Kinodynamic Planning in the Configuration Space via Admissible Velocity Propagation

Quang-Cuong Pham, Stéphane Caron, Yoshihiko Nakamura
Department of Mechano-Informatics, University of Tokyo, Japan

***Abstract*—We propose a method that enables kinodynamic planning in the configuration space (of dimension n) instead of the state space (of dimension $2n$), thereby potentially cutting down the complexity of usual kinodynamic planning algorithms by an exponential factor. At the heart of this method is a new technique – called Admissible Velocity Propagation (AVP) – which, given a path in the configuration space and an interval of reachable velocities at the beginning of that path, computes exactly and efficiently the interval of all the velocities the system can reach after traversing the path while respecting the system kinodynamic constraints. Combining this technique with usual sampling-based methods gives rise to a family of new motion planners that can appropriately handle kinodynamic constraints while avoiding the complexity explosion and, to some extent, the conceptual difficulties associated with a move to the state space.**

I. Introduction

For robots to move dynamically and safely in the real world, it is essential to consider *kinodynamic constraints*, that is, the constraints that stem from the *physical laws* the robots are subject to [4, 10]. Such constraints include for instance force and torque limits for robotic manipulators [1], dynamic balance for humanoid robots [21], or nonholonomic constraints for underactuated robots [11].

There are currently two main approaches to planning collision-free *trajectories* for robots under kinodynamic constraints. The first approach decouples the problem: first, search for a collision-free *path* in the robot *configuration space* $\mathscr{C}$ (e.g. the joint angles of a manipulator, the pose of a mobile robot, etc.) and second, find a time-parameterization of that path that satisfies the kinodynamic constraints [1]. The obvious drawback here is that the path found in the first step may have no time-parameterization at all that respects the kinodynamic constraints. A usual workaround is to include in the first step some configuration-space constraints which guarantee that the path is executable *quasi-statically* (i.e. at arbitrary low velocities, a regime where the effects of dynamics vanish). This idea is frequently used for humanoid robots: for instance, in [8], the authors first search for a path that satisfies static balance – by requiring that the projection of the center of gravity lies in the support area – and then time-parameterize this path using arbitrarily low velocities if necessary.

However, this workaround also suffers from a major limitation: the quasi-static executability condition may be too restrictive and one thus may overlook too many possible solutions. For instance, humanoid robots walking with ZMP-based control are dynamically balanced but rarely satisfy the aforementioned quasi-static condition on the center of gravity [21]. Another example is provided by an actuated pendulum subject to severe torque limits, but which can still be put into the upright position by swinging back and forth several times. It is clear that such solutions make an essential use of the system dynamics and can in no way be discovered by quasi-static methods, nor by any method that considers only configuration-space coordinates.

The second approach requires moving to the *state space* $\mathcal{X}$, which is essentially the configuration space augmented with *velocity coordinates* [10, 5]. While this approach has been proved to be probabilistically complete (i.e., if a solution exists, it can be found with probability 1 after a long enough execution [10, 5]), it results in a two-fold increase in dimension. Since the complexity of planning algorithms usually scales *exponentially* with the dimension of the search space [5], state-space kinodynamic planning may be impractical even for relatively small values of the dimension n of $\mathscr{C}$ [17]. Additional conceptual difficulties include: the design of suitable metric and extension methods [10, 12] in a larger and less intuitive space, the choice of the appropriate ranges of velocities to explore, the numerical errors associated with forward dynamics computations when sampling the control inputs, etc.

In the present paper, we propose a method to plan trajectories under kinodynamic constraints *while staying in the configuration space*. A fundamental requirement for this is the ability to efficiently assess whether a given path in $\mathscr{C}$ can be executed, or *traversed*, while respecting kinodynamic constraints. Furthermore, if one wishes to adopt an *incremental* approach (such as in RRT [7] or PRM [5]), this requirement alone is not sufficient: the concatenation of two traversable paths may not be traversable, since there may exist no common valid velocity at the junction point. Thus, one needs to be able to *propagate velocity intervals*[1]: given an interval of reachable velocities at the *beginning* of a path, determine the interval of all velocities that the system can reach *after* traversing that path while respecting the kinodynamic constraints.

We developed a new algorithm, called Admissible Velocity Propagation (AVP), to precisely solve this velocity propagation problem. The algorithm is based on the classical Time-Optimal Path Parameterization (TOPP) algorithm [1, 19, 20, 18] which,

[1] In theory, the set of valid velocities may not be an interval if we authorize multiple-valued Maximum Velocity Curves (cf. [16]). While there are no fundamental difficulties in handling multiple-valued MVCs, the practical implementation is complex and is left out in the present paper.

given a robotic manipulator and a path in its configuration space (the space of the manipulator joint angles), efficiently computes a time-parameterization of the path which minimizes the traversal time while respecting actuator torque limits. Besides robotic manipulators with torque limits, the TOPP algorithm has been extended to a large spectrum of dynamical systems and kinodynamic constraints such as manipulators with gripper and payload constraints, vehicles with friction constraints, humanoid robots with joint velocity and acceleration limits [9] or ZMP constraints [14], or certain types of nonholonomic systems [2]. All these extensions can be incorporated effortlessly into the AVP algorithm. Note finally that the idea of using the TOPP algorithm to halve the dimension of the search space was first suggested in [17], but in a non-incremental set-up.

In Section II, we summarize and reformulate the TOPP algorithm before presenting, in Section III, the AVP algorithm itself. We then show in Section IV how to combine this AVP algorithm with existing sampling-based planning methods to yield a family of new configuration-space planners that can appropriately handle kinodynamic constraints. To illustrate, we consider in Section V two motion planning problems where dynamics play a critical role: a fully actuated double pendulum with severe torque limits, and a bottle on a tray subject to static friction. We show that quasi-static planning methods are inherently unable to solve these problems, while our approach can address them in a very general and efficient way. Finally, we briefly discuss in Section VI the advantages and limitations of the proposed approach, its theoretical implications, and future developments.

Conventions and notations

- We make a distinction between a *path*, which is a geometric object devoid of any timing information, and a *trajectory*, which is a path endowed with a time-parameterization (velocity profile). Often, we may refer to a path as the *underlying path* of a given trajectory.
- A *quasi-static* trajectory is associated with *arbitrary low velocities*, which cancel the effects of dynamics.
- For convenience, we denote by D^2 the class of functions that are C^1-continuous and piecewise C^2-continuous.

II. Background on the TOPP algorithm

A. Time-parameterization of a path

Consider an n-dof manipulator with dynamics equation

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \mathbf{C}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau, \tag{1}$$

where $\mathbf{q}$ is a $n \times 1$ 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 $n \times 1$ vector of gravity forces and τ the $n \times 1$ vector of actuator torques. Assume that this manipulator is subject to torque limits expressed as: for each joint $i \in [\![1, n]\!]$ and time u,

$$\tau_i^{\min} \leq \tau_i(u) \leq \tau_i^{\max}. \tag{2}$$

Consider now a D^2 path P – represented as the underlying path of a trajectory $\mathbf{q}(u)_{u\in[0,T]}$ – in the manipulator configuration space. The Time-Optimal Path Parameterization (TOPP) algorithm, which we summarize and reformulate below, finds the fastest time-parameterization of that path starting from a given velocity v_{beg} and ending at a given velocity v_{end} while staying under the torque limits. For details, proofs and extensions to different kinds of dynamical systems, the reader is referred to [1, 19, 20, 16, 18, 9, 13, 14].

More precisely, a *time-parameterization* of P is a D^2 increasing function $s : [0, T'] \to [0, T]$. A time-parameterization can be seen alternatively as a *velocity profile*, *i.e.*, the curve $\dot{s}(s)_{s\in[0,T]}$ in the $(s, \dot{s})$ plane. We say that a time-parameterization or, equivalently, a velocity profile, is *valid* if $\dot{s}$ is always strictly positive and if the *re-timed* trajectory $\mathbf{q}(s(t))_{t\in[0,T']}$ satisfies the torque constraints (2) of the system.

Differentiating $\mathbf{q}(s(t))$ with respect to t yields

$$\dot{\mathbf{q}} = \mathbf{q}_s \dot{s}, \quad \ddot{\mathbf{q}} = \mathbf{q}_s \ddot{s} + \mathbf{q}_{ss} \dot{s}^2, \tag{3}$$

where $\mathbf{q}_s = \frac{\mathrm{d}\mathbf{q}}{\mathrm{d}s}$ and $\mathbf{q}_{ss} = \frac{\mathrm{d}^2\mathbf{q}}{\mathrm{d}s^2}$. Substituting (3) into (1) yields

$$\mathbf{M}(\mathbf{q})(\mathbf{q}_s \ddot{s} + \mathbf{q}_{ss}\dot{s}^2) + \mathbf{q}_s^\top \mathbf{C}(\mathbf{q})\mathbf{q}_s \dot{s}^2 + \mathbf{g}(\mathbf{q}) = \tau(s). \tag{4}$$

The above equation can be rewritten in the following form

$$\mathbf{a}(s)\ddot{s} + \mathbf{b}(s)\dot{s}^2 + \mathbf{c}(s) = \tau(s), \quad \text{where}$$

$$\begin{aligned} \mathbf{a}(s) &= \mathbf{M}(\mathbf{q}(s))\mathbf{q}_s(s), \\ \mathbf{b}(s) &= \mathbf{M}(\mathbf{q}(s))\mathbf{q}_{ss}(s) + \mathbf{q}_s(s)^\top \mathbf{C}(\mathbf{q}(s))\mathbf{q}_s(s), \\ \mathbf{c}(s) &= \mathbf{g}(\mathbf{q}(s)). \end{aligned}$$

The torque limits of (2) can now be expressed by the following $2n$ inequalities: for each $i \in [1, n]$

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

Next, if $a_i(s) \neq 0$ (the case $a_i = 0$ corresponds to a "zero-inertia" point, which must be dealt with specifically [18, 9]), one can write

$$\alpha_i(s, \dot{s}) \leq \ddot{s} \leq \beta_i(s, \dot{s}), \quad \text{with}$$

$$\begin{aligned} \alpha_i(s, \dot{s}) &= (\tau_i^\alpha - b_i(s)\dot{s}^2 - c_i(s))/a_i(s), \\ \beta_i(s, \dot{s}) &= (\tau_i^\beta - b_i(s)\dot{s}^2 - c_i(s))/a_i(s), \end{aligned}$$

where τ_i^α and τ_i^β are defined by

$$\begin{cases} \tau_i^\alpha = \tau_i^{\min};\ \tau_i^\beta = \tau_i^{\max} & \text{if } a_i(s) > 0, \\ \tau_i^\alpha = \tau_i^{\max};\ \tau_i^\beta = \tau_i^{\min} & \text{if } a_i(s) < 0. \end{cases}$$

Thus the bounds on $\ddot{s}$ are defined by

$$\alpha(s, \dot{s}) \leq \ddot{s} \leq \beta(s, \dot{s}), \tag{5}$$

where $\alpha(s, \dot{s}) = \max_i \alpha_i(s, \dot{s})$ and $\beta(s, \dot{s}) = \min_i \beta_i(s, \dot{s})$.

B. Maximum Velocity Curve (MVC) and Concatenated Limiting Curve (CLC)

Remark that $(\dot{s}, \alpha(s, \dot{s}))$ and $(\dot{s}, \beta(s, \dot{s}))$ can be seen 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), see Figure 1.

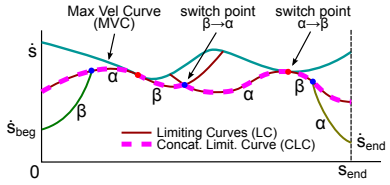

Fig. 1. Illustration for the TOPP algorithm.

Next, observe that if $\alpha(s,\dot{s}) > \beta(s,\dot{s})$, then from (5) there is no possible value for $\ddot{s}$. Thus, to be valid, any velocity profile must stay below the MVC defined by (see also footnote 1)

$$\mathrm{MVC}(s) = \begin{cases} \min\{\dot{s} \geq 0 : \alpha(s,\dot{s}) = \beta(s,\dot{s})\} & \text{if } \alpha(s,0) < \beta(s,0), \\ 0 & \text{if } \alpha(s,0) \geq \beta(s,0). \end{cases}$$

More precisely, it was shown (see e.g. [18]) 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 (see Figure 1). Next, it was shown that possible *switch points* from α to β are to be found on the MVC. More precisely, it was proposed to:

- find all the possible $\alpha \to \beta$ switch points on the MVC;
- from each of these switch points, integrate backward following α and forward following β to obtain the Limiting Curves (LC);
- construct the Concatenated Limiting Curve (CLC) by considering, for each s, the value of the lowest LC at s.

The time-optimal velocity profile can be found by integrating forward from $(0, \dot{s}_{\mathrm{beg}})$ following β and backward from $(s_{\mathrm{end}}, \dot{s}_{\mathrm{end}})$ following α, and by considering the intersection of these profiles with each other or with the CLC (see Figure 1). Note that $\dot{s}_{\mathrm{beg}}$ and $\dot{s}_{\mathrm{end}}$ are computed from the absolute velocities v_{beg} and v_{end} by

$$\dot{s}_{\mathrm{beg}} = v_{\mathrm{beg}} / \|\mathbf{q}_s(0)\|, \quad \dot{s}_{\mathrm{end}} = v_{\mathrm{beg}} / \|\mathbf{q}_s(s_{\mathrm{end}})\|. \tag{6}$$

Before going further, let us state two lemmata that we will use later on.

Lemma 1 Assume that a forward β-profile hits the MVC at $s = s_1$ and a backward α-profile hits the MVC at $s = s_2$, with $s_1 < s_2$, then there exists at least one $\alpha \to \beta$ switch point on the MVC at some position $s_3 \in [s_1, s_2]$.

Lemma 2 Either one of the LCs reaches $\dot{s} = 0$, or the CLC is *continuous*.

Proofs of these lemmata can be found in Appendix A in the Supplementary Material [15] of the present paper.

III. Admissible Velocity Propagation (AVP)

The objective in this section is to provide an algorithm AVP($P, \dot{s}^-_{\mathrm{beg}}, \dot{s}^+_{\mathrm{beg}}$), which takes as inputs:

- a path P in the configuration space, and
- an interval $[\dot{s}^-_{\mathrm{beg}}, \dot{s}^+_{\mathrm{beg}}]$ of initial velocities;

and returns the *interval* (cf. Theorem 1) $[\dot{s}^-_{\mathrm{end}}, \dot{s}^+_{\mathrm{end}}]$ of velocities that the system can reach *at the end* of P after traversing P while respecting the system constraints [2]. This algorithm is the building block that will enable kinodynamic planning in the configuration space. It comprises the following three steps:

A. Compute the limiting curves;
B. Determine the *maximum* final velocity $\dot{s}^+_{\mathrm{end}}$ by integrating *forward* from $s = 0$;
C. Determine the *minimum* final velocity $\dot{s}^-_{\mathrm{end}}$ by bisection search and by integrating *backward* from $s = s_{\mathrm{end}}$.

These three steps are detailed in the next three sections.

A. Computing the limiting curves

Under the conventions and notations introduced previously, we say that a *valid final velocity* is a velocity $\dot{s}_{\mathrm{end}}$ such that there exists a valid profile that starts at $(0, \dot{s}_0)$ for some $\dot{s}_0 \in [\dot{s}^-_{\mathrm{beg}}, \dot{s}^+_{\mathrm{beg}}]$ and ends at $(s_{\mathrm{end}}, \dot{s}_{\mathrm{end}})$.

As in Section II-B, we compute the Concatenated Limiting Curve (CLC) by integrating, from each switch point, forward following β and backward following α, and by concatenating the resulting profiles. Since the CLC is continuous (cf. Lemma 2 in Section II-B) and bounded by $s = 0$ from the left, $s = s_{\mathrm{end}}$ from the right, $\dot{s} = 0$ from the bottom and the MVC from the top (see Figure 2A), there are only five exclusive and exhaustive cases listed below. In the four of them where a solution exists, we define the maximum initial velocity $\dot{s}^*_{\mathrm{beg}}$ from the velocity curves (either MVC or CLC).

A1 One of the limiting curves hits the line $\dot{s} = 0$. In this case, the path cannot be traversed by the system without violating the kinodynamic constraints: AVP returns `Failure`. Indeed, assume for instance that a backward (α) profile hits $\dot{s} = 0$. Then any profile that goes from $s = 0$ to $s = s_{\mathrm{end}}$ must cross that profile somewhere and *from above*, which violates the α bound (see Figure 2A). Similarly, if it is a forward (β) profile that hits $\dot{s} = 0$, then this profile must be crossed somewhere and *from below*, which violates the β bound (see also Figure 2A). Thus, no valid profile can go from $s = 0$ to $s = s_{\mathrm{end}}$;

A2 The concatenated limiting curve hits the MVC while integrating backward and while integrating forward. In this case, let $\dot{s}^*_{\mathrm{beg}} = \mathrm{MVC}(0)$ and go to Section III-B. The situation where there is no switch point is assimilated to this case;

A3 The concatenated limiting curve hits $s = 0$ while integrating backward, and the MVC while integrating forward. In this case, let $\dot{s}^*_{\mathrm{beg}} = \mathrm{CLC}(0)$ and go to Section III-B;

A4 The concatenated limiting curve hits the MVC while integrating backward, and $s = s_{\mathrm{end}}$ while integrating forward. In this case, let $\dot{s}^*_{\mathrm{beg}} = \mathrm{MVC}(0)$ and go to Section III-B;

A5 The concatenated limiting curve hits $s = 0$ while integrating backward, and $s = s_{\mathrm{end}}$ while integrating forward. In this case, let $\dot{s}^*_{\mathrm{beg}} = \mathrm{CLC}(0)$ and go to Section III-B.

[2] Note that [6] also introduced a velocity interval propagation algorithm along a path but for kinematic constraints and dynamic obstacles.

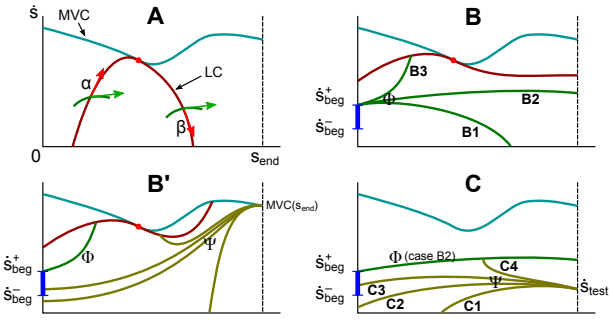

Fig. 2. Illustrations for the AVP algorithm. See the legends of Figure 1.

B. Finding the maximum final velocity

Note that, in any of the cases A2-4, $\dot{s}^*_{\text{beg}}$ was chosen so that no valid profile can start above it. Thus, if $\dot{s}^-_{\text{beg}} > \dot{s}^*_{\text{beg}}$, the path is not traversable and AVP returns `Failure`. Otherwise, the interval of valid initial velocities is $[\dot{s}^-_{\text{beg}}, \dot{s}^{+*}_{\text{beg}}]$ where $\dot{s}^{+*}_{\text{beg}} = \min(\dot{s}^+_{\text{beg}}, \dot{s}^*_{\text{beg}})$.

We argue that the maximum valid final velocity can be obtained by integrating forward from $\dot{s}^{+*}_{\text{beg}}$ following β. Let's call Φ the velocity profile obtained by doing so. Since Φ is continuous and bounded by $s = s_{\text{end}}$ from the right, $\dot{s} = 0$ from the bottom, and either the MVC or the CLC from the top, there are four exclusive and exhaustive cases:

B1 Φ hits $\dot{s} = 0$ (see Figure 2-B, profile B1). Here, as in the case A1 of Section III-A, the path is not traversable and AVP returns `Failure`. Indeed, any profile that starts below $\dot{s}^{+*}_{\text{beg}}$ and tries to reach $s = s_{\text{end}}$ must cross Φ somewhere and *from below*, thus violating the β bound;

B2 Φ hits $s = s_{\text{end}}$ (see Figure 2-B, profile B2). Then $\Phi(s_{\text{end}})$ corresponds to the $\dot{s}^+_{\text{end}}$ we are looking for. Indeed, $\Phi(s_{\text{end}})$ is reachable – precisely by Φ –, and to reach any value above $\Phi(s_{\text{end}})$, the corresponding profile would have to cross Φ somewhere and from below;

B3 Φ hits the CLC. There are two sub-cases:

- B3a If we proceed from cases A4 and A5 (in which the CLC reaches $s = s_{\text{end}}$, see Figure 2B, profile B3), then $\text{CLC}(s_{\text{end}})$ corresponds to the $\dot{s}^+_{\text{end}}$ we are looking for. Indeed, $\text{CLC}(s_{\text{end}})$ is reachable – precisely by the concatenation of Φ and the CLC –, and no value above $\text{CLC}(s_{\text{end}})$ can be valid by the definition of the CLC;
- B3b If we proceed from cases A2 and A3, then the CLC hits the MVC while integrating forward (see Figure 2-B'), say at $s = s_1$; we then proceed as in case B4 below;

B4 Φ hits the MVC, say at $s = s_1$. It is clear that $\text{MVC}(s_{\text{end}})$ is an upper bound of the valid final velocities, but we have to ascertain whether this value is reachable. For this, we use the predicate IS_VALID defined in Box 1.

- If IS_VALID($\text{MVC}(s_{\text{end}})$), then $\text{MVC}(s_{\text{end}})$ is the $\dot{s}^+_{\text{end}}$ we are looking for;
- Else, the path is not traversable: AVP returns `Failure`. Indeed, as we shall see, if $\neg$ IS_VALID($\dot{s}$), then furthermore no value below $\dot{s}$ is reachable.

C. Finding the minimum final velocity

Assume that we proceed from cases B2, B3a, B3b or B4 of Section III-B. Consider a final velocity $\dot{s}_{\text{test}}$ where $\dot{s}_{\text{test}} < \Phi(s_{\text{end}})$ if we proceed from case B2, $\dot{s}_{\text{test}} < \text{CLC}(s_{\text{end}})$ from case B3a, $\dot{s}_{\text{test}} \leq \text{MVC}(s_{\text{end}})$ from cases B3b and B4. Let us integrate backward from $(s_{\text{end}}, \dot{s}_{\text{test}})$ following α and call the resulting profile Ψ. Remark first that Ψ cannot hit the MVC before hitting either Φ or the CLC. Indeed, if we proceed from cases B2 or B3a, then it is clear that Ψ must first hit Φ (case B2) or the CLC (case B3a) before hitting the MVC. If we proceed from cases B3b or B4, assume by contradiction that Ψ hits the MVC first at a position $s = s_2$. Then by Lemma 1 (cf. Section II-B), there must exist a switch point between s_2 and the end of the CLC (in case B3b) or the end of Φ (in case B4). In both cases, there is a contradiction with the fact that the CLC must be continuous (cf. Section II-B). We can now detail in Box 1 the predicate IS_VALID which assesses whether a final velocity $\dot{s}_{\text{test}}$ is valid.

Box 1 IS_VALID($\dot{s}_{\text{test}}$)

Input: candidate final velocity $\dot{s}_{\text{test}}$

Output: `True` iff there exists a valid velocity profile with final velocity $\dot{s}_{\text{test}}$

Consider the profile Ψ constructed above. Since it must hit Φ or the CLC before hitting the MVC, the following five cases are exclusive and exhaustive:

C1 Ψ hits $\dot{s} = 0$ (see Figure 2-C, profile C1). Here, as in A1 of Section III-A or B1 of Section III-B, no velocity profile can reach s_{test}: return `False`;

C2 Ψ hits $s = 0$ for some $\dot{s}_0 < \dot{s}^-_{\text{beg}}$ (see Figure 2-C, profile C2): here, any profile that ends at $\dot{s}_{\text{test}}$ would have to hit Ψ from above, which is contradictory: return `False`;

C3 Ψ hits $s = 0$ at a point $\dot{s}_0 \in [\dot{s}^-_{\text{beg}}, \dot{s}^{+*}_{\text{beg}}]$ (see Figure 2-C, profile C3): then, $\dot{s}_{\text{test}}$ can be reached following the valid velocity profile Ψ: return `True`. (Note that, if $\dot{s}_0 > \dot{s}^{+*}_{\text{beg}}$ then Ψ must have crossed Φ somewhere before arriving at $s = 0$, which is covered by case C4 below.)

C4 Ψ hits Φ (see Figure 2-C, profile C4): then, $\dot{s}_{\text{test}}$ can be reached, precisely by the concatenation of a part of Φ and Ψ: return `True`.

C5 Ψ hits the CLC: then, $\dot{s}_{\text{test}}$ can be reached, precisely by the concatenation of Φ, a part of the CLC and Ψ: return `True`.

At this point, we have that, either the path is not traversable, or we have determined $\dot{s}^+_{\text{end}}$ in Section III-B. Remark from conditions C3, C4 and C5 that, if some $\dot{s}_0$ is a valid final velocity, then any $\dot{s} \in [\dot{s}_0, \dot{s}^+_{\text{end}}]$ is also valid. Similarly, from conditions C1 and C2, if some $\dot{s}_0$ is *not* a valid final velocity, then *no* $\dot{s} \leq s_0$ can be valid. We have thus established the following result:

Theorem 1: Under the assumption of footnote 1, the set of valid final velocities is an interval.

This interval property enables us to efficiently search for the minimum final velocity as follows. First, we test whether 0 is a valid final velocity: if IS_VALID(0), then the sought-after $\dot{s}^-_{\text{end}}$ is 0. Else, we run a standard bisection search with initial bounds $(0, \dot{s}^+_{\text{end}}]$ where 0 is not valid and $\dot{s}^+_{\text{end}}$ is valid. Thus, after executing $\log_2(1/\epsilon)$ times the routine IS_VALID, we can determine $\dot{s}^-_{\text{end}}$ with an error smaller than ϵ.

In terms of complexity, if n is the dimension of the configuration space and N the number of discretization points in the integration schemes, then the AVP algorithm has a complexity of $O(\log(1/\epsilon)Nn^2)$.

IV. Kinodynamic planning in the configuration space using AVP

A. General algorithm

The AVP algorithm presented in Section III is general and can be combined with iterative planning methods such as RRT [7], PRM [5] or dynamic programming [17]. To illustrate this, we choose here to combine it with a variant of RRT called "K nearest neighbors RRT" (KNN-RRT). The resulting algorithm, which we call AVP-RRT, is detailed in Box 2 and illustrated in Figure 3.

Box 2 AVP_RRT($\mathbf{q}_{\text{init}}, v_{\text{init}}, \mathbf{q}_{\text{goal}}, v_{\text{goal}}$)

$U_{\text{init}} \leftarrow$ NEW_VERTEX()
U_{init}.config $\leftarrow \mathbf{q}_{\text{init}}$; U_{init}.inpath $\leftarrow$ `Null`
U_{init}.vmin $\leftarrow v_{\text{init}}$; U_{init}.vmax $\leftarrow v_{\text{init}}$
$\mathcal{T}$.INITIALIZE(U_{init})
for rep $= 1$ to N_{maxrep} **do**
  $\mathbf{q}_{\text{rand}} \leftarrow$ RANDOM_CONFIG()
  $U_{\text{new}} \leftarrow$ EXTEND($\mathcal{T}, \mathbf{q}_{\text{rand}}$)
  if EXTEND succeeds **then**
    if CONNECT($U_{\text{new}}, \mathbf{q}_{\text{goal}}, v_{\text{goal}}$) succeeds **then**
      return COMPUTE_TRAJ($\mathcal{T}, \mathbf{q}_{\text{goal}}$)
    end if
  end if
end for
return `Failure`

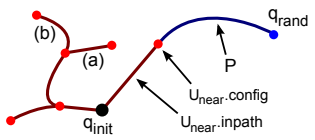

Fig. 3. Illustration for the AVP-RRT algorithm. (a) indicates a straight path and (b) indicates a path that preserves the continuity of the tangent vector, cf. Section IV-B2.

The algorithm iteratively constructs a tree $\mathcal{T}$ in the configuration space. Here, by contrast with the usual RRT, a vertex U consists of a quadruple (U.config, U.inpath, U.vmin, U.vmax) where U.config is a configuration in $\mathscr{C}$, U.inpath is a path $P \subset \mathscr{C}$ that connects the configuration of U's parent to U.config, and [U.vmin, U.vmax] is the interval of reachable velocities at U.config (the end of $P := U$.inpath).

At each step, a random configuration $\mathbf{q}_{\text{rand}}$ is generated. The EXTEND routine (see Box 3) then tries to add $\mathbf{q}_{\text{rand}}$ to the tree by connecting it to its closest – in a configuration-space metric d – vertex in $\mathcal{T}$. If this attempt fails, EXTEND tries with the second closest vertex, and so on up to the K-th closest vertex.

Box 3 EXTEND($\mathcal{T}, \mathbf{q}_{\text{rand}}$)

1: **for** $i = 1$ to K **do**
2: $U_{\text{near}} \leftarrow$ ITH_NEAREST_NEIGHBOR($\mathcal{T}, \mathbf{q}_{\text{rand}}, i$)
3: $P \leftarrow$ INTERPOLATE($U_{\text{near}}, \mathbf{q}_{\text{rand}}$)
4: **if** P is collision-free **then**
5: $(v^-, v^+) \leftarrow$ AVP(P, U_{near}.vmin, U_{near}.vmax)
6: **if** AVP succeeds **then**
7: $U_{\text{new}} \leftarrow$ NEW_VERTEX()
8: U_{new}.config $\leftarrow \mathbf{q}_{\text{rand}}$; U_{new}.inpath $\leftarrow P$
9: U_{new}.vmin $\leftarrow v^-$; U_{new}.vmax $\leftarrow v^+$
10: $\mathcal{T}$.ADD_VERTEX(U_{new})
11: **return** U_{new}
12: **end if**
13: **end if**
14: **end for**
15: **return** `Failure`

Box 4 gives more details about the routines mentioned in Boxes 2 and 3 that are not self-evident.

Box 4 Explanation of the other routines in Boxes 2 and 3

- ITH_NEAREST_NEIGHBOR($\mathcal{T}, \mathbf{q}, i$) returns the i^{th} nearest-neighbor of configuration $\mathbf{q}$ in the tree $\mathcal{T}$; for more details, see Section IV-B1;
- INTERPOLATE($U, \mathbf{q}$) returns a path P between U.config and $\mathbf{q}$; for more details, see Section IV-B2;
- AVP(P, v_0^-, v_0^+) runs the AVP algorithm from Section III to find the valid velocity interval $[v^-, v^+]$ at the end of path P starting from the initial velocity interval $[v_0^-, v_0^+]$;
- CONNECT($U, \mathbf{q}_{\text{goal}}, v_{\text{goal}}$) attempts at connecting directly U to the goal configuration $\mathbf{q}_{\text{goal}}$, using the same algorithm as in (lines 3 to 13 of) EXTEND, but with the further requirement that v_{goal} is included in the final velocity interval;
- COMPUTE_TRAJ($\mathcal{T}, \mathbf{q}_{\text{goal}}$) constructs the entire path from $\mathbf{q}_{\text{init}}$ to $\mathbf{q}_{\text{goal}}$ and computes the optimal time-parameterization of that path by applying the TOPP algorithm one last time.

B. Implementation choices

As in the case of the original kinodynamic RRT [10], some implementation choices are crucial to the performance of the algorithm.

1) Distance metric: Perhaps the most critical choice is that of the metric d which determines the nearest neighbors in ITH_NEAREST_NEIGHBOR. For the original RRT, it was remarked that finding the best metric is actually equivalent to solving the entire planning problem. Here it is also clear that

the metric has a strong impact on the overall performance of AVP-RRT. In addition to configuration space parameters, the design of a suitable metric in our setting may also include the velocity intervals and the orientation of the incoming path associated with each vertex in the tree. However, in an effort to introduce as few heuristics as possible, we only considered in our experiments the simple *Euclidean distance* in the configuration space (see Section V).

2) Interpolating path: In most configuration-space sampling-based algorithms, the path between two vertices is a straight line. Here, if 0 is not a valid velocity at a given vertex, any outgoing path from that vertex must ensure that the tangent vector is continuous at the junction point with the incoming path. Thus, we introduce the following heuristics for INTERPOLATE($U, \mathbf{q}$):

(a) If $U.\text{vmin} = 0$, return the path P which is the straight line in $\mathscr{C}$ connecting $U.\text{config}$ and $\mathbf{q}$. Then, in the subsequent call to AVP, start with $v_{\text{beg}} = 0$ (*i.e.*, call AVP($P, 0, 0$)).

(b) If $U.\text{vmin} > 0$ or if the AVP call in (a) has failed, return a path P made up of third-degree polynomials that ensure that the tangent vector is continuous at the junction between $U.\text{inpath}$ and P. The subsequent call to AVP is done normally (*i.e.*, AVP($P, U_{\text{near}}.\text{vmin}, U_{\text{near}}.\text{vmax}$)).

The two types of paths, (a) and (b), are illustrated in Figure 3.

In general, good heuristics for choosing the interpolating path may significantly improve the performance of the algorithm. However, in an effort to introduce as few heuristics as possible, we only considered the two above choices in our experiments. Note also that the choice of the interpolating path is even more crucial and difficult in the case of *nonholonomic* systems (but it is still possible, cf. [2]).

V. Applications to problems where dynamics play a critical role

A. A double pendulum with severe torque limits

1) System equations: We consider a fully actuated double pendulum (see Figure 4-A), subject to the torque limits

$$|\tau_1| \leq \tau_1^{\max}, \qquad |\tau_2| \leq \tau_2^{\max}.$$

Such a pendulum can be seen as a 2-link manipulator, so one can apply the development of Section II-A as is.

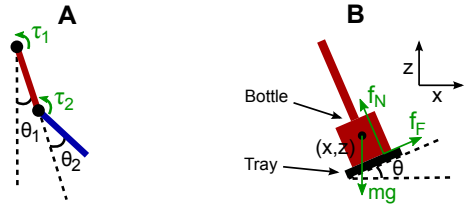

Fig. 4. **A:** A fully actuated double pendulum. **B:** A bottle on a tray. The vectors $\mathbf{f}_N$ and $\mathbf{f}_F$ represent respectively the normal reaction force and the friction force. For the bottle not to move with respect to the tray, one must ensure that $\|\mathbf{f}_F\| \leq \mu \|\mathbf{f}_N\|$, where μ is the coefficient of static friction.

2) Obstruction to quasi-static planning: The task is to bring the pendulum from its initial state $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (0, 0, 0, 0)$ towards the upright state $(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (\pi, 0, 0, 0)$, while respecting torque limits. For simplicity, we do not consider self-collision issues.

Any trajectory that achieves the task must pass through a configuration where $\theta_1 = \pi/2$. Note that the configuration with $\theta_1 = \pi/2$ that requires the smallest torque at the first joint to stay still is $(\theta_1, \theta_2) = (\pi/2, \pi)$. Let then τ_1^{qs} be this smallest torque. It is clear that, if $\tau_1^{\max} < \tau_1^{\text{qs}}$, then *no quasi-static trajectory* can achieve the task.

In our simulations, we used the following lengths and masses for the links: $l = 0.2\,\text{m}$ and $m = 8\,\text{kg}$, yielding $\tau_1^{\text{qs}} = 15.68\,\text{N·m}$. For information, the smallest torque at the second joint to keep the configuration $(\theta_1, \theta_2) = (0, \pi/2)$ stationary was $7.84\,\text{N·m}$. We carried experiments in the following scenarii: $(\tau_1^{\max}, \tau_2^{\max}) \in \{(11, 7), (13, 5), (11, 5)\}$ (N·m).

3) Implementation and results: We implemented the algorithm exactly as described in Section IV in Python. For the number of nearest neighbors to consider, we chose $K = 10$. The maximum number of repetitions was set to $N_{\text{maxrep}} = 2000$. Random configurations were sampled uniformly in $[-\pi, \pi]^2$. Inverse Dynamics computations (required by the TOPP algorithm) were performed using OpenRAVE [3]. We ran 40 simulations for each value of $(\tau_1^{\max}, \tau_2^{\max})$ on a 2 GHz Intel Core Duo computer with 2 GB RAM. The results are given in Table I and Figure 5. Note that only successful trials were taken into account in the subsequent statistics.

TABLE I
Results for the pendulum simulations

$\tau^{\max}$ (N·m)	Success rate	Configs tested	Vertices added	Search time (min)
(11,7)	100%	64±44	31±23	4.2±2.7
(13,5)	100%	92±106	29±30	5.9±6.3
(11,5)	92.5%	212±327	56±81	12.1±15.0

We compared our implementation of AVP-RRT with the usual state-space RRT [10]. To ensure that the comparison is fair, we also considered the variants of state-space RRT with K nearest neighbors, for $K \in \{1, 10, 40, 100\}$. A complete description of the experimental methods is given in Appendix B (Supplementary Material [15]). Figure 6 and Table II summarize the results.

TABLE II
Comparison of AVP-RRT and KNN-RRT

	$\tau^{\max} = (11, 7)$		$\tau^{\max} = (11, 5)$	
Planner	Success rate	Search time (min)	Success rate	Search time (min)
AVP-RRT	100%	3.3±2.6	100%	9.8±12.1
RRT-1	40%	70.0±34.1	47.5%	63.8±36.6
RRT-10	82.5%	53.1±59.5	85%	56.3±60.1
RRT-40	92.5%	44.6±42.6	87.5%	54.6±52.2
RRT-100	82.5%	88.4±54.0	92.5%	81.2±46.7

In the two problem instances, AVP-RRT was respectively 13.4 and 5.6 times faster than the best KNN-RRT in terms of

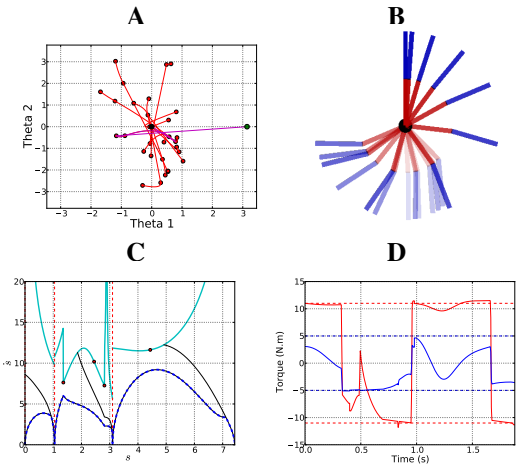

Fig. 5. A typical solution for the case $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ N·m, with trajectory duration 1.88 s (see also the attached video). **A**: The tree in the (θ_1, θ_2) space. The final path is highlighted in magenta. **B**: snapshots of the trajectory, taken every 0.1 s. Snapshots taken near the beginning of the trajectory are lighter. **C**: Velocity profiles in the $(s, \dot{s})$ space. The MVC is in in cyan. The various velocity profiles (CLC, Φ, Ψ, cf. Section III) are in black. The final, optimal, velocity profile is in dashed blue. The vertical dashed red lines correspond to vertices with discontinuous tangent vector, where the velocity must be 0 (cf. Section IV-B2). **D**: Torques profiles. The torques for joint 1 and 2 are respectively in red and in blue. The torque limits are in dotted line. Note that, in agreement with time-optimal control theory, at each time instant, at least one torque limit was saturated (the small overshoots were caused by discretization errors).

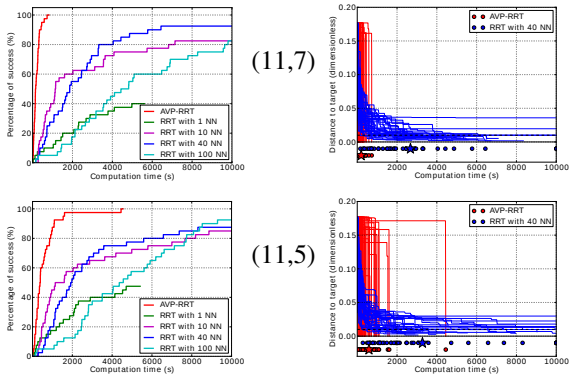

Fig. 6. Comparison of AVP-RRT and KNN-RRT for $\tau^{\max} = (11, 7)$ N·m (top) and $\tau^{\max} = (11, 5)$ N·m (bottom). **Left plots**: Percentage of trials that have reached the goal area at given time instants. **Right plots**: Individual plots for each trial. Each curve shows the distance to the goal as a function of time for a given instance (red: AVP-RRT, blue: RRT-40). Dots indicate the time instants when a trial successfully terminated. Stars show the mean values of termination times.

search time. We noted however that the search time of AVP-RRT increased significantly from instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 5)$ to instance $(\tau_1^{\max}, \tau_2^{\max}) = (11, 7)$, while that of RRT only marginally increased. This may be caused by the "superposition" phenomenon: as torque constraints become tighter, more "pumping" swings are necessary to reach the upright configuration. However, since we plan in the configuration space, configurations with different speeds (corresponding to different pumping cycles) can become indistinguishable. While this problem could easily be addressed by including the reachable velocity interval associated with each vertex in the metric computation, we chose not to do so in the present paper to preserve generality. Nevertheless, AVP-RRT still significantly over-performed KNN-RRT.

B. Moving a bottle on a tray with static friction

1) System equations and reduction to the TOPP form: A bottle is placed upon a tray, which is in turn firmly held by a waiter. The waiter can move the tray in a vertical plane (x, z) and tilt it, in the same vertical plane, by an angle θ (see Figure 4B). However, he is not allowed to touch the bottle. The task is to move the system {bottle + tray} from an initial configuration (x_0, z_0, θ_0) to a goal configuration while avoiding obstacles and in such a way that the bottle never moves with respect to the tray. To ensure the latter, the following two constraints must be satisfied (see Figure 4B):

- the normal reaction force $\mathbf{f}_N$ should be non-negative;
- the friction force should be bounded by $\|\mathbf{f}_F\| \leq \mu \|\mathbf{f}_N\|$, where μ is the static friction coefficient (cf. Coulomb's law for static friction).

Appendix C in the Supplementary Material [15] shows how to reduce the system equations to the TOPP form.

2) Obstruction to quasi-static planning: We designed an environment that precludes quasi-static planning as follows. The initial and goal configurations are respectively $(x, z, \theta) = (0, 0, 0)$ and (2, 0, 0), both with zero velocity. At $x = 0.5$ stands a high wall with a vertical opening (see Figure 7B). The height of the opening is smaller than that of the compound, so that, in order to go through the opening, the compound must be tilted by at least an angle θ_{qs}. Thus, if the static friction coefficient μ is such that $\mu < \tan(\theta_{\text{qs}})$, then *no quasi-static trajectory* can achieve the task.

In our simulations, the height of the opening was chosen such that $\theta_{\text{qs}} = 0.5$, i.e. $\tan(\theta_{\text{qs}}) = 0.55$. We tested two different values of $\mu < 0.55$: $\mu = 0.5$ and $\mu = 0.4$.

3) Implementation and results: Here we used the same implementation as for the double-pendulum of Section V-A3), the only changes consisting in the computation of the kinodynamic constraints (static friction, versus torque limits for the pendulum) in the AVP routine. We ran 40 simulations for each value of μ. Results are given in Table III and Figure 7. Note that only successful trials were taken into account in the subsequent statistics.

TABLE III
RESULTS FOR THE BOTTLE + TRAY SIMULATIONS

μ	Success rate	Configs tested	Vertices added	Search time (min)
0.5	97.5%	309±416	82±108	7.2±10.0
0.4	85%	288±295	76±71	7.4±7.7

VI. Discussion

We have presented a new approach, based on a combination of usual sampling-based planners with a new Admissible Velocity Propagation algorithm, which together enable kinodynamic planning in the configuration space. As shown theoretically and through simulations, this approach can appropriately handle kinodynamic constraints while avoiding the complexity explosion and, to some extent, the conceptual difficulties associated with a move to the state space.

This approach is general and can be applied to a wide spectrum of dynamical systems. To illustrate this point, we addressed two challenging problems involving completely

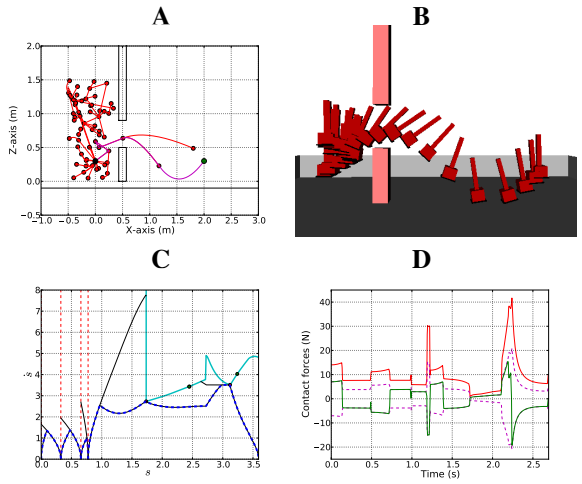

Fig. 7. A typical solution for the case $\mu = 0.4$, with trajectory duration 2.70 s (see also the attached video). **A**: The tree in the (x, z) space. The final path is highlighted in magenta. **B**: snapshots of the trajectory, taken every 0.1 s. **C**: Velocity profiles in the $(s, \dot{s})$ space, same legends as in Figure 5C. **D**: Contact forces. The norm of the reaction force ($N = \|\mathbf{f}_N\|$) is in red and that of the friction force ($F = \|\mathbf{f}_F\|$) is in green. Note that N is always positive and F is always bounded between $-\mu N$ and μN (dotted magenta lines). Furthermore, in agreement with time-optimal control theory, at each time instant, at least one bound is saturated.

different types of kinodynamic constraints (torque limits and static friction, respectively). We successfully applied the same search algorithm to both cases, relying on a minimum number of problem-specific heuristics. Using such heuristics or other heuristics found in the literature (e.g. bidirectional search [10], biased sampling, retraction-based sampling, LQR-based metric [12], etc.) would further improve the algorithm performance.

A considerable amount of work remains to be carried out to achieve a better understanding of the *complexity* of our approach. As the potential gain from planning in the configuration space versus planning in the state space is expected to increase with the dimension of the system under study, we believe that the improvements shown in Section V-A3 will scale up and make possible the resolutions of dynamical systems of higher dimensions (e.g. triple, quadruple pendulums, etc., or humanoid robots – which are inaccessible to existing kinodynamic planners). Another advantage of planning in configuration space, which we have to understand better and experimentally evaluate, lies in the space/time decoupling effect [2]: since *all* possible *trajectories* corresponding to a given collision-free *path* are "exploited" by a single run of AVP, our approach may be particularly relevant when collision-checking is expensive.

Finally, regarding the "completeness" properties, the "superposition" phenomenon mentioned in Section V-A3 is an indication that AVP-RRT in its current form might not be probabilistically complete. Taking into account the velocity interval and/or the orientation of the incoming path for the metric design (cf. Section IV-B1) should overcome this issue. How to do so without compromising the performance of our approach on problem instances arising in practice is the focus of our current investigations.

Acknowledgments: This work was supported by a JSPS postdoctoral fellowship. We are grateful to Prof. Z. Shiller for inspiring discussions about the TOPP algorithm and kinodynamic planning. We thank the Reviewers and the Area Chair for their extensive and insightful comments.

REFERENCES

[1] J.E. Bobrow, S. Dubowsky, and J.S. Gibson. Time-optimal control of robotic manipulators along specified paths. *The International Journal of Robotics Research*, 4(3):3–17, 1985.

[2] F. Bullo and K. Lynch. Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems. *IEEE Transactions on Robotics and Automation*, 17(4):402–412, 2001.

[3] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[4] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

[5] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

[6] J. Johnson and K. Hauser. Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In *IEEE International Conference on Robotics and Automation*, pages 2035–2041, 2012.

[7] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, 2000.

[8] J.J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[9] T. Kunz and M. Stilman. Time-optimal trajectory generation for path following with bounded acceleration and velocity. In *Robotics: Science and Systems*, volume 8, pages 09–13, 2012.

[10] S.M. LaValle and J.J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[11] Y. Nakamura and R. Mukherjee. Nonholonomic path planning of space robots via a bidirectional approach. *IEEE Transactions on Robotics and Automation*, 7(4):500–514, 1991.

[12] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *IEEE International Conference on Robotics and Automation*, 2012.

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

[14] 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.

[15] Q.-C. Pham, S. Caron, and Y. Nakamura. Supplementary material, 2013. URL http://www.normalesup.org/%7Epham/docs/kinodynamic-sup.pdf.

[16] Z. Shiller and S. Dubowsky. Robot path planning with obstacles, actuator, gripper, and payload constraints. *The International Journal of Robotics Research*, 8(6):3–18, 1989.

[17] 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*, 7(6):785–797, 1991.

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

[19] K. Shin and N. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[20] J.J.E. Slotine and H.S. Yang. Improving the efficiency of time-optimal path-following algorithms. *IEEE Transactions on Robotics and Automation*, 5(1):118–124, 1989.

[21] M. Vukobratovic, B. Borovac, and D. Surdilovic. Zero-moment point–proper interpretation and new applications. In *IEEE/RAS International Conference on Humanoid Robots*, 2001.