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). A key feature of this solution is that, contrary to existing planners where step timings are provided as inputs, here the timing between contact switches is computed as *output* of a fast nonlinear optimization. This is appealing to multi-contact locomotion, where proper timings depend on terrain topology and suitable heuristics are unknown. We show how to formulate legged locomotion as a TOPP problem and demonstrate the behavior of the resulting TOPP-MPC controller in simulations with a model of the HRP-4 humanoid robot.**

I. Introduction

Walking pattern generators on horizontal floors usually take single-support (SS) and double-support (DS) durations as user-defined parameters. Recent works showed how adapting step timings helps recover from disturbances [1], [2], yet using capture-point schemes that are so far limited to walking on flat surfaces. The full locomotory capabilities of humanoids appear in *multi-contact*, where wheeled robots cannot follow. In multi-contact locomotion, walking speed and step timings depend on terrain topology, as illustrated for instance by Naismith's rule: the walking *pace* (in $\mathrm{min.km}^{-1}$) depends affinely on terrain slope.

Model predictive control (MPC) is a paradigm that gives controllers the level of foresight required to tackle the issue of timed walking. Its application to multi-contact locomotion is relatively recent, and can be split in two categories. In one line of research, contact forces are jointly considered as control variables used to optimize a cost function over future whole-body motions [3], [4], [5]. In such formulations, contact feasibility 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* [6]. In [7], 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 feasibility constraints yield *bilinear* inequalities crossing COM position and acceleration. In [8], 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 [9] to formulate a linear MPC problem over 3D accelerations of the COM.

Yet, all of the works [3], [4], [7], [8], [9], [10], [11] are based on pre-defined timings. Set aside flat-floor walking, the alternative to fixed timings seems to be blackbox nonlinear optimization [12], [13], [5]. Walking on non-flat terrains was showed in [12] using a SLIP model and on-line foot-step planning. Lengagne *et al.* [13] showcased a broad set of tasks, but noted that the performance and numerical stability of nonlinear solvers were still problematic. Recent developments [11], [10] performed a few iterations of sequential quadratic programming fast enough for the control loop; yet again using pre-defined step timings.

In this work, we explore an alternative to linearize bilinear 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* of the optimization problem. Our contribution lies in (1) the formulation of legged locomotion as a TOPP problem, including COM-trajectory retiming, contact switches and swing-leg synchronization, and (2) a model predictive controller (called TOPP-MPC) that leverages this solution into a full-fledge multi-contact locomotion controller.

II. Background

A. Contact stability conditions

We wish that contacts stay *fixed* (*i.e.*, neither slip nor tilt) while the robot pushes on them to locomote. The inequality constraints that model this regime are known as *contact-stability* conditions, and can be rewritten at the centroidal level. Consider the Newton-Euler equations of motion:

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

where m is the total robot mass, and G its center of mass (COM) located at $\boldsymbol{p}_G$ (coordinates are given in the inertial frame of origin O), $\mathbf{L}_G$ is the angular momentum of the robot around G, $\boldsymbol{g}$ the gravity vector and $\boldsymbol{w}_G$ the net contact wrench taken at G:

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

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

[1] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM), CNRS–University of Montpellier, France.

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

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

Corresponding author: stephane.caron@normalesup.org

with continuous pressure distributions, see *e.g.* [14]). We can rewrite the Newton-Euler equations (1) at a fixed reference point O as:

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

Under the assumption of linearized friction cones, *feasible* contact wrenches, *i.e.*, those corresponding to contact forces that lie inside their respective 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 net contact wrench cone (CWC), which can be computed based uniquely on the positions and orientations of the contacts. See *e.g.* [15] for a review of the corresponding algorithm.

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 all dynamic constraints on the robot 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 time-optimal path parameterization (TOPP) problem, for which efficient methods have been developed (see [16] 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, contact-stability constraints for humanoid robots in single- [14] and multi-contact [15]. Constraints on redundantly actuated systems (such as humanoids in multi-contact), cannot be put into the form of (4), but rather in the more general form [17]:

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

where $\mathcal{P}(s)$ is a convex polygon in the $(\ddot{s}, \dot{s}^2)$-plane. In [18], the authors developed TOPP-Polygon, an extension of the TOPP algorithm that is able to deal with such polygonal constraints. Its computation bottleneck lied in the enumeration of polygonal constraints $\mathcal{P}(s)$, which could take up to tens of milliseconds per path discretization step. In what follows, we reduce it to less than a millisecond.

III. TOPP for multi-contact locomotion

A. Reduction of TOPP Polygons

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 kept constant ($\dot{\mathbf{L}}_G = \mathbf{0}$). Substituting the expression of $\ddot{\boldsymbol{p}}_G$ into the equation of motion (2), we get:

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

The contact-stability condition (3) thus rewrites to:

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

which has the canonical form (4) of TOPP.

Equation (8) usually contains a large number of inequalities, up to 150 for a rectangular double-support area. Existing TOPP solvers [16], [17]) take several seconds to solve problems with that many inequalities, which would make them unpractical for closed-loop control. Hence the importance of pruning redundant inequalities [17], which is algorithmically equivalent to applying a convex hull to the dual of the problem, as exploited in [9].

This observation applies to the present TOPP problem. Equation (8) can be put in the canonical form:

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

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 described contains the origin $(0, 0)$ (*i.e.*, if $\boldsymbol{c} \geq \mathbf{0}$) then running a convex hull algorithm on the rows of $\mathbf{B}$ (dual vectors) will enumerate edges of the corresponding polygon. Intersecting consecutive edges will then provide the list of vertices. Meanwhile, if $(0, 0)$ is outside of the polygon, it becomes necessary to compute an interior point of the polygon. Following [9], we use its *Chebyshev center* $\hat{\boldsymbol{x}}$, which can be computed by solving a single linear program.

Alternatively, one can compute the polygon directly from (8) using a recursive polytope projection method [19]. Table I reports an experimental comparison of the two methods[1] (convex hull versus polytope projection), in three different scenarios: single-contact in static equilibrium (SE), single-contact non-static-equilibrium (NSE) and double contact. The table reports the number of inequalities before and after pruning, averaged across multiple positions $\boldsymbol{p}_G$, with computation times averaged across 100 iterations. We observe that the convex-hull method outperforms the polytope projection method in all scenarios, and significantly so in double contact.

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[1]	B&L[1]
Single (SE)	16	3.8 ± 0.4	0.4 ms	1.0 ms
Single (NSE)	16	3.5 ± 0.5	0.6 ms	1.2 ms
Double	145	6.1 ± 1.6	0.5 ms	2.0 ms

[1] All computation times reported in this paper were obtained on an Intel(R) Core(TM) i7-6500U CPU @ 2.50 Ghz. Standard deviations for computation times in Table I are not reported as they are all smaller than 0.1 ms.

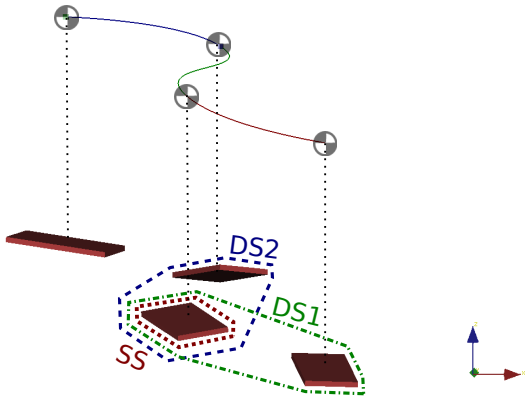

Fig. 1. Contact locations and interpolated COM paths for the test case reported in Table I. Contacts corresponding to the successive Double-Support (DS) and Single-Support (SS) phases are contoured in dotted lines.

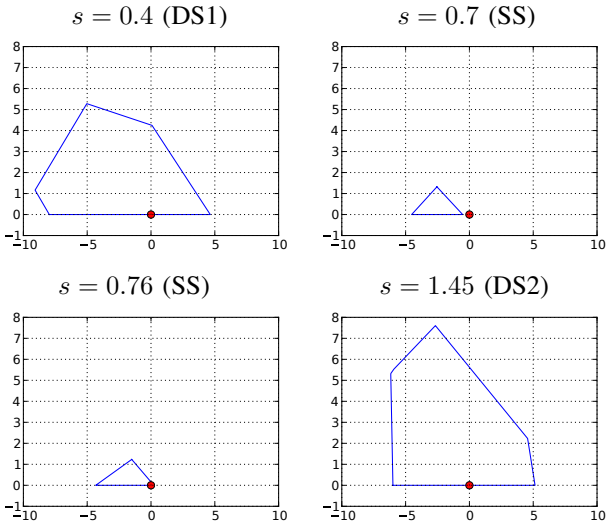

Fig. 2. Constraint polygons in the $(\ddot{s}, \dot{s}^2)$-plane for various values of s. Here, the contact switches happen at $s_1 = 0.67$ (DS1 to SS) and $s_2 = 1.33$ (SS to DS2). Note that the beginning of the single-contact phase is not in static equilibrium, as illustrated by the polygon at $s =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

B. TOPP through contact switches

Consider the smooth three-dimensional COM path $\boldsymbol{p}_G(s)_{s\in[0,1]}$ depicted in Figure 1, along with the sequence of foot stances DS1–SS–DS2. Denote by s_1 (resp. s_2) the path index of the contact switch from DS1 to SS (resp. from SS to DS2). Mind that s_1 and s_2 are two geometric positions and do not include any timing information. Consider now the TOPP problem *through contact switches*. The contact-stability matrices $\mathbf{A}_{\text{DS1}}$, $\mathbf{A}_{\text{SS}}$, $\mathbf{A}_{\text{DS2}}$ (computed only once per stance based on footstep locations [15]) are used over the path intervals $[0, s_1]$, $[s_1, s_2]$, $[s_2, 1]$ respectively. Figure 2 shows examples of polygons computed in each of the intervals. TOPP can finally be run on the full path $[0, 1]$, subject to the constraints provided by these polygons.

C. Geometric nature of contact switches

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. 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,[2] a behavior that can be observed in many walking patterns reported in the literature (see *e.g.* Figure 4 in [20], Figure 1 in [21], Figure 2 and 3 in [10], ...). However, having the COM in the SEP indicates *quasi-static* walking. It notably precludes the discovery of dynamic walking patterns, where the COM need not enter the SEP if foot swings are fast enough. TOPP can discover such motions through proper variations of the geometric parameters s_1 and s_2.

Let s_1^* and s_2^* denote respectively the first and second intersection of the COM path with the static-equilibrium prism. Choosing $s_1 = s_1^*$ and $s_2 = s_2^*$ corresponds to the "safe" quasi-static 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 traversable. However, if the path is time-parameterizable, 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 a feasible time-parameterization exists. For the same path as in Figure 1, if one chooses $s_1 = 0.51$ instead of $s_1 = 0.67$, then the path becomes non time-parameterizable.

[2] When walking on horizontal floors, this is equivalent to keeping the COM over the so-called support area.

IV. Locomotion by Retimed Predictive Control

The loop of a predictive controller can be described as follows: at each iteration, (1) generate a *preview trajectory* of future system dynamics leading to a desired configuration, then (2) apply the *first controls* of this trajectory. 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 [22], [23], a quadratic program [3], [9] or a blackbox optimal-control problem [5]. In what follows, our optimization unfolds in two steps: interpolating preview paths, and retiming them using TOPP. Our preview window foresees one footstep ahead at a time.

In single-support phases, one needs to interpolate both COM and swing-foot trajectories. Similarly to [24], we adopt a simplified dynamics model where the COM is represented by a point-mass and the swing foot by a rigid body (Figure 3). 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. The complete pipeline is summarized in Figure 4.

A. Finite state machine

The locomotion state machine underlying TOPP-MPC is simpler than those of previous works like [9] as it is

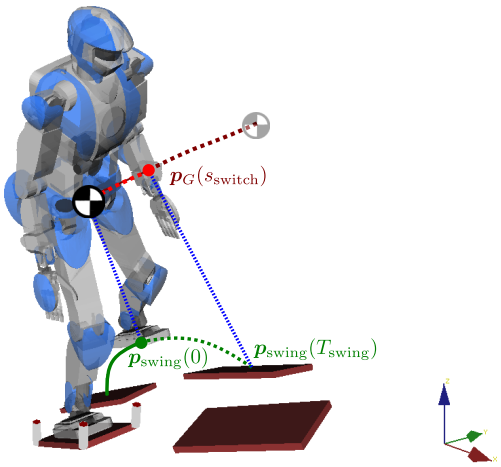

Fig. 3. **Preview of swing-foot (green) and COM (ligh and dark red) trajectories during a single-support phase.** Retiming the swing foot path under conservative constraints yields a swing duration T_{swing}. This value is then converted into retiming constraints on the COM trajectory, so that TOPP uses the single-support CWC up to $\boldsymbol{p}_G(s_1)$ (light red) and the double-support one afterwards (dark red).

does not need to take time into consideration. Transitions between single- and double-support phases are triggered by straightforward geometric 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. COM and swing-foot path interpolation

Path interpolation is constrained by state estimation from the robot: 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 $\boldsymbol{p}_{\text{swing}}$. This means that:

$$\boldsymbol{p}(0) = \boldsymbol{p}_{\text{cur}} \tag{10}$$
$$\cos\left(\boldsymbol{p}_s(0), \dot{\boldsymbol{p}}_{\text{cur}}\right) = 1 \tag{11}$$

The latter equation, a cosine of the angle between two vectors, 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 similarly given by:

$$\boldsymbol{p}(1) = \boldsymbol{p}_{\text{goal}} \tag{12}$$
$$\cos(\boldsymbol{p}_s(1), \dot{\boldsymbol{p}}_{\text{goal}}) = 1 \tag{13}$$

Like [17], 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 $\lambda > 0$ and $\mu > 0$: we can choose any path

$$I_{\lambda,\mu} = H(\boldsymbol{p}_{\text{cur}},\ \lambda\dot{\boldsymbol{p}}_{\text{cur}},\ \boldsymbol{p}_{\text{goal}},\ \mu\dot{\boldsymbol{p}}_{\text{goal}}) \tag{14}$$

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. [25] 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 but observed that, when boundary velocity vectors $\dot{\boldsymbol{p}}_{\text{cur}}$, $\dot{\boldsymbol{p}}_{\text{goal}}$ deviate significantly from the direction of motion $\Delta \stackrel{\text{def}}{=} \boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{cur}}$, OGH curves tend to start or end with very sharp accelerations. These accelerations have little impact on strain energy but jeopardize trajectory retiming.

To avoid this phenomenon, we chose to optimize a different criterion. We observed empirically that a good proxy criterion for the "retimability" of a trajectory is the uniform bound given by:

$$\arg\min_{\lambda,\mu} \max_{s\in[0,1]} \|I''_{\lambda,\mu}(s)\|^2 \tag{16}$$

The maximum over s in this formula implies that, contrary to OGH curves, the optimal solution here cannot be expressed as a single analytical formula on λ and μ. However, the optimum to a relaxation of this problem can. We call *Hermite curves with Overall Uniformly-Bounded Accelerations* (HOUBA) the polynomials given by:

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

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

(Recall that $\Delta \stackrel{\text{def}}{=} \boldsymbol{p}_{\text{goal}} - \boldsymbol{p}_{\text{cur}}$.) We show in Appendix I

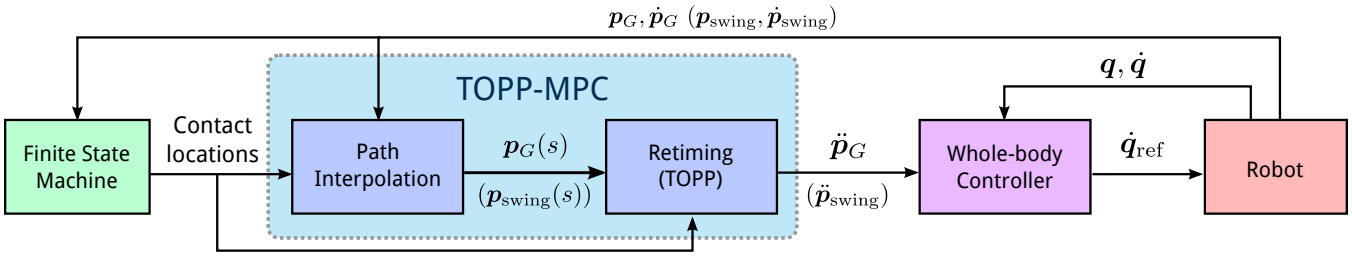

Fig. 4. **Pipeline of the predictive controller.** A finite state machine sends the current walking phase (single- or double-support) as well as contact locations to TOPP-MPC. COM and (in single support) swing-foot trajectories are then interpolated from the current robot state to a desired future state, 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.

that this formula provides a suboptimal upper bound to the uniform bound (16). As desired, it yields trajectories that are less prone to sharp boundary accelerations. In what follows, we interpolate all our COM and swing-foot paths by HOUBA curves.

C. Swing limb synchronization

COM and swing-foot path retimings are not independent: contact constraints on COM accelerations need to be computed using the single-support CWC while the swing 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. Under synchronization, this would imply 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 Figure 3:

- First, retime the swing-foot trajectory. Let T_{swing} denote the duration of the resulting trajectory.
- 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 $s \in [0, s_{\text{trans}}]$.

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 $\ddot{s} \leq \ddot{s}_{\max}$ suitable for TOPP:

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

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

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 $\ddot{s} = \pm\ddot{s}_{\max}$ and can be done readily in existing software.

D. Inverse kinematics 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 inverse kinematics solver for this, which is implemented in the open-source *pymanoid* library.[3] The solver is based on a single-layer quadratic program with weighted whole-body tasks (see [7] 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 chose joint-velocity minimization as the regularizing task. Each iteration of the solver updates a vector of joint-angle velocities $\dot{\boldsymbol{q}}_{\text{ref}}$ which is finally sent to the robot.

V. Experiments

We evaluated TOPP-MPC in simultations with a model of the HRP-4 humanoid robot. All the source code necessary to reproduce our work is publicly released [4].

A. Preview targets tuning

The main 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. 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 as we use Hermite curves for interpolation.

[3] https://github.com/stephane-caron/pymanoid
[4] https://github.com/stephane-caron/topp-mpc

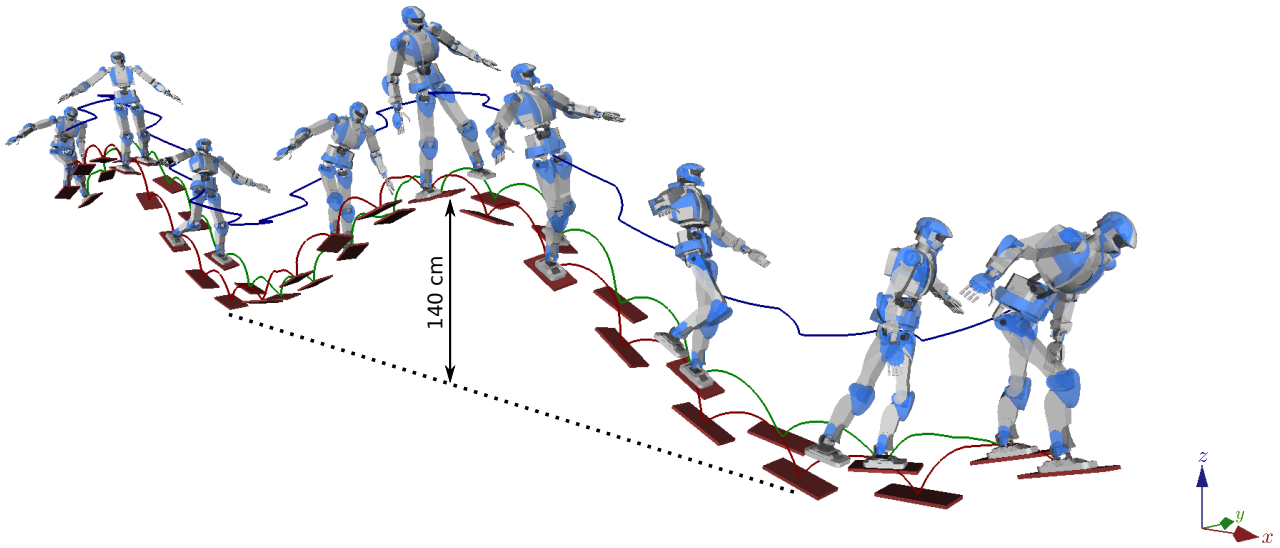

Fig. 5. **Locomotion over uneven terrain by HRP-4 running the TOPP-MPC controller.** The only external input to the system is a foothold sequence. From this, the controller automatically deduces the *timings* of its single- and double-support phases (which depend on terrain topology), along with COM 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 feasible supporting contact forces, which can be seen in the accompanying video.

B. TOPP tuning

Once a path is interpolated, we retime it using a path discretization step of $\mathrm{d}s = 0.1$. Minimum-time trajectories always saturate inequality constraints, which means in particular that output contact wrenches $\boldsymbol{w}_G$ will lie on the boundary of the CWC, a behavior that would make our controller sensitive to disturbances. We palliate this by computing the CWC matrix with sole dimensions and friction coefficients downscaled by 0.75.

C. Simulations

Figure 5 shows one application of our controller on a 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^{-2}. Phase timings derived by the controller were, on average over the complete motion, 0.88 ± 0.06 s for DS phases (these rather long durations may result from our choice of COM targets centered in the SEP), and 0.65 ± 0.22 s for SS phases. Note that these phase durations *result from* the particular terrain topology selected for the experiment.

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

Table II reports times taken to build and solve TOPP instances in this simulation. We observe that the convex-hull reduction presented in Section III-A yields results 20 to 30% faster than when using Bretl and Lall's method. Overall, TOPP-MPC fits in an update loop running at 30 Hz.

VI. Conclusion

We introduced a model predictive controller whose underlying optimization is a Time-Optimal Path Parameterization (TOPP-MPC). Despite its nonlinearity, we showed how the TOPP optimization runs fast enough for the control loop in legged locomotion. 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. We evaluated the performance of the overall controller in simulations where the humanoid model HRP-4 climbs up and down a series of hills (Figure 5 and accompanying video).

There are many avenues for further improvements. By construction, time-optimal retiming switches between maximum and minimum accelerations, causing discontinuities in the ensuing 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 [26] or jerk bounds [27]. Another direction would be to rely on admissible velocity propagation [28] to discover feasible motions throughout multiple contact changes.

References

[1] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Step timing adjustment: A step toward generating robust gaits," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 35–42.

[2] R. J. Griffin, G. Wiedebach, S. Bertrand, A. Leonessa, and J. Pratt, "Walking stabilization using step timing and location adjustment on the humanoid robot, atlas," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] S. Caron, Q.-C. Pham, and Y. Nakamura, "Zmp support areas for multicontact mobility under frictional constraints," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 67–80, 2017.

[8] 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.

[9] S. Caron and A. Kheddar, "Multi-contact walking pattern generation based on model preview control of 3d com accelerations," in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*. IEEE, 2016, pp. 550–557.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[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] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] H. Pham and Q.-C. Pham, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017.

[28] Q.-C. Pham, S. Caron, P. Lertkultanon, and Y. Nakamura, "Admissible velocity propagation: Beyond quasi-static path planning for high-dimensional robots," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 44–67, 2017.

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,

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

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

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,

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

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

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:

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

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

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

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