Multi-contact transitions and force control of humanoid robots by stability polygons reshaping and morphing

Hervé Audren, Abderrahmane Kheddar

Multi-contact transitions and force control of humanoid robots by stability polygons reshaping and morphing

Hervé Audren[1] and Abderrahmane Kheddar[1,2]

***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 slow transitions and force limitation with a level of force control. 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*— Humanoid Robots, Stability, Optimization, Multi-contact**

I. Introduction

The motivation behind this work originates from our rehearsal for the Darpa Robotics Challenge. In several tasks such as climbing stairs using handrails or egress/ingress a car, we decided to use our background in multi-contact technology that was assessed in complex tasks [1], [2]. Multi-contact technology is relevant in situations of confined spaces, high unevenness or highly unstructured environments. In these situations, the motion of legged robots must be slow. While several works have proposed efficient regulation control strategies in multi-contact configurations [3]–[6], transitions between contact stances is still to be researched[1].

Multi-contact transitions are prone to instability: because releasing or adding a contact is a discrete event, it induces an abrupt change in some tasks and constraints. One way to alleviate the effects of these changes is to use a preview controller to enhance the motion with a flavor of dynamics [7], [8], which would have hidden the problem addressed by this paper. But slow multi-contact motion can be achieved without preview [2], using only a closed-loop local task-based quadratic programming (QP) control, largely adopted in the community.

Our QP controller considers two level of hierarchy: tasks that are to be achieved are given as part of the cost function and tasks that should not be violated are given as constraints. For example, a contact that is to be added appears first in the cost function, and once achieved, it is shifted to the constraints part to be treated as a motion support contact in the next phase. The Center of Mass (CoM) was always put in the cost part of the QP controller as stability was ad-hoc, assumed to be held by tracking at best the CoM target given by the planner, while enforcing each contact force to be positive and be within its respective friction cone. Our experiments confirmed that this is not enough to warrant stable transitions. This becomes even more critical if one would limit the force applied for a given contact.

A large number of works enforce stability by controlling the Center of Pressure (CoP) at each contact to remain within the convex hull of each contact area [5]–[7], [9], [10]. This can be added to the QP controller as a constraint, replacing the non-sliding contacts and unilateral forces constraints currently in use. However, in all those approaches 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 our case, we are concerned with a position-controlled robot, and we are ideally looking for a criterion linking joint angles with stability. As multi-contact motions are quasi-static, constraining the CoM to remain within the frictional static stability polygon is an acceptable criterion (ZMP assumes infinite friction at contact). Although the CoM position is a continuous quantity, the static stability polygon abruptly changes shape when changing contacts. Thus, including it directly as a constraint in our QP controller would lead to its immediate violation upon contact change.

Therefore, we propose a novel continuation method based on optimal matching of points to interpolate between convex polygons (those resulting from 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.

As an unexpected bonus, this technique 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 or measured contact forces. This is another novel contribution, and force regulation can be obtained by implicitly moving the CoM, resulting in a modified force distribution that fulfills the target contact force limit. We note that our work applies to torque and position controlled robots, but it has more impact and value for the latter ones.

Our summarized main contributions stand as follows:

- We extended the exact stability polygon computation algorithm proposed in [11] to handle bilateral contacts;
- We propose a novel force control method based on stability polygon morphing to constrain the CoM during motion and contact transitions;

[1] CNRS-AIST Joint Robotics Laboratory UMI3218/CRT, Tsukuba, Japan
[2] CNRS-UM LIRMM, Interactive Digital Human, Montpellier, France

[1] Among the conclusions of the panelists of IEEE-RAS Humanoids 2015's workshop on Whole-Body Multi-Task Multi-Contact Humanoid Control

- We shape the stability polygon smoothly as a function of the force intensity and not only the contact geometry;
- We assess our approach using our QP multi-objective controller in HRP-2Kai humanoid robot multi-contact scenarios.

We present in subsection II-A how we adapted the polygon computation to our use case, then how we included this polygon as a constraint in objective-based controllers in subsection II-B. We introduce our new method to smoothly constrain the CoM in section III before applying it to the stairs case in section IV. To solve an arising problem, we present a way to efficiently combine tasks with this constraint in section V and present results obtained in another multi-contact scenario in section VI.

II. Background

A. 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 built iteratively from solving a series of second-order cone programs of the form:

$$\begin{aligned} \max_{z} \quad & d^T \widetilde{com} \\ \text{s.t.} \quad & A_1 x + A_2 \widetilde{com} = t \\ & ||Bx|| \leq u^T x \end{aligned} \tag{1}$$

The solution $\widetilde{com}^*$ of this problem is an extremal stable CoM position in the direction d i.e. there exists a set of forces x realizing $t = \begin{bmatrix} mg & 0 \end{bmatrix}^T$ 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 $\widetilde{com}$. For each of these problems, $\widetilde{com}^*$ is added to the inner approximation, while the half-plane defined by $\{\widetilde{com} \in \mathbb{R}^2 | d^T \widetilde{com} > d^T \widetilde{com}^*\}$ 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 σ. The authors prove that there is a strict upper bound on the number of iterations needed to reach σ.

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 $|\widetilde{com}| < r$ that will constrain the CoM to a rectangular (polygonal) region or a conic constraint of the form $||\widetilde{com}|| < r$ that will constraint 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 $||\widetilde{com} - l|| < r$ 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, $|Tx| < \tau_m$ where T represents the cross product between the contact points and the contacting link. In this case, τ_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.

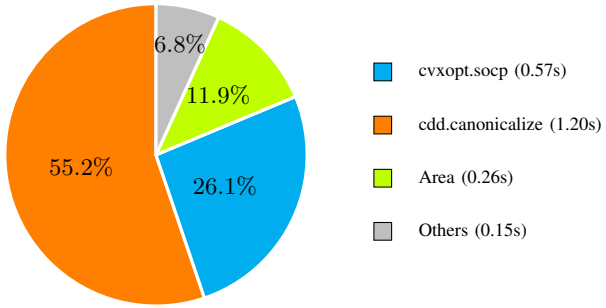

Fig. 1: Pie chart showing as angular sections the repartition of computation time for 108 iterations (Required precision of 1 cm^2): total time 2.17 s

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. The cost of solving each problem is largely offset by that of finding the correct edge for the next iteration. As we manipulate both vertices and half-planes, we use the `cddlib` [13] to change between the face and span representation of the convex polygons. Most of the algorithm time is spent removing redundancies in the polygon representation (calls to `cdd.canonicalize`) see Figure 1. Without such correction, numerical instability arises. Moreover, the cost of such a simplification is reported back on the area computation routine that switches from the half-plane to the vertex representations and simplifies the polygon anyway. Our implementation is sub-optimal as we do not explicitly keep track of edges and have to compare them after each iteration to re-compute the necessary areas only. This computational cost seems to be negligible, but increases with the number of iterations and may become a source of numerical errors when the desired precision is high.

B. Task-based controller

We control our humanoid robot using the multi-contact QP task controller described in [2], which compact form is:

$$\begin{aligned} \min_{y} \quad & \frac{1}{2} y^T Q y + c^T y \stackrel{\text{def}}{=} \sum_i \beta_i \mathcal{T}_i(y) \\ \text{s.t.} \quad & Ay \leq B \; ; \; Cy = D \end{aligned}$$

where y is the decision variable representing both the joint accelerations $\ddot{q}$, and the intensity of the forces λ applied at each contact point. The problem is written as a sum of quadratic or linear objectives weighted by β_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 maintain the CoM within a moving polygon, we use a collision constraint as in [2], [14]. 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 with respect to the CoM:

$$\dot{\delta} = n^T \cdot (J\dot{q} - \dot{p}) \tag{2}$$

$$\ddot{\delta} = \dot{n}^T \cdot (J\dot{q} - \dot{p}) + n^T \cdot \left(\dot{J}\dot{q} + J\ddot{q} - \ddot{p} \right) \tag{3}$$

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

$$u = J\dot{q} - \dot{p} \qquad e = \dot{J}\dot{q} - \ddot{p}$$

$$-\mathrm{d}t\, n^T J\ddot{q} \leq \xi \frac{\delta - \delta_s}{\delta_i - \delta_s} + \dot{\delta} + \mathrm{d}t\, [\dot{n}^T u + n^T e] \tag{4}$$

with ξ a damping coefficient, δ_i and δ_s the interaction and safety distances respectively. Activating the above constraint whenever $\delta < \delta_i$ will ensure that δ is never smaller than δ_s.

III. Morphing stability polygons

Changing contacts is a discrete event: at that time, the CoM may not be inside the appropriate region of 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.

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. [15], [16] because we only morph much simpler shapes: 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 [17]. 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 κ.

The Munkres algorithm is in $\mathcal{O}(n^3)$, 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 κ 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

Algorithm 1 Interpolation of two polygons : $f(P_s, P_d, \kappa)$

Input: P_s, P_d start and destination polygons, κ percentage
$a^{cur}, done \leftarrow \{\}, \{\}$
while $|P_s| < |P_d|$ **do**
  $P_s \leftarrow P_s \cup$ midpoints(P_s)
end while
$M \leftarrow$ zeros$(|P_s|, |P_d|)$
for $a^{tar} \in P_d$ **do**
  for $b \in P_s$ **do**
    $M_{ij} = ||a^{tar} - b||_2$
  end for
end for
$a^{cur}, done \leftarrow$ Munkres(M)
for $b \in P_s - done$ **do**
  $a^{tar} \leftarrow P^{\perp}(b, P_d)$
end for
$conv$(interpolate$(a^{cur}, a^{tar}, \kappa)$)

a 50-sided one, it takes about $0.67\,\text{s}$. Solving the assignment problem takes about $95\,\%$ of the time.

In a number of cases, the quantities $\dot{p}, \ddot{p}, n, \dot{n}$ introduced in subsection II-B can be explicitly computed. Indeed for every ordered vertex v_k of our current (convex) polygon that is the interpolate between a_k^{cur} and a_k^{tar} at $\kappa(t)$, t is the time:

$$\begin{aligned} v_k(\kappa) &= a_k^{cur}(1-\kappa) + a_k^{tar}\kappa \\ \frac{\partial^n v_k}{\partial t^n} &= (a_k^{tar} - a_k^{cur})\frac{\partial^n \kappa}{\partial t^n} \qquad \forall n \geq 1 \end{aligned}$$

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

$$\frac{\partial^n p_k}{\partial t^n} = \sum_{i=0}^{n} \binom{n}{i} \frac{\partial^i \alpha_k}{\partial t^i} \frac{\partial^{n-i} v_k}{\partial t^{n-i}} + \frac{\partial^i (1-\alpha_k)}{\partial t^i} \frac{\partial^{n-i} v_{k+1}}{\partial t^{n-i}} \tag{5}$$

$$\begin{aligned} n_k &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} (v_{k+1} - v_k) \overset{\text{def}}{=} N(v_{k+1} - v_k) && (6) \\ &= N\left[a_{k+1}^{cur}(1-\kappa) + a_{k+1}^{tar}\kappa - (a_k^{cur}(1-\kappa) + a_k^{tar}\kappa)\right] \\ &= (1-\kappa)N(a_{k+1}^{cur} - a_k^{cur}) + \kappa N(a_{k+1}^{tar} - a_k^{tar}) \\ &= (1-\kappa) n_{a_k^{cur}} + \kappa n_{a_k^{tar}} && (7) \end{aligned}$$

$$\frac{\partial^n n_k}{\partial t^n} = (n_{a_k^{cur}} - n_{a_k^{tar}})\frac{\partial^n \kappa}{\partial t^n} \tag{8}$$

Whenever two points of the current polygon map to the same point of the destination polygon, i.e. $n_{a_k^{tar}} = \begin{bmatrix} 0 & 0 \end{bmatrix}$, 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 κ in this case.

Thus, knowing the n^{th} time derivative of κ and α, one can compute the n^{th} time derivative of the segments' position and their normals. However, when κ or α depend implicitly on the time, these quantities have to be evaluated through numerical differentiation. We can make the following approximations to simplify this computation:
– As the CoM only interacts with edges closer than δ_i, 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 α_k are small during interaction: we consider that $\frac{\partial^n \alpha_k}{\partial t^n} = 0 \, \forall n > 0$.
– In our use-case, we interact when the interpolation is almost done. At this moment, $\ddot{p}$ is either zero (when we interpolate at constant speed) or negative (using quadratic interpolation, the polygon is contracting and slowing down): we set $\ddot{p} = 0$. This puts a harder bound on the constraint.

Rotation speeds are usually small compared to the homothetic part of the transformation, but we can not neglect $\dot{n}$. 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.

IV. 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).

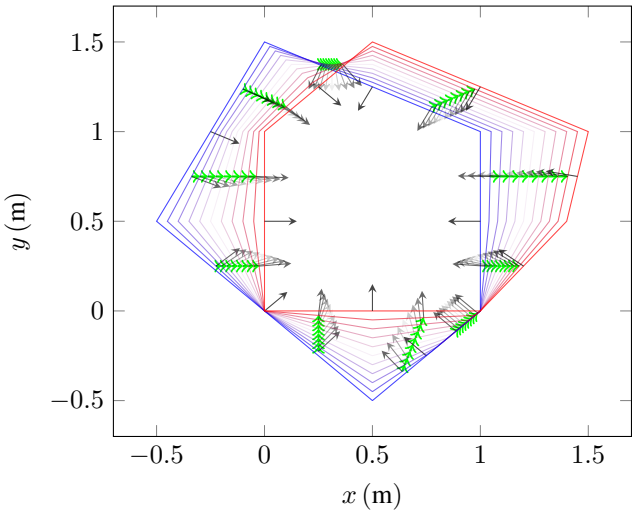

Fig. 2: Stability 2D polygon morphing: P_s in blue, P_d in red, the green and black arrows show the motion and the normals directions of the edges respectively.

A. Presentation of the stairs task

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 $\min ||S_{\text{gripper}}\lambda||^2$ 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).

From intuition, maintaining the CoM in a shape that is similar to the static stability polygon of the feet should limit the force applied on the grippers. To assess this, 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. We can then compare an intermediate polygon, the interpolation of the "full" and "restricted" polygon at κ, with the real static stability polygon computed with an additional force constraint. We thus compute for a set of real numbers $\gamma \in [0, 1]$ the static stability polygon as described in subsection II-A with an additional constraint on the forces applied on each contact point of the grippers:

$$||f_i|| \leq \gamma m g \tag{9}$$

We then find by dichotomy which κ yields the intermediate 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 γ and κ 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

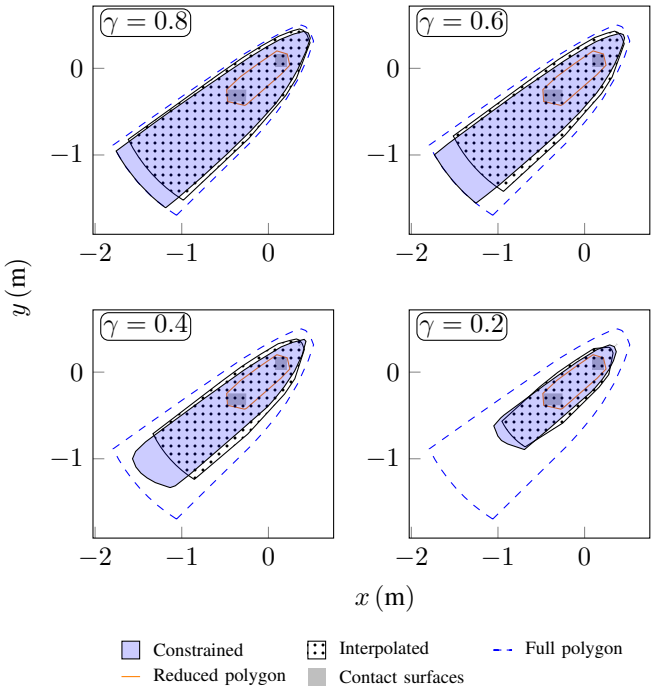

Fig. 3: Comparison between direct computation of the constrained static stability polygon and interpolation.

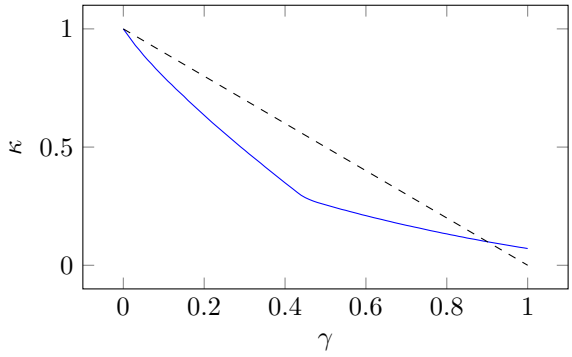

Fig. 4: Interpolation coefficient as a function of the force constraint coefficient for a 3-contact scenario.

static stability polygon.

- κ is a non-linear function of γ. Comprehensive analysis of this relationship is beyond the scope of this paper.
- Interpolation is more accurate for extreme values of γ which is a direct consequence of using interpolation.

Thus, maintaining the CoM inside a polygonal region using the constraint presented in subsection II-B 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 δ_s acts as a stability margin.

B. Results of stairs climbing

To limit the stability region, we use the constraints of subsection II-A 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:

$$|f_i| = |S_i x| \leq 5mg; |\tau_i| = |T_i S_i x| \leq 5\,\mathrm{N\,m}; ||\widetilde{com}|| \leq 1\,\mathrm{m}$$

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 triangular 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 interpolate at $90\,\%$: we need to choose a high κ because we want to strongly limit the forces applied on the gripper. This κ corresponds to an upper force of about $34\,\mathrm{N}$, which means that in the worst case the torque applied on the gripper is lower than our torque limit.

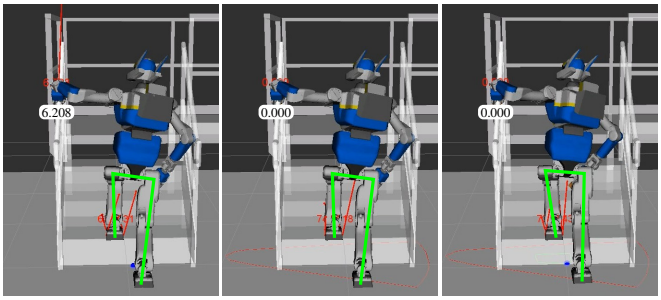

Fig. 5: Climbing the stairs with HRP-2 Kai: with neither the constraint nor gripper torque task activated (left), with only the task (middle) and with both (right)

We can see in Figure 5 and Figure 6 that adding this constraint does indeed modify the CoM position:

- In region `b`, the robot has contact with the handrail, the left foot is on the first step and it moves the right foot onto the first step. We can see that our constraint pushes the CoM back onto the left foot, that should reduce the strain applied on the hand. This difference is shown in Figure 5 and will be used as the interest region in Figure 7. Other similar events take place in regions `a`, `e` and `f`.
- In regions `c` and `d` the robot is entirely on the stairs and removes its hand contact without its CoM being inside the feet support polygon. The stability region constraint gets activated and the CoM moves forward onto the feet.

To validate our approach, we add a high-gain task in order to minimize the torque applied on the gripper. In this case, one of the few times we got non-zero torque on the gripper was in region `b`, when moving the right foot from the floor to the first step. The CoM was out of the support region during the leg swing: our approach enables us to move the CoM onto the stable region and then execute the movement

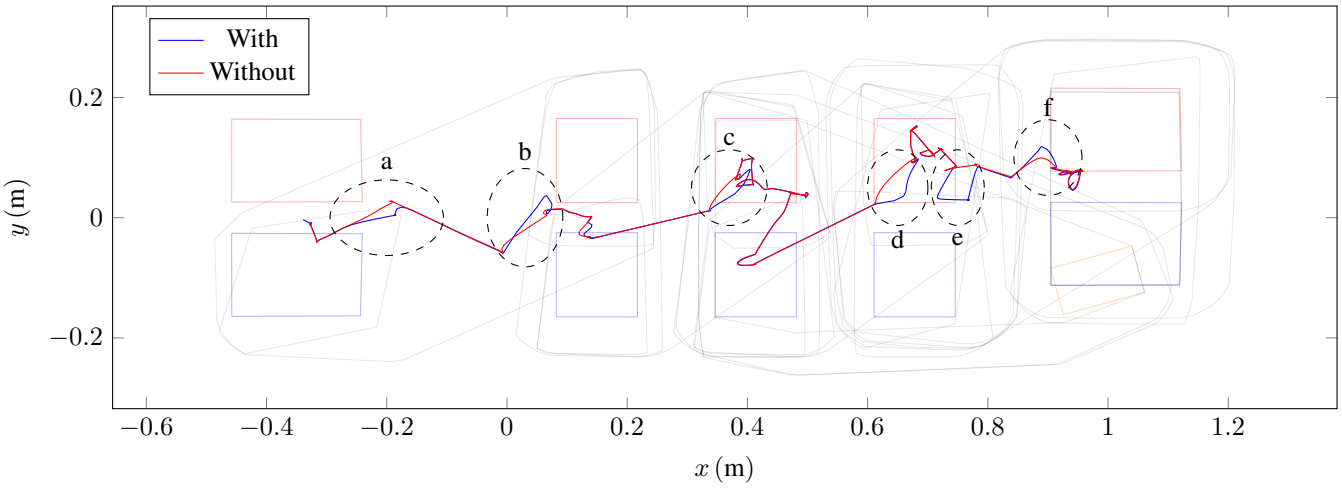

Fig. 6: Comparison of CoM trajectories with and without the polygonal constraint (Full movement). The red and blue rectangles are contact areas between the left and right foot, respectively, on the stairs projected on the ground.

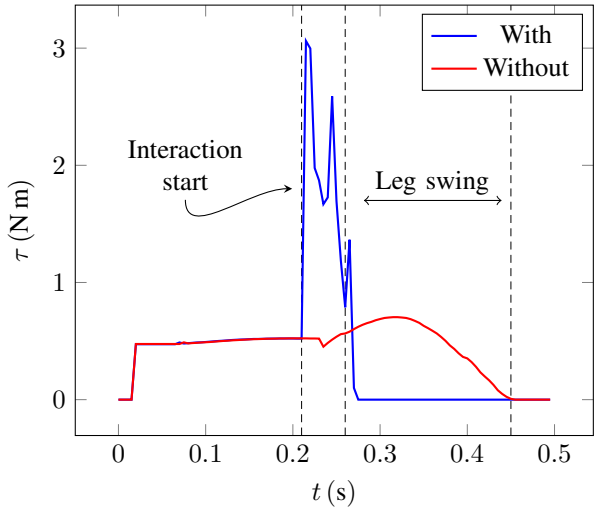

Fig. 7: Comparison of torque applied on left gripper with and without the polygonal constraint (region of interest)

at gripper 0-torque, as shown in Figure 7. Note that what was a continuous torque being applied during the leg swing is now replaced by a torque spike applied before movement. This result may seem worse than the original, but in fact it enforces stability. Firstly, the angular impulse of the torque spike does not differ much from the original, with both being within 5 % of relative error, around $0.20\,\mathrm{N\,m\,s}$. Secondly, this torque is still well within the torque limits of the gripper. Most importantly, we recomputed the static stability polygon in both cases: once with only two contacts and once using all three contacts but constraining the torque to be between 2 and $3\,\mathrm{N\,m}$. In the latter case, the stability polygon is still twice as wide as in the former. Although this shows that we could almost get rid of the CoM objective given by the planner, using only the constraint to move the CoM is 'coarse' if not used in conjunction with the tasks. In the next section we build a set of tasks that allow smooth removal of contacts and force control in a multi-contact setting.

V. Combining Tasks and Constraints

In this section we present how to combine a CoM task, the CoM constraint and a force task to drive smooth removal of contacts. This is in fact a particular case of potentially more complex robot force control tasks.

To perform force control on n contacts, we may set a force f_d of contact c_i. The remainder force is $f_r = mg - f_d$. The force at contact c_k at time t is $f_k(t)$. Then we can compute two geometry stability polygons:

$$P_s = \bigsqcup(\{c_k\}_{k\in[0\cdots n]}) \tag{10}$$

$$P_d = \bigsqcup(\{c_k\}_{k\in[0\cdots n] \quad k\neq i}) \tag{11}$$

Then, we can create a new CoM objective that is the barycenter of the start and end polygons centroid weighted by the intensity of desired force and remainder force:

$$f_{\%} = \frac{||f_d||}{||f_d|| + ||f_r||} \tag{12}$$

$$\mathrm{CoM}_d = f_{\%}\mathcal{C}(P_d) + (1 - f_{\%})\mathcal{C}(P_s) \tag{13}$$

where $\mathcal{C}$ is the centroid; we can then define task error at any time t by $||\mathrm{CoM}_d - \mathrm{CoM}(t)||$. We normalize it by the error at $t = 0$ and invert it to reach its maximum value when the task is fully realized, that is:

$$\epsilon(t) = 1 - \frac{||\mathrm{CoM}_d - \mathrm{CoM}(t)||}{||\mathrm{CoM}_d - \mathrm{CoM}(0)||} \tag{14}$$

As the CoM task has the highest priority in our case, its error is strictly decreasing when the robot is not falling i.e. $\epsilon(t) \in [0, 1]\ \forall t > 0$. If $\mathrm{CoM}(0)$ is close to CoM_d compared to the numerical precision of the controller, we do not update

the objective. From this, we can derive the current force objective and constraint polygon:

$$f_\epsilon(t) = \epsilon f_d + (1-\epsilon) f_i(0)$$
$$P(t) = \text{interpolate}(P_s, P_d, f_{\%}\epsilon(t))$$

VI. Application to a multi-contact setting

In this simulation, we place the robot in a non-coplanar contact configuration. The robot has three unilateral contacts with its environment: one between the right foot and an inclined ramp, one with its left foot and the rear flat platform, the last one is with its wrist and an elevated flat block. Using these contacts, we ask the robot to apply, in succession, 50 N on its right foot, 100 N on its left foot. In a second test, we switch the desired force on each foot.

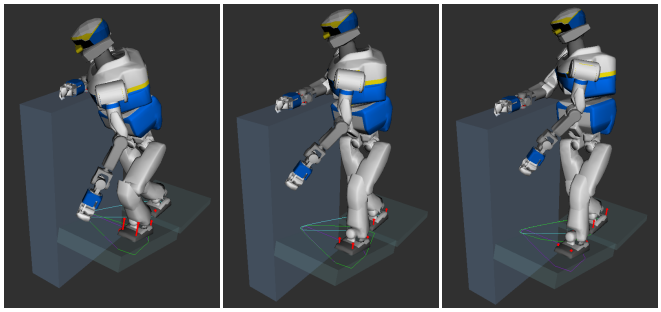

Fig. 8: Snapshots of the force control simulation with HRP-2. In green, the current polygon, in cyan and purple the polygons using respectively the right and left foot.

We use a high-weight (high-gain) 2D CoM task, and a low weight joint position task to ensure that all joints are controlled throughout the movement. We also add a force task with low-weight which role is to try matching the force on the controlled contact to the desired one. This does not allow us to check that the forces on the real robot will be close to the desired ones, but rather that such a distribution of forces is possible. Note that without this objective, as the Hessian of the problem is positive definite, the controller will try to minimize $||\lambda||^2$ i.e. an equi-distribution. The robot alternates between the postures presented in Figure 8. The force results are presented in Figure 9. The interpolated force does not reach the exact desired one because the desired CoM is not reachable: this is worth noticing as the controller behaves well when requiring an infeasible force.

The main reason for the error remainder is the collision constraint with the right block. Indeed, the actual CoM objective is too close to the block to be reached without having the robot's torso penetrating it. Notice that the spike torque behavior in Figure 7 doesn't occur here. This is because we combined the CoM task, the force task and the stability region constraint.

In this scenario, we did not fully remove the contacts in order to do a cyclical check of the feasibility of our method. In others, we fully removed the contact (when desired). Indeed, whenever we reach the CoM target (CoM_d), the

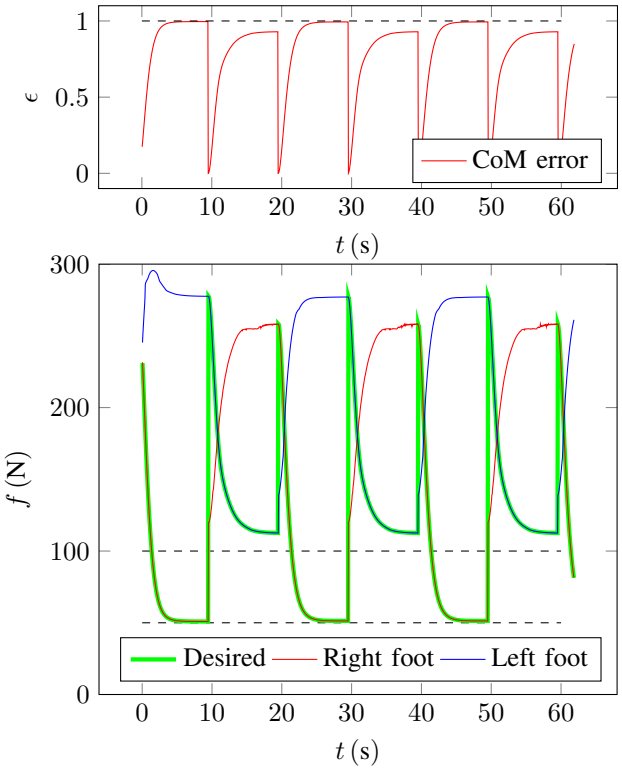

Fig. 9: Force control of feet in multi-contact.

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 we continuously reduce the force applied on the contact that can be removed.

In order to check that this technique would be applicable on the robot, we used it in an open-loop scenario, simply replaying the joint trajectory in the Choreonoid dynamics simulator [18], which is very reliable (it embeds flexibilities at the ankle and noise). The Figure 9 shows that the force "tracking" can still be improved: the actual repartition of forces is not linear in the distance between the CoM and the polygon centroids, a result that was expected from Figure 4. However, we get a very good precision when the desired force is small (around 50 N): this shows that indeed moving the robot in this way reduces the force applied on the contact in a continuous way. This confirms that when the CoM is inside the reduced stability polygon, or even better, above its centroid, one can safely remove the other contact as we are close to a zero-force solution.

VII. 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

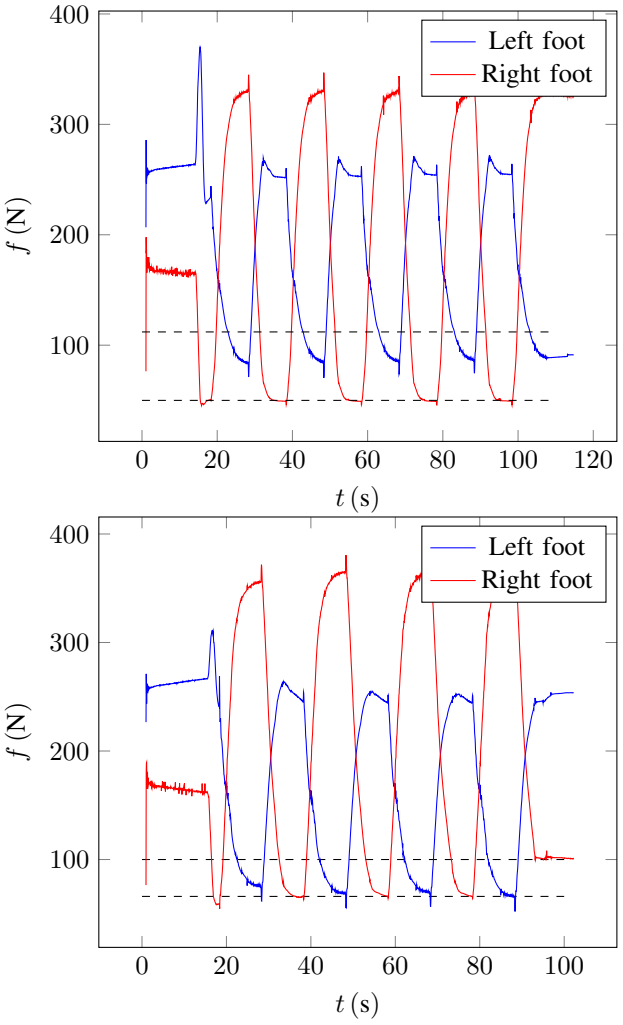

Fig. 10: Results of dynamic simulation. Top: $f_{d_r} = 50\,\text{N}$, $f_{d_l} = 100\,\text{N}$. Bottom: $f_{d_r} = 100\,\text{N}$, $f_{d_l} = 50\,\text{N}$. The dashed lines represent the actual asymptotes of $f_\epsilon(t)$.

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. 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 a given position.

As future work, we will devise a closed-loop controller based upon this theory, by deforming the CoM region from the forces that are actually measured or estimated and compare it to an impedance/admittance control on each contact. To improve force tracking performance, we are considering integrating an impedance controller at a lower level. Prior to that, 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.

References

[1] K. Bouyarmane and A. Kheddar, "Humanoid robot locomotion and manipulation step planning," *Advanced Robotics*, vol. Special Issue on Cutting edge robotics in Japan, no. 26, pp. 1099–1126, July/September 2012.

[2] J. Vaillant, A. Kheddar, H. Audren, F. Keith, S. Brossette, K. Kaneko, M. Morisawa, E. Yoshida, and F. Kanehiro, "Vertical Ladder Climbing by HRP-2 Humanoid Robot," in *14th IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 671–676.

[3] C. Ott, M. A. Roa, and G. Hirzinger, "Posture and balance control for biped robots based on contact force optimization," in *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011, pp. 26–33.

[4] S.-H. Lee and A. Goswami, "A momentum-based balance controller for humanoid robots on non-level and non-stationary ground," *Autonomous Robots*, pp. 1–16, 2012.

[5] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[6] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, Illinois, 14-18 September 2014.

[7] 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*, Chicago, Illinois, 14-18 September 2014.

[8] H. Dai, A. V. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, Madrid, Spain, 18-20 November 2014, pp. 295–302.

[9] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *IEEE Transactions on Robotics*, 2010.

[10] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization Based Full Body Control for the Atlas Robot," *IEEE-RAS International Conference on Humanoid Robots*, pp. 120—-127, 2014.

[11] T. Bretl and S. Lall, "Testing static equilibrium for legged robots," *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[12] M. Andersen, J. Dahl, and V. Lieven, "CVXOPT: Python Software for Convex Optimization."

[13] K. Fukuda and A. Prodon, "Double Description Method Revisited," *Combinatorics and Computer Science (LNCS 1120)*, vol. 1, pp. 91–111, 1996.

[14] F. Kanehiro, M. Morisawa, W. Suleiman, K. Kaneko, and E. Yoshida, "Integrating geometric constraints into reactive leg motion generation," *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 4069–4076, 2010.

[15] T. W. Sederberg and E. Greenwood, "A physically based approach to 2-D shape blending," *Proceedings of the 19th annual conference on Computer graphics and interactive techniques - SIGGRAPH '92*, pp. 25–34, 1992.

[16] M. Alexa, D. Cohen-Or, and D. Levin, "As-rigid-as-possible shape interpolation," *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pp. 157–164, 2000.

[17] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[18] S. Nakaoka, "Choreonoid: Extensible virtual robot environment built on an integrated GUI framework," in *2012 IEEE/SICE International Symposium on System Integration, SII 2012*, 2012, pp. 79–85.