When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC

Stéphane Caron, Quang-Cuong Pham

When to make a step? Tackling the timing problem in multi-contact locomotion by TOPP-MPC

Stéphane Caron[1] and Quang-Cuong Pham[2]

***Abstract*— We present a Model-Predictive Controller (MPC) for multi-contact locomotion where predictive optimizations are realized by Time-Optimal Path Parameterization (TOPP). The key feature of this design is that, contrary to existing planners where step timings are provided as inputs, here the timing between contact switches is computed as *output* to a linear optimization problem based on a dynamic model of the robot. This is particularly appealing to multi-contact locomotion, where proper timings depend on the terrain topology and suitable heuristics are unknown. Thanks to recent advances in multi-contact stability computations, we improve the performance of TOPP for COM trajectories, which allows us to integrate it into a fast control loop. We implement the complete control pipeline and showcase it in simulations where a model of the HRP-4 humanoid climbs up and down a series of hills.**

I. Introduction

For walking on horizontal floors, step timing was mostly a question of parameter tuning: the ground being isotropic, it was sufficient to tune fixed durations for single-support (SS) and double-support (DS) phases, which would work for any future number of steps. However, in order to fully exploit the locomotory capabilities of humanoids, current research is now moving away from this isotropic assumption. In general environments, the ability to step in a given time depends both on the terrain topology and robot dynamics, which led *e.g.*the authors of [1] to conclude that, in multi-contact, "time parameterization of the contact formation/release and their transition phases can hardly be left for tuning".

Model-predictive control (MPC) is a paradigm that can give controllers the level of foresight required to tackle this question. Its applications to multi-contact locomotion are relatively recent, and can be split in two groups. In one line of research, contact forces are jointly considered as control variables used to optimize a quadratic cost function on future whole-body motions [1], [2], [3]. In such formulations, inequality constraints (namely, that contact wrenches lie inside their wrench cone) are straightforward to calculate, at the cost of a large number of control variables.

Another line of research seeks to reduce both control variables and contact constraints at the center of mass (COM), *i.e.*, focusing on *centroidal dynamics* [4]. In [5], ZMP support areas were generalized to multi-contact and applied to whole-body motion generation, but it was observed that these areas vary with the position of the COM. This is indeed a general phenomenon: once reduced at the COM, contact stability constraints yield *quadratic* inequalities, with a product between COM position and acceleration. In [6], these inequalities were kept linear by bounding variations of a nonlinear term, resulting in a ZMP controller that can raise or lower its COM. Polyhedral boundaries were also used in [7] to formulate a linear MPC problem controlling 3D COM accelerations to locomot: in multi-contact.

Yet, all of the works [1], [2], [5], [6], [7], [8], [9] are based on pre-defined timings. In the literature, the alternative to fixed timings seems to be nonlinear optimization [10], [11], [3]. Walking on non-flat terrains was showed in [10] using a SLIP model and on-line foot-step planning, solved using the Covariance Matrix Adaptation method. Meanwhile, [11] showcased a broad set of tasks, but noted that the performance and numerical stability of nonlinear solvers were still problematic. Latest developments [9], [8] reported that a few iterations of a Sequential Quadratic Programming solver can be performed fast enough for the control loop; yet both works used pre-defined timings.

In the present work, we explore another way to linearize quadratic constraints at the center of mass; namely, by alternating path interpolation with trajectory retiming. A key feature of this approach is that all timings (step durations, COM transfer durations, etc.) are produced as *output* to our optimization problem, which allows the controller to discover suitable timings automatically from the terrain topology and robot dynamics used to formulate its constraints.

Our contribution is twofold. First, building upon recent advances in contact-stability cone computations, we develop a new formulation for TOPP of COM trajectories which solve faster than the state-of-the-art. Second, we leverage these fast computations to design a multi-contact model-predictive controller based on TOPP (TOPP-MPC for short).

II. Background

A. Contact stability

Let m denote the mass of the robot and G its center of mass (COM). Denote by O the origin of the inertial frame and $\boldsymbol{p}_A$ the coordinate of a point in this frame (so that $\boldsymbol{p}_O = \mathbf{0}$). The Newton-Euler equations of movement of the robot are

$$\begin{bmatrix} m\ddot{\boldsymbol{p}}_G \\ \dot{\mathbf{L}}_G \end{bmatrix} = \begin{bmatrix} m\boldsymbol{g} \\ \mathbf{0} \end{bmatrix} + \boldsymbol{w}_G, \tag{1}$$

where $\mathbf{L}_G$ denotes the angular momentum of the robot around G, $\boldsymbol{g}$ the gravity vector and $\boldsymbol{w}_G$ the net contact

*This work is supported in part by H2020 EU project COMANOID http://www.comanoid.eu/, RIA No 645097.

[1]Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), CNRS / Université de Montpellier, France.

[2]School of Mechanical and Aerospace Engineering, Nanyang Technological University, Singapore.

Corresponding author: stephane.caron@normalesup.org

wrench taken at G. The latter is given by:

$$\boldsymbol{w}_G := \sum_{i=0}^{K} \begin{bmatrix} \boldsymbol{f}_i \\ \overrightarrow{GC_i} \times \boldsymbol{f}_i \end{bmatrix},$$

where $\boldsymbol{f}_i$ is the contact force exerted onto the robot at the i-th contact point C_i (this formulation includes surface contacts, see *e.g.* [12]). Next, one can rewrite the Newton-Euler equations (1) at a fixed reference point O as:

$$\boldsymbol{w}_O = \begin{bmatrix} m(\ddot{\boldsymbol{p}}_G - \boldsymbol{g}) \\ \dot{\mathbf{L}}_G + m\boldsymbol{p}_G \times (\ddot{\boldsymbol{p}}_G - \boldsymbol{g}) \end{bmatrix}. \tag{2}$$

Next, under the assumption of linearized friction cones, *valid* contact wrenches (*i.e.*, corresponding to contact forces that lie inside their respective linearized friction cones) are exactly characterized by

$$\mathbf{A}_O \boldsymbol{w}_O \leq \mathbf{0}, \tag{3}$$

where $\mathbf{A}_O$ is the matrix of the Contact Wrench Cone (CWC), which can be computed based uniquely on the positions and orientations of the contacts [13].

B. TOPP and TOPP-Polygon

Consider a robot with n degrees of freedom and a path $\boldsymbol{q}(s)_{s\in[0,1]}$ in its configuration space. Assume that constraints on the robot motion along the path can be expressed in the form:

$$\ddot{s}\boldsymbol{a}(s) + \dot{s}^2\boldsymbol{b}(s) + \boldsymbol{c}(s) \leq 0. \tag{4}$$

Finding the time-optimal parameterization $s(t)_{t\in[0,T]}$ subject to constraints (4) is the classical TOPP problem in robotics. Efficient methods have been developed to address this problem, see *e.g.* [14] for a historical review.

A wide range of constraints can be put into the form of (4), including pure acceleration bounds, torque bounds for serial manipulators [15], contact-stability constraints for humanoid robots in single- and multi-contact [12], [13], etc. Constraints on overactuated systems, such as closed-chain manipulators and humanoid robots in multi-contact, cannot be put into the form of (4). Rather, such constraints can be expressed in a more general form [16] as:

$$(\ddot{s}, \dot{s}^2) \in \mathcal{P}(s), \tag{5}$$

where $\mathcal{P}(s)$ is a convex polygon in the $(\ddot{s}, \dot{s}^2)$-plane. In [17], the authors developed TOPP-Polygon, an extension of the numerical integration method [15], [14] to the case of "polygon constraints" of the form (5). Because it is based on direct integration, this method can find time-optimal parameterizations orders of magnitude faster than its counterparts based on convex optimization [16], [18].

Historically, these methods have been used in the contex of motion planning rather than control. The main bottleneck that prevented the application of TOPP-Polygon to real-time applications is the polygon reduction step (computing polygons $\mathcal{P}(s)$ from contact-stability constraints), which could take up to tens of milliseconds per path discretization step. In this paper, we reduce this time to less than a millisecond.

III. TOPP FOR MULTI-CONTACT LOCOMOTION

A. Reduction of TOPP Polygons

We start from the TOPP reduction presented in [13]. Consider a path $\boldsymbol{p}_G(s)_{s\in[0,1]}$ of the center of mass. Differentiating twice, one obtains

$$\ddot{\boldsymbol{p}}_G = \boldsymbol{p}_{G_s}\ddot{s} + \boldsymbol{p}_{G_{ss}}\dot{s}^2, \tag{6}$$

where the subscript s denotes differentiation with respect to the path parameter.

Assume that the angular momentum at the center of mass is regulated to a constant value ($\dot{\mathbf{L}}_G = \mathbf{0}$), which corresponds to the Linear Pendulum Mode. Substituting the expression of $\ddot{\boldsymbol{p}}_G$ into the equation of motions (2), one has

$$\boldsymbol{w}_O = \begin{bmatrix} m(\boldsymbol{p}_{G_s}\ddot{s} + \boldsymbol{p}_{G_{ss}}\dot{s}^2 - \boldsymbol{g}) \\ \boldsymbol{p}_G \times m(\boldsymbol{p}_{G_s}\ddot{s} + \boldsymbol{p}_{G_{ss}}\dot{s}^2 - \boldsymbol{g}) \end{bmatrix}. \tag{7}$$

Thus, the contact-stability condition (3) can be rewritten as

$$\ddot{s}\mathbf{A}_O \begin{bmatrix} \boldsymbol{p}_{G_s} \\ \boldsymbol{p}_G \times \boldsymbol{p}_{G_s} \end{bmatrix} + \dot{s}^2\mathbf{A}_O \begin{bmatrix} \boldsymbol{p}_{G_{ss}} \\ \boldsymbol{p}_G \times \boldsymbol{p}_{G_{ss}} \end{bmatrix} \leq \mathbf{A}_O \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{p}_G \times \boldsymbol{g} \end{bmatrix}, \tag{8}$$

which has the canonical form (4) of TOPP. Equation (8) usually contains a large number of inequalities (up to 150 in double rectangular contact). Usual TOPP solvers (either based on numerical integration [14] or on convex optimization [16]) cannot solve this large path parameterization problem in less than a few seconds, which makes them unpractical for MPC. In [16], the author proposed an iterative method to prune the inequalities before applying TOPP. The complexity of his algorithm is $O(KN)$, where N is the initial number of inequalities and K is the maximum number of intermediate inequalities.

In the context of computing ZMP support areas and COM acceleration cones, the authors of [7] remarked that pruning inequalities as in [16] actually amounts to finding the convex hull of the dual of these inequalities. They could then compute their volumes using state-of-the-art convex hull algorithms, with complexity $O(N \log N)$ and for which fast implementations are readily available. Here, we apply this idea to TOPP-Polygon reduction.

Specifically, Equation (8) can be put in the canonical form used in [7]

$$\mathbf{B}[\ddot{s}\ \dot{s}^2]^\top \leq \boldsymbol{c} \tag{9}$$

where the matrix $\mathbf{B}$ and vector $\boldsymbol{c}$ are defined by:

$$\mathbf{B}, \boldsymbol{c} := \mathbf{A}_O \begin{bmatrix} \boldsymbol{p}_{G_s} & \boldsymbol{p}_{G_{ss}} \\ \boldsymbol{p}_G \times \boldsymbol{p}_{G_s} & \boldsymbol{p}_G \times \boldsymbol{p}_{G_{ss}}, \end{bmatrix}, \mathbf{A}_O \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{p}_G \times \boldsymbol{g} \end{bmatrix}.$$

If the polygon thus describes contains the origin $(0, 0)$ in its interior, *i.e.*, if $\boldsymbol{c} \geq \mathbf{0}$, then a convex hull algorithm can be run on the rows of $\mathbf{B}$ (dual vectors) to enumerate polygon edges [7]. Intersecting consecutive edges in the counter counterclockwise enumeration provided by off-the-shell 2D convex hull algorithms finally provides the list of vertices. When $(0, 0)$ is not in the interior of the polygon, [7] noted that it is necessary to compute an interior point in order to apply this algorithm. The condition $\boldsymbol{c} \geq \mathbf{0}$ describes the Static-Equilibrium Prism (SEP) [19], which contains all

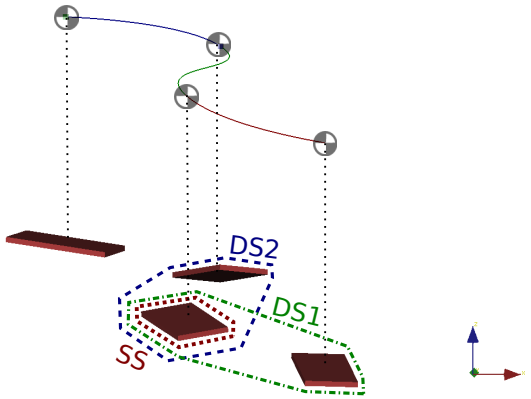

Fig. 1. Contact locations and interpolated COM paths for the test case reported in Table I. Contacts corresponding to the successive Double-Support (DS) and Single-Support (SS) phases are contoured in dotted lines.

TABLE I

POLYGON SIZES AND COMPUTATION TIMES FOR POLYGON REDUCTION BY CONVEX HULL (CH) AND BRETL & LALL'S METHOD (B&L)

Contact	# ineq. bef.	# ineq. aft.	Hull	B&L
Single (SE)	16	3.6 ± 0.5	0.46 ms	1.51 ms
Single (NSE)	16	3.0 ± 0.0	0.90 ms	1.27 ms
Double	145	6.1 ± 1.5	0.60 ms	6.42 ms

COM positions where the robot can stop. When $\boldsymbol{p}_G$ belongs to the SEP, the convex-hull algorithm can be applied readily, but one needs to:

- find a point $\mathring{\boldsymbol{x}}$ such that $\mathbf{B}\mathring{\boldsymbol{x}} < \boldsymbol{c}$;
- apply the convex-hull algorithm to the polygon defined by $\mathbf{B}\boldsymbol{x}' \leq \boldsymbol{c}'$, where $\boldsymbol{c}' := \boldsymbol{c} - \mathbf{B}\mathring{\boldsymbol{x}}$;
- translate the resulting polygon by $\mathring{\boldsymbol{x}}$.

To find an interior point, we use the following procedure inspired from [19], [16]: first, we find three extremal points $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3$ of the polygon by solving the linear program:

$$\max_{\boldsymbol{x}} \ \boldsymbol{v}_i^\top \boldsymbol{x} \quad \text{subject to} \quad \mathbf{B}\boldsymbol{x} \leq \boldsymbol{c},$$

where $\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3$ are three planar vectors pointing in different directions; then, we choose $\mathring{\boldsymbol{x}} := (\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3)/3$.

Alternatively, one can compute the polygon directly from (8) using the polytope projection method [19], [16]. Table I reports an experimental comparison of the two methods (convex hull versus polytope projection), in three different scenarios: single-contact in static equilibrium (SE), single-contact non-static-equilibrium (NSE) and double contact. We report the number of inequalities before and after pruning, averaged across multiple positions $\boldsymbol{p}_G$. Computation times are averaged across 100 iterations. The convex-hull method outperforms the polytope projection method in all scenarios, and significantly so in double contact. Note that the convex-hull method took more time in the non-static-equilibrium scenario than in the other scenarios because of the overhead associated with the search for the interior point. Contact locations and implementation details are distributed in our public code repository [20].

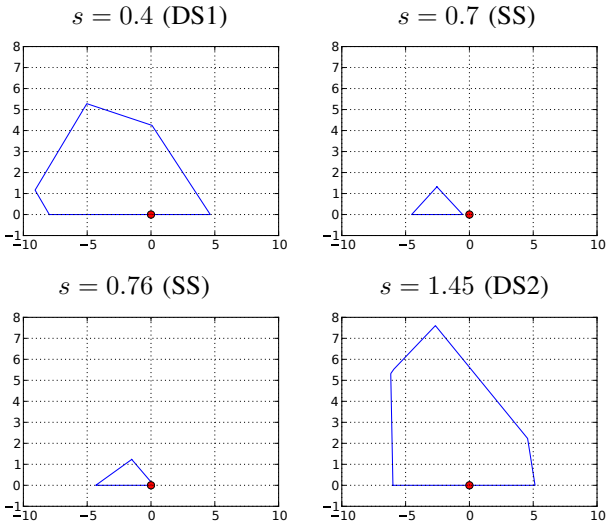

Fig. 2. Constraint polygons in the $(\ddot{s}, \dot{s}^2)$-plane for various values of s. Here, the contact switches happen at $s_1 = 0.67$ (DS1 to SS) and $s_2 = 1.33$ (SS to DS2). Note that the beginning of the single-contact phase is not in static equilibrium, as illustrated by the polygon at $s = 0.7$ which does not contain the origin (red dot).

B. TOPP through contact changes

Consider a smooth three-dimensional COM path $\boldsymbol{p}_G(s)_{s\in[0,1]}$ and a sequence of foot stances DS1–SS–DS2, where DS and SS stand respectively for double- and single-support, see Fig. 1. Assume that one switches from DS1 to SS at path position s_1 and from SS to DS2 at path position s_2. Note that s_1 and s_2 correspond to two geometric positions and do not include any timings.

Consider now the TOPP problem *through contact switches*. The contact-stability matrices (computed only once per stance based on footstep locations [13]) $\mathbf{A}_{\text{DS1}}$, $\mathbf{A}_{\text{SS}}$, $\mathbf{A}_{\text{DS2}}$ are used in the path portions $[0, s_1]$, $[s_1, s_2]$, $[s_2, 1]$ respectively. Fig. 2 shows examples of polygons computed in each portion. Next, TOPP can be run on the full path $[0, 1]$, subject to the constraints provided by the polygons. The Maximum Velocity Curv and profiles computed by TOPP are shown in Fig. 3.

C. Geometric nature of contact switches

As noted above, the contact switches are given a priori by the positions s_1 and s_2 on the paths, which are geometric objects. This is a key feature of our approach.

Single-support phases allow bipeds to transfer one foot (the swing foot) to a new foothold location while the other (the support foot) stays fixed to ensure contact stability. A central question is then: how should the COM move during these phases? One answer is to maintain it inside the Static-Equilibrium Prism (SEP) of the support foot,[1] a behavior that can be observed in many walking patterns reported in the literature (see *e.g.*Fig. 4 in [21], Fig. 1 in [22], Fig. 2 and 3 in [8], ...). However, having the COM in the SEP is an

[1] When walking on horizontal floors, this is equivalent to keeping the COM over the single-foot support area.

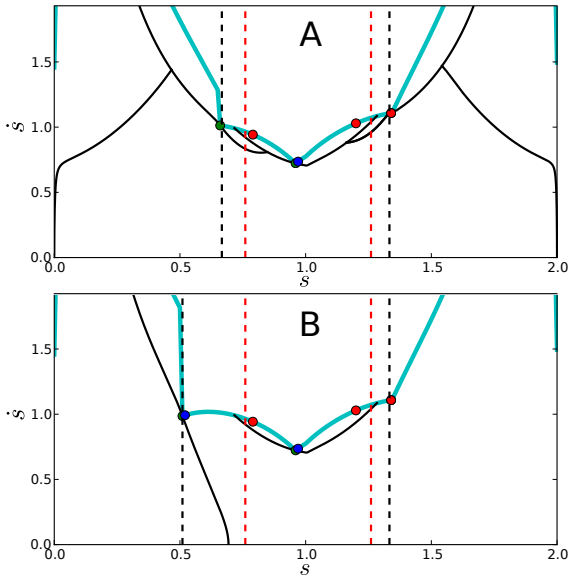

Fig. 3. **A:** Integrated profiles in the $(s, \dot{s})$-plane. By Pontryagin's Maximum Principle, the time-optimal profile is obtained by integrating alternatively maximum and minimum accelerations (black curves). The switches (colored dots) between max and min accelerations are to be found on the Maximum Velocity Curve (cyan). For more details about the TOPP algorithm, the reader is referred to [14]. Black vertical dashed lines indicate the stance changes ($s_1 = 0.67$ and $s_2 = 1.33$). Red vertical dashed lines indicate the boundaries of the static equilibrium portion ($s_1^* = 0.76$ and $s_2^* = 1.26$): any position between the two red lines is in static equilibrium. Note that the path is time-parameterizable despite having non-quasi-statically stable portions ($[0.67, 0.76]$ and $[1.26, 1.33]$). **B:** Switching to the single-contact stance too early (black vertical dashed line at $s_1 = 0.51$) leads to the path being non time-parameterizable: the max acceleration profile hits zero around $s = 0.7$.

indicator of *quasi-static* walking. It precludes the discovery of dynamic walking patterns, where the COM need not even enter the SEP of the support foot if foot swings are fast enough. Our framework allows the discovery of dynamic motions where the COM leaves the SEP even during single-support phases. We achieve this geometrically by varying the parameters s_1 and s_2 to achieve different behaviors.

Let s_1^* and s_2^* denote respectively the first and the second intersection of the COM path with the Static-Equilibrium Prism (s_1^* and s_2^* are represented by the red vertical lines in Fig. 3A). Choosing $s_1 = s_1^*$ and $s_2 = s_2^*$ corresponds to the "safe" quasi-static executable option. By contrast, choosing $s_1 < s_1^*$ and $s_2 > s_2^*$ gives rise to portions of the path that are not quasi-statically executable. However, if the path is time-parameterizable, then there will exist a dynamically-stable execution. Furthermore, as s_1 gets smaller and and s_2 gets larger, the time-parameterized behavior will be more aggressive, up to a point when no valid time-parameterization exists. For the same path as in Fig. 1, if one chooses $s_1 = 0.51$ instead of $s_1 = 0.67$, then the path becomes non time-parameterizable, see Fig. 3B.

IV. Locomotion by Retimed Predictive Control

The generic control loop of a predictive controller can be described as follows: at each iteration,

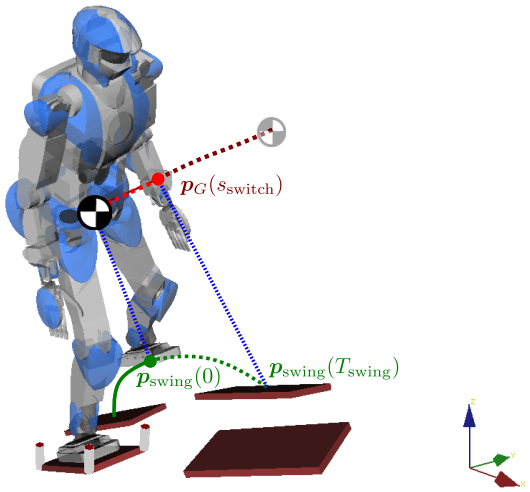

Fig. 4. Preview of swing-foot (green) and COM (ligh and dark red) trajectories during a single-support phase. Retiming the swing foot path under conservative constraints yields a swing duration T_{swing}. This value is then converted into retiming constraints on the COM trajectory, so that TOPP uses the single-support CWC up to $\boldsymbol{p}_G(s_1)$ (light red) and the double-support one afterwards (dark red).

- generate or update a trajectory of future system dynamics (the *preview trajectory*) leading it to a desired configuration, then
- apply the *first controls* of this trajectory until the next iteration.

To keep up with the high rates of a control loop, the preview trajectory is commonly found as the solution to an optimization problem, be it a linear-quadratic regulator [23], [24], a quadratic program [1], [7] or a generic optimal-control problem [25]. Here, our optimization unfolds in two steps: interpolating preview paths, and retiming them using TOPP.

For humanoid walking, there are two kinds of trajectories to interpolate: COM trajectories, and swing-foot trajectories in single-support phases. Similarly to [26], we adopt a simplified dynamics model where the COM is represented by a point-mass and the swing foot by a rigid body (Fig. 4). Contact dynamics are reduced at the COM under zero angular momentum as described in Section III-A, while joint kinematic and dynamic constraints are modeled by workspace velocity and acceleration limits on the swing foot. All of these constraints are taken into account when retiming by TOPP. Retimed foot and COM accelerations are then sent as reference to a whole-body controller that converts them into joint accelerations sent to the robot. Our complete pipeline is summarized in Fig. 5. We provide technical details on its consecutive steps in the following subsections.

A. Finite State Machine

Our locomotion state machine is simpler than those of previous works like [7] as it is does not need to take time into consideration. Transitions between single- and double-support phases are triggered by straightforward geometric

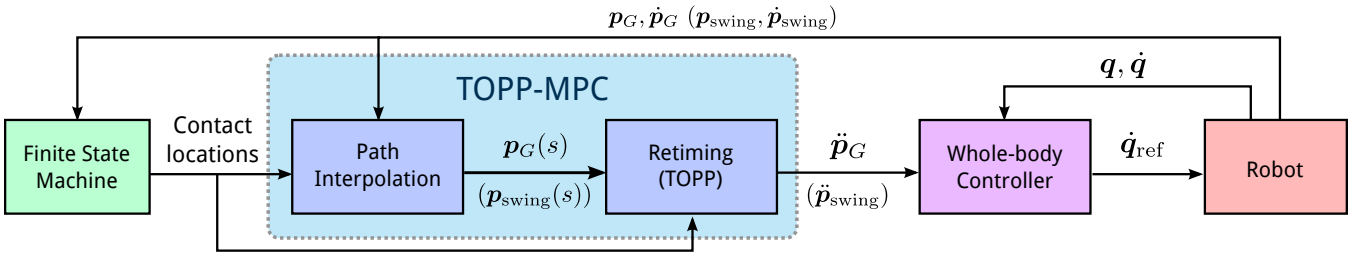

Fig. 5. Overview of the predictive control pipeline. A finite state machine sends the current walking phase (single- or double-support) as well as contact locations to TOPP-MPC. COM and (in single support) swing-foot trajectories are then interpolated from the current robot state to a desired future state, and sent to TOPP. The initial COM and foot accelerations of the retimed trajectory are finally converted to joint accelerations by a whole-body controller and sent to the robot.

conditions, namely:

- SS $\rightarrow$ DS when $\|\boldsymbol{p}_{\text{swing}} - \boldsymbol{p}_{\text{swing}}^{\text{goal}}\| \leq \epsilon$, *i.e.*, when the swing-foot touches down on its target foothold.
- DS $\rightarrow$ SS when $\|\boldsymbol{p}_G - \boldsymbol{p}_G^{\text{goal}}\| \leq d_{\text{trans}}$, *i.e.*, when the COM enters the vicinity of its preview target.

While ϵ should be a small value scaled upon the ability of the robot to estimate its foot displacements, d_{trans} depends on the contact geometry and preview target velocity. We used $d_{\text{trans}} = 5$ cm in our experiments, which corresponds roughly to the half-width of static-equilibrium polygons.

B. Path Interpolation

Path interpolation is constrained by our closed feedback loop: the beginning of the path $\boldsymbol{p}(s)$ needs to coincide with the current robot state $(\boldsymbol{p}_{\text{cur}}, \dot{\boldsymbol{p}}_{\text{cur}})$, where we use $\boldsymbol{p}$ to refer equivalently to the COM position $\boldsymbol{p}_G$ or swing-foot position $\boldsymbol{p}_{\text{swing}}$. Then,

$$\boldsymbol{p}(0) = \boldsymbol{p}_{\text{cur}} \tag{10}$$

$$\cos\left(\widehat{\boldsymbol{p}_s(0), \dot{\boldsymbol{p}}_{\text{cur}}}\right) = 1 \tag{11}$$

The latter states that the initial path tangent must be positively aligned with the current velocity (the norms of the two vectors will be matched as well at the retiming stage, so that the output trajectory velocity $\dot{\boldsymbol{p}}(0) = \boldsymbol{p}_s(0)\dot{s}(0) = \dot{\boldsymbol{p}}_{\text{cur}}$).

Target positions and velocities $(\boldsymbol{p}_{\text{goal}}, \dot{\boldsymbol{p}}_{\text{goal}})$ by the end of the preview window are also provided:

- for swing-foot paths, the target position is taken at the contact location and the target velocity is zero;
- for COM paths, the target position is taken at the center of the Static-Equilibrium Prism for the next single-support phase, while the target velocity is a fixed-norm vector parallel to the contact surface and going forward in the direction of motion.

Boundary conditions at the end of the preview path are then:

$$\boldsymbol{p}(1) = \boldsymbol{p}_{\text{goal}} \tag{12}$$

$$\cos\left(\widehat{\boldsymbol{p}_s(1), \dot{\boldsymbol{p}}_{\text{goal}}}\right) = 1 \tag{13}$$

Like [16], we interpolate the path $\boldsymbol{p}(s)$ by a cubic Hermite curve, *i.e.*, a third-order polynomial $H(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{p}_1, \boldsymbol{v}_1)$ such that $H(0) = \boldsymbol{p}_0$, $H'(0) = \boldsymbol{v}_0$, $H(1) = \boldsymbol{p}_1$ and $H'(1) = \boldsymbol{v}_1$. This construction ensures that position constraints (10) and (12) are satisfied. However, velocity constraints (11) and (13) are not vector equalities: they only impose vector directions and signs, leaving norms as a free parameter. Our path interpolation problem therefore has two degrees of freedom: we can select any path $I_{\lambda,\mu}(s)$ given by

$$I_{\lambda,\mu} = H(\boldsymbol{p}_0,\, \lambda\boldsymbol{v}_0,\, \boldsymbol{p}_1,\, \mu\boldsymbol{v}_1) \tag{14}$$

where $\lambda, \mu > 0$, $(\boldsymbol{p}_0, \boldsymbol{p}_1) = (\boldsymbol{p}_{\text{cur}}, \boldsymbol{p}_{\text{goal}})$ and $(\boldsymbol{v}_0, \boldsymbol{v}_1) = (\dot{\boldsymbol{p}}_{\text{cur}}, \dot{\boldsymbol{p}}_{\text{goal}})$. The values of λ and μ can be selected so as to optimize additional path smoothness criteria. This problem has been studied in computer graphics, where Yong et al. [27] introduced the Optimized Geometric Hermite (OGH) curves that minimize *strain energy* $\int_0^1 \|\boldsymbol{p}_{ss}(s)\|^2 \mathrm{d}s$ of the path:

$$\text{OGH}(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{p}_1, \boldsymbol{v}_1) = \arg\min_{\lambda,\mu} \int_0^1 \|I''_{\lambda,\mu}(s)\|^2 \mathrm{d}s \tag{15}$$

An interesting feature of this problem is that the values $\lambda^*_{\text{OGH}}, \mu^*_{\text{OGH}}$ that yield this minimum are found analytically from boundary conditions $(\boldsymbol{p}_{\text{cur}}, \dot{\boldsymbol{p}}_{\text{cur}}, \boldsymbol{p}_{\text{goal}}, \dot{\boldsymbol{p}}_{\text{goal}})$, so that there is no need for numerical optimization at runtime.

We experimented with OGH curves in our framework, but found them to be rather unfit for multi-contact trajectory retiming. The reason behind this is that, when boundary velocity vectors $\dot{\boldsymbol{p}}_{\text{cur}}$, $\dot{\boldsymbol{p}}_{\text{goal}}$ deviate significantly from the direction of motion $\Delta = \boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{cur}}$, OGH curves tend to start or end with very sharp accelerations. Such accelerations have little impact on strain energy but jeopardize trajectory retiming.

To avoid this phenomenon, we optimize another criterion. We call *Hermite curves with Optimized Uniformly-Bounded Accelerations* (HOUBA) the polynomials defined by:

$$\text{HOUBA}(\boldsymbol{p}_0, \boldsymbol{v}_0, \boldsymbol{p}_1, \boldsymbol{v}_1) := \arg\min_{\lambda,\mu} \max_{s\in[0,1]} \|I''_{\lambda,\mu}(s)\|^2 \tag{16}$$

Because this bound is uniform rather than integral, it is less prone to sharp boundary accelerations. Optimum values λ^*_{HOUBA} and μ^*_{HOUBA} corresponding to this criterion are given by (see Appendix I for calculations):

$$\lambda^*_{\text{HOUBA}} = 6 \cdot \frac{3(\Delta\cdot\boldsymbol{v}_0)(\boldsymbol{v}_1\cdot\boldsymbol{v}_1) - 2(\Delta\cdot\boldsymbol{v}_1)(\boldsymbol{v}_0\cdot\boldsymbol{v}_1)}{9\|\boldsymbol{v}_0\|^2\|\boldsymbol{v}_1\|^2 - 4(\boldsymbol{v}_0\cdot\boldsymbol{v}_1)^2} \tag{17}$$

$$\mu^*_{\text{HOUBA}} = 6 \cdot \frac{-2(\Delta\cdot\boldsymbol{v}_0)(\boldsymbol{v}_0\cdot\boldsymbol{v}_1) + 3(\Delta\boldsymbol{v}_1)\|\boldsymbol{v}_0\|^2}{9\|\boldsymbol{v}_0\|^2\|\boldsymbol{v}_1\|^2 - 4(\boldsymbol{v}_0\cdot\boldsymbol{v}_1)^2} \tag{18}$$

We note how these two formulas have the same structure as

those reported for λ^*_{OGH} and μ^*_{OGH} [27], yet with different integer coefficients. In what follows, we interpolate all our swing-foot position and COM paths by HOUBA curves.

C. Retiming with Contact Switches

Let us assume that the robot is undergoing a single-support phase. Retiming the COM and swing-foot paths are not two independent operations: TOPP contact constraints for COM retiming should be computed using the single-support CWC while the foot is in the air, and the double-support CWC once contact is made.

One way to take this coupling into account is to *synchronize* both path retimings. Denote by s and s_{swing} the indexes of the COM and swing-foot paths respectively. Synchronization amounts to setting $s_{\text{swing}} = s/s_{\text{trans}}$ and reformulating all foot constraints (limited workspace velocity and acceleration) on $(\ddot{s}_{\text{swing}}, \dot{s}^2_{\text{swing}})$ as constraints on $(\ddot{s}, \dot{s}^2)$. This approach has the merit of conciseness and computational efficiency, but we chose not to do so due to an undesired side effect: at the end of the swing-foot trajectory, we want the foot velocity to go to zero so as to avoid impacts, but under synchronization this implies that the COM velocity goes to zero as well, and thus that the contact switch is quasi-static.

To avoid this issue, we adopted the two-stage retiming strategy depicted in Fig. 4:

1) first, retime the swing-foot trajectory, and let T_{swing} denote the duration of the resulting trajectory;
2) second, retime the COM trajectory under the additional constraint that $t(s_{\text{trans}}) > T_{\text{swing}}$, *i.e.*, that the retimed COM trajectory will spend *at least* T_{swing} of its time in the single-support section $s \in [0, s_{\text{trans}}]$.

We will now see that, given the initial path velocity $\dot{s}_0 = \|\dot{\boldsymbol{p}}_{\text{cur}}\|/\|\boldsymbol{p}_s(0)\|$, the constraint $t(s_{\text{trans}}) > T_{\text{swing}}$ can be formulated as a path acceleration constraint $\ddot{s} \leq \ddot{s}_{\text{max}}$ suitable for TOPP.

Property 1: A sufficient condition for $t(s_{\text{trans}}) > T_{\text{swing}}$ is that $\ddot{s} \leq \ddot{s}_{\text{max}}$, with

$$\ddot{s}_{\text{max}} \stackrel{\text{def}}{=} \frac{1}{2s_{\text{trans}}} \left[\left(\frac{s_{\text{trans}}}{T_{\text{swing}}} \right)^2 - \dot{s}_0^2 \right] \tag{19}$$

The proof of this property is given in Appendix II.

In the $(\ddot{s}, \dot{s}^2)$-plane where TOPP-polygons are computed, Equation (19) amounts to adding two vertical boundaries at $\ddot{s} = \pm\ddot{s}_{\text{max}}$ and can be done readily using the existing software.

D. Whole-body Controller

The last step of our pipeline converts COM and foot reference accelerations $\ddot{\boldsymbol{p}}_G, \ddot{\boldsymbol{p}}_{\text{swing}}$ into joint commands that are finally sent to the robot's motor controllers. We used our own differential inverse kinematics solver for this, which is implemented in the *pymanoid* library.[2] The solver is based on a single-layer Quadratic Program (QP) with weighted tasks (see [5] for details). We used the following tasks, by decreasing priority: support foot contact, swing foot tracking, COM tracking, constant angular-momentum and joint-velocity minimization, with respective weights $10^4, 100, 10, 1.$. We used joint-velocity minimization as the regularizing task. Each iteration of the QP solver updates a vector of joint-angle velocities $\dot{\boldsymbol{q}}_{\text{ref}}$ which is finally sent to the robot.

[2] https://github.com/stephane-caron/pymanoid

V. Experiments

We implemented the whole pipeline described so far and ran simultations with a model of the HRP-4 humanoid robot. All our code is publicly released at [20]. The reader may refer to it for any technical point that would not be detailed below.

A. Preview targets tuning

The only external input required by our controller is a sequence of foothold locations, which we assume to be provided by a parallel perception-and-planning module. From these footholds, target COM and swing-foot positions and velocities are computed as follows. Let $(\boldsymbol{t}_i, \boldsymbol{b}_i, \boldsymbol{n}_i)$ denote the frame of the i^{th} contact in the sequence, where $\boldsymbol{n}_i$ is pointing upward. Then, for the DS phase ending on this contact and the SS phase of the $(i-1)^{\text{th}}$ contact, preview targets are set to:

- $\boldsymbol{p}_G^{\text{goal}}$ is the center of the SEP of the next SS footstep
- $\dot{\boldsymbol{p}}_G^{\text{goal}} = v_{\text{ref}}\, \boldsymbol{t}_i$, where $v_{\text{ref}} = 0.4$ m.s^{-1}
- $\boldsymbol{p}_{\text{swing}}^{\text{goal}}$ is the position of the i^{th} contact
- $\dot{\boldsymbol{p}}_{\text{swing}}^{\text{goal}} = \alpha \boldsymbol{t}_i - (1-\alpha)\boldsymbol{n}_i$, where α tunes the forward inclination of the foot landing velocity.

Similarly, we used $\boldsymbol{v}_0 = \beta \boldsymbol{t}_{i-1} + (1-\beta)\boldsymbol{n}_{i-1}$ for the foot takeoff direction. In the accompanying video, we set $\alpha = 0.5$ and $\beta = 0.3$. Recall that only the signs and directions of these velocity vectors matter since we are using HOUBA curves for interpolation.

B. TOPP tuning

Once a path is interpolated, we retime it by TOPP using a discretization step $\mathrm{d}s = 0.1$. As a consequence of the minimum-time criterion, retimed trajectories always saturate their inequality constraints, which means in particular that output contact wrenches $\boldsymbol{w}_G$ will lie on the boundary of the CWC. This behavior is undesirable as it makes our controller sensitive to disturbances. To palliate this, when computing the CWC matrix $\mathbf{A}_O$ given to TOPP, we scale foothold dimensions and friction coefficients by 0.75. This results in a conservative contact-stability criterion, leaving room for error compensation at runtime.

C. Simulations

Fig. 6 shows one application of our controller on an artificial scenario where the robot has to climb up and down a series of meter-high hills, with slopes ranging from 0° to 30°. In this example, swing-foot accelerations were bounded by $\|\ddot{\boldsymbol{p}}_{\text{swing}}\| \leq 5$ m.s^{-1}. Phase timings were automatically derived by the controller. On average, DS phases in this setting last 0.88 ± 0.06 s, while SS phases last 0.65 ± 0.22 s.

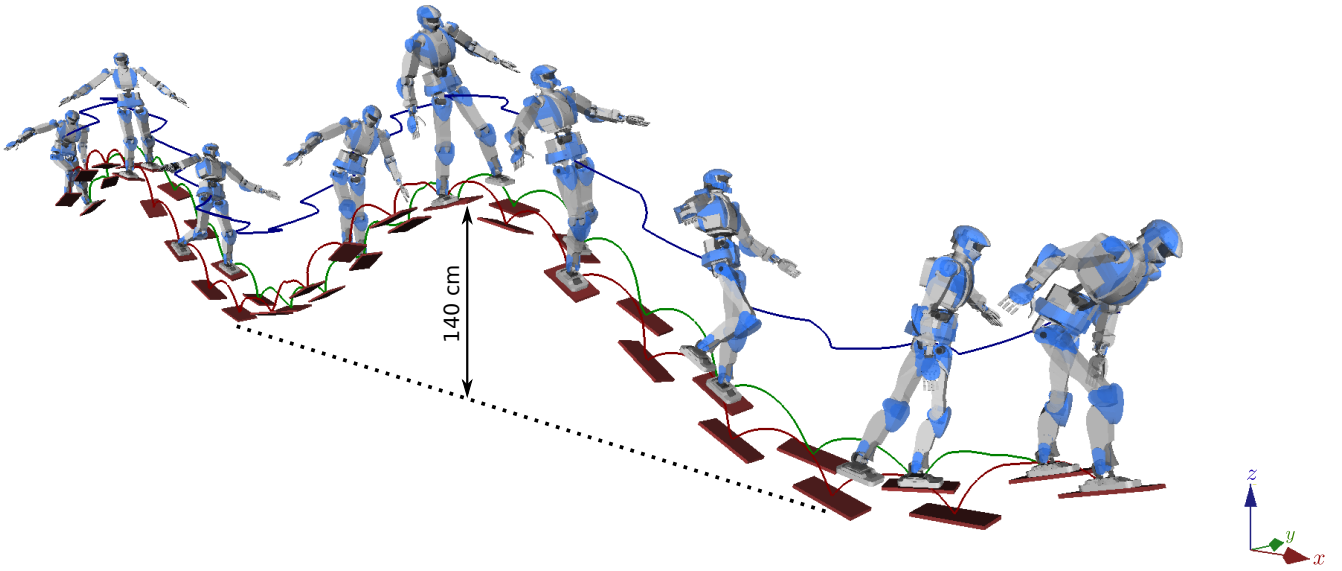

Fig. 6. **Locomotion over uneven terrain by HRP-4 running the TOPP-MPC controller.** The only external input to the system is a foothold sequence. From this, the controller automatically deduces the *timings* of its single- and double-support phases (which depend on terrain topology), along with COM and swing-foot motions that guarantee contact stability. TOPP previews are run in a closed feedback loop with an update period of 40 ms. In the simulations depicted above, the robot climbs up and down two meter-high hills with slopes ranging from 0° to 30°. Feasibility of the motion has been checked by computing at each time instant valid supporting contact forces, which can be seen in the accompanying video.

TABLE II
TOPP PERFORMANCE IN THE CLOSED MPC FEEDBACK LOOP.

Phase	Convex Hull	Bretl & Lall
DS	18.6 ± 5.5 ms	26.2 ± 7.0 ms
SS	30.0 ± 5.6 ms	38.9 ± 7.5 ms

These statistics are only provided for information, as proper timings depend on terrain topology.

Table II reports times taken to build and solve TOPP instances in this simulation. All computations were run on a laptop computer equiped with an Intel(R) Core(TM) i7-6500U CPU @ 2.50 Ghz. We compare the convex-hull reduction presented in Section III-A with Bretl and Lall's method [19], and observe that it is 20 to 30% faster in practice.

VI. CONCLUSION

We have presented a Model-Predictive Controller whose underlying optimization routine is based on Time-Optimal Path Parameterization (TOPP-MPC). The key feature of TOPP-MPC is that it determines by itself proper timings between contact switches, based on terrain topology and its model of system dynamics. By keeping the problem formulation linear and exploiting recent advances in contact-stability cone computations, we were able to run TOPP computations fast enough for the control loop. We showcased the performance of the overall controller in simulations where the humanoid model HRP-4 climbs up and down a series of hills (Fig. 6 and accompanying video).

There are many avenues for further improvements. By construction, time-optimal parameterizations switch between maximum and minimum acceleration, which causes discontinuities in accelerations and, hence, in contact forces. To mitigate this effect, an active line of research seeks to extend the underlying TOPP routine so as to enforce continuity constraints on accelerations [28] or jerk bounds [29].

REFERENCES

[1] 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 *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4030–4035.

[2] A. Herzog, N. Rotella, S. Schaal, and L. Righetti, "Trajectory generation for multi-contact momentum control," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 874–880.

[3] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3555–3561.

[4] P. M. Wensing and D. E. Orin, "Improved computation of the humanoid centroidal dynamics and application for whole-body control," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1550039, 2016.

[5] S. Caron, Q.-C. Pham, and Y. Nakamura, "ZMP support areas for multi-contact mobility under frictional constraints," *IEEE Transactions on Robotics*, to appear.

[6] C. Brasseur, A. Sherikov, C. Collette, D. Dimitrov, and P.-B. Wieber, "A robust linear mpc approach to online generation of 3d biped walking motion," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 595–601.

[7] S. Caron and A. Kheddar, "Multi-contact Walking Pattern Generation based on Model Preview Control of 3D COM Accelerations," Jul. 2016, submitted. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01349880

[8] K. V. Heerden, "Real-time variable center of mass height trajectory planning for humanoids robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 135–142, Jan 2017.

[9] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–17, 2017.

[10] I. Mordatch, M. De Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 71, 2010.

[11] S. Lengagne, J. Vaillant, E. Yoshida, and A. Kheddar, "Generation of whole-body optimal dynamic multi-contact motions," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1104–1119, 2013.

[12] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015.

[13] ——, "Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics," in *Robotics: Science and System*, 2015.

[14] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, pp. 1533–1540, 2014.

[15] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[16] K. Hauser, "Fast interpolation and time-optimization with contact," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1231–1250, 2014.

[17] Q.-C. Pham and O. Stasse, "Time-optimal path parameterization for redundantly actuated robots: A numerical integration approach," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 3257–3263, 2015.

[18] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

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

[20] [Online]. Available: https://github.com/stephane-caron/topp-mpc

[21] T. Buschmann, R. Wittmann, M. Schwienbacher, and H. Ulbrich, "A method for real-time kineto-dynamic trajectory generation," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE, 2012, pp. 190–197.

[22] C. Santacruz and Y. Nakamura, "Analytical real-time pattern generation for trajectory modification and footstep replanning of humanoid robots," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 2095–2100.

[23] 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 *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1620–1626.

[24] R. Tedrake, S. Kuindersma, R. Deits, and K. Miura, "A closed-form solution for real-time zmp gait generation and feedback stabilization," in *Proceedings of the International Conference on Humanoid Robotics*, 2015.

[25] J. Carpentier, S. Tonneau, M. Naveau, O. Stasse, and N. Mansard, "A versatile and efficient pattern generator for generalized legged locomotion," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 3555–3561.

[26] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1084–1091.

[27] J.-H. Yong and F. F. Cheng, "Geometric hermite curves with minimum strain energy," *Computer Aided Geometric Design*, vol. 21, no. 3, pp. 281–301, 2004.

[28] A. K. Singh and K. M. Krishna, "A class of non-linear time scaling functions for smooth time optimal control along specified paths," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5809–5816.

[29] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*. IEEE, 1993, pp. 725–730.

APPENDIX I
CALCULATION OF HOUBA PARAMETERS

In this problem, we seek to minimize an upper-bound M and the accelerations $\|H''(s)\|^2$ of the curve. By definition of Hermite curves,

$$3H''(0) = -3\Delta + \lambda \boldsymbol{v}_0 + 2\mu \boldsymbol{v}_1 \quad (20)$$

$$3H''(1) = 3\Delta - 2\lambda \boldsymbol{v}_0 - \mu \boldsymbol{v}_1 \quad (21)$$

By convexity of the cost function $\|H''(s)\|^2$, extrema are realized at the boundaries $s \in \{0, 1\}$ of the interval: $\forall s \in [0,1], \|H''(s)\|^2 \leq \max(\|H''(0)\|^2, \|H''(1)\|^2)$. Our problem is therefore the joint minimization of (20)-(21).

By Minkowski inequality,

$$3\|H''(0)\|^2 \leq \|3\Delta - \lambda \boldsymbol{v}_0 - \mu \boldsymbol{v}_1\|^2 + \|\mu \boldsymbol{v}_1\|^2 \quad (22)$$

$$3\|H''(1)\|^2 \leq \|3\Delta - \lambda \boldsymbol{v}_0 - \mu \boldsymbol{v}_1\|^2 + \|\lambda \boldsymbol{v}_0\|^2 \quad (23)$$

Using the symmetry in λ and μ, we reformulate this as the minimization of $E(\lambda, \mu) := \|3\Delta - \lambda \boldsymbol{v}_0 - \mu \boldsymbol{v}_1\|^2 + \frac{1}{2}\lambda \|\boldsymbol{v}_0\|^2 + \frac{1}{2}\mu \|\boldsymbol{v}_1\|^2$. Differentiating with respect to λ and μ yields:

$$\frac{\partial E}{\partial \lambda} = -6(\Delta \cdot \boldsymbol{v}_0) + 9\lambda \|\boldsymbol{v}_0\|^2 + 6\mu(\boldsymbol{v}_0 \cdot \boldsymbol{v}_1) \quad (24)$$

$$\frac{\partial E}{\partial \mu} = -6(\Delta \cdot \boldsymbol{v}_1) + 6\lambda(\boldsymbol{v}_0 \cdot \boldsymbol{v}_1) + 9\mu \|\boldsymbol{v}_1\|^2 \quad (25)$$

Finally, solving for critical points the linear system given by $(24) = 0$, $(25) = 0$ yields the formulas (17) and (18).

APPENDIX II
PROOF OF PROPERTY 1

Using the same notations as [18], let us define $a(s) \stackrel{\text{def}}{=} \ddot{s}$, $b(s) \stackrel{\text{def}}{=} \dot{s}^2$, and denote by $b'(s) \stackrel{\text{def}}{=} \frac{\mathrm{d}b}{\mathrm{d}s}$ and $\dot{b}(s) \stackrel{\text{def}}{=} \frac{\mathrm{d}b}{\mathrm{d}t}$. The definitions of a and b imply that $b'(s) = 2a(s)$, so that

$$b(s) = \dot{s}_0^2 + \int_0^s b'(s)\mathrm{d}s = \dot{s}_0^2 + 2\int_0^s a(s)\mathrm{d}s. \quad (26)$$

An upper bound $a(s) \leq \ddot{s}_{\max}$ then implies that

$$b(s_{\text{trans}}) \leq \dot{s}_0^2 + 2 s_{\text{trans}} \ddot{s}_{\max}. \quad (27)$$

Next, the output switching time $t(s_{\text{trans}})$ can be written as:

$$t(s_{\text{trans}}) = \int_0^{s_{\text{trans}}} \frac{\mathrm{d}s}{\sqrt{b(s)}} \geq \frac{s_{\text{trans}}}{\sqrt{\dot{s}_0^2 + 2 s_{\text{trans}} \ddot{s}_{\max}}} \quad (28)$$

A necessary condition for $t(s_{\text{trans}}) > T_{\text{swing}}$ is thus that $s_{\text{trans}}^2 \geq T_{\text{swing}}(\dot{s}_0^2 + 2 s_{\text{trans}} \ddot{s}_{\max})$. Equation (19) is finally a rewriting of the latter inequality.