The 2nd IFToMM Asian Conference on Mechanism and Machine Science
November 7-10, 2012, Tokyo, Japan

Planning manipulator trajectories under dynamics constraints using minimum-time shortcuts

Quang-Cuong Pham*

Planning minimum-time, dynamically-feasible, collision-free trajectories for robotic manipulators is essential to improve industrial productivity. Yet this difficult problem is still waiting for robust and efficient algorithms. Here we propose to tackle it by bringing together two methods : fixed-path time-minimization under dynamics constraints and shortcuts-based smoothing. We implement our algorithm using the OpenRAVE platform, and demonstrate its efficiency through simulations on a dynamic model of the 4-dof Barrett Whole-Arm Manipulator.

Keywords: *minimum time, dynamics, torques, sampling, shortcut, motion planning*

1. Introduction

Following a great thinker of the 19th century – who once wrote that "the more the productivity of labour increases, the more can the working-day be shortened and, [in a future society], the greater the time for the free development, intellectual and social, of the individual" – it is easy to conceive that time-minimization of robotic motions is bound to play a major role in social progress. Yet because of the nonlinearity and the large number of degrees of freedom associated with industrial robotic manipulators, finding a minimum-time, dynamically-feasible, collision-free trajectories for these systems is still an open problem.

If a *path* is given between two manipulator configurations $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{end}}$, efficient algorithms for computing a *time-parameterization* of that path which minimizes the traversal time while respecting dynamic constraints exist [2, 14, 15, 13]. Extensions of these algorithms to the non-fixed-path case are mainly made by varying the path (using iterative path modification [1] or extensive grid search [12]) and then running the fixed-path algorithm on the so-modified path. However, as noted in e. g. [6], iterative path-modification methods are ill-adapted to highly cluttered environments because a costly collision-checker must be called at each path-modification iteration and because of the issue of local minima. On the other hand, grid-search-based methods are ill-adapted to manipulators with large numbers of degrees of freedom.

More recently, approaches combining *sample-based planners* (see e.g. [9]) and *shortcuts-based smoothers* have proved to be particularly adapted to problems involving cluttered environments and manipulators with large numbers of degrees of freedom [5, 6]. Indeed, one advantage of making shortcuts is that the collision-checker need only be called on *portions* of the trajectory. In addition, the random and non-local nature of shortcuts allow effectively exploring high-dimensional spaces and, to some extent, alleviating the issue of local minima.

However, the works just mentioned did not explicitly consider time-minimization [5], or did so under *kinematics* (such as velocity and acceleration limits), and not dynamics, constraints [6]. Yet dynamics constraints such as torque limits are the actual constraints that apply on robots in the physical world. Taking into account these constraints as such is thus essential if planning algorithms are to find their ways into industrial applications. Some recent approaches [11, 7] allow taking into account dynamics-based *cost* (e. g. by minimizing the integrated square torque) but cannot satisfactorily handle dynamics hard *constraints* (such as torque limits).

Here we bring together the two ideas mentioned previously, namely fixed-path time-minimization under dynamics constraints and shortcuts-based smoothing, in order to provide an efficient method to plan time-*optimized*, dynamically-feasible, collision-free trajectories. Note that we used the term "time-optimized" and not "minimum-time" since the method is not guaranteed to provide the global minimum-time trajectory.

The article is organized as follows : section 2. recalls the necessary backgrounds and outlines the main algorithm, section 3. presents simulation results for a 4-dof Barrett Whole-Arm Manipulator (WAM), and section 4. offers a brief conclusive discussion.

2. Proposed algorithm

2.1 Background : minimum-time parameterization of a fixed path

The main ingredient of our algorithm is the time-minimization under dynamics constraints along a fixed path developed in the 80's and the beginning of the 90's [2, 15, 13], which we summarize below.

Consider the dynamics equation of a general n-dof

*Q.-C. Pham is with Nakamura-Takano Laboratory, Department of Mechano-Informatics, University of Tokyo, Japan. Email : cuong.pham@normalesup.org.

manipulator

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

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. We consider dynamics constraints of the following form : for all $i \in [1, n],\ t \in [0, T]$

$$\tau_i^{\min} \leq \tau_i(t) \leq \tau_i^{\max} \quad (2)$$

Assume that we are given a C^1-continuous, piecewise C^2 trajectory $\mathbf{q}(t)_{t\in[0,T]}$ (which does not necessarily respect the torque limits). The goal is to find a time-parameterization of its underlying path which minimizes the traversal time while respecting the torque limits. A time-parameterization is a C^1-continuous, piecewise C^2, increasing function $s : [0, T'] \to [0, T]$.

First, we can express $\mathbf{q}$ as a function of the parameterization s as $\mathbf{q} = \mathbf{q}(s)$, which in turn yields

$$\dot{\mathbf{q}} = \mathbf{q}_s \dot{s} \quad (3)$$

and

$$\ddot{\mathbf{q}} = \mathbf{q}_s \ddot{s} + \mathbf{q}_{ss}\dot{s}^2 \quad (4)$$

Substituting (3) and (4) into (1) yields

$$\mathbf{M}(\mathbf{q})(\mathbf{q}_s\ddot{s}+\mathbf{q}_{ss}\dot{s}^2)+\mathbf{q}_s^\top \mathbf{C}(\mathbf{q})\mathbf{q}_s\dot{s}^2+\mathbf{g}(\mathbf{q}) = \tau(s), \quad (5)$$

The above equation can be rewritten in the following form

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

where

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

Now the torque limits of equations (2) can be expressed by the $2n$ equations : for all $i \in [1, n]$

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

Next, if $a_i(s) \neq 0$, one can write

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

with

$$\alpha_i(s, \dot{s}) = (\tau^\alpha - b_i(s)\dot{s}^2 - c_i(s))/a_i(s)$$
$$\beta_i(s, \dot{s}) = (\tau^\beta - b_i(s)\dot{s}^2 - c_i(s))/a_i(s),$$

where τ_i^α and τ_i^β are defined by

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

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

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

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

Remark that, if $\alpha(s, \dot{s}) = \beta(s, \dot{s})$, then there is only one possible value for $\ddot{s}$, namely, $\ddot{s} = \alpha(s, \dot{s}) = \beta(s, \dot{s})$. If $\alpha(s, \dot{s}) > \beta(s, \dot{s})$, then there is no possible value for $\ddot{s}$, and the trajectory must be stopped. Thus, following [2, 13], one defines the *maximum velocity curve* $\gamma(s)_{s\in[0,T]}$ by

$$\gamma(s) = \sup\{v : \forall v' < v,\ \alpha(s, v') \leq \beta(s, v)\}.$$

It can be proved [2] that the minimum-time parameterization s is obtained by integrating alternatively the vector fields β and α in the phase plane $(s, \dot{s})$, while always staying below the curve γ (see Fig. 1). To find the switching points, i. e. the points where the integration switches from the vector field β to α and vice versa, several methods have been proposed, see section 2.2.3 for a more detailed discussion.

2.2 Implementation details for the fixed-path algorithm

We give here some implementation details for the above algorithm that were not clearly available in the previous works.

2.2.1 Computing the vectors **a**, **b** and **c**

To compute the vectors $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ at a given s, it is actually *not* necessary to evaluate completely the matrix $\mathbf{M}$ and the tensor $\mathbf{C}$. Using the Recursive Euler-Newton algorithm [17, 3], one can indeed evaluate directly the needed quantities, that is $\mathbf{M}\mathbf{q}_s$, $\mathbf{M}\mathbf{q}_{ss}$ and $\mathbf{q}_s^\top \mathbf{C}\mathbf{q}_s$, without computing explicitly the full $\mathbf{M}$ and $\mathbf{C}$.

Note also that the cases of closed-link mechanisms and of time-varying structure can be handled within the same framework using [10].

2.2.2 Including joint velocity limits

Usually, in addition to the torque limits of (2), the joints must also respect velocity limits of the form: for all $i \in [1, n],\ t \in [0, T]$

$$|\dot{q}_i(t)| \leq \dot{q}_i^{\max}. \quad (6)$$

From equation (3), one has $\dot{q}_i(s) = (\mathbf{q}_s)_i \dot{s}$, which in turn yields $\dot{s} \leq \frac{\dot{q}_i^{\max}}{|(\mathbf{q}_s)_i|}$. Thus, if one defines the *velocity-limits-generated maximum velocity curve* $\gamma^v(s)_{s\in[0,T]}$ by

$$\gamma^v(s) = \min_{i\in[1,n]} \frac{\dot{q}_i^{\max}}{|(\mathbf{q}_s)_i|},$$

then the phase plane trajectory must also always stay below γ^v.

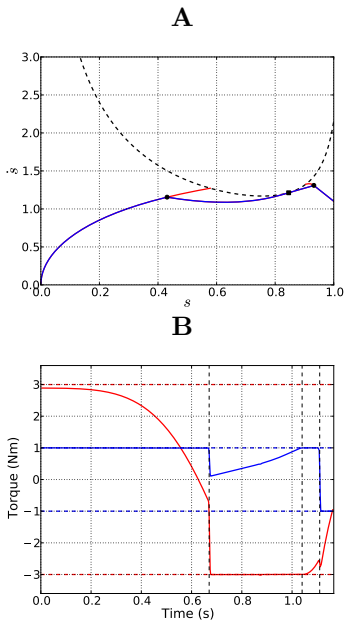

Fig.1 Minimum-time parameterization for a planar two-links manipulator on a fixed path. The two links have the same length (0.2m) and mass (8kg). The torque limits were ±3Nm for joint 1 (proximal) and ±1Nm for joint 2 (distal). The initial trajectory of the joint values was $(t + 0.5, t^2 + 2t)_{t\in[0,1]}$. **A**: The $(s, \dot{s})$ phase plane: the maximum velocity curve is plotted in black dashed line. The red curves represent (a) the trajectory integrated forward from the initial condition $(0, 0)$ following the vector field β, (b) the trajectory integrated backward from the final condition $(1, 1.1)$ following α, (c) the trajectory integrated forward from the tangent switching point (square dot) following β and (d) the trajectory integrated backward from the tangent point following α. The blue trajectory is the minimum-time phase plane trajectory obtained by following successively portions of the red trajectories (a), (d), (c), (b), in this order. The corresponding minimum-time parameterization yields a movement duration of 1.2s. **B**: The torque history corresponding to the minimum-time parameterization: red for joint 1 and blue for joint 2. The dashed red and blue lines represent the torque limits. The vertical black dashed lines correspond to the switching time instants. Note that, in agreement with the theory, at least one actuator saturates at every moment in time: actuator 2 between $t = 0$ and the first switching point, actuator 1 between the first switching point and the second switching point (the tangent switching point), etc. At the tangent switching point (near $t = 1.1$), both actuators saturate tangentially to their constraints (cf. [15]).

In practice, when the phase plane trajectory touches γ^v during the integration of α or β then this trajectory will be allowed to "slide" along γ^v as long as the tangent of γ^v at s is comprised between the minimum and maximum acceleration vectors defined by $\alpha(s)$ and $\beta(s)$, see Fig. 1 and e.g. [8] for more details. However, we note along with [18] that, by following γ^v, the algorithm might sometimes get stuck in a "trap" region. This point is currently under investigation.

2.2.3 Finding the switching points

In [15], a method is proposed to find the switching points without explicitly evaluating the maximum velocity curve γ. While this method has helped gaining insights into the problem at hand, we found, along with [13], that it has in practice the same complexity as a direct search along γ, while being less robust to discretization issues.

To compute γ at a given s, we propose to solve the quadratic equation $\alpha_i(s, v) = \beta_j(s, v)$ in the unknown v for $i \neq j$, which gives (at most) one positive solution $v_{ij}(s)$. Then $\gamma(s)$ can be defined by $\gamma(s) = \min_{i,j} v_{ij}(s)$. The complexity of this procedure is $O(n^2)$.

2.3 The global algorithm

We propose the following algorithm to find a global (i. e. non-fixed-path) time-optimized, dynamically-feasible, collision-free trajectory between two configurations $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{end}}$.

1. find a collision-free path between $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{end}}$, using e. g. a sampling-based planner;
2. time-parameterize this path to obtain a C^1-continuous, piecewise C^2 trajectory, then run the algorithm of section 2.1 to obtain the minimum-time parameterization $\mathbf{q}(t)_{t\in[0,T]}$ of the underlying path;
3. pick two random points $t_1, t_2 \in [0, T]$ and generate a candidate shortcut between these two points;
4. test if this shortcut is collision-free, if not, return to step 3;
5. run the algorithm of section 2.1 on the shortcut to obtain the minimum-time parameterization of the shortcut path;
6. if the new time duration is smaller than $t_2 - t_1$, then replace the original trajectory portion by the shortcut;
7. repeat steps 3 to 6 until the allotted time is over.

We now discuss some implementation details.

- In step 3, we generate a candidate shortcut by interpolating, for each i, a 3rd-degree polynomial of time duration $t_2 - t_1$ between $(q_i(t_1), \dot{q}_i(t_1))$ and $(q_i(t_2), \dot{q}_i(t_2))$. Since a 3rd-degree polynomial has 4 free coefficients and there are 4 linear conditions

(position and velocity at the beginning and the end of the shortcut), the interpolation problem is well-defined.

- In step 5, to ensure the C^1-continuity of the resulting trajectory, we use the initial and final values $\dot{s}_{\text{init}} = \dot{s}_{\text{end}} = 1$ when integrating in the $(s, \dot{s})$ plane (cf. the end of section 2.1).
- Steps 4 and 5 should be switched if the algorithm of section 2.1 happens to be faster than collision-checking (for instance in highly cluttered environments).

3. Simulation results

We ran simulations on a dynamic model of the 4-dof Barrett WAM using OpenRAVE [4]. The torque limits were set at ±10Nm, ±15Nm, ±6Nm, ±6Nm respectively for the shoulder yaw, pitch and roll joints and the elbow joint. The velocity limits were set at $\pm 4 \text{rad.s}^{-1}$ for all joints. We used the same reaching-under-the-table paradigm as in [6].

Instead of step 1 of the global algorithm just presented, we manually set three intermediate configurations such that a collision-free path can be easily obtained by interpolating between the initial and final configurations and theses via-points (trajectory 1 : left column of Fig. 2). We then ran steps 2 to 7 of the presented algorithm as they are. The minimum-time, dynamically-feasible parameterization of the initial path (obtained after step 2) had time duration 2.22s (trajectory 2 : middle column of Fig. 2). The computation time allotted for each execution of the algorithm was 15s on a 2GHz Intel Core Duo computer with 2GB RAM. Note finally that the algorithm is currently prototyped in Python : we thus expect a significant gain in performance after transcription into C++.

The trajectory obtained from the best execution of the shortcutting algorithm (trajectory 3) is shown in the right column of Fig. 2. Six shortcuts were effectively made, out of the 49 attempted within the 15s time limit (a shortcut was rejected because either it involved collisions, or was not dynamically feasible or had a time duration longer than that of the original portion). The resulting trajectory had time duration 1.03s, which was 46.4% of the time duration before shortcutting. The velocity and torque profiles for trajectories 2 and 3 are shown in Fig. 3.

Fig. 4 next shows statistics over 100 executions of the shortcutting algorithm. The average final trajectory duration (± standard deviation) across the 100 runs was 1.21±0.07s (54.5% of the initial fixed-path minimum-time trajectory). One can observe that, after about 6 effective shortcuts or about 30 shortcut attempts, no significant improvements were further made. Thus, in the present experiment, stopping the algorithm after about 10s of execution time (instead

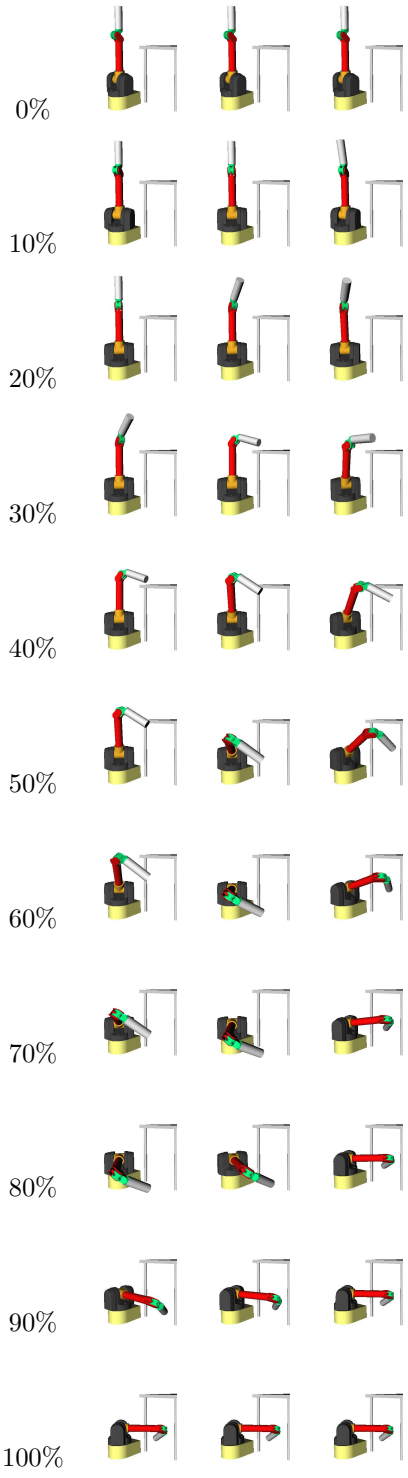

Fig.2 Simulations of a reaching-under-the-table movement using a dynamic model of the Barrett WAM (a video is available at `http://www.normalesup.org/~pham/shortcuts.mov`). **Left** : snapshots of the original interpolated trajectory. **Middle** : minimum-time parameterization of the original path, with time duration 2.22s. **Right** : the trajectory obtained from the best execution of the shortcutting algorithm, with time duration 1.03s.

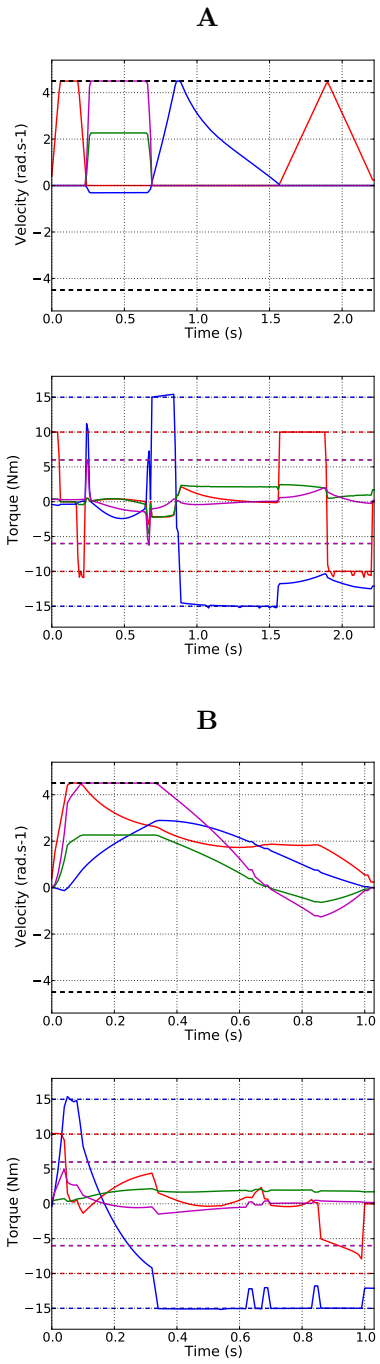

Fig.3 **A** : velocity and torque profiles for the initial fixed-path minimum-time trajectory (2.22s). The red, blue, green and magenta curves represent respectively the torques of the shoulder yaw, pitch, roll and the elbow joints. The velocity and torque limits were plotted in dashed lines. **B** : velocity and torque profiles for the best trajectory out of 100 executions of the shortcutting algorithm (1.03s). Note that, in agreement with the theory, at every moment in time, either a velocity limit or a torque limit is saturated, except for some short periods because of discretization issues.

of 15s as in the current implementation) would not significantly impair the performances. Note nevertheless that, in each run, the current trajectory did not converge to the global optimal trajectory (which had time duration 1.03s). One workaround could consist of restarting the shortcutting algorithm several times from the initial fixed-path minimum-time trajectory and choosing the best solution. For instance, if we put the 100 runs into 10 bins and consider the best result in each bin, then the average time duration becomes 1.12±0.04s, which is very close to the overall best (1.03s).

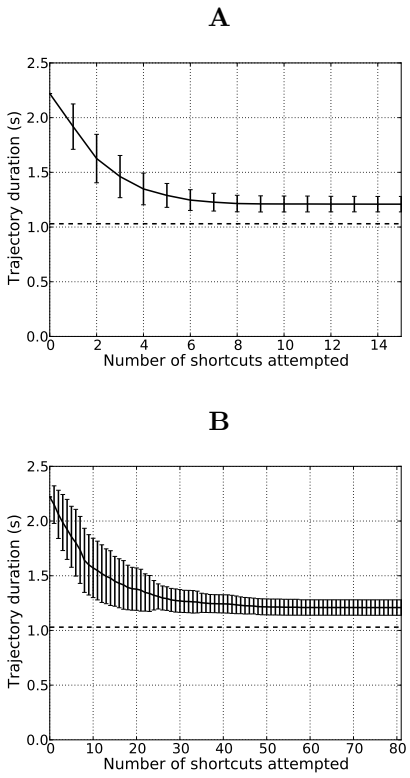

Fig.4 Statistics over 100 runs. **A** : average time duration (± standard deviation) of the trajectory after k *effective* shortcuts. The average time duration at the end of the shortcutting algorithm was 1.21s (54.5% of the original duration). The dashed horizontal line represents the time duration of the best of the 100 runs (1.03s). **B** : average time duration (± standard deviation) of the trajectory after k *attempted* shortcuts, i.e. including those rejected because they involved collisions, were not dynamically feasible or had a time duration longer than that of the original portion.

4. Discussion

We have presented an algorithm to find time-optimized, dynamically-feasible, collision-free trajectories for robotic manipulators based on two main ingredients : fixed-path time-minimization under dy-

namics constraints and shortcuts-based smoothing. This algorithm was implemented in OpenRAVE and tested on a dynamic model of the Barrett WAM. The simulation results show that the algorithm is very efficient.

We are investigating several heuristics to improve the performance of the global algorithm, such as: how to generate better candidate shortcuts between two configurations (other than simply interpolating a polynomial as currently done), how to choose random t_1 and t_2 in step 3 of the algorithm, how to deal with the issue of local minima, etc.

Regarding the fixed-path algorithm, there is also much room for improvements. In particular, the issue of "trap" regions when including the joint velocity limits (cf. section 2.2.2) is still waiting for a robust and efficient solution. Taking into account third-order dynamic constraints, such as jerk limits, is also very important to avoid torque jumps. Future implementations of our algorithm will include this aspect, building e.g. from [16].

Finally, we are planning to test the algorithm on an actual industrial robot.

Acknowledgments

We would like to thank Dr. R. Diankov for stimulating discussions and his help with OpenRAVE, and Prof. Y. Nakamura, Prof. Z. Shiller and two anonymous reviewers for their valuable suggestions. This work was supported by "Grants-in-Aid for Scientific Research" for JSPS fellows and by a JSPS postdoctoral fellowship.

References

[1] J.E. Bobrow. Optimal robot plant planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*, 4(4):443–450, 1988.

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

[3] P.I. Corke. A robotics toolbox for matlab. *IEEE Robotics & Automation Magazine*, 3(1):24–32, 1996.

[4] 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`.

[5] R. Geraerts and M.H. Overmars. Creating high-quality paths for motion planning. *The International Journal of Robotics Research*, 26(8):845–863, 2007.

[6] K. Hauser and V. Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *IEEE International Conference on Robotics and Automation*, pages 2493–2498, 2010.

[7] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation*, pages 4569–4574, 2011.

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

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

[10] Y. Nakamura and K. Yamane. Dynamics computation of structure-varying kinematic chains and its application to human figures. *IEEE Transactions on Robotics and Automation*, 16(2):124–134, 2000.

[11] N. Ratliff, M. Zucker, J.A. Bagnell, and S. Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation*, pages 489–494, 2009.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] M.W. Walker and D.E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104: 205, 1982.

[18] L. Zlajpah. On time optimal path control of manipulators with bounded joint velocities and torques. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1572–1577. IEEE, 1996.