we have the following implication:

$$\|Bf\| \leq u^T f + l - s \Rightarrow \|Bf_i\| \leq u^T f_i \tag{8}$$

Proof. $\|Bf\| \leq u^T f + l - s$ writes $\|Bf\| + s \leq u^T f + l$ then $\|Bf_i\| \leq \|Bf\| + s \leq u^T f + l \leq u^T f_i$ from eq. (6) and eq. (7). □

In what follows, we show that bounds l and s exist and subsequently eq. (8) holds.

We need intermediary results that will be used in expressing both l and s.

Lemma 1. *The only possible* f_i *are given by:*

$$f_i = f + A_1^\dagger\,(T(\tilde{g}_i) - A_2(\tilde{g}_i)c) + (I - A_1^\dagger A_1)w \tag{9}$$

with $.^\dagger$ *the Moore-Penrose pseudo inverse, and* w *a vector having the size of contact forces.*

Proof. We first exploit the linearity of A_2 (screw operator) and T (stacking acceleration with zero angular momentum):

$$eq.\ (2) \text{ and } eq.\ (4) \Leftrightarrow A_1 \cdot (f_i - f) + A_2(\tilde{g}_i)c = T(\tilde{g}_i) \tag{10}$$

Since A_1 is full row-rank[2], the only possible f_i is given by:

$$f_i = f + A_1^\dagger\,(T(\tilde{g}_i) - A_2(\tilde{g}_i)c) + (I - A_1^\dagger A_1)w \tag{11}$$

□

Theorem 1. *A suitable lower bound* l *is given by:*

$$u^T f_i - u^T f \geq l = -\mu r m \bar{\sigma}(A_1^\dagger)(1 + \|c\|) \tag{12}$$

Proof. We develop $u^T f_i$ using Lemma 1, considering the minimal norm solution: $w = 0$.

$$u^T f_i = u^T f + u^T A_1^\dagger\,(T(\tilde{g}_i) - A_2(\tilde{g}_i)c) \tag{13}$$

$$= u^T f + u^T A_1^\dagger T(\tilde{g}_i) - u^T A_1^\dagger A_2(\tilde{g}_i)c \tag{14}$$

Now, we will use the following properties:

---

[1] The said CoM accelerations can result from a perturbation, i.e. applied external forces, or from the control, i.e. from the actuators.

[2] Except the degenerate unlikely case where all the contact points are aligned.

- For any vectors a and b, $\left|a^T b\right| \leq \|a\| \|b\|$ (Cauchy-Schwarz inequality);
- $\|u\| = \mu$ (by definition);
- $\|Ax\| \leq \bar{\sigma}(A) \|x\|$ with $\bar{\sigma}(A)$ the largest singular value of A.

Firstly,

$$u^T A_1^\dagger T(\tilde{g}_i) \geq -\|u\| \left\|A_1^\dagger T(\tilde{g}_i)\right\| \tag{15}$$

$$\geq -\mu\bar{\sigma}(A_1^\dagger) mr \tag{16}$$

Secondly,

$$u^T A_1^\dagger A_2(\tilde{g}_i)c \leq \|u\| \left\|A_1^\dagger A_2(\tilde{g}_i)c\right\| \tag{17}$$

$$\leq \mu\bar{\sigma}(A_1^\dagger) mr \|c\| \tag{18}$$

$$-u^T A_1^\dagger A_2(\tilde{g}_i)c \geq -\mu\bar{\sigma}(A_1^\dagger) mr \|c\| \tag{19}$$

Thus, from eq. (16) and eq. (19), we have

$$u^T f_i \geq u^T f - \mu r m \bar{\sigma}(A_1^\dagger)(1 + \|c\|) \tag{20}$$

□

Theorem 2. *A suitable upper bound s is given by:*

$$\|Bf_i\| - \|Bf\| \leq s = rm\bar{\sigma}(BA_1^\dagger)(1 + \|c\|) \tag{21}$$

Proof. We develop $||Bf||$ using Lemma 1, considering the minimal norm solution: $w = 0$.

$$\|Bf_i\| = \left\|Bf + BA_1^\dagger T(\tilde{g}_i) + BA_1^\dagger A_2(\tilde{g}_i)c\right\| \tag{22}$$

We first apply the triangular inequality:

$$\|Bf_i\| \leq \|Bf\| + \left\|BA_1^\dagger T(\tilde{g}_i)\right\| + \left\|BA_1^\dagger A_2(\tilde{g}_i)c\right\| \tag{23}$$

Firstly:

$$\left\|BA_1^\dagger T(\tilde{g}_i)\right\| \leq rm\bar{\sigma}(BA_1^\dagger) \tag{24}$$

Secondly:

$$\left\|BA_1^\dagger A_2(\tilde{g}_i)c\right\| \leq rm\bar{\sigma}(BA_1^\dagger) \|c\| \tag{25}$$

We thus have:

$$\|Bf_i\| - \|Bf\| \leq rm\bar{\sigma}(BA_1^\dagger)(1 + \|c\|) \tag{26}$$

□

Corollary 1. *For a given CoM position c, if there exist contact forces f that satisfy:*

$$A_1 f + A_2(g)c = T(g) \tag{27}$$

$$\|Bf\| \leq u^T f - rm(\bar{\sigma}(BA_1^\dagger) + \mu\bar{\sigma}(A_1^\dagger))(1 + \|c\|) \tag{28}$$

Then c is in robust static equilibrium.

Proof. The proof is straightforwardly obtained by combining Theorem 2 and Theorem 1 using Proposition 1. □

Unfortunately, eq. (28) is in general *not* tight, and the converse of Corollary 1 is not true. Indeed, let us examine under which conditions we achieve tightness:

- $\|Aa\| \leq \bar{\sigma} \|a\|$ is tight whenever a is aligned with the singular vector associated with the maximum singular value, say $\bar{v}$. This gives us:
  - $\tilde{g}_i = r\bar{v}$ thus $\tilde{g}_i$ is parallel to $\bar{v}$.
  - $A_2(\tilde{g}_i)c = k\bar{v}$ i.e. $\tilde{g}_i \times c = \alpha r \|c\| \bar{v}$ that is to say a vector perpendicular to $\tilde{g}_i$ is parallel to $\bar{v}$.
- $\|A_2(\tilde{g}_i)c\| \leq mr \|c\|$ is equal iff $c \perp \tilde{g}_i$

The three above conditions are incompatible in general because $\tilde{g}_i$ has to be colinear with $\bar{v}$ and perpendicular to it. Moreover, c will be different at each iteration, but $\bar{v}$ is a constant.

Even if this constraint is the tightest conic constraint (in terms of L_2-norm) we found, it is sill conservative. However, it allows to reduce the dimensionality of the problem substantially as we only need to find a single set of forces associated to a CoM position.

C. Discretizing the hypersphere

Instead of the conservative bound of Definition 1 described previously, we approximate the sphere $\|\tilde{g}_i\| < r$ by a polytope whose k vertices are selected among $\tilde{g}_i$, that is:

$$g_i = g + \tilde{g}_i \qquad i \in [0 \cdots k] \tag{29}$$

The closer this polytope is to a sphere of radius r, the closer the computed polyhedron will be to the real $\mathcal{P}$.

The projection of the acceleration on the horizontal directions is not null. Hence, we still cannot use the method in [9]. Instead, we need to find 3D CoM positions that realize every k accelerations. That is, can we find k sets of forces $F = \left[f_0^T \cdots f_k^T\right]^T$ that produce $G = \left[g_0^T \cdots g_k^T\right]$ at a given CoM position c?

Proposition 2. *If a CoM position c is in robust static equilibrium ($c \in \mathcal{P}$), there exists F that verifies:*

$$\begin{aligned} \Phi F + \Psi c &= \Gamma \\ \|\Lambda F\| &\leq \Upsilon F \end{aligned} \tag{30}$$

With:

$$\Phi = \text{diag}(A_1 \cdots A_1) \tag{31}$$

$$\Psi = \left[A_2(g_0)^T \cdots A_2(g_k)^T\right]^T \tag{32}$$

$$\Gamma = \left[T(g_0)^T \cdots T(g_k)^T\right]^T \tag{33}$$

$$\Lambda = \text{diag}(B \cdots B) \tag{34}$$

$$\Upsilon = \begin{bmatrix} u & \cdots & u \end{bmatrix}^T \tag{35}$$

Proof. We stack k problems given by Definition 1, one for each $\tilde{g}_i$. As a robust CoM position verifies the constraints of Definition 1 for any $\tilde{g}_i$, it does also for any k of them. □

With this approach, the robustness of the static equilibrium $\mathcal{P}$ is obtained w.r.t. the polytope $\mathcal{G}$ defined by the $k\tilde{g}_i$.

In the next section, we introduce the algorithms that leverage those new constraints in order to compute $\mathcal{P}$.

IV. COMPUTING THE ROBUST POLYHEDRON

We are using the *double description* in our algorithms, therefore we briefly recall its principles.

A. Double description

The Weyl-Minkowski theorem [45], [31] states that any convex region defined by a finite number of inequalities is a convex polyhedron with a finite number of vertices. Hence, any convex region can be equivalently represented in two ways:

- *hyperplanes representation* (H-rep): a matrix H composed of its normals and a vector b of associated offsets. A point a is interior to this region iff $Ha \leq b$;
- *vertices representation* (V-rep): a set of vertices V. A point a is interior to this region iff there exists a vector $\vec{\alpha}$ having the size of V and with positive coefficients, such that $a = V^T \alpha$ with $\sum_i \alpha(i) = 1$.

A number of algorithms and implementations are available to perform the H-rep to V-rep mapping and *vice-versa*. They are referred to as the *double description* algorithms, and vary in their capabilities and performance. Yet, none of them is inherently superior to the others. Notable examples are `Qhull` [4] that computes convex hull and half-space representations from vertices using the Beneath-Beyond algorithm; the `Cddlib` [19] and the Parma Polyhedra Library (`PPL`) [3] both of which build on the Fourier-Motzkin elimination algorithm [18], [33], the dual of the Beneath-Beyond algorithm. Unfortunately, both algorithms have super-exponential complexity in the worst case [8], [25].

B. Direct projection

To compute the direct projection of a set defined by convex constraints, it is necessary to perform four steps:

1) Reduce the problem to a set of linear inequalities and equalities;
2) Enumerate the vertices bounding that set (i.e. find the V-rep from an the H-rep);
3) Project the V-rep in the given space, resulting in another set of vertices.
4) Compute the convex hull of the previously obtained set of vertices.

Following these steps, we can first reduce eq. (2) by linearizing the friction cones and $\mathcal{G}$. Unfortunately, a linear $\mathcal{G}$ with N vertices, leads to searching for N sets of forces. This is not an option since the double description has a worst case complexity that is exponential is the number of dimensions [22]. Plus, all iterative conversion algorithms are super-polynomial in the combined size of input and output [8], [25].

Moreover, eq. (28) has non-linear terms in both sides; its linearization is far from being trivial. We thus investigate an alternative: the *recursive projection*.

C. Recursive projection

1) Principle: The recursive projection technique consists in approximating the projection of a convex set by linear boundaries [27]. We seek for the approximation of $\mathcal{P}$ as a projection in the 3D space, of a higher dimensional space. The core process of the recursive projection is to iteratively select directions d. The furthest point found along d (the CoM c^*) will be part of the set ($\mathcal{P}$). The others, i.e. $\{c | d^T c > c^*\}$ will be outside of the set ($\mathcal{P}$). The collection of c^* will form an inner (i.e. conservative) approximation of the set ($\mathcal{P}$), while the complementary of the collection of unacceptable points will be a convex outer (i.e. over) approximation. The boundary of the real set lies in-between those two approximations, and they match exactly whenever *all* directions have been enumerated. In practice, the number of iterations will be limited.

We also associate a stopping criterion that depends on a measure of the error between the inner and outer approximation. Also, for fast convergence, we have to choose suitable search directions.

We now present two ways to formulate an optimization problem that yields the appropriate c^*: (i) using bounds in subsection III-B and (ii) linearization in subsection III-C.

2) Using bounds: Unfortunately, the formulation (eq. (28)) obtained in subsection III-B is not suitable to an optimization problem. To turn it into a second-order cone program (SOCP), we need to transform eq. (28) into two single conic inequalities by introducing an additional slack variable s:

$$\|c\| \leq s \tag{36}$$

$$\|Bf\| \leq u^T f - rm(1+s)(\bar{\sigma}(BA_1^\dagger) + \mu\bar{\sigma}(A_1^\dagger)) \tag{37}$$

Then, solving the following optimization problem gives us an extremal point:

$$\begin{aligned} \max_{c,f,s} \quad & d^T c \\ \text{s.t.} \quad & A_1 f_i + A_2(g) c = T(g) \\ & \|c\| \leq s \\ & \|Bf\| \leq u^T f - rm(1+s)(\bar{\sigma}(BA_1^\dagger) + \mu\bar{\sigma}(A_1^\dagger)) \end{aligned} \tag{38}$$

3) Using linearization: It is much simpler as the constraints presented in subsection III-C are already in a form that can be combined with a linear cost function to form a SOCP:

$$\begin{aligned} \max_{c,F} \quad & d^T c \\ \text{s.t.} \quad & \Phi F + \Psi c = \Gamma \\ & \|\Lambda F\| \leq \Upsilon F \end{aligned} \tag{39}$$

4) The projection algorithm: Given that we are dealing with convex 3-dimensional sets, a natural measure of size is the volume. Thus we use as a stopping criterion the volume difference between the inner and outer approximations.

To maximize the error reduction at each step, we took the inspiration from [9] and choose the search direction to be perpendicular to the facet of the inner approximation. We associate to each facet of the inner approximation a convex polyhedron called an *uncertainty volume* (ν_σ for short): it is the set of points that are outside that facet but inside the outer approximation. This ν_σ can be seen as a cut of the outer approximation by the facet. A facet is defined by its normal n and its offset o as $\{c \in \mathbb{R}^3 | n^T \leq o\}$. Thus, the ν_σ is easily computed if an H-rep of $\mathcal{P}_{\text{outer}}$, $(H_{\text{outer}}, b_{\text{outer}})$ is available:

$$\begin{aligned} \nu_\sigma(\texttt{facet}) &= \{c \in \mathbb{R}^3 | c \in \mathcal{P}_{\text{outer}}, n^T c > o\} \\ &= \{c \in \mathbb{R}^3 | H_{\text{outer}} c \leq b_{\text{outer}}, n^T c > o\} \\ &= \textsc{cut}(\mathcal{P}_{\text{outer}}, \texttt{facet}) \end{aligned} \tag{40}$$

In [9], the difference between the inner and outer approximation was a disjoint union of triangles. Therefore, computing their areas and adding and removing points and hyperplanes was easy. Indeed, a simple list of ordered vertices is sufficient and all operations can be performed without a dedicated algorithm.

In our case, we need to use the double description. Indeed, one iteration starting from the simplest volume, a tetrahedron, does not generate a union of disjoint tetrahedrons, but an union of intersecting prismatoids, see Figure 2.

For a given facet of the outer approximation, if we cut its ν_σ with a parallel (to facet) hyperplane, there is no guarantee that this hyperplane does not intersect another uncertainty volume.

Instead, we propose the following Algorithm 1:

- The input is the 6D contact positions and their friction coefficients.
- We initialize $\mathcal{P}_{\text{inner}}$, $\mathcal{P}_{\text{outer}}$ and the list ν of uncertainty volumes, by solving eq. (39) or eq. (38). ν associates to each facet its uncertainty volume.
- For all facets, we check if they are invalid. This can happen in two cases: (i) the facet k was added at the last iteration and has no ν_k or (ii) its ν_k was intersected by the plane added at the previous iteration. In both cases ν_k needs to be computed.
- We then select the direction d to be the normal to the facet having the biggest uncertainty volume.
- Solve the SOCP eq. (39) or eq. (38).
- Resulting c^* is added to the $\mathcal{P}_{\text{inner}}$, and the plane normal to this direction passing by c^* is added to $\mathcal{P}_{\text{outer}}$.
- Invalidate the uncertainty volumes ν_i that are cut by the plane we just added.

A note on the polyhedral volume computation: computing the volume of a set of points is equivalently difficult to finding their convex hull. Indeed, to compute the volume of a random set of points, it is necessary to decompose it into a set of elementary volumes, the simplest being tetrahedrons. This is exactly what a convex hull of a set of points does. Similarly, finding the volume of a convex set defined by inequalities is equivalently difficult to enumerating its vertices. The convex hull operation is very well implemented in `Qhull` [4] when taking vertices as input. But, we found that its performance when taking inequalities as input is poor, so we used `CDD` [19]. However, `Qhull` was found to be slightly more numerically stable, and can be used in incremental mode, which makes it an attractive fallback. Still, numerical issues are inherent to the double description computations, and in pathological cases we switch to the `PPL` [3] that uses exact integer arithmetic and is much faster than `CDD` in this setting.

5) Proof of convergence:

Lemma 2. *$(\mathcal{P}_{outer})^n$ is monotonous non-increasing.*

Proof. $\mathcal{P}^{n+1}_{\text{outer}} = \text{CUT}(\mathcal{P}^n_{\text{outer}}, plane)$ thus, $\forall a \in \mathcal{P}^{n+1}_{\text{outer}}, a \in \mathcal{P}^n_{\text{outer}}$ and $\mathcal{P}^{n+1}_{\text{outer}} \subseteq \mathcal{P}^n_{\text{outer}}$. □

Lemma 3. *$(\mathcal{P}_{inner})^n$ is monotonous non-decreasing.*

Proof. $\mathcal{P}^{n+1}_{\text{inner}} = \text{CONV}(\mathcal{P}^n_{\text{inner}}, c^*)$ thus, $\forall a \in \mathcal{P}^n_{\text{inner}}, a \in \mathcal{P}^{n+1}_{\text{inner}}$ and $\mathcal{P}^n_{\text{inner}} \subseteq \mathcal{P}^{n+1}_{\text{inner}}$. □

Algorithm 1 Robust polyhedron computation by recursive projection

```
contacts ← the set of contacts
$\epsilon$ ← the approximation difference precision
procedure ROBUST(contacts, $\epsilon$)
    $\mathcal{P}_{\text{inner}}, \mathcal{P}_{\text{outer}}, \nu$ ← INITIALIZE(contacts)
    while $\mathcal{V}(\mathcal{P}_{\text{outer}}) - \mathcal{V}(\mathcal{P}_{\text{inner}}) > \epsilon$ do
        for facet $\in \mathcal{P}_{\text{inner}}$ do
            UVf ← $\nu$[facet]
            if INVALID(UVf) then
                if UVf exists then
                    $\nu$[facet] ← CUT(UVf, plane)
                else
                    $\nu$[facet] ← CUT($\mathcal{P}_{\text{outer}}$, facet)
                end if
            end if
            $volumes$[facet] ← $\mathcal{V}(\nu$[facet]$)$
        end for
        $d \leftarrow normal(\text{argmax}(volumes))$
        $c^* \leftarrow$ OPTIM($d$)
        plane ← PLANE($d, c^*$)
        $\mathcal{P}_{\text{inner}} \leftarrow \mathcal{P}_{\text{inner}} \cup c^*$
        $\mathcal{P}_{\text{outer}} \leftarrow \mathcal{P}_{\text{outer}} \cap$ plane
        INVALIDATE($\nu$, plane)
    end while
    return $\mathcal{P}_{\text{inner}}, \mathcal{P}_{\text{outer}}$
end procedure
```

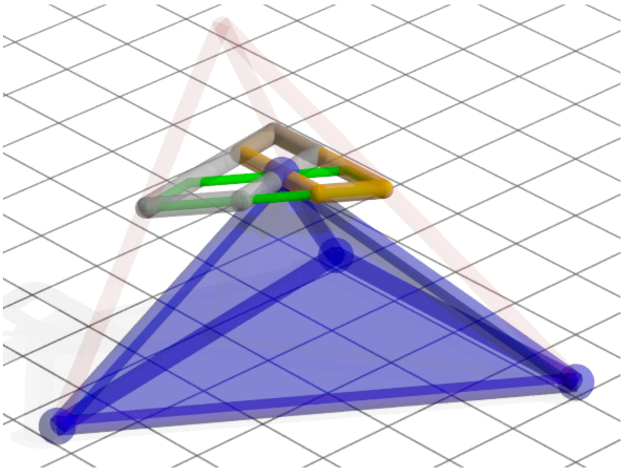

Fig. 2. One iteration of the algorithm: the outer tetrahedron is cut by a plane parallel to its base, forming an inner tetrahedron (transparent blue) and three prismatoids with intersecting trapezoidal bases (thick orange, thin green, and transparent grey)

Lemma 4. $\forall n \in \mathbb{N}, \mathcal{P}^n_{inner} \subseteq \mathcal{P} \subseteq \mathcal{P}^n_{outer}$

Proof. $\forall n \in \mathbb{N}, c^* \in \mathcal{P}$. We denote $\pi_n = \{c | d_n^T c \leq c_n^*\}$ Thus, $\mathcal{P}^n_{\text{inner}} = \text{CONV}(c_i^* | i \in [0 \cdots n]) \subseteq \mathcal{P}$. $\forall n \in \mathbb{N}, \pi_n^c \subseteq \mathcal{P}^c$. Thus $\mathcal{P}^n_{\text{outer}} = \bigcap_{0 \cdots n} \pi_n \supseteq \mathcal{P}$. □

Theorem 3. *The sequences $(\mathcal{P}_{outer})_n$ and $(\mathcal{P}_{inner})_n$ converge towards $\mathcal{P}$.*

Proof. By the Lemmas 2, 3 and 4, the sequences are monotonous and bounded: they converge. Thus, we can consider their limits, $\mathcal{P}^{\infty}_{\text{outer}}$ and $\mathcal{P}^{\infty}_{\text{inner}}$. At this limit, we know that:

- $c^* \in \mathcal{P}_{\text{inner}}\infty$
- $\pi \cap \mathcal{P}_{\text{outer}}\infty = \mathcal{P}_{\text{outer}}\infty$

Thus, in that direction d, $\mathcal{V}(\nu_d) = 0$. As we choose the maximum volume for the stepping direction, $\max_{\texttt{facet}} \mathcal{V}(\nu) = 0$. Then, $\forall \texttt{facet}, \mathcal{V}(\nu_k) = 0$. As the difference between $\mathcal{P}_{\text{outer}}$ and $\mathcal{P}_{\text{inner}}$ is exactly $\mathcal{V}(\bigcup \nu_k)$, $\mathcal{V}(\mathcal{P}^{\infty}_{\text{outer}} \setminus \mathcal{P}^{\infty}_{\text{inner}}) = 0$. As both $\mathcal{P}^{\infty}_{\text{inner}}$ and $\mathcal{P}^{\infty}_{\text{outer}}$ are non-empty convex sets $\overline{\mathcal{P}^{\infty}_{\text{inner}}} = \overline{\mathcal{P}^{\infty}_{\text{outer}}}$. $\mathcal{P}^{\infty}_{\text{outer}} = \left(\bigcup_{i \in \mathbb{N}} \pi^c_n\right)^c$, thus $\mathcal{P}^{\infty}_{\text{outer}}$ is closed, and we have $\overline{\mathcal{P}^{\infty}_{\text{inner}}} = \mathcal{P}^{\infty}_{\text{outer}}$. As the sequence of c^* is included in $\mathcal{P}$, it is bounded, and thus compact. Then, as $\mathcal{P}^{\infty}_{\text{inner}} = \text{CONV}(c^*_n | n \in \mathbb{N})$, $\mathcal{P}^{\infty}_{\text{inner}}$ is also compact, and we get $\overline{\mathcal{P}^{\infty}_{\text{inner}}} = \mathcal{P}^{\infty}_{\text{inner}} = \mathcal{P}^{\infty}_{\text{outer}}$, which entails $\mathcal{P}^{\infty}_{\text{inner}} = \mathcal{P}^{\infty}_{\text{outer}} = \mathcal{P}$ by Lemma 4. □

D. An intersection of prisms

The 3-dimensional shape described in Equation 38 is the intersection of k shapes. We show that each of these shapes is a prism, whose base can be computed as in subsection II-A.

Let us consider $\tilde{g}_i$ and the 3D vector, c_0 that describes the 2D CoM position in the plane $c_z = 0$:

$$c_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} c \tag{41}$$

Let's find which shape is described by the constraints:

$$\begin{aligned} &A_1 f + A_2(g + \tilde{g}_i)c = T(g + \tilde{g}_i) \\ &\|Bf\| \leq u^T f \end{aligned} \tag{42}$$

Note that A_2 is a function of $g_i = g + \tilde{g}_i$, and it is the only part of the equation that affects c. We know that:

$$A_2 c = \begin{bmatrix} 0 \\ -T(mg)c \end{bmatrix} = -m \begin{bmatrix} 0 \\ g \times c \end{bmatrix} \tag{43}$$

Thus, for any $c = \begin{bmatrix} c_x & c_y & c_z \end{bmatrix}^T$:

$$\begin{aligned} T(mg)c &= m \begin{bmatrix} -g_i^z c_y + g_i^y c_z \\ g_i^z c_x - g_i^x c_z \\ -g_i^y c_x + g_i^x c_y \end{bmatrix} & (44) \\ &= m \begin{bmatrix} -g_i^z c_y \\ g_i^z c_x \\ -g_i^y c_x + g_i^x c_y \end{bmatrix} + m \begin{bmatrix} g_i^y c_z \\ -g_x^z c_z \\ 0 \end{bmatrix} & (45) \\ &= T(mg)c_0 + m \begin{bmatrix} g_i^y c_z \\ -g_i^x c_z \\ 0 \end{bmatrix} & (46) \end{aligned}$$

Thus we can show that the solution at any altitude is the translated of the solution at $c_z = 0$. Indeed,

$$T(mg)\left(c_0 - \begin{bmatrix} \frac{g_i^x}{g_i^z} c_z \\ \frac{g_i^y}{g_i^z} c_z \\ 0 \end{bmatrix}\right) = T(mg)c_0 + m \begin{bmatrix} g_i^y c_z \\ -g_i^x c_z \\ 0 \end{bmatrix} = T(mg)c \tag{47}$$

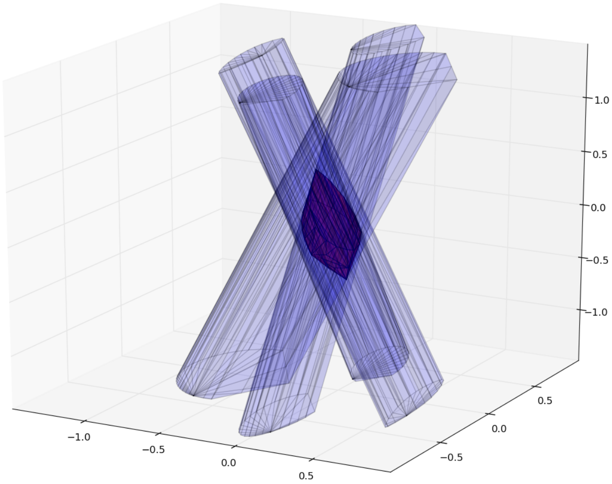

Fig. 3. The robust static stability polyhedron as an intersection of non-right prisms, for a residual radius $3.55\,\mathrm{m\,s^{-2}}$

Thus, for any $\tilde{g}_i$, the shape described by Equation 42 is an infinite prism, whose base can be computed by finding the 2D polyhedron described with a slightly different problem:

$$\begin{aligned} \max_{c_2, f} \quad & d^T c_2 \\ \text{s.t.} \quad & A_1 f + A_2 \phi c_2 = T(g + \tilde{g}_i) \\ & \|Bf\| \leq u^T f \end{aligned} \tag{48}$$

Where c_2 is the 2-dimensional CoM position in the plane $c_z = 0$ and ϕ is a projection matrix:

$$\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^T \tag{49}$$

The axis of this prism is thus given by the ρ_i, colinear to the g_i:

$$\rho_i = \begin{bmatrix} \frac{g_i^x}{g_i^z} & \frac{g_i^y}{g_i^z} & 1 \end{bmatrix}^T = \frac{1}{g_i^z} g_i \tag{50}$$

Then, the resulting intersection of prisms can be easily computed as follows:

1) Compute an approximation of each base using eq. (48) and the recursive projection in subsection II-A.
2) Compute the convex hull of each prism using a fixed height (possibly large) for the prism.
3) Compute the intersection of the prisms. It is the V-rep of the stacked H-rep of each prism.

V. COMPARATIVE RESULTS

We tested the computation on a simple 3-contact scenario, using an acceleration polytope $\mathcal{G}$ with 4 vertices. It is a "lozenge" whose vertices are generated from adding and removing an acceleration g_m to each lateral component of the gravity, that is $g_i = [\pm g_m, \pm g_m, g]$. We use Algorithm 1 with the linearization (subsection III-C): at each step, the problem contains 39 variables; 36 linear constraints that limit the maximum amplitude of each force component; 13 conic

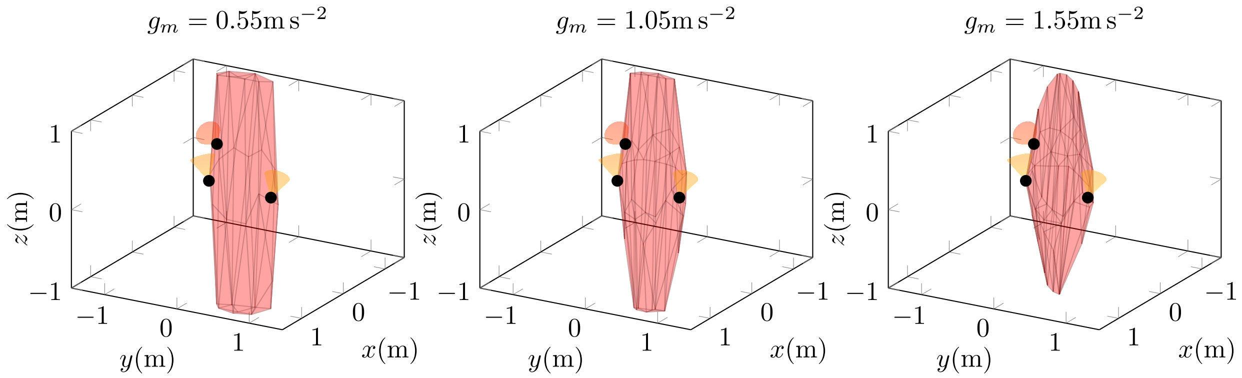

Fig. 4. Comparison of different $\mathcal{P}$ for different acceleration polytopes $\mathcal{G}$ generated by g_m. Only the inner approximations $\mathcal{P}_{\text{inner}}$ are rendered (in red), but are almost superimposed with $\mathcal{P}_{\text{outer}}$. The black dots with yellow cones represent contact points with associated friction cones.

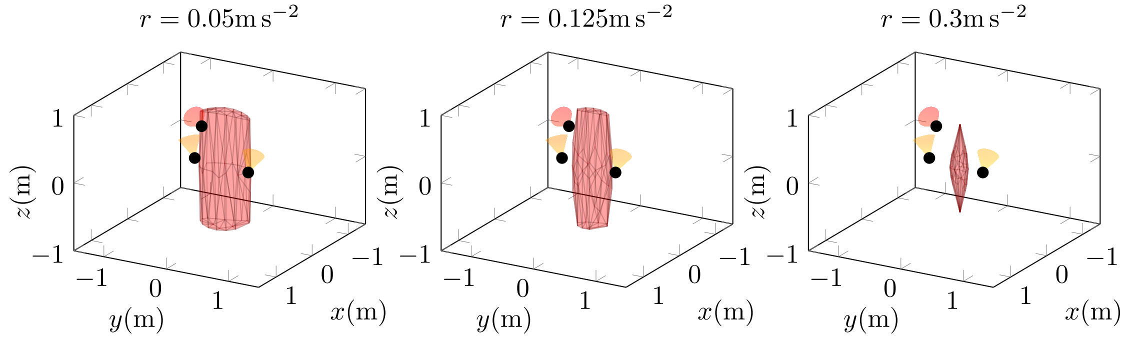

Fig. 5. Comparison of different $\mathcal{P}$ for different acceleration spheres of radius r. Only the inner approximations $\mathcal{P}_{\text{inner}}$ are rendered (in red), but are almost superimposed with $\mathcal{P}_{\text{outer}}$. The black dots with yellow cones represent contact points with associated friction cones.

constraints that enforce friction and limit the CoM to a sphere of unit radius.

The resulting $\mathcal{P}$s are presented in Figure 4. The obtained $\mathcal{P}$s are not right prisms as those of the non-robust case (displayed for reference in Figure 6); they have a diamond-like shape instead. When using the acceleration sphere developed in subsection III-B, the polyhedron shrinks faster, see Figure 5. Indeed, the constraint eq. (28) is isotropic: only the norm of c intervenes. Hence, as the residual radius increases, the polyhedron shrinks in all directions at the same rate. Because eq. (28) uses maximum singular values, this shrinking rate is high. This shows that eq. (28) is far from being tight, and should only be used when fast computation is paramount.

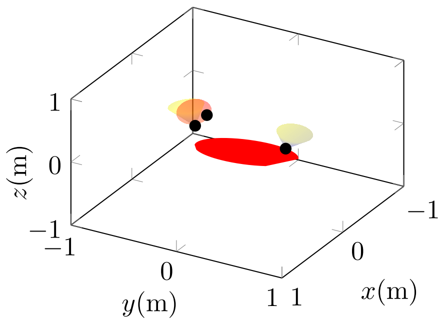

Fig. 6. Stability polygon in 2D (in red). The black dots with yellow cones represent contact points with associated friction cones.

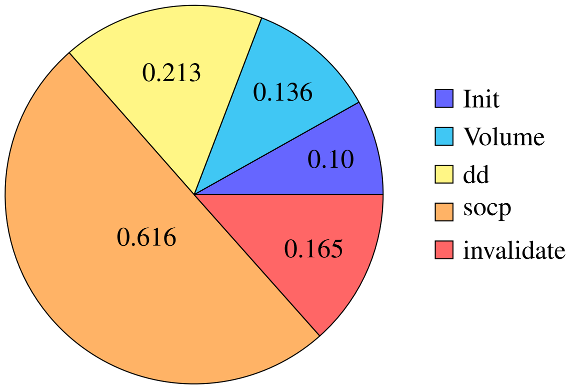

Fig. 7. Computation times for 3 contacts, polytope with 4 vertices and 3 point contacts. 50 iterations, total time 1.29s (clock time, without interpreter set-up and teardown). Final error: $0.043\,\mathrm{m}^3$.

The Algorithm 1 converges in around $1\,\mathrm{s}$ to an acceptable precision of $0.043\,\mathrm{m}^3$ $(1,02\%)$, see Figure 7. The computation time can be split in two halves: (i) solving the optimization problems, and (ii) other operations. This proves that it is very important to limit the number of the double-description operations, that we do in the invalidation function in Algorithm 1.

What is not apparent in Figure 7 is that the computation time increases with the number of iterations. Indeed, as the algorithm progresses, more volumes have to be compared and potentially recomputed to find the best direction.

To assess the convergence characteristics of our algorithm, we devised a worst-case scenario: $\mathcal{P}$ would be a sphere (we took 6 contacts, one per each axis plus a limitation on the CoM or the contact forces). The results in Figure 8 show that the convergence is almost linear in the number of iterations. This does not compare to the nice quadratic result of [9], which is expected because our problem is 3D and not 2D.

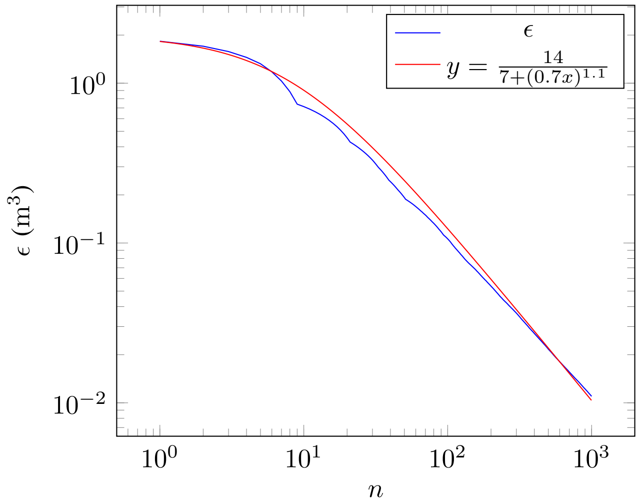

Fig. 8. Precision reached as a function of the number of iterations while approximating a sphere.

It is faster to compute $\mathcal{P}$ using the prism intersection method (subsection IV-D). First, at each iteration we solve k SOCP, each of which is approximately $\mathcal{O}(n^3)$ in the number of variables; whereas in Algorithm 1, we solve at each iteration a single problem whose size grows with k and has a complexity of about $\mathcal{O}((nk)^3)$. Second, we need $\mathcal{O}(\frac{1}{\epsilon})$ iterations to reach the desired precision ϵ in Algorithm 1, while only $\mathcal{O}(\frac{1}{\sqrt{\epsilon}})$ is need using the prism intersection method. Last, with the prism intersection method, we do not need to keep track of the uncertainty volumes ν. In practice, it takes about $0.83\,\mathrm{s}$ to compute the same example as Figure 7 with a similar precision using the prism intersection method; in this case, most of the time is spent solving optimization problems. The intersection computation is not significant. Another advantage in using prisms is its robustness to numerical errors. In the other hand, it is not possible to target exactly a desired precision ϵ. Indeed, we do not know how large the intersection of prisms will be before computing it.

VI. INCREMENTAL PROJECTION FOR TESTING

A. Why incremental projection?

Testing if a particular point c_t is statically stable, can be done by solving the SOCP derived from Equation 1 as follows:

$$\begin{aligned} \max_{f} \quad & 1 \\ \text{s.t.} \quad & A_1 f = T(g) - A_2 c_t \\ & \|Bf\| \leq u^T f \end{aligned} \tag{51}$$

Hence, we would need to solve as many SOCP as there are CoM to be tested. To circumvent this, one can leverage the fact that a CoM position is in (resp. robust) static equilibrium iff it is inside the (resp. robust) static polygon (resp. polyhedron). To do so, a TEST-SAMPLE routine was introduced in [9] for the static case. Their idea is to test if points are inside/outside of the static stability polygone, or fall in-between the inner and outer approximation. In that latter case, it is necessary to refine the approximations until the position of c_t is determined.

The difference between TEST-SAMPLE and the polygon computation in subsection II-A is only in the choice of the search direction: the point being tested is outside one (and only one) of the facets of the inner approximation. Hence, the search direction is chosen perpendicular to that facet.

By doing so, all the points are tested by two matrix multiplications. The result of the latter operation can tell us if points are in or out the stability polygon, otherwise they reside in-between and need refinement. This leads to an interesting observation: we can start with a very rough approximation of the stability polygon and refine it with the tests as needed.

We present how to adapt this methodology to $c_t \in \mathcal{P}$.

B. Polyhedral case

We need to update Algorithm 1 for testing. In 3D, testing $c_t \in \mathcal{P}$ can also be done by two matrix multiplications. However, when we need to update the approximations, c_t may lie outside of *multiple* facets of the $\mathcal{P}_{\text{inner}}$. Randomly selecting a facet gives us a direction. Hence, we get rid of a the bookkeepings related to maintaining a map of the facets to their corresponding uncertainty volumes ν.

Alternatively, the direction can be selected to be perpendicular to the facet of $\mathcal{P}_{\text{inner}}$ forming the greatest uncertainty volume ν containing c_t. This however does not present any significant reduction in the number of iterations (i.e. the number of optimization problems being solved), but adds overheads due to the volume computations.

We thus choose a random selection for its efficiency.

C. Prism case

To test a point while using the prisms intersection method in subsection IV-D, we apply Algorithm 2, derived from Algorithm 1. It takes as input a set of polygons, $\{p_i\}$ that contain an inner and outer approximation. Each of these polygons uses a different $\tilde{g}_i$ and is computed using Equation 48.

For every operation, we do *not* compute the intersection of prisms; we only store the polygons that form their bases.

To test if a point is inside or outside one of the prisms, we compute its oblic projection along the prism axis, ρ_i, on the plane $c_z = 0$. We denote ψ the projection operator:

$$c_2 = \phi(c_t - \rho_i c_t^z) = \phi c_t - \frac{c_t^z}{g + \tilde{g}_i^z} \begin{bmatrix} \tilde{g}_i^x \\ \tilde{g}_i^y \end{bmatrix} \triangleq \psi(\tilde{g}_i) c_t \tag{52}$$

ϕ is defined in Equation 49.

Then we check if c_2 is inside every polygon, or outside any polygon. If it is neither inside nor outside, we refine the appropriate polygons until c_2 is determined. If it can not, within a predefined iterations `maxIter`, we reject it. In fact `maxIter` is reached only when a point is exactly on the boundary. Alternatively, the area of the current triangle can be used as a stopping criteria whenever it becomes very small.

Algorithm 2 Algorithm to test a sample while iteratively improving the underlying polygons

```
procedure TEST-PRISMS($\{p_i\}, \{\tilde{g}_i\}, c_t$, maxIter)
    nrIter ← 0
    while nrIter ≤ maxIter do
        if $c_t \in \cap \{p_i^{\text{inner}}\}$ then
            return true, $\{p_i\}$
        else if $c_t \notin \cup \{p_i^{\text{outer}}\}$ then
            return false, $\{p_i\}$
        else
            for $p_i, \tilde{g}_i \in (\{p_i\}, \{\tilde{g}_i\})$ do
                $c_2 \leftarrow \psi(\tilde{g}_i) c_t$
                if $c_2 \in p_i^{\text{outer}}$ and $c_2 \notin p_i^{\text{inner}}$ then
                    $p_i \leftarrow$ TEST-SAMPLE($p_i, c_2$)
                end if
            end for
        end if
        nrIter++
    end while
    return false, $\{p_i\}$
end procedure
```

D. Results

We use the examples defined in section V for testing one million uniformly sampled random points in a box of dimensions $1\,\mathrm{m} \times 1\,\mathrm{m} \times 3\,\mathrm{m}$. We used the incremental version of Algorithm 1 presented in subsection VI-B. The results are illustrated in Figure 9. Note that as the number of query points increases, the number of the iterations –to determinate whether they are robustly stable or not– increases but at a sub-linear rate. The horizontal lines represent the number of iterations necessary to reach a certain precision ϵ using Algorithm 1.

For testing, it is faster to use the polyhedral version rather than the prism intersection one. Indeed, when testing many samples, the performance bottleneck is no longer the number of iterations, but rather the time necessary to test a sample. In the polyhedral case, two matrix multiplications at most are necessary to test a point, while in the prism intersection, at most $2k$ multiplications are necessary. Moreover, to test a point in the prism intersection, it is necessary to refine all polygons in which the projected point lands between the inner and outer approximation until it is outside of at least one of them or all of them contain it.

On a mono-core, Python implementation, it takes about $40\,\mathrm{s}$ to test one million points, but this could of course be greatly improved using parallelization. Parallel computing is impractical to build $\mathcal{P}$ in Algorithm 1 because each iteration will modify the approximations, but in the testing case, most samples can be tested independently.

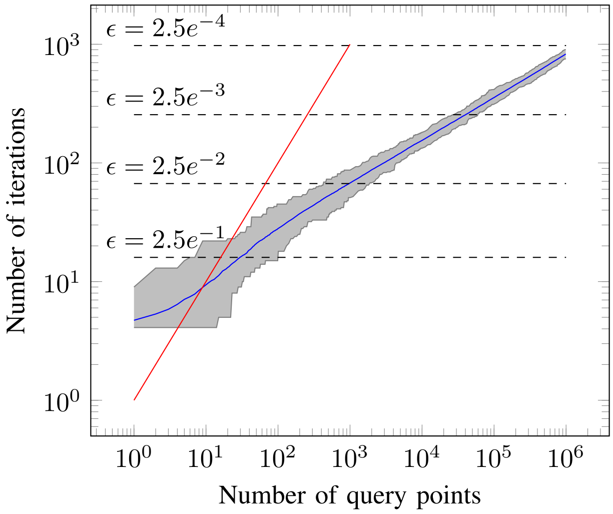

Fig. 9. Number of iterations required to test random points compared to computing directly

VII. EXTENSIONS AND DISCUSSIONS

A. Morphing and change in robustness

We know how to compute $\mathcal{P}$ corresponding to a given $\mathcal{G}$, which allows us to known which CoM positions are robust w.r.t this acceleration polytope.

Now, having $(\mathcal{G}_0, \mathcal{P}_0)$ and $(\mathcal{G}_1, \mathcal{P}_1)$, can we compute $\mathcal{P}_\lambda$ from $\mathcal{G}_\lambda = \lambda_0 \mathcal{G}_0 + (1-\lambda)\mathcal{G}_1$, $\lambda \in [0,1]$?

Both $\mathcal{P}_0, \mathcal{P}_1$ are the projection of a high-dimensional convex shapes that lives in the manifold of the contact forces and the CoM position. Thus, deforming $\mathcal{G}$ induces non-linear changes that are projected in the 3-dimensional space.

We propose to use interpolation (linear morphing) between convex polyhedrons to approximate changes in the acceleration polytope $\mathcal{G}_\lambda$. Hence, we consider scaling of the acceleration polyhedron, i.e. changes in the residual radius r but not in the linearization. To compute $\mathcal{P}_\lambda$, we use the morphing algorithm presented in [28]. Although the latter presents strategies to morph non-convex polyhedrons, at its core the algorithm is a 5-step process to find a common topology of two convex polyhedrons:

1) Get two polyhedrons (vertices and associated topology) and translate them to barycentric coordinates. Project them on the unit sphere.
2) Using neighbouring information to limit the search space, find every intersection of all spherical edges of the projections.
3) Order the intersections according to topological information and find in which facet of one polyhedron lie the vertices of the other.
4) For each vertex on the unit sphere, use barycentric coordinates to find its position on the original polyhedron.
5) Output the combined topology.

Then, we obtain intermediate polyhedrons by linearly interpolating between matching topologies. The Figure 10 shows an example of the results obtained by this algorithm.

We also compare the volumes obtained in Figure 4 with those resulting from the morphing. To compare the volumes,

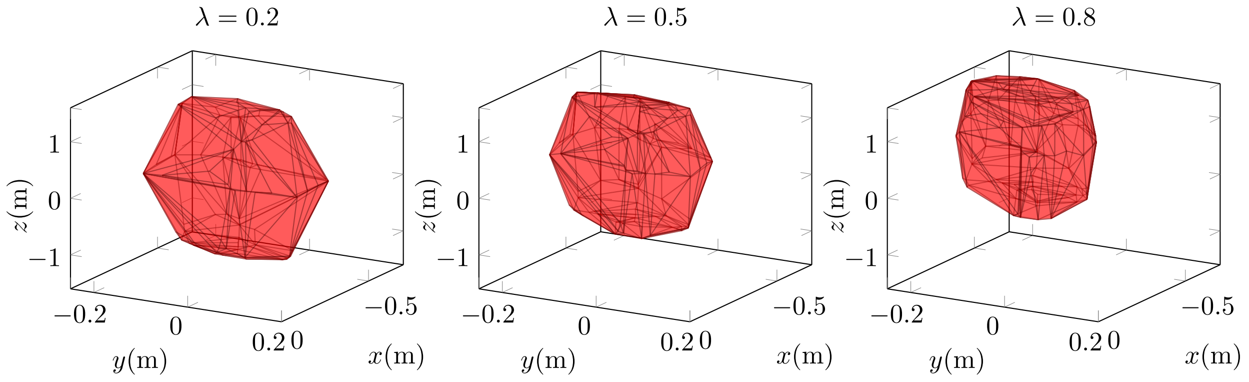

Fig. 10. Morphing between two different robust static stability polyhedrons.

we select an interval $[r_l, r_u]$ and a number of percentages λ. For each point we compute the exact polyhedron $\mathcal{P}_e$ corresponding to the discretization of the robust sphere of radius $r = (1-\lambda)r_l + \lambda r_u$ and the interpolated polyhedron at λ, $\mathcal{P}_\lambda$. We can then compute an error metric, δ, such that:

$$\delta = \frac{\mathcal{V}(\mathcal{P}_e) + \mathcal{V}(\mathcal{P}_\lambda) - 2\,\mathcal{V}(\mathcal{P}_e \cap \mathcal{P}_\lambda)}{\mathcal{V}(\mathcal{P}_e) + \mathcal{V}(\mathcal{P}_\lambda)} \tag{53}$$

This metric is the volume of the symmetric difference $\mathcal{P}_e \ominus \mathcal{P}_\lambda$ normalized by the sum of volumes. We show in Figure 11 how δ evolves as a function of r, given different interpolation keypoints. The key main result is that morphing is a good fit (less than $12\,\%$) error as long as the distance $r_u - r_l$ between keypoints is less than $2\,\mathrm{m\,s^{-2}}$. In this case, the problem becomes infeasible above $r = 4.0\,\mathrm{m\,s^{-2}}$ meaning that computing 3 keypoints is enough to cover the whole acceptable range of residual radiuses. However, a drawback of this morphing is that the obtained polyhedron is *bigger* than the actual one, which may lead to false positives when testing for robust stability.

B. Robust 2D and robust 3D

We present a serious limitation of the 2-dimensional robust approaches [36], [39]: the robust static region is a 3-dimensional polyhedron, but they consider a 2-dimensional region, and then extend it to a right prism. We show that this approximation is a poor fit of the real volume, that only gets worse as the residual radius increases.

In order to compare our method to 2-dimensional approaches, we compare the volumetric computation to right (vertical) prisms whose bases are polygons. The right prism will have the same height as the diameter of the limiting sphere we use to bound the CoM accessible region section V.

We first ensure that the volume that we compute is always included in the prism formed by the non-robust static stability polygon. We then compute a “robust static stability polygon” at height h, i.e. the recursive projection defined by:

$$\Phi F + \Psi \left(\phi c_2 + \begin{bmatrix}0 & 0 & h\end{bmatrix}^T\right) = \Gamma \tag{54}$$

$$||\Lambda F|| \leq \Upsilon F \tag{55}$$

where c_2 is a 2D vector, ϕ is defined in Equation 49.

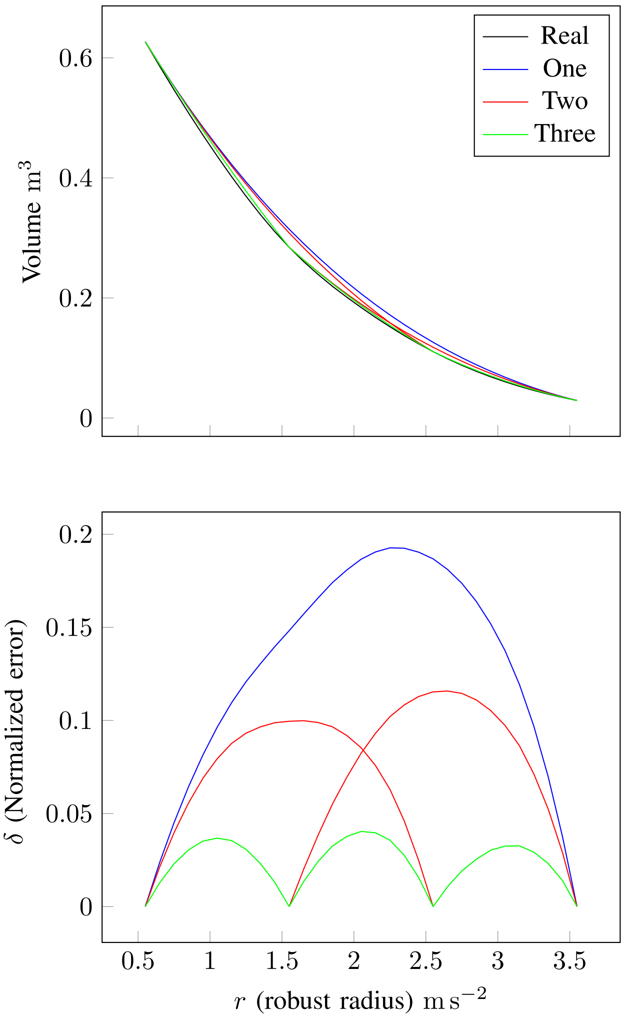

Fig. 11. Comparison of morphing with direct computation for different keypoints.

We use our previous examples, for 20 residual radiuses linearly spaced in the interval $[0.55, 3.55]$, and for five different heights we compute both forms (right prism and $\mathcal{P}$). We then compute the error metric δ between the right prism and the 3D volume at each residual radius. We ensure that both computations lead to the same result by comparing the intersection of $\mathcal{P}$ and the plane $c_z = h$ with the robust static stability polygon (2D). We use the same error metric in eq. (53), replacing the volume with the area. An example of the objects being compared is in Figure 12 while results are shown in Figure 13. The area error remains low (less than $3\,\%$) at any margin while the volume error shoots towards $100\,\%$.

This means that using a 2D approach is equivalent to approximating the 3D polyhedron by a prism formed by its section at some constant height $c_z = h$. This leads to a poor approximation of the real volume because it is *not* similar to a prism, and this similarity only diminishes with the increase in the stability margin. Note that at any margin the robust 3D polyhedron is *not* included in the prism formed by the "robust polygon" so the "robust prism" is neither a conservative nor optimistic approximation.

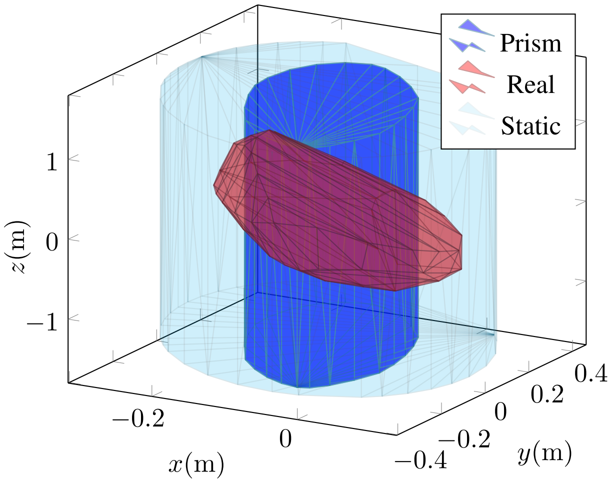

Fig. 12. Comparison between: the statically stable prism in light blue, the robust static polyhedron with a $\mathcal{G}$ generated using $g_m = 2.6\,\mathrm{m\,s^{-2}}$ in red and the prism formed by the robust polygon at the same margin and height $0.1\,\mathrm{m}$ in dark blue. Note that the scale is non-uniform across axes.

C. Recent work

Our approach is in a way the dual of the that shown in [12]: instead of limiting the CoM positions to a known convex polytope and finding the envelope of acceptable CoM accelerations, we propose to limit the CoM acceleration and find all acceptable CoM positions.

It is also more general than what is presented in [39]: they only consider errors on the contact forces without considering that the resulting acceleration could change.

Very recent works, developed in parallel to our approach have pointed at ways to compute the proposed region: [35] showed that when the CoM acceleration is not null, the CoM accessible region, nicknamed CFR was a slanted prism, while [15] has shown how to use a polyhedral or ellipsoidal region, but this time to limit the CoM position.

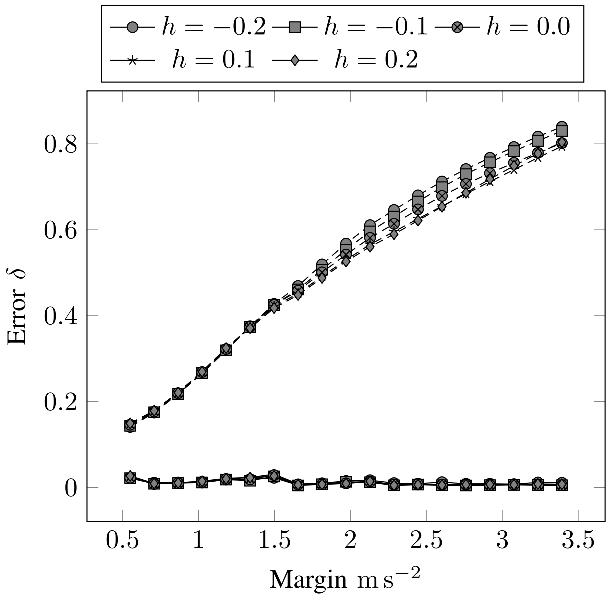

Fig. 13. Compare 2D robust and 3D robust at different heights (in meters). Dashed lines represent volumes error, while solid lines represent areas error.

VIII. CASE STUDY IN MULTI-CONTACT POSTURE GENERATION

A. Posture generation

We use the posture generation framework (PG) developed in [10], [11] and add an extra constraint to maintain the CoM in the robust static stability polyhedron. The core of the PG is to solve a non-linear optimization problem in the generalized robot coordinates q, and the contact forces f. We can thus add our constraint by specifying:

$$Hc(q) \leq b \tag{56}$$

Where H and b represent the H-rep of the robust static polyhedron. Other constraints encode:

- Contacts fixed to pre-defined locations;
- Forces in friction cones;
- Both torque and position limits for every joint;
- Self-collision constraints.

The objective function is the distance to a usual posture.

It is also possible to extend the state of the posture generator to (q, F) and directly encode the robust stability by specifying that for every $f_i \in F$:

$$A_1(q)f_i + A_2(\tilde{g}_i)c(q) = T(\tilde{g}_i) \tag{57}$$

This is beneficial for the contact positions to depend on q and thus be automatically determined. However, dimensionality remains a problem in non-linear programming, and extending the state increases drastically the computation times. Therefore, we only only consider the effects of adding a constraint of the form eq. (56).

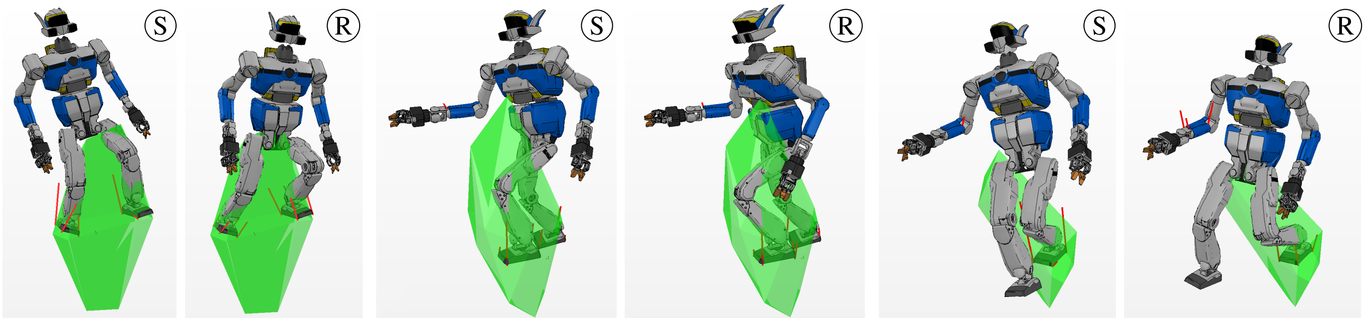

Fig. 14. Results obtained using posture generation. From left to right, both feet, feet and right forearm, left foot and right forearm, each pair is first non-robust, labelled S, then robust, labelled R.

B. Results

We generated three random contact positions: one for each foot and one for the right arm. Then we computed three pairs of postures using:

- Both feet;
- Both feet and the arm;
- The left foot and the arm.

In each pair, the first posture is generated using the static stability constraint while the second one is generated using the same problem plus a constraint of the form eq. (56). The results are shown in Figure 14. They qualitatively highlight a problem with the regular static stability: the solver has a tendency to stop on the edge of the constraint which is an *unstable* equilibrium position. The postures generated with the robust static stability look more resilient to external perturbations.

IX. CONCLUSION

In this paper, we have thoroughly explored the notion of robust static stability polyhedron: its nature and properties, how to compute it and how to use it for testing robust static equilibrium. We have also shown how to approximate variations in the residual radius with polyhedral morphing.

An interesting property is that it is no longer a right prism: approximating this shape by simply shrinking the static stability polygon is not correct as the height of the CoM now plays a role.

However, a lot of work remains in perspective. We first have to integrate this computation in our planning and control frameworks. We believe that adding the notion of robustness margins will greatly increase the quality of planning and will make our control more resilient to external perturbations. In both cases the location of the contacts will vary by small increments: it would be paramount to find a way to recompute, in a few milliseconds, a new polyhedron based on the contact displacement.

In general, the computation times needs to be improved and it would be interesting to better exploit the particular characteristics of the problem to compute it faster. Indeed, the high-dimensional polyhedron whose projection we compute is a cartesian product of cones cut by a 6-dimensional hyperplane, but we only leverage the fact that it is a convex shape. Similarly, it could prove highly beneficial to parallelize some parts of our algorithm, most notably during sampling.

Finally, we have shown how to perform iterative sampling, but this opens a new venue of research: we can choose the search direction according to the problem rather than always choosing the optimal direction. This would allow us to only refine (or even only compute) the parts of the robust stability polyhedron that are relevant to the problem being solved.

REFERENCES

[1] H. Audren, A. Kheddar, and P. Gergondet. Stability polygons reshaping and morphing for smooth multi-contact transitions and force control of humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, pages 1037–1044, Cancun, Mexico, Nov. 2016.

[2] H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida. Model preview control in multi-contact motion– application to a humanoid robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4030–4035, 2014.

[3] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[4] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[5] S. Barthélemy and P. Bidaud. Stability measure of postural dynamic equilibrium based on residual radius. *Advances in Robot Kinematics: Analysis and Design*, pages 399–407, 2008.

[6] M. W. Bern, D. Eppstein, L. J. Guibas, J. Hershberger, S. Suri, and J. Wolter. The centroid of points with approximate weights. In *Proc. ESA*, volume 979, pages 460–472, 1995.

[7] K. Bouyarmane and A. Kheddar. Humanoid Robot Locomotion and Manipulation Step Planning. *Advanced Robotics*, 26(10):1099–1126, 2012.

[8] D. Bremner, K. Fukuda, and A. Marzetta. Incremental Convex Hull Algorithms Are Not Output Sensitive. *Discrete & Computational Geometry*, 21(1):57–68, 1999.

[9] T. Bretl and S. Lall. Testing static equilibrium for legged robots. *IEEE Transactions on Robotics*, 24(4):794–807, 2008.

[10] S. Brossette, A. Escande, G. Duchemin, B. Chretien, and A. Kheddar. Humanoid posture generation on non-euclidean manifolds. In *IEEE-RAS 15th International Conference on Humanoid Robots*, pages 352–358, Nov 2015.

[11] S. Brossette, A. Escande, and A. Kheddar. Multi-Contact Postures Computation on Manifolds. *IEEE Transactions on Robotics*, Submitted, under review, December 2016.

[12] S. Caron and A. Kheddar. Multi-contact walking pattern generation based on model preview control of 3d com accelerations. In *IEEE-RAS International Conference on Humanoid Robots*, pages 550–557, Cancun, Mexico, November 2016.

[13] S. Caron, Q.-c. Pham, and Y. Nakamura. Leveraging Cone Double Description for Multi-contact Stability of Humanoids with Applications to Statics and Dynamics. In *Robotics: Science and Systems*, 2015.

[14] S. Caron, Q.-C. Pham, and Y. Nakamura. Zmp support areas for multi-contact mobility under frictional constraints. *IEEE Transactions on Robotics*, 2017.

[15] H. Dai and R. Tedrake. Planning robust walking motion on uneven terrain via convex optimization. In *IEEE-RAS International Conference on Humanoid Robots*, pages 579–586, Cancun, Mexico, 2016.

[16] A. Escande, A. Kheddar, and S. Miossec. Planning contact points for humanoid robots. *Robotics and Autonomous Systems*, 61(5):428–442, 2013.

[17] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer US, Boston, MA, 2008.

[18] J. Fourier. Solution d'une question particulière du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomathique de Paris*, pages 317–319, 1826.

[19] K. Fukuda and A. Prodon. Double description method revisited, 1996.

[20] E. Garcia, J. Estremera, and P. G. de Santos. A comparative study of stability margins for walking machines. *Robotica*, 20(06):595–606, 2002.

[21] A. Goswami. Postural stability of biped robots and the foot-rotation indicator (fri) point. *The International Journal of Robotics Research*, 18(6):523–533, 1999.

[22] M. Henk, J. Richter-Gebert, and G. M. Ziegler. Basic properties of convex polytopes. In *Handbook of Discrete and Computational Geometry*, pages 243–270. CRC Press, Boca, 1997.

[23] H. Hirukawa, S. Hattori, K. Harada, S. Kajita, K. Kaneko, F. Kanehiro, K. Fujiwara, and M. Morisawa. A universal stability criterion of the foot contact of legged robots - Adios ZMP. *IEEE International Conference on Robotics and Automation*, pages 1976–1983, June 2006.

[24] S.-H. Hyon and G. Cheng. Disturbance rejection for biped humanoids. In *IEEE International Conference on Robotics and Automation*, pages 2668–2675, April 2007.

[25] M. Joswig and N. Takayama, editors. *Algebra, Geometry and Software Systems*, chapter Beneath-and-Beyond Revisited, pages 1–21. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[26] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1620–1626, 2003.

[27] J. E. J. Kelley. The Cutting-Plane Method For Solving Convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[28] J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *ACM Transactions on Computer Graphics (SIGGRAPH)*, pages 47–54, New York, NY, USA, 1992. ACM.

[29] R. Mattikalli, D. Baraff, and P. Khosla. Finding All Stable Orientations of Assemblies with Friction. *IEEE Transactions on Robotics and Automation*, 12(2):290–301, 1996.

[30] R. McGhee and A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, 1968.

[31] H. Minkowski. Allgemeine lehrsätze über die convexen polyeder. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, pages 198–219, 1897.

[32] H. Mosemann, F. Röhrdanz, and F. M. Wahl. Stability analysis of assemblies considering friction. *IEEE Transactions on Robotics and Automation*, 13(6):805–813, 1997.

[33] T. Motzkin. *Beitrage zur theorie der Linearen Ungleichungen*. PhD thesis, Universität Basel, 1936.

[34] F. Nori, S. Traversaro, J. Eljaik, F. Romano, A. Del Prete, and D. Pucci. iCub whole-body control through force regulation on rigid non-coplanar contacts. *Frontiers in Robotics and AI*, 2(March):1–18, 2015.

[35] S. Nozawa, M. Kanazawa, Y. Kakiuchi, K. Okada, T. Yoshiike, and M. Inaba. Three-dimensional Humanoid Motion Planning Using COM Feasible Region and its Application to Ladder Climbing Tasks. In *IEEE-RAS International Conference on Humanoid Robots*, pages 49–56, Cancun, Mexico, 2016.

[36] Y. Or and E. Rimon. Computation and Graphical Characterization of Robust Multiple-Contact Postures in Two-Dimensional Gravitational Environments. *The International Journal of Robotics Research*, 25(11):1071–1086, 2006.

[37] Y. Or and E. Rimon. Characterization of Frictional Multi-Legged Equilibrium Postures on Uneven Terrains. *The International Journal of Robotics Research*, submitted, 2016.

[38] D. E. Orin, A. Goswami, and S. H. Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2-3):161–176, 2013.

[39] A. D. Prete, S. Tonneau, and N. Mansard. Fast Algorithms to Test Robust Static Equilibrium for Legged Robots. In *IEEE International Conference on Robotics and Automation*, 2016.

[40] T. Saida, Y. Yokokohji, and T. Yoshikawa. FSW (feasible solution of wrench) for multi-legged robots. *IEEE International Conference on Robotics and Automation*, pages 3815–3820, 2003.

[41] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(5):630–637, Sept 2004.

[42] L. Sentis, J. Park, and O. Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *IEEE Transactions on Robotics*, 26(3):483–501, jun 2010.

[43] M. Vukobratović and B. Borovac. Zero-Moment Point — Thirty Five Years of Its Life. *International Journal of Humanoid Robotics*, 01(01):157–173, 2004.

[44] P. M. Wensing, G. Bin Hammam, B. Dariush, and D. E. Orin. Optimizing Foot Centers of Pressure Through Force Distribution in a Humanoid Robot. *International Journal of Humanoid Robotics*, 10(03):1350027, 2013.

[45] H. Weyl. Elementare theorie der konvexen polyeder. *Commentarii Mathematici Helvetici*, 7:290–306, 1934.

Hervé Audren received the MS degree in science and executive engineering from École des Mines de Paris, France, in 2013. He is currently working toward the PhD Degree in Robotics with the CNRS-UM Laboratory of Informatics, Robotics, and Microelectronics, Montpellier, France (LIRMM) and the CNRS-AIST Joint Robotics Laboratory, Tsukuba, Japan (JRL). His research interests encompass whole-body motion and planning, numerical optimization and human-robot interaction.

Abderrahmane Kheddar received the BSCS degree from the Institut National d'Informatique (ESI), Algiers, the MSc and PhD degrees in robotics, both from the University of Pierre and Marie Curie, Paris 6. He is presently Directeur de Recherche at the CNRS. He is the Codirector of the CNRS-AIST Joint Robotic Laboratory (JRL), UMI3218/RL, Tsukuba, Japan; and leader of the Interactive Digital Humans (IDH) team at CNRS-UM2 LIRMM at Montpellier, France. His research interests include humanoid robotics, haptics, and recently thought-based control using brain machine interfaces. He is a founding member of the IEEE/RAS chapter on haptics (acting also as a senior advisor), the co-chair and co-founding member of the IEEE/RAS Technical committee on model-based optimization, and a member of the steering committee of the IEEE Brain Initiative. He is presently Editor of the IEEE Transactions on Robotics, the Journal of Intelligent and Robotic Systems, and Frontiers in Bionics; he is a founding member of the IEEE Transactions on Haptics and served in its editorial board during three years (2007-2010), he also served as associate editor in the MIT Press Presence journal. He is a member of the National Academy of Technologies of France (NATF), knight in the National Order of Merit, and a Senior Member of the IEEE Society.