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 [3, 10]. Such constraints include for instance force and torque limits for robotic manipulators [3], dynamic balance for humanoid robots [21], or nonholonomic constraints for underactuated robots [17].

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 $\mathcal{X}$ (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 may thus 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 $\mathcal{X}$ [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 $\mathcal{X}$ can be executed, or traversed, while respecting kinodynamic constraints. Furthermore, if one wishes to adopt an incremental approach (such as in RRT [2] 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, 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,
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 [12], 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

$$M(q)\ddot{q} + \dot{q}^T C(q)\dot{q} + g(q) = \tau,$$

where $q$ is a $n \times 1$ vector of joint values, $M$ the $n \times n$ manipulator inertia matrix, $C$ the $n \times n \times n$ Coriolis tensor, $g$ the $n \times 1$ vector of gravity forces and $\tau$ 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 $u$,

$$\tau_{i}^{\min} \leq \tau_i(u) \leq \tau_{i}^{\max}.$$  

Consider now a $D^2$ path $P$ — represented as the underlying path of a trajectory $q(u)_{u \in [0, T]}$ — 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 $v_{\text{beg}}$ and ending at a given velocity $v_{\text{end}}$ while staying under the torque limits. For details, proofs and extensions to different kinds of dynamical systems, the reader is referred to [11, 13, 21, 22, 15, 18, 14, 13, 14].

More precisely, a time-parameterization of $P$ is a $D^2$ increasing function $s : [0, T'] \rightarrow [0, T]$. 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 $q(s(t))_{t \in [0, T']}$ satisfies the torque constraints (2) of the system.

Differentiating $q(s(t))$ with respect to $t$ yields

$$\ddot{q} = q_s \dot{s}, \quad \dot{q} = q_s \ddot{s} + q_{ss} \dot{s}^2,$$  

where $q_s = \frac{dq}{ds}$ and $q_{ss} = \frac{d^2q}{ds^2}$. Substituting (3) into (1) yields

$$M(q)(q_s \dot{s} + q_{ss} \dot{s}^2) + \dot{q}^T C(q)q_s \dot{s}^2 + g(q) = \tau(s).$$  

The above equation can be rewritten in the following form

$$a(s) \dot{s} + b(s) \dot{s}^2 + c(s) = \tau(s),$$  

where

$$a(s) = M(q(s))q_s, \quad b(s) = M(q(s))q_{ss} + q_s^T C(q(s))q_s, \quad c(s) = g(q(s)).$$

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

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

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, 3]), one can write

$$\alpha_i(s, \dot{s}) \leq \dot{s} \leq \beta_i(s, \dot{s}),$$

with

$$\alpha_i(s, \dot{s}) = \frac{(\tau_{i}^{\beta} - b_i(s) \dot{s}^2 - c_i(s))}{a_i(s)}, \quad \beta_i(s, \dot{s}) = \frac{(\tau_{i}^{\alpha} - b_i(s) \dot{s}^2 - c_i(s))}{a_i(s)},$$

where $\tau_{i}^{\alpha}$ and $\tau_{i}^{\beta}$ are defined by

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

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

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

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, $\alpha$, in short) to obtain minimum acceleration profiles (or $\alpha$-profiles),
or following the field $\beta$ to obtain maximum acceleration profiles (or $\beta$-profiles), see Figure 3.

Fig. 1. Illustration for the TOPP algorithm.

Next, observe that if $\alpha(s, \dot{s}) > \beta(s, \dot{s})$, then from (3) there is no possible value for $\dot{s}$. Thus, to be valid, any velocity profile must stay below the MVC defined by (see also footnote 2)

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

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 $\beta$ and $\alpha$ fields while always staying below the MVC (see Figure 3). Next, it was shown that possible switch points from $\alpha$ to $\beta$ 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 $\alpha$ and forward following $\beta$ 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, $s_{\text{beg}}$) following $\beta$ and backward from ($s_{\text{end}}$, $s_{\text{end}}$) following $\alpha$, and by considering the intersection of these profiles with each other or with the CLC (see Figure 3). Note that $\dot{s}_{\text{beg}}$ and $\dot{s}_{\text{end}}$ are computed from the absolute velocities $v_{\text{beg}}$ and $v_{\text{end}}$ by

$$\dot{s}_{\text{beg}} = v_{\text{beg}}/\|q_s(0)\|, \quad \dot{s}_{\text{end}} = v_{\text{end}}/\|q_s(s_{\text{end}})\|.$$  \hfill (6)

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

**Lemma 1** Assume that a forward $\beta$-profile hits the MVC at $s = s_1$ and a backward $\alpha$-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 [18] of the present paper.

### III. Admissible Velocity Propagation (AVP)

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

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

and returns the interval (cf. Theorem 1) $[\dot{s}_{\text{end}}, \dot{s}_{\text{end}}]$ of velocities that the system can reach at the end of $P$ after traversing $P$ while respecting the system constraints. 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 $\dot{s}_{\text{end}}$ by integrating forward from $s = 0$;

C. Determine the minimum final velocity $\dot{s}_{\text{end}}$ by bisection search and by integrating backward from $s = s_{\text{end}}$.

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 $\dot{s}_{\text{end}}$ such that there exists a valid profile that starts at $(0, \dot{s}_0)$ for some $\dot{s}_0 \in [\dot{s}_{\text{beg}}, \dot{s}_{\text{beg}}]$ and ends at $(s_{\text{end}}, \dot{s}_{\text{end}})$.

As in Section 3, we compute the Concatenated Limiting Curve (CLC) by integrating, from each switch point, forward following $\beta$ and backward following $\alpha$, and by concatenating the resulting profiles. Since the CLC is continuous (cf. Lemma 2 in Section 3) and bounded by $s = 0$ from the left, $s = s_{\text{end}}$ from the right, $\dot{s} = 0$ from the bottom and the MVC from the top (see Figure 4A), 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 $\dot{s}_{\text{beg}}^*$ 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 ($\alpha$) profile hits $\dot{s} = 0$. Then any profile that goes from $s = 0$ to $s = s_{\text{end}}$ must cross that profile somewhere and from above, which violates the $\alpha$ bound (see Figure 4A). Similarly, if it is a forward ($\beta$) profile that hits $\dot{s} = 0$, then this profile must be crossed somewhere and from below, which violates the $\beta$ bound (see also Figure 4A). Thus, no valid profile can go from $s = 0$ to $s = s_{\text{end}}$.

A2. The concatenated limiting curve hits the MVC while integrating backward and while integrating forward. In this case, let $\dot{s}_{\text{beg}} = \text{MVC}(0)$ and go to Section 3.

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 $\dot{s}_{\text{beg}} = \text{CLC}(0)$ and go to Section 3.

A4. The concatenated limiting curve hits the MVC while integrating backward, and $s = s_{\text{end}}$ while integrating forward. In this case, let $\dot{s}_{\text{beg}} = \text{MVC}(0)$ and go to Section 3.

A5. The concatenated limiting curve hits $s = 0$ while integrating backward, and $s = s_{\text{end}}$ while integrating forward. In this case, let $\dot{s}_{\text{beg}} = \text{CLC}(0)$ and go to Section 3.

2Note that 2 also introduced a velocity interval propagation algorithm along a path but for kinematic constraints and dynamic obstacles.
Failure. Indeed, as we shall see, if $\neg$ IS_VALID($\delta$), then furthermore no value below $\delta$ is reachable.

C. Finding the minimum final velocity

Assume that we proceed from cases B2, B3a, B3b or B4. Consider a final velocity $\delta_{test}$ such that $\delta_{test} < \Phi(s_{end})$ if we proceed from case B2, $\delta_{test} < \text{CLC}(s_{end})$ from case B3a, $\delta_{test} \leq \text{MVC}(s_{end})$ from cases B3b and B4. Let us integrate backward from $(s_{end}; \delta_{test})$ following $\alpha$ and call the resulting profile $\Psi$. Remark first that $\Psi$ cannot hit the MVC before hitting either $\Phi$ or the CLC. Indeed, if we proceed from cases B2 or B3a, then it is clear that $\Psi$ must first hit $\Phi$ (case B2) or the CLC (case B3a) before hitting the MVC. If we proceed from cases B3b or B4, assume by contradiction that $\Psi$ 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 $\Phi$ (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 II the predicate IS_VALID which assesses whether a final velocity $\delta_{test}$ is valid.

**Box 1 IS_VALID($\delta_{test}$)**

**Input:** candidate final velocity $\delta_{test}$

**Output:** True iff there exists a valid velocity profile with final velocity $\delta_{test}$

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

C1 $\Psi$ hits $s = 0$ (see Figure 2-C, profile C1). Here, as in A1 of Section III-A, the path is not traversable and AVP returns Failure. Indeed, no velocity profile can reach $\Phi(s_{end})$ from below, thus violating the $\beta$ condition.

C2 $\Psi$ hits $s = 0$ for some $\delta_0 < \delta_{beg}$ (see Figure 2-C, profile C2). Here, any profile that ends at $\delta_{test}$ will have to hit $\Phi$ from below, which is contradictory: return False.

C3 $\Psi$ hits $s = 0$ at a point $\delta_0 \in [\delta_{beg}; \delta_{beg}^-]$ (see Figure 2-C, profile C3); then, $\delta_{test}$ can be reached following the valid velocity profile $\Psi$: return True. (Note that, if $\delta_0 > \delta_{beg}$ then $\Psi$ must have crossed $\Phi$ somewhere before reaching at $s = 0$, which is covered by case C4 below.)

C4 $\Psi$ hits $\Phi$ (see Figure 2-C, profile C4); then, $\delta_{test}$ can be reached, precisely by the concatenation of a part of $\Phi$ and $\Psi$: return True.

C5 $\Psi$ hits the CLC; then, $\delta_{test}$ can be reached, precisely by the concatenation of $\Phi$, a part of the CLC and $\Psi$: return True.

At this point, we have that, either the path is not traversable, or we have determined $\delta_{beg}$ in Section II.B. Remark from conditions C3, C4 and C2 that, if some $\delta_0$ is a valid final velocity, then any $\delta \in [\delta_0; \delta_{beg}^-]$ is also valid. Similarly, from conditions II and C4, if some $\delta_0$ is not a valid final velocity, then no $\delta \leq s_0$ can be valid. We have thus established the following result:

**Theorem 1:** Under the assumption of footnote II, 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 $s_{end}$ is 0. Else, we run a standard bisection search with initial bounds $(0, s_{end})$ where 0 is not valid and $s_{end}$ is valid. Thus, after executing $\log_2(1/\epsilon)$ times the routine IS_VALID, we can determine $s_{end}$ with an error smaller than $\epsilon$.

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 [11] is general and can be combined with iterative planning methods such as RRT [2], PRM [3] 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($q_{init}$, $v_{init}$, $q_{goal}$, $v_{goal}$)

Start the algorithm from $q_{init}$ and $v_{init}$.

1. \[ U_{init} \leftarrow \text{NEW_VERTEX}() \]
2. \[ U_{init}.config \leftarrow q_{init}; U_{init}.inpath \leftarrow \text{Null} \]
3. \[ U_{init}.vmin \leftarrow v_{init}; U_{init}.vmax \leftarrow v_{init} \]
4. \[ \mathcal{T}.\text{INITIALIZE}(U_{init}) \]
5. \[ \text{for rep = 1 to } N_{\text{maxrep}} \text{ do} \]
6. \[ q_{rand} \leftarrow \text{RANDOM_CONFIG()} \]
7. \[ U_{new} \leftarrow \text{EXTEND}() \]
8. \[ U_{new}.config \leftarrow q_{rand}; U_{new}.inpath \leftarrow \text{null} \]
9. \[ U_{new}.vmin \leftarrow v_{init}; U_{new}.vmax \leftarrow v_{init} \]
10. \[ \mathcal{T}.\text{ADD_VERTEX}(U_{new}) \]
11. \[ \text{end for} \]
12. \[ \text{return Failure} \]

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

Box 3 EXTEND($\mathcal{T}$, $q_{rand}$)

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

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

- \[ \text{ITH_NEAREST_NEIGHBOR}() \]
- \[ \text{INTERPOLATE}(U, q) \]
- \[ \text{AVP}(P, v^-_0, v^+_0) \]
- \[ \text{CONNECT}(U, q_{goal}, v_{goal}) \]
- \[ \text{COMPUTE_TRAJ}(\mathcal{T}, q_{goal}) \]

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 \ref{sec:Nonholonomic}.

2) Interpolating path: In most configuration-space sampling-based algorithms, the path between two vertices is a straight line. Here, if \( \theta \) 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 \textsc{Interpolate}(U, q):

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

(b) If \( U.vmin > 0 \) 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 \( U.inpath \) and \( P \). The subsequent call to AVP is done normally (i.e., \textsc{AVP}(P, U\_near.vmin, U\_near.vmax)).

The two types of paths, (a) and (b), are illustrated in Figure \ref{fig:paths}.

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. \cite{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 \ref{fig:double_pendulum}A), subject to the torque limits

\[ |\tau_1| \leq \tau_1^{max}, \quad |\tau_2| \leq \tau_2^{max} \]

Such a pendulum can be seen as a 2-link manipulator, so one can apply the development of Section \ref{sec:Kinematic} as is.

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 \((\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2) = (\pi/2, \pi/2)\), 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 \( \tau_{1}^{qs} \) be this smallest torque. It is clear that, if \( \tau_1^{max} < \tau_1^{qs} \), 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 \( \tau_1^{qs} = 15.68 \text{ N-m} \). 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 scenario: \((\tau_1^{max}, \tau_2^{max}) \in \{(11, 7), (13, 5), (11, 5)\} \text{ (N-m)}\).

3) Implementation and results: We implemented the algorithm exactly as described in Section \ref{sec:Implementation} in Python. For the number of nearest neighbors to consider, we chose \( K = 10 \). The maximum number of repetitions was set to \( N_{maxrep} = 2000 \). Random configurations were sampled uniformly in \([-\pi, \pi]\). Inverse Dynamics computations (required by the TOPP algorithm) were performed using OpenRAVE \cite{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 \ref{table:pendulum_results} and Figure \ref{fig:results}. Note that only successful trials were taken into account in the subsequent statistics.

![Double Pendulum](image)

![Bottle on a Tray](image)

**Fig. 4.** A: A fully actuated double pendulum. B: A bottle on a tray. The vectors \( f_N \) and \( f_F \) 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 \( \|f_F\| \leq \mu \|f_N\| \), where \( \mu \) is the coefficient of static friction.

<table>
<thead>
<tr>
<th>( \tau^{max} ) (N-m)</th>
<th>Success rate</th>
<th>Configs tested</th>
<th>Vertices added</th>
<th>Search time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(11, 7)</td>
<td>100%</td>
<td>61±22</td>
<td>29±13</td>
<td>4.2±1.7</td>
</tr>
<tr>
<td>(13, 5)</td>
<td>100%</td>
<td>92±10</td>
<td>29±13</td>
<td>5.9±6.3</td>
</tr>
<tr>
<td>(11, 5)</td>
<td>92.5%</td>
<td>212±237</td>
<td>56±81</td>
<td>12.1±15.0</td>
</tr>
</tbody>
</table>

We compared our implementation of AVP-RRT with the usual state-space RRT \cite{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 \cite{13}). Figure \ref{fig:results} and Table \ref{table:comparison} summarize the results.

**TABLE II**

<table>
<thead>
<tr>
<th>Planner</th>
<th>Success rate</th>
<th>Search time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVP-RRT</td>
<td>100%</td>
<td>9.8±12.1</td>
</tr>
<tr>
<td>RRT-1</td>
<td>40%</td>
<td>63.8±36.6</td>
</tr>
<tr>
<td>RRT-10</td>
<td>82.5%</td>
<td>36.3±60.1</td>
</tr>
<tr>
<td>RRT-40</td>
<td>92.5%</td>
<td>54.6±52.2</td>
</tr>
<tr>
<td>RRT-100</td>
<td>82.5%</td>
<td>81.2±46.7</td>
</tr>
</tbody>
</table>

In the two problem instances, AVP-RRT was respectively 13.4 and 5.6 times faster than the best KNN-RRT in terms of
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 \(\theta\). However, he is not allowed to touch the bottle. The task is to move the system \{bottle + tray\} from an initial configuration \((x_0, z_0, \theta_0)\) 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 BB):

- the normal reaction force \(f_N\) should be non-negative;
- the friction force should be bounded by \(||f_T\| \leq \mu ||f_N||\), where \(\mu\) is the static friction coefficient (cf. Coulomb’s law for static friction).

Appendix C in the Supplementary Material [13] 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 BB). 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 \(\theta_{\text{OB}}\). Thus, if the static friction coefficient \(\mu\) is such that \(\mu < \tan(\theta_{\text{OB}})\), then no quasi-static trajectory can achieve the task.

In our simulations, the height of the opening was chosen such that \(\theta_{\text{OB}} = 0.5\), i.e. \(\tan(\theta_{\text{OB}}) = 0.55\). 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.A.2, 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 \(\mu\). Results are given in Table II and Figure A. Note that only successful trials were taken into account in the subsequent statistics.

<table>
<thead>
<tr>
<th>(\mu)</th>
<th>Success rate</th>
<th>Configs tested</th>
<th>Vertices added</th>
<th>Search time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>97.5%</td>
<td>399±416</td>
<td>82±108</td>
<td>7.2±10.0</td>
</tr>
<tr>
<td>0.4</td>
<td>85%</td>
<td>288±295</td>
<td>76±71</td>
<td>7.4±7.7</td>
</tr>
</tbody>
</table>

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

search time. We noted however that the search time of AVP-RRT increased significantly from instance \((\tau_{\text{max}}^1, \tau_{\text{max}}^2) = (11, 5)\) to instance \((\tau_{\text{max}}^1, \tau_{\text{max}}^2) = (11, 7)\), 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.

Fig. 5. A typical solution for the case \((\tau_{\text{max}}^1, \tau_{\text{max}}^2) = (11, 5)\) N-m, with trajectory duration 1.88 s (see also the attached video). A: The tree in the \((\theta_1, \theta_2)\) 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, \(\Phi, \Psi\), cf. Section IB) 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 V.B). 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 \(\tau_{\text{max}} = (11, 7)\) N-m (top) and \(\tau_{\text{max}} = (11, 5)\) 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.

TABLE III

RESULTS FOR THE BOTTLE + TRAY SIMULATIONS

<table>
<thead>
<tr>
<th>(\mu)</th>
<th>Success rate</th>
<th>Configs tested</th>
<th>Vertices added</th>
<th>Search time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>97.5%</td>
<td>399±416</td>
<td>82±108</td>
<td>7.2±10.0</td>
</tr>
<tr>
<td>0.4</td>
<td>85%</td>
<td>288±295</td>
<td>76±71</td>
<td>7.4±7.7</td>
</tr>
</tbody>
</table>
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.A 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.A 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 V.B) 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 post-doctoral 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