Stability polygons reshaping and morphing for smooth multi-contact transitions and force control of humanoid robots

Hervé Audren, Abderrahmane Kheddar, Pierre Gergondet

To cite this version:
Stability polygons reshaping and morphing for smooth multi-contact transitions and force control of humanoid robots

Hervé Audren¹, Abderrahmane Kheddar¹,² and Pierre Gergondet¹

Abstract—In this paper we provide hindsight on how to use the information provided by the explicit computation of the stability polygon and build on task-based QP controllers to achieve both stable multi-contact smooth transitions and force control without jeopardizing stability. This entails computing stability polygons with unilateral and bilateral contacts and devising a method to continuously constrain the CoM position throughout multi-contact motion in order to effectively regulate the forces applied on the environment while avoiding hard constraints on the CoM and discontinuity between stances.

Index Terms—Body balancing, Locomotion, Humanoid Dynamics

I. INTRODUCTION

Multi-contact technology [1], [2] can be used in several tasks such as climbing stairs using handrails or egress/ingress a car, or more generally access confined spaces. While several works have proposed efficient regulation control strategies in multi-contact configurations [3]–[6], transitions between contact stances is still to be researched¹.

One can enforce stability by controlling the Center of Pressure (CoP) at each contact to remain within the convex hull of its area [5]–[9]. This criterion can be added to the QP controller as a constraint. Yet, in these implementations, the CoM was given as a high-level task, and the CoP or contact forces were regulated often in the null-space or in conjunction with the CoM task. Moreover, CoP are ill-conditioned upon contact removal because they only exist when the normal contact force is non-zero. All papers using CoP treat this issue with an ad-hoc counter-measure.

In order to smoothly transition from contacts, we want to specify only a desired force profile on a given contact without endangering stability. Whole body controllers typically solve an optimization problem that aims at finding computable joint accelerations, \( \tilde{\dot{q}} \) and contact forces \( \lambda \) minimizing a set of task errors under constraints. Stability objectives typically only depend on the current joint state (position, velocity and acceleration) \( q, \dot{q}, \ddot{q} \). Thus, adding a task to minimize the 2-norm of \( \lambda \) will not result in modification of angular values: the best local solution is redistributing the internal torques to fall as little as possible.

One usual way to circumvent the local-view issue is to use a preview controller to enhance the motion with a flavor of dynamics [8], [10], but slow multi-contact motion can be achieved without preview [2], using only a closed-loop local task-based controller.

To ensure that the desired force is indefinitely sustainable, we want the controller to move the CoM to a stable region. However, this approach is not directly applicable as the CoM position only depends on \( q \): a task error \( v(q, \lambda) \) cannot be turned into a quadratic objective \( T \) that depends only on \( \dot{q} \) and \( \lambda \). Indeed, to do so one must differentiate \( v \) with respect to time, but \( \lambda \) is our optimization variable.

Forcefully setting the CoM to a precise location will require to set gains that will interfere with almost all the other tasks, such as end-effectors trajectories. This is generally a very conservative measure. Thus, we may want the CoM objective to be among the tasks to be achieved at best when this is needed, but it should be allowed to freely move in a known region of stability, that can be a hard constraint.

In this study, we will limit our developments to multi-contact with quasi-static motions. Hence, constraining the CoM to remain within the frictional static stability polygon is an acceptable criterion (ZMP assumes infinite friction at contact). The static stability is a polygon and allow us to have a better comprehension before moving to the general 3D case in future work. Yet this shape is neither modified according to desired contact forces nor continuous across contact changes. For the latter point, including it directly as a constraint in our QP controller would lead to its immediate violation upon contact transition.

To deal with the latter drawbacks, our main contribution is to propose a novel continuation method based on optimal matching of points to interpolate between convex polygons (those resulting from multi-contact stability region computation). This allows us to smoothly reshape the region of stability during contact removal and addition and hence make QP controllers robust to contact changes.

By formulating a pure CoM based constraint, we effectively decouple the CoM placement from the force regulation problem which allows us to perform force control in multi-contact. Indeed, we can deform the static stability polygon obtained from the geometrical and frictional information of a stance to take into account the desired contact forces. Then, using a simple impedance control scheme, we regulate the applied force to the objective one without endangering stability. This is another novel contribution, as we do not move the CoM to a predetermined location, but inside a formally defined polygon that guarantees stability with respect to a certain force limitation. We note that our work applies to torque and position controlled robots, but it has more impact and value for the latter ones as redistributing...
internal torques has no effect on them.  

We recall in section II the static stability polygon computation, then how we included it as a constraint our controller in section IV. We introduce our idea to smoothly constrain the CoM in section III before applying it to the stairs case in section V. We present a way to efficiently combine tasks with this constraint in section VI and present results obtained in another multi-contact scenario in section VII.

II. COMPUTATION OF THE STABILITY POLYGON

We use the algorithm in [11] to compute the stability polygon for static postures provided by the multi-contact planner. The exact stability polygon lies between an inner and an outer approximation, that are build iteratively from solving a series of second-order cone programs of the form:

$$\max_{\vec{c}_{\text{com}}} d^{T} \vec{c}_{\text{com}} \quad \text{s.t.} \quad A_{1}x + A_{2}\vec{c}_{\text{com}} = t$$

The solution $\vec{c}_{\text{com}}^{*}$ of this problem is an extremal stable CoM position in the direction $d$ i.e. there exists a set of forces $x$ realizing $t = [mg \quad 0]^{T}$ under the non-linearized friction cones constraints defined by $B$ and $u$. This wrench is composed of the total wrench generated by a set of forces $x$, at the origin, computed using the transformation matrix $A_{1}$ and the gravity wrench computed with $A_{2}$ given a CoM position $\vec{c}_{\text{com}}$. For each of these problems, $\vec{c}_{\text{com}}^{*}$ is added to the inner approximation, while the half-plane defined by $\{\vec{c}_{\text{com}} \in \mathbb{R}^{2} | d^{T} \vec{c}_{\text{com}} > d^{T} \vec{c}_{\text{com}}^{*}\}$ is added to the outer approximation. At each step, the difference between these two approximations is a set of triangles. The search direction for the next step is perpendicular to the edge of the inner approximation forming the triangle of maximum area. The algorithm stops when the difference between the outer and inner approximations area is less than a given precision $\sigma$. The authors prove that there is a strict upper bound on the number of iterations needed to reach $\sigma$.

In its original expression, this method is not applicable to the case of mixed unilateral-and-bilateral contacts, that we are using in our experiments [2]. The support region may be unlimited in some directions. Thus, we need additional constraints limiting the search region. As our model of bilateral contacts is simply a set of points with non-aligned friction cones, we still can use Equation 1 providing additional constraints on the acceptable CoM positions. Accordingly, we can either use a constraint $|\vec{c}_{\text{com}}| < r$ that will constrain the CoM to a rectangular (polygonal) region or a conic constraint of the form $|\vec{c}_{\text{com}}| < r$ that will constrain the CoM to a circle. A circular constraint is difficult to approximate by a polygon. However, it may be closer to the real geometric constraint: using a maximum limb length per contact, we can write a series of constraints $|\vec{c}_{\text{com}} - l| < r$ that ensure that the CoM never goes further than $d$ from the contact at position $l$.

The stability region is unlimited is some directions because we can apply arbitrary forces on opposite contact points, allowing us to compensate for any momentum. In fact, the robot cannot apply infinite torques. Translating the torque constraint into a precise force limit of the form $|x| < f_{m}$ requires setting $f_{m}$ from the robot posture, whereas we reason on the CoM model to compute the stability polygon. Instead, we use a reduced torque constraint, $|T_{z}| < \tau_{m}$ where $T_{z}$ represents the cross product between the contact points and the contacting link. In this case, $\tau_{m}$ can be obtained from the characteristics of the link that is in contact (e.g. gripper actuators or feet/ankle actuators). This constraint is linear because we consider non-sliding contacts.

This method is quite fast but not enough for real-time. It takes few seconds for over a hundred iterations. To solve the second-order cone programs, Equation 1, we use the convex optimisation package CVXOPT [12] that natively supports conic constraints. Our implementation is sub-optimal as we recompute the area spanned by every edge at every iteration but the cost for solving each problem is by far the most consuming task. This means that even if we curb the time spent finding the next best edge, the overall runtime of the algorithm will not be significantly improved.

III. MORPHING STABILITY POLYGONS

Now, we are able to compute the stability region for each contact configuration. It is a polygon that can be reshaped in a more conservative form when needed. But more importantly, the static stability can be integrated in our controller as a hard constraint through ‘CoM in Polygon’ equations. When the robot changes contact configuration, the stability polygon shape for the following configuration is different from the current one. We do not want to reset the problem or to design a specific CoM task. Instead, we would like the constraints to continuously deform from a stability shape to the other while subscribing the CoM all along the deformation. In this case, the transition from a contact configuration to another is made smoothly while keeping hard constraints on the stability.

If we do not enforce the CoM to be maintained within a moving/morphing polygon, the QP controller might often start from an infeasible constraint set and fails. Controller failure is solved by our new Algorithm 1 to smoothly morph the stability polygons between stances and this is done within the QP controller.

![Fig. 1: Pie chart showing as angular sections the repartition of computation time for 150 iterations: total time 1.87 s](image-url)
Algorithm 1 Interpolation of two polygons : \( f(P_s, P_d, \kappa) \)

Input: \( P_s, P_d \) start and destination polygons, \( \kappa \) percentage
\( a^{cur}, \text{done} \leftarrow \{\}, \{\} \)

\[ \text{while } |P_s| < |P_d| \text{ do} \]
\[ P_s \leftarrow P_s \cup \text{midpoints}(P_s) \]

\[ \text{end while} \]

\[ M \leftarrow \text{zeros}(|P_s|, |P_d|) \]

\[ \text{for } a^{tar} \in P_d \text{ do} \]
\[ \text{for } b \in P_s \text{ do} \]
\[ M_{ij} = ||a^{tar} - b||_2 \]

\[ \text{end for} \]

\[ \text{end for} \]

\[ a^{cur}, \text{done} \leftarrow \text{Munkres}(M) \]

\[ \text{for } b \in P_s \text{ - done do} \]
\[ a^{tar} \leftarrow P^\perp(b, P_d) \]

\[ \text{end for} \]

\[ \text{conv} (\text{interpolate}(a^{cur}, a^{tar}, \kappa)) \]

It is not necessary to use state-of-the-art 2D or 3D shape morphing found in the computer graphics and animation community, see e.g. [13], [14] because in this study we only morph 2D convex polygons. Furthermore, as we are interested in adding and removing contacts one at a time, successive stability polygons have a non-empty intersection. In particular, when adding a contact, the previous stability polygon is entirely included in the new one. Thus, our successive polygons will be both convex and have non-empty intersections, which allows us to use a simpler, faster, morphing algorithm. Morphing polygons is typically done in three steps:

- Adding points to the shapes;
- Finding a correspondence between the points from the start and target shape;
- Generating intermediate shapes by interpolating each of these couples.

The main difficulty when morphing polygons is that if corresponding points are badly chosen, the resulting polygon will self-intersect during interpolation.

To solve the problem of self-intersection during morphing, we use the Munkres (Hungarian) algorithm [15]. This algorithm solves the assignment problem, i.e. finds a minimum weight matching in a weighted bipartite graph. In our case, we want to match every point from the target polygon to one from the current polygon while minimizing the sum of distances between each couple of matched points. In order to warrant that the current polygon has more points than the target polygon, we first extend the current polygon with the midpoint of each edge until it contains more points than the target. The result of the Munkres algorithm is the minimal, in terms of distance travelled, set of points \( a^{cur} \) from the current polygon matching all points \( a^{tar} \) from the target polygon. The remaining \( b \) points of the current polygon are mapped to their orthogonal projections onto the target polygon. The intermediate shape is generated by interpolating each couple of points by a percentage \( \kappa \). Figure 2 presents the results obtained on simple shapes.

The Munkres algorithm is in \( O(n^3) \), yet pre-computations are possible. The start and target polygons only depend on the geometry and friction properties of each stance. Then, although the algorithm presented here directly gives back the interpolation, it is much more interesting to save the result, that is the set of correspondences \( a^{cur} \rightarrow a^{tar} \) and then only compute the interpolation between each couple of points for a given morphing percentage \( \kappa \) at each time-step. Moreover, as we usually use a few tens of points, running the Munkres algorithm is not very costly. To map a 25-sided polygon onto a 50-sided one, it takes about 0.67 s. Solving the assignment problem takes about 95% of the time.

Fig. 2: Stability 2D polygon morphing: \( P_s \) in blue, \( P_d \) in red, the green and black arrows show the motion and the normals directions of the edges respectively.

IV. TASK-BASED CONTROLLER

Now we discuss how this morphing is integrated in the controller. We control our humanoid robot using the multi-contact QP task controller described in [2], which compact form is:

\[
\min_y \quad \frac{1}{2} y^T Q y + c^T y \quad \text{def} \sum_i \beta_i \mathcal{T}_i(y) \\
\text{s.t.} \quad Ay \leq B \quad Cy = D
\]

where \( y \) is the decision variable representing both the joint accelerations \( \ddot{q} \), and the intensity of the forces \( \lambda \) applied at each contact point. The problem is written as a sum of quadratic or linear objectives weighted by \( \beta_i \) (task gains). Encoded constraints include joint position, velocity and torque limits, non-sliding contacts, and non-desired collisions avoidance. Typical tasks are written either directly or using the set-point objective formulation and comprise a wide variety of objectives [2]. Here, we focus on the CoM task.

To subscribe the CoM within a moving polygon, we use a collision constraint as in [2], [16]. However, in our case
the CoM interacts with a changing shape (see next section), which means that we have to take into account the speed and acceleration of the edges w.r.t the CoM, that is:

\[ \dot{\delta} = n^T \cdot (J \dot{q} - \dot{p}) \]  

\[ \ddot{\delta} = n^T \cdot (J \ddot{q} - \ddot{p}) + n^T \cdot (\dot{J} \dot{q} + J \dddot{q} - \dddot{p}) \]

where \( p, n \) are respectively the position and normal of the CoM projection on one such edge, \( \delta \) the resulting distance, and \( J \) the CoM Jacobian. We formulate a constraint:

\[ u = J \dddot{q} - \dddot{p} \quad e = J \dddot{q} - \dddot{p} \]

\[ -\text{dt} n^T J \dddot{q} \leq \frac{\delta - \delta_i}{\delta_i - \delta_s} + \ddot{\delta} + \text{dt} [\dot{n}^T u + n^T e] \]

with \( \xi \) a damping coefficient, \( \delta_i \) and \( \delta_s \) the interaction and safety distances respectively. Activating the above constraint whenever \( \delta < \delta \) will ensure that \( \delta \) is never smaller than \( \delta_s \).

In a number of cases, the quantities \( \dot{p}, \ddot{p}, n, \dot{n} \) can be explicitly computed from the mapping obtained by applying Algorithm 1. Indeed for every ordered vertex \( v_k \) of our current (convex) polygon that is the interpolate between \( a_k^{\text{cur}} \) and \( a_k^{\text{tar}} \) at \( \kappa(t) \), \( t \) being the time:

\[ v_k(\kappa) = a_k^{\text{cur}}(1 - \kappa) + a_k^{\text{tar}} \kappa \]

\[ \frac{\partial^n v_k}{\partial t^n} = (a_k^{\text{tar}} - a_k^{\text{cur}}) \frac{\partial^n \kappa}{\partial t^n} \quad \forall n \geq 1 \]

Thus, for every point \( p_k \) of the segment \([v_k, v_{k+1}]\) at a distance \( \alpha_k \) from the \( v_k \) and the associated normal \( n_k \):

\[ \frac{\partial^n p_k}{\partial t^n} = \sum_{i=0}^{n} \binom{n}{i} \kappa^i (1 - \kappa)^{n-i} \frac{\partial^n \alpha_k}{\partial t^n} \frac{\partial^{n-i} v_k}{\partial t^{n-i}} + \frac{\partial^i (1 - \alpha_k)}{\partial t^i} \frac{\partial^{n-i} v_{k+1}}{\partial t^{n-i}} \]

\[ n_k = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (v_{k+1} - v_k) \]

\[ n_k = N \cdot [a_k^{\text{cur}}(1 - \kappa) + a_k^{\text{tar}} \kappa - (a_k^{\text{cur}}(1 - \kappa) + a_k^{\text{tar}} \kappa)] \]

\[ = (1 - \kappa)N(a_k^{\text{cur}}(1 - \kappa) - a_k^{\text{tar}}(1 - \kappa)) \]

\[ = (1 - \kappa)n_{a_k^{\text{cur}}} + \kappa n_{a_k^{\text{tar}}} \]

\[ \frac{\partial^n n_k}{\partial t^n} = (n_{a_k^{\text{cur}}} - n_{a_k^{\text{tar}}}) \frac{\partial^n \kappa}{\partial t^n} \]

Whenever two points of the current polygon map to the same point of the destination polygon, i.e. \( n_{a_k^{\text{cur}}} = 0 \), the direction of \( n_k \) is constant but its norm is strictly decreasing. To avoid numerical issues whenever \( ||n_k|| \) is small, we consider \( n_k \) constant with respect to \( \kappa \) in this case.

Thus, knowing the \( n \)th time derivative of \( \kappa \) and \( \alpha \), one can compute the \( n \)th time derivative of the segments’ position and their normals. However, when \( \kappa \) or \( \alpha \) depend implicitly on the time, these quantities have to be evaluated through numerical differentiation. We can make the following approximations to neglect high-order terms:

- As the CoM only interacts with edges closer than \( \delta_s \), it usually interacts with few edges at a time. Because all points are interpolated uniformly and because the constraint pushes the CoM perpendicularly to the edge, variations of \( \alpha_k \) are small during interaction: we consider that \( \frac{\partial^n \alpha_k}{\partial t^n} = 0 \forall n > 0 \).

- In most of our use-cases, we interact when the interpolation is almost done (the CoM is generally nearby the barycenter of the polygons. At this moment, \( \dot{p} \) is either zero (when we interpolate at constant speed) or negative (using quadratic interpolation, the polygon is contracting and slowing down): we set \( \dot{p} = 0 \). This puts a harder bound on the constraint.

Overall rotation speeds are usually small compared to the homothetic part of the transformation, but we cannot neglect \( \dot{\delta} \). Depending on the shape of the support polygons, local rotations can be important, such as in the top corner of the polygon in Figure 2.

### V. Limiting the CoM Region During Motion

For the sake of clarity and without loss of generality, we illustrate our approach using the stair climbing with handrail task by our humanoid robot (HRP-2Kai). In our experiment trials the robot applied too much forces on the handrail. Although part of the excess of force is due to positioning error and would be solved by applying lower-level compliance, part of it comes from the fact that our QP controller does not explicitly reduce the force applied on the gripper. Adding a task of the form \( \min ||S_{\text{gripper}}||^2 \) did minimize the force and torque computed, but kept the posture unchanged meaning that the force applied by the real robot would remain high (this was also observed for a torque controlled robot in [5], p. 288, last sentence). Thus, we want to modify the trajectory such that the force applied on the gripper diminishes without endangering stability.

### A. Linking force limitation and interpolation

From intuition, maintaining the CoM in a shape that is similar to the static stability polygon of a restricted set of contacts should reduce the force applied on the others. We can compute the real static stability polygon with an additional force constraint: for a set of real numbers \( \gamma \in [0, 1] \) we compute the static stability polygon as described in section II with an additional constraint on the forces applied on each contact point of the grippers:

\[ ||f_i|| \leq \gamma mg \]

Then we compute two stability polygons at each step of the climbing, i.e. every time the active set of contacts is changed: one using all contacts, the other using all contacts but those between grippers and rail, and their interpolation. We then find by dichotomy which \( \kappa \) yields the the interpolation of the “full” and “restricted” polygon with the closest area to the real constrained static stability polygon. Captions of this process are shown in Figure 3 while a numerical comparison of values of \( \gamma \) and \( \kappa \) are shown in Figure 4.

This shows us that interpolation is a good approximation of the force constraint with restrictions:

- Because we limit the search region, the interpolation retains the shape of this limit contrarily to the constrained static stability polygon.

- \( \kappa \) is a non-linear function of \( \gamma \). Indeed, as a consequence of the above point, there is a “dead-zone” when the force constraint results in a shape modification smaller
Contact surfaces

\[ \gamma \] represents the constraint coefficient for a 3-contact scenario. Dashed line

Fig. 4: Interpolation coefficient as a function of the force

Fig. 3: Comparison between direct computation of the con-

strained static stability polygon and interpolation.

\[ |f_i| = |S_ix| \leq 5mg; |\tau_i| = |T_iS_ix| \leq 5 \text{ N m}; ||\text{com}|| \leq 1 \text{ m} \]

With \( S_i \) a selection matrix of forces at the \( i \)th contact. The interpolation between the current and target polygon takes place over the course of one second. To avoid strong discontinuities when the constraint gets activated, we use quadratic interpolation, i.e. a trapezoidal speed profile. We also use relatively high damping, \( \xi = 0.1 \), to further avoid discontinuity when activating the constraint. To make sure the QP does not fail, we allow a small violation of the constraint. The intermediate polygon is the interpolated at \( \gamma \), which is a direct consequence of using interpolation.

Thus, maintaining the CoM inside a polygonal region using the constraint presented in section IV corresponds to limiting the force applied on the gripper. This polygonal region is smoothly interpolated from the current static stability polygon, to an intermediate one. We can still maintain a task of the CoM in the cost function with low gains and limit the CoM motion in the constraint part of the QP. Note that we can use a static stability polygon because the accelerations and speeds generated by our controller are small and because the safety distance \( \delta_s \) acts as a stability margin.

B. Results of stairs climbing

To limit the stability region, we use the constraints of section II in the stability polygon computation. We strictly limit the torque applied on the gripper, and loosely the forces and CoM position using the following values:

\[ |f_i| = |S_ix| \leq 5mg; |\tau_i| = |T_iS_ix| \leq 5 \text{ N m}; ||\text{com}|| \leq 1 \text{ m} \]

With \( S_i \) a selection matrix of forces at the \( i \)th contact. The interpolation between the current and target polygon takes place over the course of one second. To avoid strong discontinuities when the constraint gets activated, we use quadratic interpolation, i.e. a trapezoidal speed profile. We also use relatively high damping, \( \xi = 0.1 \), to further avoid discontinuity when activating the constraint. To make sure the QP does not fail, we allow a small violation of the constraint. The intermediate polygon is the interpolated at 90\%: we need to choose a high \( \kappa \) because we want to strongly limit the forces applied on the gripper. This \( \kappa \) corresponds to an upper force of about 34 N, which means that in the worst case the torque applied on the gripper is lower than our torque limit.

We first designed the motion, and checked that introducing the polygonal constraint in our controller does indeed modify the posture as shown in Figure 5. Moreover, introducing this constraint allowed us to check that our controller can indeed generate a trajectory without applying any force on the gripper.

We then modified the CoM objectives in each phase to ensure that the robot’s flexibilities would not be too excited
VI. COMBINING TASKS AND CONSTRAINTS FOR FORCE CONTROL

In this section we present how to combine the CoM constraint and an impedance force control task to drive smooth removal of contacts. This is in fact a particular case of potentially more complex robot force control tasks. We also show that it is possible to recompute on the fly, but not in real-time the important polygons.

In a $n$ contact scenario, we want to force control the $i$th contact, $c_i$. We can then compute two geometry stability polygons:

$$P_s = \bigcup \{ \{c_k\}_{k \in [0 \ldots n]} \}$$  \hspace{1cm} (12)

$$P_d = \bigcup \{ \{c_k\}_{k \in [0 \ldots n]} \ k \neq i\}$$  \hspace{1cm} (13)

We can then interpolate between $P_s$ and $P_d$ at a rate $\kappa(t)$. We choose $\kappa(t)$ to have a trapezoidal speed profile of total duration 30s. Denoting $f_k$ the force at the $k^{th}$ contact, we can derive an objective force at that contact, $\bar{f}_k$ that is a function of $\kappa$. Knowing that we only control one contact, we choose:

$$\bar{f}_k = mg(1 - \kappa)$$  \hspace{1cm} (14)

We can then use an impedance force control task such as the one presented in section V to regulate the actual $f_k$ around $\bar{f}_k$ by setting $f_{obj}$ to $\bar{f}_k$ at each timestep. In reality, we only modify the impedance task objective whenever $f_k > \bar{f}_k$ to only maintain $f_k$ under the maximum desired force.

This linear objective in $\kappa$ is in general sub-optimal as lower forces could be attained, but without an explicit relationship between $\kappa$ and $\gamma$, we have to remain conservative.

VII. APPLICATION TO A MULTI-CONTACT SETTING

In this experiment, the controller establishes three contacts with the environment. Each of the two contacts that are added use the force control task to ensure parallelism of the contacting body and surface. When done, the robot has established three unilateral contacts with its environment: one between the right foot and the rear flat platform, one with its left foot and an inclined ramp and the last one is with its hand and an elevated flat block.

After establishing the contacts, we reposition the CoM before starting the experiment. During those few seconds, we asynchronously compute the corresponding polygons: $P_s$ and $P_d$ for every contact as well as every interpolator.

While maintaining those contacts, we interpolate $P_s$ towards $P_d$ for every contact: first for removing the right foot, then for removing the left foot and finally for removing the right hand. Another shorter experiment presented in Figure 9 and the attached video consists in doing only one interpolation, but then entirely removing the right foot.

In order to check that this technique would be applicable on the robot, we first tested it in the Choreonoid dynamics simulator [20], which is very reliable (it embeds flexibilities to the fact that a low-force solution only exists, but it is not the one we get when doing raw position control.

In this experiment, the controller establishes three contacts with the environment. Each of the two contacts that are added use the force control task to ensure parallelism of the contacting body and surface. When done, the robot has established three unilateral contacts with its environment: one between the right foot and the rear flat platform, one with its left foot and an inclined ramp and the last one is with its hand and an elevated flat block.

After establishing the contacts, we reposition the CoM before starting the experiment. During those few seconds, we asynchronously compute the corresponding polygons: $P_s$ and $P_d$ for every contact as well as every interpolator.

While maintaining those contacts, we interpolate $P_s$ towards $P_d$ for every contact: first for removing the right foot, then for removing the left foot and finally for removing the right hand. Another shorter experiment presented in Figure 9 and the attached video consists in doing only one interpolation, but then entirely removing the right foot.

In order to check that this technique would be applicable on the robot, we first tested it in the Choreonoid dynamics simulator [20], which is very reliable (it embeds flexibilities
at the ankle and noise) to ensure that the trajectory is indeed feasible and that the robot does avoid both environment and self-collisions.

In this experiment, we do not use a CoM task, only the constraint with wide safety margin. This ensures smoothness and stability. We also add a very low weight posture task to avoid having uncontrolled joints. The last task is the impedance control task that is only inserted when interpolating for the first contact. This will allow us to confirm its importance and role. Note that without an explicit force regulation task, as the Hessian of the problem is positive definite, the controller will try to minimize $||\lambda||^2$ i.e. an equal distribution that does not reflect the real forces being applied.

Whenever we finish the interpolation (i.e. $\kappa = 1$), the CoM is inside the reduced support polygon of the supporting contacts: there exists a zero-force solution –at the contact to be removed– to support the robot in this configuration. Of course, we do not know a priori if this solution fulfills the torque limits of the robot, but as we started from a feasible configuration and smoothly drove the robot to the desired one while continuously reducing the force applied on the contact, it is to be expected.

We show in Figure 10 how the force did decrease continuously, both in simulation and in reality albeit with slightly different profiles. We reach a zero-force configuration whenever the interpolation ends for the first contact. Note that for successive contacts, this is no longer true. When we do not explicitly aim for the desired force distribution, the robot establishes itself in a configuration that balances the contact forces with the joint torques generated by the lower level PD controller. Indeed, the worst case is shown in the third part, when interpolating to remove the hand: as the CoM is already almost inside the reduced polygon, the force applied on the left hand is only slightly reduced. However, it could safely be controlled to smoothly decrease towards zero as the interpolation index decreases as was done for the foot.

**VIII. Conclusion**

In this paper, we show that it is possible to enforce stability as a constraint in low dynamics multi-contact transitions. We devised a continuous morphing of the stability polygon as part of the constraint in QP controllers. We use this technique to limit the force applied on a given contact, whether by directly restraining the accessible region of the CoM throughout the movement or by specifying the force to be applied on a contact and displacing the CoM accordingly. Once the CoM is correctly placed, we can be sure that impedance or admittance control is usable to select the appropriate force distribution. Although we still use postures generated from planning, it is an interesting addition to it,
as it allows us to modify the generated motion according to our needs without any expensive re-computation. This is even more interesting as it allows us to specify that the CoM has to remain in a deformable stability region instead of explicitly forcing the CoM to remain at the planned position.

As future work, we are working on the following improvement: extending this work to dynamic motion, and closed-loop control. For the former, we need to extend the theoretical part to take into account accelerations explicitly instead of relying on safety margins (this is almost done). For the latter, we need to curb the computation time of the morphing as it will depend on online measured/estimated forces and that of the computation of the geometric stability polygon to run with the controller on the robot embedded computer. Similarly, to account for differences between actual surface shapes and planned ones we need to efficiently compute the actual shape from the off-line planned one, possibly by using the current robot state to narrow down the search area.

REFERENCES

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Distance along the segment, see equation (8), page 4</td>
<td></td>
</tr>
<tr>
<td>$f_k$</td>
<td>Objective force at contact k, see equation (13), page 6</td>
<td></td>
</tr>
<tr>
<td>$\beta$</td>
<td>Task weight, see equation (1), page 3</td>
<td></td>
</tr>
<tr>
<td>$\co{\text{m}}$</td>
<td>CoM position, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$a^{\text{cur}}$</td>
<td>Corresponding points in $P_s$, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$\delta f$</td>
<td>Force error for compliance task, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>Distance to a line, see equation (3), page 4</td>
<td></td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Morphing percentage, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Error threshold to stop the algorithm, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$a^{\text{tar}}$</td>
<td>Points in $P_d$, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$\xi$</td>
<td>Damping coefficient, see equation (4), page 4</td>
<td></td>
</tr>
<tr>
<td>$A, C/B, D$</td>
<td>Constraint matrices/vectors, see equation (1), page 3</td>
<td></td>
</tr>
<tr>
<td>$A_1, A_2$</td>
<td>Matrices projecting the wrenches, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$b$</td>
<td>Points in $P_s$, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$B, u$</td>
<td>Matrices describing the friction cones, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$c$</td>
<td>Gradient vector, see equation (1), page 3</td>
<td></td>
</tr>
<tr>
<td>$c_k$</td>
<td>$k$-th contact, see equation (13), page 6</td>
<td></td>
</tr>
<tr>
<td>$d$</td>
<td>Search direction, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$e$</td>
<td>Normal acceleration, see equation (4), page 4</td>
<td></td>
</tr>
<tr>
<td>$f_k$</td>
<td>Force at contact k, see equation (13), page 6</td>
<td></td>
</tr>
<tr>
<td>$f_{\text{mes}}$</td>
<td>Measured force at the sensor, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$f_{\text{obj}}$</td>
<td>Compliance task objective force, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$J$</td>
<td>CoM jacobian, see equation (3), page 4</td>
<td></td>
</tr>
<tr>
<td>$K_d$</td>
<td>Compliance task derivative gain matrix, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$K_p$</td>
<td>Compliance task proportional gain matrix, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$l$</td>
<td>Contact position, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$N$</td>
<td>Normal matrix, see equation (8), page 4</td>
<td></td>
</tr>
<tr>
<td>$n$</td>
<td>Normal to a line, see equation (3), page 4</td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>Position of a point on the line, see equation (3), page 4</td>
<td></td>
</tr>
<tr>
<td>$Q$</td>
<td>Hessian matrix, see equation (1), page 3</td>
<td></td>
</tr>
<tr>
<td>$r$</td>
<td>CoM region radius, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$S$</td>
<td>Selection matrix for contact forces, see equation (9), page 5</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>Target acceleration, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$u$</td>
<td>Relative speed, see equation (4), page 4</td>
<td></td>
</tr>
<tr>
<td>$v$</td>
<td>Vertex of the interpolate, see equation (8), page 4</td>
<td></td>
</tr>
<tr>
<td>$X$</td>
<td>Compliance task position objective, see equation (11), page 6</td>
<td></td>
</tr>
<tr>
<td>$x$</td>
<td>Set of forces, see equation (1), page 2</td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>Optimisation variable, see equation (1), page 3</td>
<td></td>
</tr>
</tbody>
</table>