A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm

Article *in* IEEE Transactions on Robotics · December 2013

DOI: 10.1109/TRO.2014.2351113 · Source: arXiv

A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm

Quang-Cuong Pham
School of Mechanical and Aerospace Engineering
Nanyang Technological University, Singapore

***Abstract*—Finding the Time-Optimal Parameterization of a given Path (TOPP) subject to kinodynamic constraints is an essential component in many robotic theories and applications. The objective of this article is to provide a general, fast and robust implementation of this component. For this, we first derive a general formulation of the TOPP problem and show how diverse robot dynamics and constraints can be cast into this formulation. Next, we give a complete solution to the issue of dynamic singularities, which are the main cause of failure in existing implementations. Based on these two contributions, we provide an open-source implementation of the algorithm in C++/Python and demonstrate its efficiency in some complex kinodynamic planning problems.**

I. Introduction

Time-optimal motion planning plays a key role in many areas of robotics and automation, from industrial to mobile, to service robotics. While the problem of (optimal) *path planning under geometric constraints* can be considered as essentially solved in both theory and in practice (see e.g. [1]), general and efficient solutions to the (optimal) *trajectory planning under kinodynamic* [1] *constraints* [2] are still lacking. We argue that Time-Optimal Path Parameterization [2] (TOPP) may constitute an efficient tool to address the latter problem.

Importance of Time-Optimal Path Parameterization (TOPP): There are at least three types of kinodynamic motion planning problems where TOPP is useful or even indispensable. First, some applications such as painting or welding require specifically tracking a predefined path. Other applications, such as car parts assembly, involve so cluttered environments that finding a single collision-free path is already challenging. In both cases, since the path is given – either by the very nature of the task or by a previous path planner – the only "degree of freedom" that one can optimize indeed consists of the time law to traverse the path.

Second, even when there is no *a priori* necessity to track a predefined path, it can be efficient to *decouple* the optimal trajectory planning problem into two simpler, more tractable sub-problems: rather than searching directly for a trajectory in the state space (which is very complex and mostly intractable for high-dofs robots), one would first generate a set of paths in the robot configuration space, optimally time-parameterize these paths and finally pick the path with the fastest optimal parameterization [3], [4], [5].

Third, it was recently suggested in [6] that TOPP can also be used to address the *feasibility* problem, i.e. finding *one* feasible trajectory in a challenging context, as opposed to choosing an optimal trajectory in a context where it is relatively easy to find many feasible trajectories. At the heart of the mentioned reference is the so-called "Admissible Velocity Propagation" (AVP): given an interval of reachable velocities at the beginning of a path, the authors compute, using TOPP, the interval of all velocities that the system can reach *after* traversing that path while respecting the kinodynamic constraints. Combining AVP with usual configuration-space sampling-based planners then gives rise to a family of very efficient kinodynamic planners which significantly over-performed usual state-space planners.

Overview of existing TOPP methods: Since the path is constrained, the only "degree of freedom" to optimize is the scalar function $t \mapsto s(t)$ which represents the "position" on the path at each time instant. If the system dynamics and constraints are of second-order, one can next search for the optimal function in the 2-dimensional space $(s, \dot{s})$. There are basically three families of methods to do so.

Dynamic programming – The first family of methods divide the $(s, \dot{s})$ plane into a grid and subsequently uses a dynamic programming approach to find the optimal trajectory in the $(s, \dot{s})$ plane [7]. If N denotes the number of discretization points on each axis, then the dynamic programming approach has a complexity that scales as N^3.

Convex optimization – The second family of methods discretize only the s-axis (into N segments) and subsequently convert the original problem into a convex optimization problem in $O(N)$ variables and $O(N)$ equality and inequality constraints [8], [9], [10], [11]. These methods have the advantage of being versatile (they can for instance trade off time duration with other objectives such as energy) and can rely on existing efficient convex optimization packages. However, since the size of the optimization problem is fairly large – typically, a TOPP instance for a 6-dof robot under torque constraints and using a grid size of 100 points involves about 800 variables and 1500 equality and inequality constraints – and that the resolution time of a convex optimization problem is super-linear in its size, this approach can be slow compared to the numerical integration approach described below (see experimental comparisons in section IV-B).

Numerical integration – The third family of methods are based on the Pontryagin Maximum Principle: the optimal trajectory in the $(s, \dot{s})$ plane is known to be "bang-bang" and can thus be found by integrating successively the maximum and minimum accelerations $\ddot{s}$ [12], [13], [14], [15], [16], [17], see section III-A for details. This approach is theoretically faster than the previous two since it exploits the bang-bang structure of the optimization problem. However, to our knowledge, there is no available general and efficient implementation, perhaps because of the programming difficulties this approach may involve and of the robustness issues associated with the so-called *dynamic singularities* [15], [16], [17], see section III.

Finally, note that all three families can be applied to a wide variety of robot dynamics and constraints, such as manipulators subject to torque bounds [12], [18], [13], [14], [15], [8], [17], humanoids subject to joint velocity and accelerations bounds [16], [10], mobile robots or humanoids subject to balance and friction constraints [19], [9], [20], non-holonomic robots [21], etc.

Contributions and organization of this article: The goal of this article is to provide a *general*, *fast* and *robust* implementation of TOPP. First, even though TOPP has been applied to a wide variety of robot dynamics and constraints as noted previously, each paper came up with its own derivation of the dynamic equations and computer implementation. Our first contribution, in section II, consists of deriving a general formulation of the TOPP problem and showing how diverse dynamics and constraints can be cast into this formulation. Combined with Object-Oriented Programming, this allows addressing new kinodynamic problems or combining kinodynamic constraints of different types using only a dozen of lines of code in C++ or Python.

Next, we address a critical issue in the numerical integration approach, namely the existence of *dynamic singularities*. Such singu-

[1] Geometric constraints – such as joints limits or obstacle avoidance – depend only on the configuration of the robot, while kinodynamic constraints – such as bounds on joint velocity, acceleration and torque, or dynamic balance – involve also higher-order time derivatives of the robot configuration.

[2] *Parameterizing* a given geometric path consists of finding a time law to traverse the path, thereby transforming it into a *trajectory*. *Time-optimal* parameterization seeks to minimize the traversal time under given kinodynamic constraints.

larities arise in almost all real-world problem instances of TOPP, and are known to dramatically affect the robustness of existing implementations. Yet, in most references devoted to the TOPP algorithm, from the original articles [13], [14], [15], [22], [16] to reference textbooks [23], the characterization and treatment of these singularities have not yet been done completely correctly. In section III, we derive a complete characterization of dynamic singularities and suggest how to appropriately address these singularities. The development extends our previous contribution in the case of torque bounds (presented at IROS 2013 [17]) to the general case.

In section IV, based on the previous two contributions, we provide a general, fast and robust open-source implementation of TOPP in C++/Python based on the numerical integration approach. We show in particular that our implementation is between 6 and 18 times faster than the currently fastest implementation of the convex optimization approach [10]. This improvement is particularly crucial for the global trajectory optimization or feasibility problems mentioned earlier, which require calling the TOPP routine tens or hundreds of thousands times. As illustration, we present a method for planning fast collision-free, dynamically-balanced trajectories for a humanoid robot based on TOPP.

Finally section V concludes by briefly discussing the obtained results and future research directions.

II. General formulation of the TOPP problem

A. General dynamics equations and constraints

Let $\mathbf{q}$ be a n-dimensional vector representing the configuration of a robot system. Consider second-order inequality constraints of the form

$$\mathbf{A}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}(\mathbf{q}) \leq 0, \tag{1}$$

where $\mathbf{A}(\mathbf{q})$, $\mathbf{B}(\mathbf{q})$ and $\mathbf{f}(\mathbf{q})$ are respectively a $M \times n$ matrix, a $n \times M \times n$ tensor and a M-dimensional vector.

Inequality (1) is very general and may represent a large variety of second-order systems with actuation bounds, or subject to balance or equilibrium constraints. Section II-B gives examples of reduction of several popular systems to this form.

Note that pure velocity bounds of the form

$$\dot{\mathbf{q}}^\top \mathbf{B}_v(\mathbf{q})\dot{\mathbf{q}} + \mathbf{f}_v(\mathbf{q}) \leq 0,$$

can also be taken into account [24].

Consider next a path P – represented as the underlying path of a trajectory $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ – in the configuration space. Assume that $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ is C^1- and piecewise C^2-continuous. We are interested in *time-parameterizations* of P – or time-*re*parameterizations of $\mathbf{q}(s)_{s\in[0,s_{\text{end}}]}$ – which are increasing *functions* $s : [0, T'] \to [0, s_{\text{end}}]$. Differentiating $\mathbf{q}(s(t))$ with respect to t yields

$$\dot{\mathbf{q}} = \dot{s}\mathbf{q}_s, \quad \ddot{\mathbf{q}} = \ddot{s}\mathbf{q}_s + \dot{s}^2\mathbf{q}_{ss}, \tag{2}$$

where dots denote differentiations with respect to the time parameter t and $\mathbf{q}_s = \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}s}$ and $\mathbf{q}_{ss} = \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}s^2}$. Substituting (2) into (1) then leads to

$$\ddot{s}\mathbf{A}(\mathbf{q})\mathbf{q}_s + \dot{s}^2\mathbf{A}(\mathbf{q})\mathbf{q}_{ss} + \dot{s}^2\mathbf{q}_s^\top \mathbf{B}(\mathbf{q})\mathbf{q}_s + \mathbf{f}(\mathbf{q}) \leq 0,$$

which can be rewritten as

$$\ddot{s}\mathbf{a}(s) + \dot{s}^2\mathbf{b}(s) + \mathbf{c}(s) \leq 0, \quad \text{where} \tag{3}$$

$$\begin{aligned} \mathbf{a}(s) &= \mathbf{A}(\mathbf{q}(s))\mathbf{q}_s(s), \\ \mathbf{b}(s) &= \mathbf{A}(\mathbf{q}(s))\mathbf{q}_{ss}(s) + \mathbf{q}_s(s)^\top \mathbf{B}(\mathbf{q}(s))\mathbf{q}_s(s), \\ \mathbf{c}(s) &= \mathbf{f}(\mathbf{q}(s)). \end{aligned} \tag{4}$$

Equation (3) constitutes another level of abstraction, which is particularly convenient for computer implementation: it suffices indeed to evaluate the M-dimensional vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ along the path and give these vectors as input to the optimization algorithm. From this formulation, one can also remark that it is *not necessary* to evaluate the full matrices $\mathbf{A}$ and tensors $\mathbf{B}$ (which are of sizes $M \times n$ and $n \times M \times n$), but only their *products* (of sizes M) with $\mathbf{q}_s$ and $\mathbf{q}_{ss}$. In the case of torque bounds, the Recursive Newton-Euler method for inverse dynamics [25] allows computing these products without ever evaluating the full $\mathbf{A}$ and $\mathbf{B}$. Finally, this formulation allows very easily *combining* different types of constraints: for each s, it suffices to concatenate the vectors $\mathbf{a}(s)$ corresponding to the different constraints, and similarly for the vectors $\mathbf{b}(s)$ and $\mathbf{c}(s)$.

B. Examples of reduction

We now show how very different dynamics and constraints can be cast into the form of equations (1) or (3).

1) Manipulator with torque bounds: This is the classical problem that motivated the development of TOPP [12], [18]. The dynamics equations of a n-dof manipulator is given by

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}, \tag{5}$$

where $\mathbf{q}$ is the n-dimensional vector of joint values, $\mathbf{M}$ the $n \times n$ manipulator inertia matrix, $\mathbf{C}$ the $n \times n \times n$ Coriolis tensor, $\mathbf{g}$ the n-dimensional vector of gravity forces and $\boldsymbol{\tau}$ the n-dimensional vector of actuator torques. Assume that there are lower and upper bounds on the torques, that is, for any joint i and time t,

$$\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max}.$$

Clearly, inequalities (5) can then be put in the form of (1) with

$$\mathbf{A}(\mathbf{q}) = \begin{pmatrix} \mathbf{M}(\mathbf{q}) \\ -\mathbf{M}(\mathbf{q}) \end{pmatrix}, \mathbf{B}(\mathbf{q}) = \begin{pmatrix} \mathbf{C}(\mathbf{q}) \\ -\mathbf{C}(\mathbf{q}) \end{pmatrix}, \mathbf{f}(\mathbf{q}) = \begin{pmatrix} \mathbf{g}(\mathbf{q}) - \boldsymbol{\tau}^{\max} \\ -\mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}^{\min} \end{pmatrix},$$

where $\boldsymbol{\tau}^{\max} = (\tau_1^{\max}, \ldots, \tau_n^{\max})^\top$ and $\boldsymbol{\tau}^{\min} = (\tau_1^{\min}, \ldots, \tau_n^{\min})^\top$.

2) Pure acceleration bounds: Assume that the acceleration of each joint is bounded from above and below, i.e., for every i

$$a_i^{\min} \leq \ddot{q}_i \leq a_i^{\max}.$$

These constraints can be put in the form of (1) with

$$\mathbf{A}(\mathbf{q}) = \begin{pmatrix} \mathbf{I} \\ -\mathbf{I} \end{pmatrix}, \quad \mathbf{B} = 0, \quad \mathbf{f}(\mathbf{q}) = \begin{pmatrix} -\mathbf{a}^{\max} \\ \mathbf{a}^{\min} \end{pmatrix},$$

where $\mathbf{a}^{\max} = (a_1^{\max}, \ldots, a_n^{\max})^\top$ and $\mathbf{a}^{\min} = (a_1^{\min}, \ldots, a_n^{\min})^\top$.

3) ZMP constraints: Consider a legged robot composed of interconnected rigid links. A well-known condition for the robot dynamic balance is that the ZMP stays within the convex hull of the ground contact points at any time instant [26].

For each link i, denote by $\mathbf{r}^i$ the position of its center of mass in the laboratory reference frame. Next, let $\mathbf{h}$ be the result of gravity and inertia forces [9]

$$\mathbf{h} = \sum_{i=0}^{N} m_i(\mathbf{g} - \ddot{\mathbf{r}}^i),$$

and $\boldsymbol{\tau}$ be the moment of $\mathbf{h}$ around the origin

$$\boldsymbol{\tau} = \sum_{i=0}^{N} m_i \mathbf{r}^i \times (\mathbf{g} - \ddot{\mathbf{r}}^i) - \mathbf{L}. \tag{6}$$

To avoid lengthy formula, we shall omit the angular momentum term $\mathbf{L}$ in what follows, for a complete derivation, see [20].

Next, the X- and Y-coordinates of the ZMP are given by

$$x_{\mathbf{ZMP}} = \frac{-(\boldsymbol{\tau})_2}{(\mathbf{h})_3}, \qquad y_{\mathbf{ZMP}} = \frac{(\boldsymbol{\tau})_1}{(\mathbf{h})_3}, \tag{7}$$

where $(\boldsymbol{\tau})_1$ denotes the first coordinate of $\boldsymbol{\tau}$, and so forth. Note that $(\mathbf{h})_3 < 0$ so long as the robot is not free-flying.

Assume that the convex hull of the ground contact points is a rectangle, such that a condition for balance is given by

$$\left\{ \begin{array}{ccccc} x_{\min} & \leq & x_{\mathbf{ZMP}} & \leq & x_{\max} \\ y_{\min} & \leq & y_{\mathbf{ZMP}} & \leq & y_{\max} \end{array} \right. . \tag{8}$$

From the expressions (7) of $x_{\mathbf{ZMP}}$ and $y_{\mathbf{ZMP}}$, inequalities (8) can be rewritten as

$$\left\{ \begin{array}{rcl} (\boldsymbol{\tau})_2 + x_{\max} \cdot (\mathbf{h})_3 & \leq & 0 \\ -(\boldsymbol{\tau})_2 - x_{\min} \cdot (\mathbf{h})_3 & \leq & 0 \\ -(\boldsymbol{\tau})_1 + y_{\max} \cdot (\mathbf{h})_3 & \leq & 0 \\ (\boldsymbol{\tau})_1 - y_{\min} \cdot (\mathbf{h})_3 & \leq & 0 \end{array} \right. . \tag{9}$$

Remark next that one can in fact express $\mathbf{r}^i$ as a non-linear function of the joint angles as $\mathbf{r}^i = \mathbf{r}^i(\mathbf{q})$, which yields

$$\dot{\mathbf{r}}^i = \mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}}, \quad \ddot{\mathbf{r}}^i = \mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}}.$$

Thus, inequalities (9) can be put in the form of (3) with

$$\mathbf{a} = \begin{pmatrix} \sum_i m_i \left[-(\mathbf{r}^i \times \mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_2 - x_{\max}(\mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[(\mathbf{r}^i \times \mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_2 + x_{\min}(\mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[(\mathbf{r}^i \times \mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_1 - y_{\max}(\mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[-(\mathbf{r}^i \times \mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_1 + y_{\min}(\mathbf{r}^i_{\mathbf{q}} \dot{\mathbf{q}})_3 \right] \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} \sum_i m_i \left[-(\mathbf{r}^i \times (\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}}))_2 - x_{\max}(\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[(\mathbf{r}^i \times (\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}}))_2 + x_{\min}(\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[(\mathbf{r}^i \times (\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}}))_1 - y_{\max}(\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}})_3 \right] \\ \sum_i m_i \left[-(\mathbf{r}^i \times (\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}}))_1 + y_{\min}(\mathbf{r}^i_{\mathbf{q}} \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{r}^i_{\mathbf{qq}} \dot{\mathbf{q}})_3 \right] \end{pmatrix}$$

$$\mathbf{c} = \begin{pmatrix} \sum_i m_i \left[(\mathbf{r}^i \times \mathbf{g})_2 + x_{\max}(\mathbf{g})_3 \right] \\ \sum_i m_i \left[-(\mathbf{r}^i \times \mathbf{g})_2 - x_{\min}(\mathbf{g})_3 \right] \\ \sum_i m_i \left[-(\mathbf{r}^i \times \mathbf{g})_1 + y_{\max}(\mathbf{g})_3 \right] \\ \sum_i m_i \left[(\mathbf{r}^i \times \mathbf{g})_1 - y_{\min}(\mathbf{g})_3 \right] \end{pmatrix}.$$

4) Grasp equilibrium constraints: Consider a grasp with n fingers, and where each fingertip can exert r independent forces/moments on the grasped object: for instance, $r = 4$ if a fingertip can apply a normal and two tangential forces and a moment about the point of contact (soft fingertip). A condition for the grasp to be in (static or dynamic) equilibrium is given by (cf. equation (19) of [27])

$$\mathbf{A}\mathbf{f} \leq 0, \tag{10}$$

where $\mathbf{A}$ is a $6n \times rn$ block-diagonal matrix and $\mathbf{f}$ is a rn-dimensional vector representing the forces/moments exerted by the fingertips on the object. Next, $\mathbf{f}$ can be decomposed as

$$\mathbf{f} = \mathbf{f}_h + \mathbf{f}_p,$$

where $\mathbf{f}_h$ represents the grasp internal forces/moments and $\mathbf{f}_p$ represents the forces/moments that oppose the external load $\mathbf{f}_{\text{ext}}$ [27], [28]. Consider a grasp synthesis scheme where $\mathbf{f}_h$ is constant and $\mathbf{f}_p$ is given by (cf. equation (31) of [28])

$$\mathbf{f}_p = \mathbf{G}\mathbf{f}_{\text{ext}},$$

with $\mathbf{G}$ being defined as in equation (15) of [28]. Assume finally that the external load $\mathbf{f}_{\text{ext}}$ consists of gravity and inertial forces/moments, such that it can be expressed as [29]

$$\mathbf{f}_{\text{ext}} = \mathbf{M}\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}\dot{\mathbf{q}} + \mathbf{g},$$

where $\mathbf{q}$ is a 6-dimensional vector of generalized coordinates that describe the position and orientation of the grasped object in the laboratory frame. Clearly, inequalities (10) can be put in the form of (1) as follows

$$(\mathbf{AGM})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top (\mathbf{AGC})\dot{\mathbf{q}} + (\mathbf{AGg} + \mathbf{A}\mathbf{f}_h) \leq 0.$$

III. Improving the robustness of the numerical integration approach

A. Review of the numerical integration approach

Each row i of equation (3) is of the form

$$a_i(s)\ddot{s} + b_i(s)\dot{s}^2 + c_i(s) \leq 0.$$

There are three cases:

- if $a_i(s) > 0$, then one has $\ddot{s} \leq \frac{-c_i(s) - b_i(s)\dot{s}^2}{a_i(s)}$. Define the *upper bound* $\beta_i = \frac{-c_i(s) - b_i(s)\dot{s}^2}{a_i(s)}$;
- if $a_i(s) < 0$, then one has $\ddot{s} \geq \frac{-c_i(s) - b_i(s)\dot{s}^2}{a_i(s)}$. Define the *lower bound* $\alpha_i = \frac{-c_i(s) - b_i(s)\dot{s}^2}{a_i(s)}$;
- if $a_i(s) = 0$, then s is a "*zero-inertia*" point [13], [15].

One then has a certain number of α_p and β_q where $P + Q \leq M$, with equality when s is not a zero-inertia point. One can next define for each $(s, \dot{s})$

$$\begin{aligned} \alpha(s, \dot{s}) &= \max_p \alpha_p(s, \dot{s}), \\ \beta(s, \dot{s}) &= \min_q \beta_q(s, \dot{s}). \end{aligned}$$

From the above transformations, one can conclude that $\mathbf{q}(s(t))_{t \in [0, T']}$ satisfies the constraints (1) if and only if

$$\forall t \in [0, T'] \quad \alpha(s(t), \dot{s}(t)) \leq \ddot{s}(t) \leq \beta(s(t), \dot{s}(t)). \tag{11}$$

Note that $(s, \dot{s}) \mapsto (\dot{s}, \alpha(s, \dot{s}))$ and $(s, \dot{s}) \mapsto (\dot{s}, \beta(s, \dot{s}))$ can be viewed as two vector fields in the $(s, \dot{s})$ plane. One can integrate velocity profiles following the field $(\dot{s}, \alpha(s, \dot{s}))$ (from now on, α in short) to obtain *minimum acceleration* profiles (or α-profiles), or following the field β to obtain *maximum acceleration* profiles (or β-profiles).

Next, observe that if $\alpha(s, \dot{s}) > \beta(s, \dot{s})$ then, from (11), there is no possible value for $\ddot{s}$. Thus, to be valid, any velocity profile must stay below the maximum velocity curve (MVC in short) defined by

$$\text{MVC}(s) = \left\{ \begin{array}{ll} \min\{\dot{s} \geq 0 : \alpha(s, \dot{s}) = \beta(s, \dot{s})\} & \text{if } \alpha(s, 0) \leq \beta(s, 0), \\ 0 & \text{if } \alpha(s, 0) > \beta(s, 0). \end{array} \right.$$

It was shown (see e.g. [15]) that the time-minimal velocity profile is obtained by a *bang-bang*-type control, i.e., whereby the optimal profile follows alternatively the β and α fields while always staying below the MVC. More precisely, the algorithm to find the time-optimal parameterization of P starting and ending with velocities v_{beg} and v_{end} is as follows (see Fig. 1):

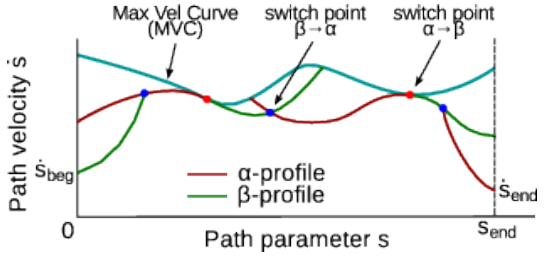

Fig. 1. Illustration for the numerical integration approach.

1) In the $(s, \dot{s})$ plane, start from $(s = 0, \dot{s} = v_{\text{beg}}/\|\mathbf{q}_s(0)\|)$ and integrate forward following β until hitting either
   (i) the MVC, in this case go to step 2;
   (ii) the horizontal line $\dot{s} = 0$, in this case the path is not dynamically traversable;
   (iii) the vertical line $s = s_{\text{end}}$, in this case go to step 3.
2) Search forward along the MVC for the next candidate $\alpha \to \beta$ switch point (cf. section III-C). From such a switch point:

a) integrate *backward* following α, until *intersecting* a forward β-profile (from step 1 or recursively from the current step 2). The intersection point constitutes a $\beta \to \alpha$ switch point;
b) integrate *forward* following β. Then continue as in step 1.

The resulting forward profile will be the concatenation of the intersected forward β-profile, the backward α-profile obtained in (a), and the forward β-profile obtained in (b).

3) Start from $(s = s_{\text{end}}, \dot{s} = v_{\text{end}}/\|\mathbf{q}_s(s_{\text{end}})\|)$ and integrate *backward* following α, until intersecting a forward profile obtained in steps 1 or 2. The intersection point constitutes a $\beta \to \alpha$ switch point. The final profile will be the concatenation of the intersected forward profile and the backward α-profile just computed.

B. Finding $\alpha \to \beta$ switch points

From the above presentation, it appears that finding the switch points is crucial for the numerical integration approach. It was shown in [13], [14], [15], [22] that a given point s is a $\alpha \to \beta$ switch point only in the following three cases:

- the MVC is *discontinuous* at s. In this case s is labeled as a "discontinuous" switch point;
- the MVC is *continuous* but *undifferentiable* at s. In this case s is labeled as a "singular" switch point or a "*dynamic singularity*" (previous works labeled such switch points as "zero-inertia points" [14]; however we shall see below that not all zero-inertia points are singular);
- the MVC is *continuous* and *differentiable* at s and the *tangent vector* to the MVC at $(s, \text{MVC}(s))$ is collinear with the vector $(\text{MVC}(s), \alpha(s, \text{MVC}(s)))$ [or, which is the same since we are on the MVC, collinear with the vector $(\text{MVC}(s), \beta(s, \text{MVC}(s)))$]. In this case s is a labeled as a "tangent" switch point.

Finding discontinuous and tangent switch points does not involve particular difficulties since it suffices to construct the MVC and examine whether it is discontinuous or whether the tangent to the MVC is collinear with α for all discretized points s along the path. Regarding the undifferentiable switch points, one approach could consist of checking whether the MVC is continuous but undifferentiable at s. However, this approach is seldom used in the literature since it is comparatively more prone to discretization errors. Instead, it was proposed (cf. [13], [14], [15], [23], [16]) to equate undifferentiable points with *zero-inertia points*, i.e. the points s where $a_k(s) = 0$ for one of the constraints k, and to consequently search for zero-inertia points.

This method is however not completely correct: we shall see in section III-C that *not all zero-inertia points are undifferentiable*. Thus, only zero-inertia points that are undifferentiable properly constitute *singular* switch points or *dynamic singularities*. Characterizing and addressing such switch points are of crucial importance since they occur in almost all real-world TOPP instances – they are in particular much more frequent than discontinuous and tangent switch points together. Furthermore, since the α and β fields are divergent near these switch points (see Fig. 2), they are the cause of most failures in existing implementations of TOPP.

C. Characterizing dynamic singularities

Consider a zero-inertia point s^*, and assume that it is triggered by the k-th constraint, i.e. $a_k(s^*) = 0$. Without loss of generality, we make the assumption that $a_k(s) < 0$ in a neighborhood to the left of s^* and $a_k(s) > 0$ in a neighborhood to the right of s^* (the case when a_k switches from positive to negative can be treated similarly by changing signs at appropriate places in the rest of the development).

We prove in the Appendix that, if the path is traversable, then $c_k(s^*) < 0$. We next distinguish two cases (see Fig. 2A).

Case 1: $b_k(s^*) > 0$. Define $\dot{s}^* = \sqrt{\frac{-c_k(s^*)}{b_k(s^*)}}$. Next, let $\dot{s}^\dagger$ be the value of the MVC, had we removed constraint k. We prove in the Appendix that

- If $\dot{s}^\dagger < \dot{s}^*$, then constraint k does *not* trigger a dynamic singularity at s^*;
- If $\dot{s}^\dagger > \dot{s}^*$, then the MVC is indeed undifferentiable at s^*, and s^* is a dynamic singularity.

Case 2: $b_k(s^*) < 0$. We prove that, in this case, constraint k does *not* trigger a dynamic singularity at s^*.

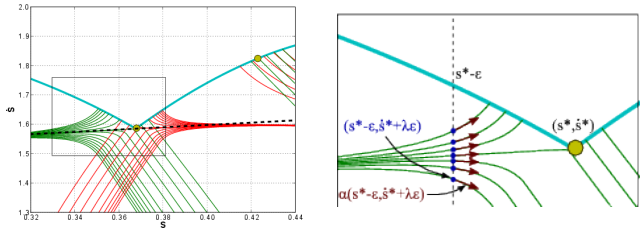

Fig. 2. **A**: α- and β-profiles (in green and red respectively) near zero-inertia points (yellow points). The left zero-inertia point is a singular switch point, while the right zero-inertia point is not singular. Note that in agreement with the definitions, at any point in the plane the slope of the red profile is higher than the slope of the green profile, except on the MVC where the two slopes are equal. The dotted line is the line that goes through the switch point and has slope λ computed by equation (14) (cf. section III-D). **B**: close-up view (zoomed in the black box of **A**) centered around the singular switch point.

D. Addressing dynamic singularities

1) Previous solutions: The next difficulty consists in the selection of the optimal acceleration to initiate the backward and forward integrations from a dynamic singularity s^*: indeed the fields α and β are not *naturally* defined at these points because of a division by $a_k(s^*) = 0$. In [13], no indication was given regarding this matter. In [14], it was stated that "*[this] acceleration is not uniquely determined*" and suggested to choose *any* acceleration to initiate the integrations. In [15], [22] (and also in [23] which reproduced the reasoning of [15], [22]), the authors suggested to select the following acceleration to initiate the backward integration (and a similar expression for the forward integration):

$$\min(\alpha^-, \alpha^+, \alpha_{\text{MVC}}), \text{ where} \tag{12}$$

$$\alpha^- = \lim_{s \uparrow s^*} \alpha(s, \text{MVC}(s^*)), \ \alpha^+ = \lim_{s \downarrow s^*} \alpha(s, \text{MVC}(s^*)), \tag{13}$$

and α_{MVC} is computed from the *slope* of the MVC on the left of s^*.

However, observing the α-profiles near the dynamic singularity of Fig. 2A, it appears that the definition of α^- in equation (13) is arbitrary. Indeed, depending on the *direction* from which one moves towards $(s^*, \text{MVC}(s^*))$ in the plane $(s, \dot{s})$ the limit of α is different: for instance, in Fig. 2A, if one moves from the top left, the limit, if it exists, would be positive, and it would be negative if one moves from the bottom left. In this context, the choice of equation (13) consisting of moving towards $(s^*, \text{MVC}(s^*))$ *horizontally* is no more justified than any other choice. More generally, it is impossible to extend α by continuity towards $(s^*, \text{MVC}(s^*))$ from the left because the α-profiles *diverge* when approaching $(s^*, \text{MVC}(s^*))$ from the left.

In practice, because of this flow divergence, choosing a slightly incorrect value for α and β at s^* may result in strong oscillations (see Fig. 4), which in turn can make the algorithm incorrectly terminate

(because the the velocity profile goes above the MVC or below the line $\dot{s} = 0$). In fact, this is probably one of the main reasons of failure in existing implementations.

2) Proposed new solution: Fig. 2B shows in more detail the α-profiles near a singular switch point s^*.

We first show in the Appendix that, if the singularity is triggered by constraint k, then $\alpha = \alpha_k$ on the left of s^* and $\beta = \beta_k$ on the right of s^*. Consider next the intersections of the vertical line $s = s^* - \epsilon$, where ϵ is an arbitrary small positive number, with the α-profiles. An α-profile can reach $(s^*, \dot{s}^*)$ only if its *tangent vector* at the intersection *points towards* $(s^*, \dot{s}^*)$. This can be achieved if there exists a real number λ such that

$$\frac{\alpha_k(s^* - \epsilon, \dot{s}^* + \lambda\epsilon)}{\dot{s}^* + \lambda\epsilon} = \lambda.$$

Replacing α_k by its expression yields the condition

$$\frac{-b_k(s^* - \epsilon)(\dot{s}^* + \lambda\epsilon)^2 - c_k(s^* - \epsilon)}{a_k(s^* - \epsilon)(\dot{s}^* + \lambda\epsilon)} = \lambda, \text{ i.e.}$$

$$-b_k(s^* - \epsilon)(\dot{s}^* + \lambda\epsilon)^2 - c_k(s^* - \epsilon) = \lambda(\dot{s}^* + \lambda\epsilon)a_k(s^* - \epsilon).$$

Computing the Taylor expansion of the above equation at order 1 in ϵ and recalling that $-b_k(s^*)\dot{s}^{*2} - c_k(s^*) = 0$ and $a_k(s^*) = 0$, one obtains the condition

$$b'_k(s^*)\dot{s}^{*2} - 2\lambda b_k(s^*)\dot{s}^* + c'_k(s^*) = \lambda\dot{s}^* a'_k(s^*).$$

Solving for λ, one finally obtains

$$\lambda = -\frac{b'_k(s^*)\dot{s}^{*2} + c'_k(s^*)}{[2b_k(s^*) + a'_k(s^*)]\dot{s}^*}. \tag{14}$$

Following the same reasoning on the right of s^*, one has to solve

$$\frac{\beta_k(s^* + \epsilon, \dot{s}^* + \lambda\epsilon)}{\dot{s}^* + \lambda\epsilon} = -\lambda,$$

which leads to the same value as in equation (14). Thus the optimal backward and forward acceleration at $(s^*, \dot{s}^*)$ is given by equation (14). One can observe in Fig. 2A that the black dotted line, whose slope is given by λ, indeed constitutes the "neutral" line at $(s^*, \dot{s}^*)$.

3) About Kunz and Stilman's conjecture: Kunz and Stilman [16] were first to remark – in the particular case of time-optimal path parameterization with velocity and acceleration bounds and paths made of straight segments and circular arcs – that the algorithm proposed in [15], [22] could not satisfactorily address all dynamic singularities. From equation (14) of [16], the correspondences between the parameters of [16] and those of the present article are given in Table I.

TABLE I
PARAMETERS CORRESPONDENCES

This article		Kunz and Stilman [16]
$a_k(s)$	$\leftrightarrow$	$f'_k(s)$
$b_k(s)$	$\leftrightarrow$	$f''_k(s)$
$c_k(s)$	$\leftrightarrow$	$-\ddot{q}_k^{\max}$ or $\ddot{q}_k^{\max}$

Remark next that the zero-inertia points in [16] are all located in the circular portions. In such portions, the coefficients a_k and b_k have the following form (using our notations):

$$\begin{aligned} a_k(s) &= -\frac{C_1}{r}\sin\left(\frac{s}{r}\right) + \frac{C_2}{r}\cos\left(\frac{s}{r}\right), \\ b_k(s) &= \frac{C_1}{r^2}\cos\left(\frac{s}{r}\right) - \frac{C_2}{r^2}\sin\left(\frac{s}{r}\right), \end{aligned}$$

where r, C_1 and C_2 are two constants independent of s. Differentiating b_k next yields

$$b'_k(s) = -\frac{C_1}{r^3}\sin\left(\frac{s}{r}\right) + \frac{C_2}{r^3}\cos\left(\frac{s}{r}\right) = \frac{1}{r^2}a_k(s).$$

One thus has $b'_k(s^*) = 1/r^2 a_k(s^*) = 0$ at a zero-inertia point. If this zero-inertia point is actually a dynamic singularity then, from equation (14), one obtains that $\lambda = 0$, which corresponds to the conjecture made in [16].

E. Algorithm and experiments

Based on the previous development, we propose the following algorithm when encountering a zero-inertia point s^*, with $a_k(s^*) < 0$ on the left of s^* and $a_k(s^*) > 0$ on the right of s^*:

- If $b_k(s^*) < 0$, then constraint k does not trigger a singularity;
- If $b_k(s^*) > 0$, then compute $\dot{s}^*$ by equation (15) and $\dot{s}^\dagger$ by removing constraint k and evaluating again the MVC at s^*.
  - If $\dot{s}^* > \dot{s}^\dagger$, then constraint k does not trigger a singularity;
  - If $\dot{s}^* < \dot{s}^\dagger$, then s^* is a dynamic singularity. Compute next λ by equation (14) and
    - integrate the constant field $(\dot{s}^*, \lambda\dot{s}^*)$ *backward* for a small number of time steps. Then continue by following α, as in the original algorithm;
    - integrate the constant field $(\dot{s}^*, \lambda\dot{s}^*)$ *forward* for a small number of time steps. Then continue by following β.

Note that after moving a small number of steps away from s^*, the fields α and β become smooth, so that there is no problem in the integration.

We tested the above algorithm on a model of the 4-dof Barrett WAM. The torque bounds were set at ± 6 N·m, ± 15 N·m, ± 5 N·m, ± 4 N·m respectively for the shoulder yaw, pitch and roll joints and the elbow joint. The results show a smooth behavior around the undifferentiable switch point, both for the $(s, \dot{s})$ profile and for the torque profiles, see Fig. 3.

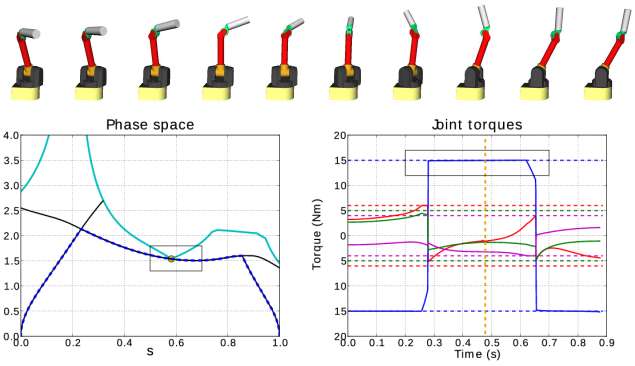

Fig. 3. Simulations for a 4-link manipulator. **Top**: snapshots of the time-parameterized trajectory taken every 10% of trajectory duration. **Left**: $(s, \dot{s})$ space. Same legends as in Fig. 2. The superimposed dotted blue line indicates the final $(s, \dot{s})$ profile, which follows parts of the computed α- and β-profiles (black). **Right**: torque profiles. The torques for the shoulder yaw, pitch, roll and elbow joints were plotted in red, blue, green and magenta respectively. The torque bounds are indicated by horizontal dotted lines. Note that, in agreement with time-optimal control theory, at least one torque limit is saturated at any time instant. The vertical dotted yellow line indicates the time instant of the zero-inertia point.

Next, to illustrate more clearly the improvements permitted by our algorithm, we compared the results given by our algorithm and that given by the algorithm of [15], [22], which use incorrect values for α and β at dynamic singularities. The results suggest that, in the numerical integration of the $(s, \dot{s})$ profiles, the algorithm of [15]

needed to use a time step at least 5 times finer (corresponding thus to an execution time potentially 5 times longer) than our algorithm to achieve the same precision, see Fig. 4.

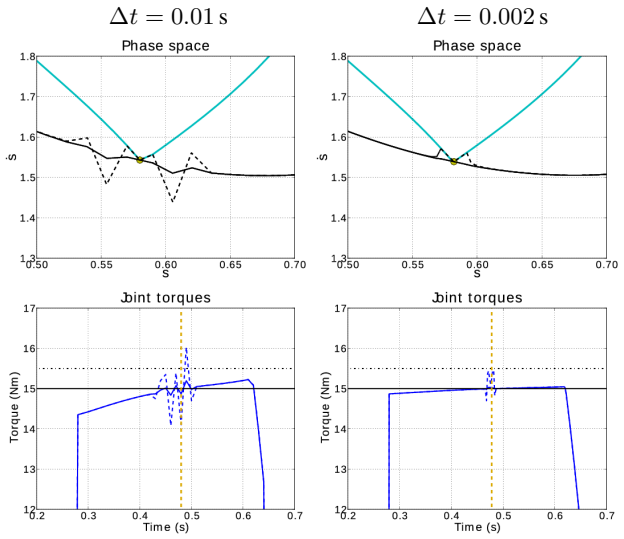

Fig. 4. Close-up views (zoomed in the black boxes of Fig. 3) of the $(s, \dot{s})$ profiles and of the shoulder pitch torque profiles around the switch point, computed using two different integration time steps Δt. Solid lines: results of our algorithm, dotted lines: results of the algorithm proposed by [15]. The vertical dotted yellow line indicates the time instant of the zero-inertia point. Note that our algorithm yielded torques that were within the $+0.5\,\mathrm{N{\cdot}m}$ tolerance even for $\Delta t = 0.01$ while the algorithm of [15] needed to use $\Delta t = 0.002$ (5 times finer) to get within the same tolerance.

IV. General implementation and evaluation

A. Open-source implementation

We provide an implementation of TOPP in C++ that integrates the developments of sections II and III. We also provide an interface in Python for an easy and interactive use. Currently, our implementation supports pure velocity and acceleration bounds, torque bounds, ZMP constraints, and any combination thereof. The dynamics computations are handled by OpenRAVE [30]. Thanks to the general formulation of section II, new systems and constraints can be implemented using a dozen of lines of code. The implementation and test cases are open-source and available at http://www.ntu.edu.sg/home/cuong/software.html.

B. Experimental comparisons with the convex optimization approach

We compared the performance of our implementation with Mintos, K. Hauser's implementation of the convex optimization approach to TOPP [10], which is, to our knowledge, the fastest implementation currently available. We considered the problem of TOPP under pure velocity and acceleration bounds, since those are the only constraints currently available in Mintos.

The test cases are as follows. For a given n, we generated 30 random paths of dimension n. We tested $n = 6$ and $n = 30$, which are the dimensions of usual systems such as industrial manipulators or humanoid robots. Each of the n degrees of freedom of each path is a cubic Bezier curve whose four control points are randomly sampled from a uniform distribution in $[-\pi, \pi]$. For each path, we computed the TOPP with acceleration bounds $\mathbf{a}^{\max} = (1, \ldots, 1)^\top$, $\mathbf{a}^{\min} = (-1, \ldots, -1)^\top$ and $\mathbf{v}^{\max} = (v, \ldots, v)^\top$, $\mathbf{v}^{\min} = (-v, \ldots, -v)^\top$ where $v = 1.2$ for $N = 6$ and $v = 1.5$ for $N = 30$. These values of v were chosen such that both velocity and acceleration bounds can be saturated in the optimal parameterization.

We ran our implementation and Mintos on the test cases. Both implementations were in C++ and compiled and executed on the same computer (Intel Core i5 3.2GHz with 3.8GB memory). The comparison results are given in Fig. 5. The two implementations gave almost the same results: at grid size $N = 1000$, the difference was $< 0.4\%$ of the optimal trajectory duration. Both implementations converged quickly: our implementation came within 1% of the optimal time (taken as the trajectory duration at $N = 1000$) for a grid size $N = 100$ while Mintos did so for $N = 150$. Regarding the computation time, at $N = 150$, our implementation spent on average 0.0011s per path for $n = 6$ and 0.0070s per path for $n = 30$, while Mintos spent respectively 0.0197s and 0.0412s (respectively 18 times and 6 times slower).

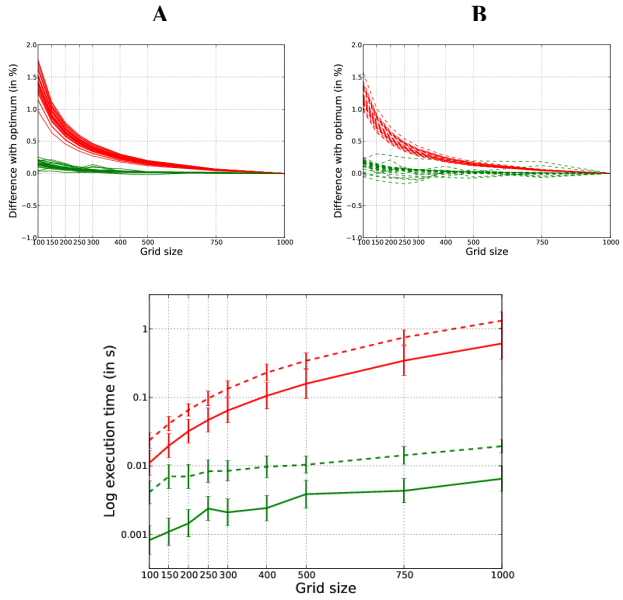

Fig. 5. Comparisons of our implementation with Mintos. **A**: Difference between the time duration of the optimal trajectory at a given grid size and at grid size 1000 (in % of the trajectory duration) for dimension $n = 6$. Mintos in red and our implementation in green. **B**: Same legends as **A** for dimension $n = 30$. **Bottom**: Computation time in seconds (log scale) as a function of grid size. Mintos in red and our implementation in green, $n = 6$ in plain lines and $n = 30$ in dotted lines.

C. Planning fast trajectories for a humanoid robot

Planning whole-body motions for humanoid robots is currently one of the most challenging problems in robotics. The difficulty stems from two main constraints: obstacles and self-collision avoidance on one hand and dynamic balance on the other. The first difficulty can be efficiently addressed by methods such as classical configuration-space planning (e.g. [31]) or hierarchical quadratic programming [32]. However such methods cannot satisfactorily take into account dynamic balance constraints because of the prohibitive cost of moving into the state-space [6]. The second difficulty can be addressed by ZMP- (see e.g. [33]) or capture-point- [34] based methods, but such methods have only been applied on simplified models such as the inverted pendulum or the reaction mass, again because of the high dimensionality of humanoid robots. Some authors have attempted addressing both difficulties by a decoupling approach: first plan a collision-free, *statically*-stable trajectory, and then time-scale it to satisfy dynamic balance [35], [36]. Here, by leveraging the speed of our TOPP implementation, we can accelerate the method of [35], [36] and supplement it with a "kinodynamic shortcutting stage" [5]

to further enhance the quality of the trajectory. Our method is as follows:

(1) Find a collision-free, statically-stable trajectory using RRT;
(2) Post-process with linear shortcuts [37];
(3) Run TOPP on each linear segment, with starting and ending with zero velocity (in order to ensure the C^1 continuity of the overall trajectory), subject to velocity, torques and ZMP constraints;
(4) Post-process with kinodynamic shortcuts under velocity, torques and ZMP constraints.

Step (4) consists of repetitively applying kinodynamic shortcuts as follows:

(a) Pick two random points s_1, s_2 along the trajectory;
(b) Interpolate a cubic polynomial path s_1 and s_2 respecting C^1-continuity at the end-points;
(c) Run TOPP on the interpolated path subject to velocity, torques and ZMP constraints;
(d) If the time duration of the parameterized trajectory is smaller than $s_2 - s_1$, then check for collision (we followed this order because checking for collision is more time-consuming than TOPP). If the path is collision-free then replace the original portion with the new portion. Go back to (a) if the allocated time is not up.

We applied this algorithm on a model of the HRP4 robot, see Fig. 6.

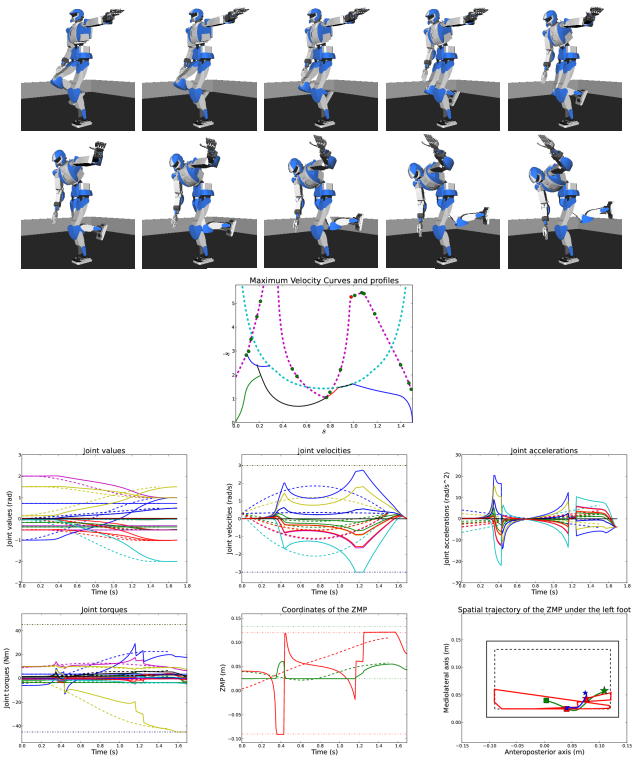

Fig. 6. TOPP on a model of the HRP4 robot with torque, ZMP, and velocity constraints.

V. Conclusion

We have provided a *general*, *fast* and *robust* implementation for solving the Time-Optimal Path Parameterization (TOPP) problem, based on two theoretical contributions.

The first contribution consisted of deriving a general and implementation-friendly formulation of TOPP and showing how diverse problems can be cast into that formulation. While some of these problems (such as acceleration, torque or ZMP constraints) have been addressed with TOPP before, the new formulation allowed very easily extending TOPP to the new problem of general grasp equilibrium, and – we believe – to more new kinodynamic problems in the future.

The second contribution consisted of rigorously characterizing and addressing dynamic singularities, which arise in the numerical integration approach to TOPP. This fully completes the celebrated line of research on TOPP that started in the 1980's with the seminal papers of Bobrow et al. [12] and Shin and McKay [18] and which has since then received contributions from many prominent research groups (e.g. [13], [14], [15], [16], just to cite a few).

Our implementation is robust and fast: on typical test cases, it was between 6 and 18 times faster than the fastest currently available implementation of the convex optimization approach. Using this implementation as a building block, we presented a method to plan collision-free, dynamically balanced, fast trajectories for a humanoid robot. As our implementation is open-source and has been designed so as to facilitate the integration of new systems dynamics and constraints, we hope that it will be useful to robotics researchers interested in kinodynamic motion planning.

Our next immediate goal is to attack the *feasibility* problem (i.e. finding a collision-free, dynamically balanced trajectory in a challenging context, in particular, where quasi-static trajectories are impossible) for humanoid robots using Admissible Velocity Propagation (AVP) [6], which itself is based on TOPP. We believe that the power of AVP combined with the speed of the present TOPP implementation can make possible the planning of unprecedentedly dynamic motions for humanoid robots in challenging environments.

Finally, from a theoretical perspective, we are investigating how higher-order constraints, such as jerk [38], [39] or other types of optimization objectives [8], can be integrated in the TOPP framework.

Acknowledgments

We would like to thank Stéphane Caron, Zvi Shiller and Yoshihiko Nakamura for stimulating discussions regarding TOPP. We are also grateful to S. C. and Rosen Diankov for their helps with the implementation. This work was supported by "Grants-in-Aid for Scientific Research" for JSPS fellows and by a JSPS postdoctoral fellowship.

Appendix

A. Characterizing dynamic singularities

In line with section III-C, consider a zero-inertia point s^* and assume that $a_k(s^*) = 0$ and that $a_k(s) < 0$ in a neighborhood to the left of s^* and $a_k(s) > 0$ in a neighborhood to the right of s^*. We have the following definition and proposition.

Proposition 1. Define, for all $(s, \dot{s})$,

$$\tilde{\alpha}(s,\dot{s}) = \max_{i \neq k} \alpha_i(s,\dot{s}) \ ; \ \tilde{\beta}(s,\dot{s}) = \min_{i \neq k} \beta_i(s,\dot{s}).$$

There exists a neighborhood $]s^* - \epsilon, s^*[$ to the *left* of s^* such that

$$\forall (s,\dot{s}) \in]s^* - \epsilon, s^*[\times [0, \infty[, \ \beta(s,\dot{s}) = \tilde{\beta}(s,\dot{s}).$$

and a neighborhood $]s^*, s^* + \epsilon[$ to the *right* of s^* such that

$$\forall (s,\dot{s}) \in]s^*, s^* + \epsilon[\times [0, \infty[, \ \alpha(s,\dot{s}) = \tilde{\alpha}(s,\dot{s}),$$

Note also that $\tilde{\alpha}$ and $\tilde{\beta}$ are continuous and differentiable in a neighborhood *around* s^*.

Proof : From our assumption that $a_k(s) < 0$ on the left of s^* and $a_k(s) > 0$ on the right of s^*, constraint k gives rise to an α_k on the left of s^* and to a β_k on the right of s^*. It thus does not contribute

to the value of β on the left of s^* or to that of α on the right of s^* $\square$

Proposition 2. If $c_k(s^*) > 0$ then there exists a neighborhood $]s^* - \epsilon, s^*[$ such that $\mathrm{MVC}(s) = 0$ for all $s \in]s^* - \epsilon, s^*[$ (which in turn implies that the path is not traversable).

Proof : Suppose that $c_k(s^*) = \eta > 0$. By continuity of c_k, there exists a neighborhood $]s^* - \epsilon_1, s^*[$ such that

$$\forall s \in]s^* - \epsilon_1, s^*[,\ -c_k(s) < -\eta/2.$$

On the other hand, one has $a_k(s) \uparrow 0$ when $s \uparrow s^*$. Thus, $\alpha_k(s, 0) = \frac{-c_k(s)}{a_k(s)} \to \infty$ when $s \uparrow s^*$. Since $\alpha = \max_i \alpha_i$, we have that $\alpha(s, 0) \to +\infty$ when $s \uparrow s^*$. Next, from Proposition 1, β is continuous, hence *upper-bounded*, in a neighborhood to the left of s^*. Thus, there exists a neighborhood to the left of s^* in which $\alpha(s, 0) > \beta(s, 0)$, which in turn implies that $\mathrm{MVC}(s) = 0$ in that neighborhood $\square$

In light of Proposition 2, we assume from now on that $c_k(s^*) < 0$. We next distinguish two cases according to the value of $b_k(s^*)$.

Case $b_k(s^*) > 0$: Define

$$\dot{s}^* = \sqrt{\frac{-c_k(s^*)}{b_k(s^*)}}. \tag{15}$$

Note that, since $c_k(s^*) < 0$ and $b_k(s^*) > 0$, the expression under the radical sign is indeed positive. Next, let $\dot{s}^\dagger$ be the smallest velocity $\dot{s}$ that satisfies $\tilde{\alpha}(s^*, \dot{s}) = \tilde{\beta}(s^*, \dot{s})$ ($\dot{s}^\dagger = +\infty$ if no such $\dot{s}$ exists).

We now distinguish two sub-cases.

Sub-case $\dot{s}^\dagger < \dot{s}^*$: Let $\dot{s}^\ddagger = (\dot{s}^\dagger + \dot{s}^*)/2$. By the definition of $\dot{s}^*$ and the assumption that $b_k(s^*) > 0$, there exists $\eta > 0$ such that, in a neighborhood to the left of s^*

$$\forall \dot{s} \leq \dot{s}^\ddagger,\ -b_k(s)\dot{s}^2 - c_k(s) > \eta.$$

On the other hand, one has $a_k(s) \uparrow 0$ when $s \uparrow s^*$. Thus, $\alpha_k(s, \dot{s}) = \frac{-b_k(s)\dot{s}^2 - c_k(s)}{a_k(s)} \to -\infty$ when $s \uparrow s^*$ and $\dot{s} \leq \dot{s}^\ddagger$. As a consequence, constraint k does *not* contribute to α in a neighborhood to the left of s^* and for $\dot{s} \leq \dot{s}^\ddagger$. Thus, one has $\alpha = \tilde{\alpha}$ in a neighborhood to the left of s^* and for $\dot{s} \leq \dot{s}^\ddagger$.

By the same argument, one can show that $\beta = \tilde{\beta}$ in a neighborhood to the right of s^* and for $\dot{s} \leq \dot{s}^\ddagger$. Combined with Proposition 1, one has thus obtained that $\alpha = \tilde{\alpha}$ and $\beta = \tilde{\beta}$ in a neighborhood *around* s^* and for $\dot{s} \leq \dot{s}^\ddagger$. This shows that $\mathrm{MVC}(s^*) = \dot{s}^\dagger$, and that the MVC is entirely determined by $\tilde{\alpha}$ and $\tilde{\beta}$ around $(s^*, \dot{s}^\dagger)$. One can thus conclude that the MVC is continuous and differentiable at s^*, which in turn implies that constraint k does *not* trigger a singularity at s^*.

Sub-case $\dot{s}^\dagger > \dot{s}^*$: Remark first that one can find a neighborhood $]s^* - \epsilon, s^* + \epsilon[\times]\dot{s}^* - \eta, \dot{s}^* + \eta[$ in which $\tilde{\alpha}$ is given by a unique α_q, $q \neq k$ and $\tilde{\beta}$ is given by a unique β_p, $p \neq k$. In that neighborhood, define

$$u(s) = \frac{-a_k(s)c_q(s) + a_q(s)c_k(s)}{a_k(s)b_q(s) - a_q(s)b_k(s)};$$

$$v(s) = \frac{-a_k(s)c_p(s) + a_p(s)c_k(s)}{a_k(s)b_p(s) - a_p(s)b_k(s)}.$$

From the assumption that $a_k(s^*) = 0$, one has

$$\lim_{s \uparrow s^*} u(s) = \frac{-c_k(s^*)}{b_k(s^*)} = \dot{s}^{*2}.$$

Thus, in a neighborhood to the left of s^*, one has $0 < u(s) < \dot{s}^{\ddagger 2}$. Next, remark that by definition $\dot{s} = \sqrt{u(s)}$ satisfies $\alpha_k(s, \dot{s}) = \beta_q(s, \dot{s})$. The above two statements together imply that $\mathrm{MVC}(s) = \sqrt{u(s)}$.

One can show similarly that there exists a neighborhood to the right of s^* in which $\mathrm{MVC}(s) = \sqrt{v(s)}$. Combining the results concerning the left and the right of s^*, one obtains that the MVC is *continuous* at s^*, since

$$\lim_{s \uparrow s^*} \mathrm{MVC}(s) = \lim_{s \uparrow s^*} \sqrt{u(s)} = \dot{s}^* = \lim_{s \downarrow s^*} \sqrt{v(s)} = \lim_{s \downarrow s^*} \mathrm{MVC}(s).$$

However, the MVC is *undifferentiable* at s^* since, in general,

$$\lim_{s \uparrow s^*} \mathrm{MVC}'(s) = \lim_{s \uparrow s^*} \left(\sqrt{u(s)}\right)' \neq \lim_{s \downarrow s^*} \left(\sqrt{v(s)}\right)' = \lim_{s \downarrow s^*} \mathrm{MVC}'(s).$$

Thus, in this sub-case, s^* is indeed a dynamic singularity.

Case $b_k(s^*) < 0$: From the assumptions that $c_k(s^*) < 0$ and $b_k(s^*) < 0$, one has that $-b_k(s^*)\dot{s}^2 - c_k(s^*) > 0$ for all $\dot{s}$. Thus, by the same argument as in sub-case $\dot{s}^\dagger < \dot{s}^*$, there exists a neighborhood $]s^* - \epsilon, s^*[$ where $\alpha = \tilde{\alpha}$ and a neighborhood $]s^*, s^* + \epsilon'[$ where $\beta = \tilde{\beta}$. One can thus conclude that constraint k does *not* trigger a singularity at s^*.

References

[1] S. LaValle, *Planning algorithms*. Cambridge Univ Press, 2006.

[2] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[3] J. Bobrow, "Optimal robot plant planning using the minimum-time criterion," *IEEE Journal of Robotics and Automation*, vol. 4, no. 4, pp. 443–450, 1988.

[4] Z. Shiller and S. Dubowsky, "On computing the global time-optimal motions of robotic manipulators in the presence of obstacles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 785–797, 1991.

[5] Q.-C. Pham, "Planning manipulator trajectories under dynamics constraints using minimum-time shortcuts," in *Second IFToMM ASIAN Conference on Mechanism and Machine Science*, 2012.

[6] Q.-C. Pham, S. Caron, and Y. Nakamura, "Kinodynamic planning in the configuration space via velocity interval propagation," in *Robotics: Science and System*, 2013.

[7] K. Shin and N. McKay, "Selection of near-minimum time geometric paths for robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 501–511, 1986.

[8] 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.

[9] W. Suleiman, F. Kanehiro, E. Yoshida, J. Laumond, and A. Monin, "Time parameterization of humanoid-robot paths," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 458–468, 2010.

[10] K. Hauser, "Fast interpolation and time-optimization on implicit contact submanifolds," in *Robotics: Science and Systems*, 2013.

[11] T. Lipp and S. Boyd, "Minimum-time speed optimization over a fixed path," 2013.

[12] 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.

[13] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal of Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[14] J. Slotine and H. Yang, "Improving the efficiency of time-optimal path-following algorithms," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 1, pp. 118–124, 1989.

[15] Z. Shiller and H. Lu, "Computation of path constrained time optimal motions with dynamic singularities," *Journal of dynamic systems, measurement, and control*, vol. 114, p. 34, 1992.

[16] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Robotics: Science and Systems*, vol. 8, 2012, pp. 09–13.

[17] Q.-C. Pham, "Characterizing and addressing dynamic singularities in the time-optimal path parameterization algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[18] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[19] Z. Shiller and Y. Gwo, "Dynamic motion planning of autonomous vehicles," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, 1991.
[20] Q.-C. Pham and Y. Nakamura, "Time-optimal path parameterization for critically dynamic motions of humanoid robots," in *IEEE-RAS International Conference on Humanoid Robots*, 2012.
[21] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.
[22] Z. Shiller, "On singular time-optimal control along specified paths," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 4, pp. 561–566, 1994.
[23] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.
[24] L. Zlajpah, "On time optimal path control of manipulators with bounded joint velocities and torques," in *IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1996, pp. 1572–1577.
[25] M. Walker and D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, p. 205, 1982.
[26] M. Vukobratovic, B. Borovac, and D. Surdilovic, "Zero-moment point–proper interpretation and new applications," in *IEEE/RAS International Conference on Humanoid Robots*, 2001.
[27] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.
[28] A. Bicchi, "On the problem of decomposing grasp and manipulation forces in multiple whole-limb manipulation," *Robotics and Autonomous Systems*, vol. 13, no. 2, pp. 127–147, 1994.
[29] E. V. Zahariev, "Generalized inertia forces of rigid and flexible bodies," *PAMM*, vol. 4, no. 1, pp. 177–178, 2004.
[30] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Carnegie Mellon University, Robotics Institute, August 2010. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf
[31] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000.
[32] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, 2013, to appear. [Online]. Available: https://sites.google.com/site/adrienescandehomepage/publications/2013_IJRR_Escande.pdf
[33] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1404–1409.
[34] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.
[35] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-stable motion planning for humanoid robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, 2002.
[36] F. Kanehiro, W. Suleiman, F. Lamiraux, E. Yoshida, and J. Laumond, "Integrating dynamics into motion planning for humanoid robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 660–667.
[37] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning," *The International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.
[38] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 725–730.
[39] D. Constantinescu and E. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.