Time-optimal Motion Planning for n-DOF Robot Manipulators using a Path-Parametric System Reformulation*

Robin Verschueren[1], Niels van Duijkeren[2], Jan Swevers[2], Moritz Diehl[1]

Abstract— Time-optimal motion planning for robotic manipulators consists of moving the robot along a path in Cartesian space as fast as possible. In contrast to time-optimal path following, small deviations from a predefined path are acceptable and can be exploited to further reduce the overall motion time. In this paper, we present a new method to compute time-optimal motions *around* a path. By employing an appropriate change of variables for the robot dynamics to path coordinates, geometric constraints enter the optimal control formulation in a convenient way. The reformulation of the robot dynamics and the path constraints is illustrated with numerical examples.

I. INTRODUCTION

Time-optimal motion is of significant importance for maximizing the productivity of robot systems. Frequently, the essence of a task for a robot is described by the path it should follow. Examples include control of CNC machines, welding robots, but also control of autonomous aerial, land, underwater vehicles. From previous work we learn that generating time-optimal motion starting from a given geometric path can be beneficial. The operator can choose a path that ensures collision avoidance and satisfies other geometric constraints in advance. Given a fixed path in the configuration space of the robot (which may require an inverse kinematics step), the remainder of the trajectory generation problem consists of the significantly simpler step of choosing the optimal timing *where* to be *when* on the given path [6], [16]. We will call this the *decoupled* approach to time-optimal motion generation. An example of this is [8], an online two-level method, where the secondary level modifies a nominal trajectory by performing a time scaling during motion. This idea is elaborated in [15] with a decoupling between lateral and longitudinal control. Other online approaches to path following go one step further and consider a geometric path solely as reference and generate an optimal feedback law that decides on both the timing and the distance to the reference using a nonlinear model predictive control (NMPC) approach [10], [14].

The geometric path in the workspace of the robot may not be completely strict and, in fact, limited deviations may be perfectly acceptable, for instance in tasks with a certain machining tolerance, or in orientation invariant tasks. The work in [9] presents a time-optimal approach which allows deviation from the reference path, by projecting the robot dynamics on the reference path and introducing constraints on the forward position kinematics. As in [9], we propose a trajectory generation methodology closely related to the decoupled approach. The main difference is that we do not project the robot dynamics onto a predefined path. Rather, we reformulate the dynamics *around* it; by doing so, we allow the end-effector of the robot to deviate from the path. This method relies on a so-called *spatial reformulation* (or *time transformation*) previously proposed in [12], [11] for planar motions. The concept of this reformulation is to introduce a variable s that measures the progress of the motion in the workspace of our system and to transform the system dynamics to evolve with this progress instead of time. This approach is appealing, since it allows all static geometric aspects in the optimization problem to appear explicitly in the horizon of the optimal control problem. The price to pay is a nonlinear transformation of the system dynamics and the loss of the ability to explicitly define changing geometric properties in time. Examples of implementations of the spatial reformulation include research on driver assistance functions for high way driving of long heavy vehicle combinations (LHVCs) [19], and time-optimal nonlinear model predictive control (NMPC) on small-scale race cars [20].

In this paper, we present a generalization of the time-transformation for paths in three-dimensional Euclidean space, applied to a reference path in the workspace of a serial-link robotic arm. The spatial reformulation enables us to express a natural formulation of an optimal control problem (OCP) for time and energy optimal motion, since travel time becomes a state variable of the system equations.

This paper is organized in the following way. First we introduce and derive the so-called spatial reformulation for motion of the end-effector of a robotic arm with respect to a reference curve in the Euclidean workspace of the robot. Secondly, an OCP is introduced that is solved with two equivalent formulations; the first approach uses the novel technique presented in this paper, the second takes the traditional approach of scaling the horizon and system dynamics linearly by an optimization variable. We illustrate the efficacy of the method to describe geometric constraints to facilitate e.g., collision avoidance. Thereafter we briefly elaborate on the implementation and the tools that were

[1]Systems Control and Optimization Laboratory, Department of Microsystems Engineering, University of Freiburg, Germany `robin.verschueren@imtek.uni-freiburg.de`
[2]Division of Production engineering, Machine design and Automation, Department of Mechanical Engineering, KU Leuven, Belgium `Niels.vanDuijkeren@kuleuven.be`

employed. The paper concludes with a discussion on the simulation results and a brief preview of future work.

In the remainder of this paper, we will make use of the notation $(\cdot)' = \dfrac{\mathrm{d}(\cdot)}{\mathrm{d}s}$ and $\dot{(\cdot)} = \dfrac{\mathrm{d}(\cdot)}{\mathrm{d}t}$.

II. SPATIAL REFORMULATION OF ROBOT DYNAMICS

To illustrate the spatial reformulation, we apply it to an optimal motion planning task for a robotic arm. Let us consider a rigid-body n-DOF serial-link robotic manipulator. Recall that the motion equations of this kind of systems can be written in the form, cf. [18]:

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \dot{q} \tag{1a}$$

$$M(q)\frac{\mathrm{d}\dot{q}}{\mathrm{d}t} = \tau - C(q,\dot{q})\dot{q} - G(q), \tag{1b}$$

where $q, \dot{q}, \tau \in \mathbb{R}^n$ are the joint angles, joint velocities and actuation torques in the joints, respectively. $M(q)$, $C(q,\dot{q})$ denote the mass matrix and a matrix accounting for Coriolis and centrifugal effects, $G(q)$ is a vector of torques due to gravitation. Note that Coulomb friction and viscous friction are neglected as they are not readily taken into account in applying the spatial reformulation.

In this section, we will first present the way in which we represent the path and establish a formula for the progress along the path. We then use this relation to apply the spatial reformulation of the dynamics.

A. Path Representation

Let $\gamma(t)$ be a continuous, sufficiently often differentiable curve in three-dimensional Euclidean space, assuming that the velocity vector $\dot{\gamma}(t) \neq 0$. We introduce the arc length $s(t)$ as the distance traveled along the path. The path $\Gamma = \{\gamma(s) \in \mathbb{R}^3 : s \in [0,l] \to \gamma(s)\}$ is parametrized by its arc length

$$s(t) = \int_0^t \|\dot{\gamma}(x)\|_2 \, \mathrm{d}x. \tag{2}$$

Local properties of the curve are characterized by the curvature κ and the torsion σ. At each point s on Γ we define an orthonormal basis frame of three vectors $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{B}$, referred to as the tangent, normal and binormal unit vectors. These unit vectors are defined by $\mathcal{T}(s) := \gamma'(s)$, $\mathcal{N}(s) := \mathcal{T}'(s)/\|\kappa(s)\|_2$ and $\mathcal{B}(s) := \mathcal{T}(s) \times \mathcal{N}(s)$ and satisfy the Frenet-Serret formulas, cf. [13]:

$$\mathcal{T}' = \kappa\mathcal{N}, \quad \mathcal{N}' = -\kappa\mathcal{T} + \sigma\mathcal{B}, \quad \mathcal{B}' = -\sigma\mathcal{N}. \tag{3}$$

Furthermore, let $p(t)$ be the vector of positional coordinates at fixed time t in the inertial world frame (forward position kinematics of the robotic manipulator), then the point on the path γ closest to $p(t)$ is $\gamma(s^\star)$, where

$$r(s,t) = p(t) - \gamma(s) \tag{4}$$

$$s^\star = \arg\min_s \frac{1}{2}\|r(s,t)\|_2^2. \tag{5}$$
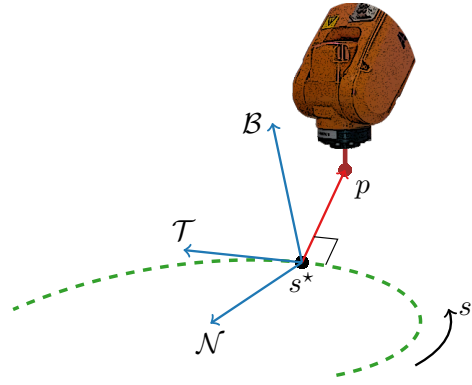
See Fig. 1 for an illustration of the concept.

Fig. 1: Illustration of the position of the end-effector with respect to the closest position of the path.

As is clear from (4)-(5), finding $s^\star(t)$ involves an optimization problem. Since it is undesired to embed this into a higher-level optimization problem, we attempt to find the temporal evolution of $s^\star(t)$ by looking at the optimality conditions. Recall that for unconstrained optimization, the first order necessary condition for optimality is:

$$0 = \frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{1}{2}\|r(s,t)\|_2^2\right) \tag{6a}$$

$$= r(s,t)^T\gamma'(s). \tag{6b}$$

Consider that the position $s^\star$ is known at an initial time-point, we can enforce the solution to be optimal in time by setting the time derivative of the necessary first order optimality condition (6) to zero, i.e.,

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(r(s,t)^T\gamma'(s)\right) \tag{7a}$$

$$= (v(t) - \gamma'(s)\dot{s}(t))^T\gamma'(s) + r(s,t)^T\gamma''(s)\dot{s}(t), \tag{7b}$$

where $v(t) = \frac{\mathrm{d}p(t)}{\mathrm{d}t}$. This ultimately gives us a closed formula for the velocity of the point on the path closest to $p(t)$,

$$\dot{s}(t) = \frac{v(t)^T\mathcal{T}(s)}{1 - \kappa(s)r(s,t)^T\mathcal{N}(s)}. \tag{8}$$

B. Spatial reformulation

We augment the state vector with $r = [r_x, r_y, r_z]^T$. Additionally, the state $t(s)$ is included to keep track of the evolution of time. The state vector then reads $\xi = \left[q^T, \dot{q}^T, r^T, t\right]^T \in \mathbb{R}^l$. Using the established representation for the dynamics of the position of the end-effector $p(t)$ with respect to the path, we perform a spatial transformation of the equations of motion:

$$\xi' := \frac{\mathrm{d}\xi}{\mathrm{d}s} = \frac{\mathrm{d}\xi}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}s}, \tag{9}$$

with the state vector ξ. For $\dot{s}(t) \neq 0$, we have that $\frac{\mathrm{d}t}{\mathrm{d}s} = \frac{1}{\dot{s}(t)}$, and therefore

$$\xi' = \frac{1}{\dot{s}(t)}\dot{\xi}. \tag{10}$$

The resulting equations of motion are:

$$\frac{\mathrm{d}q}{\mathrm{d}s} = \frac{\dot{q}}{\dot{s}} \tag{11a}$$

$$M(q)\frac{\mathrm{d}\dot{q}}{\mathrm{d}s} = \frac{\tau - C(q,\dot{q})\dot{q} - G(q)}{\dot{s}} \tag{11b}$$

$$\frac{\mathrm{d}r}{\mathrm{d}s} = \frac{\dot{p}}{\dot{s}} - T(s) \tag{11c}$$

$$\frac{\mathrm{d}t}{\mathrm{d}s} = \frac{1}{\dot{s}}, \tag{11d}$$

$$\tag{11e}$$

where the velocity of the end-effector can be written as $\dot{p} = J(q)\dot{q}$, with $J(q)$ the robot Jacobian, and $\dot{s}$ is obtained from (8).

The time transformation applied above is nonlinear, but is nevertheless appealing for two reasons. First, with the newly obtained state variable $t(s)$ and the according first-order differential equation $\frac{\mathrm{d}t}{\mathrm{d}s}$, time-optimal motion is equivalent to minimizing t over the motion along the path. Secondly, the required knowledge about the temporal evolution of the $\mathcal{T}$, $\mathcal{N}$ and $\mathcal{B}$ vectors describing the local Frenet-Serret frame and many other geometric properties (such as obstacles) at time t become explicitly available in the integration scheme for ξ'.

III. OPTIMAL CONTROL PROBLEM FORMULATION

In order to illustrate the benefits of the spatial reformulation of the robot dynamics, we formulate a time-optmimal trajectory generation problem. The OCP we intend to solve is stated as

$$\underset{\substack{\xi(\cdot)\in\mathbb{R}^l,\\\tau(\cdot)\in\mathbb{R}^m}}{\text{minimize}} \quad t_{\mathrm{f}} = \int_{t=0}^{t_{\mathrm{f}}} \mathrm{d}t \tag{12a}$$

$$\text{subject to} \quad \dot{\xi}(t) = f(\xi(t),\tau(t)) \quad \forall t \in [0,t_{\mathrm{f}}] \tag{12b}$$

$$g(p(t)) \leq 0 \quad \forall t \in [0,t_{\mathrm{f}}] \tag{12c}$$

$$\underline{\tau(t)} \leq \tau(t) \leq \overline{\tau(t)} \quad \forall t \in [0,t_{\mathrm{f}}] \tag{12d}$$

with system dynamics, path constraints and torque bounds as constraints, respectively. This formulation is not readily passed to an optimization routine, as the time interval $[0,t_{\mathrm{f}}]$ on which we solve the OCP is not independent of the optimization variables. Therefore, we proceed to pose two reformulations of the above OCP, which we will compare in the subsequent sections. The first is a time-optimal formulation using a rescaling of the time variable, with geometric constraints as nonlinear constraints. The second one makes use of our proposed spatial reformulation.

For both formulations, $\xi = [q^T, \dot{q}^T, r^T, t]^T$ is the state vector and τ are the controls.

A. Time-scaled time-optimal control problem

We introduce $\theta = t/t_{\mathrm{f}}$ as a scaling of the time variable. Furthermore, for the formulation in time, we do not make use of a predefined path, so we take $\gamma(s) = 0$; this results in $r = p$. In this way, notation is consistent with the spatial

formulation. Using the linear time scaling $\theta = t/t_{\mathrm{f}}$, the OCP in (12) becomes

$$\underset{\substack{\xi(\cdot)\in\mathbb{R}^l,\\\tau(\cdot)\in\mathbb{R}^m}}{\text{minimize}} \quad t_{\mathrm{f}} = \int_{\theta=0}^{1} t_{\mathrm{f}}\,\mathrm{d}\theta \tag{13a}$$

$$\text{subject to} \quad \frac{\mathrm{d}\xi(\theta)}{\mathrm{d}\theta} = t_{\mathrm{f}} \cdot f(\xi(\theta),\tau(\theta)) \quad \forall \theta \in [0,1] \tag{13b}$$

$$g_\theta(r(\theta)) \leq 0 \quad \forall \theta \in [0,1] \tag{13c}$$

$$\underline{\tau(\theta)} \leq \tau(\theta) \leq \overline{\tau(\theta)} \quad \forall \theta \in [0,1] \tag{13d}$$

Using the above linear rescaling, we are indeed capable of making the integration interval $[0,1]$ independent of the decision variables ξ,τ.

B. Spatial time-optimal control problem

From (10), we get the spatial dynamics as $f(\xi(s),\tau(s))/\dot{s}$. The optimal control formulation then reads as

$$\underset{\substack{\xi(\cdot)\in\mathbb{R}^l,\\\tau(\cdot)\in\mathbb{R}^m}}{\text{minimize}} \quad t_{\mathrm{f}} = \int_{s=0}^{s_{\mathrm{f}}} \frac{\mathrm{d}t}{\mathrm{d}s}\,\mathrm{d}s \tag{14a}$$

$$\text{subject to} \quad \xi'(s) = \frac{f(\xi(s),\tau(s))}{\dot{s}} \quad \forall s \in [0,s_{\mathrm{f}}] \tag{14b}$$

$$g_s(r(s)) \leq 0 \quad \forall s \in [0,s_{\mathrm{f}}] \tag{14c}$$

$$\underline{\tau(s)} \leq \tau(s) \leq \overline{\tau(s)} \quad \forall s \in [0,s_{\mathrm{f}}] \tag{14d}$$

where we obtain $\dot{s}$ from (8).

Note that both OCPs (13) and (14) are equivalent to (12), as the objective function and the constraints remain equivalent after transformation of variables.

IV. NUMERICAL ALGORITHMS

To numerically solve the optimal control problems of the previous section, we adopt the open-source CasADi [3] software framework, which has been proven to solve OCPs reliably and efficiently [5]. More specifically, we use the Python front-end to formulate the OCP as a nonlinear program (NLP) using Bock's multiple shooting method [7] as a discretization method. To this end, we relied on the integrators `cvodes` and `idas` inside the sundials suite [1].

The resulting NLP is passed to the open-source solver IPOPT [21]. It implements a primal-dual interior point method suited for solving large-scale NLPs. The linear algebra subroutine calls were passed to the sparse solver `ma57` from the HSL library [2]. Note that all of the software packages mentioned above can be conveniently called from within the CasADi framework.

For the second part of the simulations, we use a 6-DOF robot model, which is constructed with the Python-based toolbox SympyBotics [17].

V. SIMULATION RESULTS

We show the performance of our time-optimal OCP with spatial reformulation of the dynamics on a simple three link robot manipulator of which the Denavit-Hartenberg parameters are shown in Table I. We consider two different motion experiments: the first is a time-optimal point-to-point

TABLE I: Denavit-Hartenberg parameters for the three-link robot used in simulation

	$d\,[\mathrm{m}]$	$\theta\,[\mathrm{rad}]$	$a\,[\mathrm{m}]$	$\alpha\,[\mathrm{rad}]$
Link 1	0.1	0	0	$\pi/2$
Link 2	0	$\pi/2$	1	0
Link 3	0	$-\pi/2$	0.7	0

motion, where we compare the results of both the spatial reformulation and the linear time scaling. The second is a problem where no end effector position is specified, and we introduce a static obstacle.

Note that in the following, a small control regularization of the form $r_\tau \sum_i \tau_i^2$, with $r_\tau = 10^{-4}$ is added to the objective function in order to obtain smooth control trajectories.

A. Time-optimal point-to-point motion

To illustrate the equivalence of both methods discussed in Section III, we solve the OCPs (13)-(14) for a simple time-optimal motion planning task. The robot end-effector travels from a starting point p_0 to a fixed final position p_f, staying inside a certain space, defined as the space between two cilinders, see Fig. 2.

This constraint might be posed for the time-dependent formulation as

$$\underline{R} \le \|[r_x(\theta), r_y(\theta)]^T\|_2 \le \overline{R}, \quad \forall \theta \in [0,1], \tag{15a}$$
$$\underline{Z} \le r_z(\theta) \le \overline{Z}, \qquad \forall \theta \in [0,1]. \tag{15b}$$

Recall that, for the time-domain formulation, r denotes the vector from the origin to the end-effector. For the spatial formulation the constraints read as

$$\underline{R_s} \le r^T \mathcal{N}(s) \le \overline{R_s}, \quad \forall s \in [0, \pi/2], \tag{16a}$$
$$\underline{Z} \le r_z(\theta) \le \overline{Z}, \qquad \forall s \in [0, \pi/2], \tag{16b}$$

with

$$\mathcal{T}(s) = [-\sin(s), \cos(s), 0]^T,$$
$$\mathcal{N}(s) = [-\cos(s), -\sin(s), 0]^T,$$
$$\mathcal{B}(s) = [0, 0, 1]^T,$$
$$\kappa = 1.0\,\mathrm{m}^{-1},$$
$$\sigma = 0.0\,\mathrm{m}^{-1}.$$

The graph in Fig. 2 corresponds with the values $\underline{R} = 0.8\,\mathrm{m}$, $\overline{R} = 1.2\,\mathrm{m}$, $\underline{R_s} = 0.2\,\mathrm{m}$, $\overline{R_s} = 0.2\,\mathrm{m}$, $\underline{Z} = -0.1\,\mathrm{m}$, $\overline{Z} = 0.1\,\mathrm{m}$.

One advantage of the spatial reformulation follows from the comparison of (16a) and (15a): in the latter case, the path constraint is convex, in the first it is not. If convexity holds for the constraint functions, convergence of the NLP solver is often accelerated.

We compare the solution of the two different formulations in Fig. 3. The solution trajectories of both methods are shown as a function of time. The torque trajectories clearly show that the methods result in different discretizations. The spatial formulation yields a solution that takes longer time-steps in the beginning of the interval than the time-based formulation,
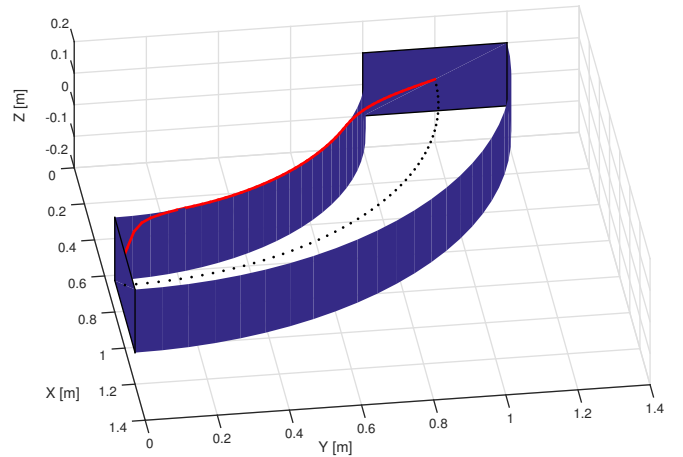

Fig. 2: Time-optimal solution (red) of optimal control problems (13) and (14) with cylindrical path constraints. The constraint space is delimited with dark blue faces, the centerline is shown in black.

which takes equidistant steps. It is clear from the figure that the resulting trajectories for the lateral and vertical deviation from the centerline of the cylindrical path are the same, up to the different discretizations. For a finer discretization grid, these differences vanish.

Another interesting remark is that the constraint on the first joint torque is active throughout the entire interval, as we would expect in a time-optimal motion. In addition, the path constraints are seen to be active most of the times as well, except to meet the initial and end point constraints. To conclude the comparison, we examine the NLP solver convergence. In this particular example, the time-optimal OCP was solved faster with the spatial formulation than the time formulation: the interior point solver needed 36 iterations (0.094 s) instead of 57 (0.191 s). More experiments need to be carried out to confirm this as a general rule.

B. Time-optimal obstacle avoidance

A major advantage of employing the time transformation instead of the conventional formulation, is the natural way to insert geometric constraints. Also, this method is suited for arbitrary paths in space, not just analytical ones as in the above example. Both properties are made apparent in the following simulation result, where we consider a 6-DOF ABB IRB120 industrial robot (Fig. 4) [4]. The simulation experiments are based on a dynamic model of this robot that was made available by ABB, albeit without taking friction into account.

In the following example, we perform a time-optimal motion planning task. The robot has to follow a given path within given tolerances as fast as possible, avoiding a static cylindrical obstacle at the end of the path. Additionally, we constrain the end-effector to point vertically down by imposing additional constraints on the forward orientation kinematics. The path specified (cf. dotted line in Fig. 5a) is a planar path with piecewise constant curvature and given initial tangent and normal vectors, both normalized. The
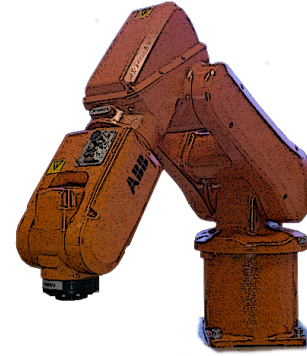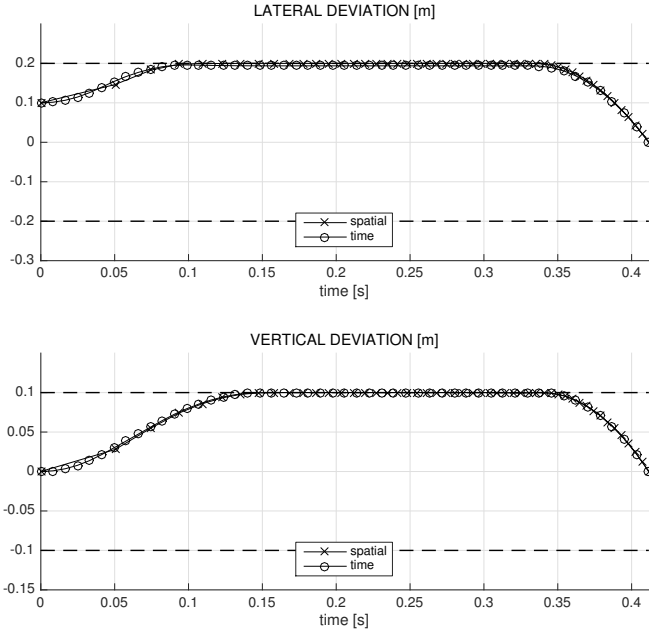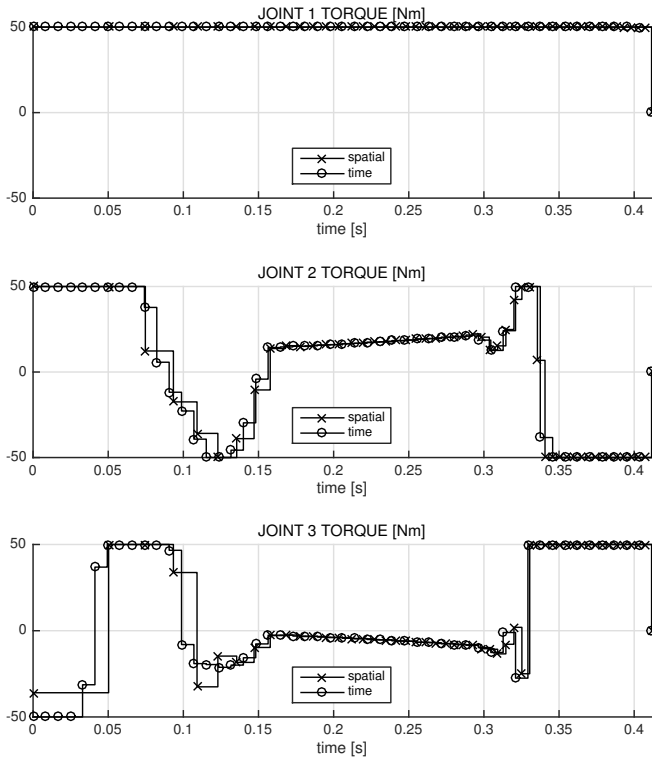
(a) The lateral and vertical deviation are taken with respect to the centerline between the two cilinders.

(b) The torques in the robot joints.

Fig. 3: Comparison of the solution of the two OCP formulations (13) and (14). The joint torque bounds are taken to be $-50\,\mathrm{Nm} \le \tau \le 50\,\mathrm{Nm}$ for both formulations.

Fig. 4: Sketch of the ABB IRB120 industrial robot. During the simulation, the end-effector points down (as shown).

binormal, which is vertical at all points, is taken arbitrarily as $\mathcal{B}_0 = [0, 0, 1]$. The cartesian coordinates of the path can be retrieved by integrating the Frenet-Serret formulas (3) forward in the path coordinate s, starting at $s_0 = 0\,\mathrm{m}$. Note that this integration can be incorporated in the state equations (which are already in function of s) by augmenting the dynamics with (3).

At the end of the prespecified path, there is a static cylindrical obstacle narrowing down the maneuvering possibilities of the end-effector. In the spatial formulation, this constraint will amount to increasing the lower bound of the constraint (16a); it remains a simple bound. The resulting time-optimal path can be seen in Fig. 5a. Again, we see that the path touches the inner path constraint as we expect for time-optimal trajectories. Note that in this case, the final position of the end effector is not completely specified, only constrained to lie in the plane spanned by $\mathcal{N}$ and $\mathcal{B}$. The vertical deviation is depicted in Fig. 5b. Also here the constraints on the vertical deviation become active at some point in the motion.

Note that the time-domain formulation (13) is not readily applicable in this example, because the path constraints are not straightforward to compute in the time domain for arbitrary paths, in contrast with paths with an analytical description, as in the first example. This shortcoming originates from the fact that we do not know on beforehand where in the configuration space the end effector will be at which point in time, i.e. the relation between t and s is not stated explicitly. An expression of the path constraints in the time domain results in possibly very nonlinear constraints; this holds even more so for static obstacles as in the above example.

VI. CONCLUSION

In this paper, we introduced a path-parametric system reformulation for robotic manipulators. It is aimed at convenience for path specification, and exhibits a natural way to impose geometric constraints. These advantages have been shown in relation to time-optimal control using a time-scaling approach, which does not posess these properties.

The work presented here forms the groundwork for future developments. A first follow-up of the results showed is to assess the novel method on an experimental testbed. Joint
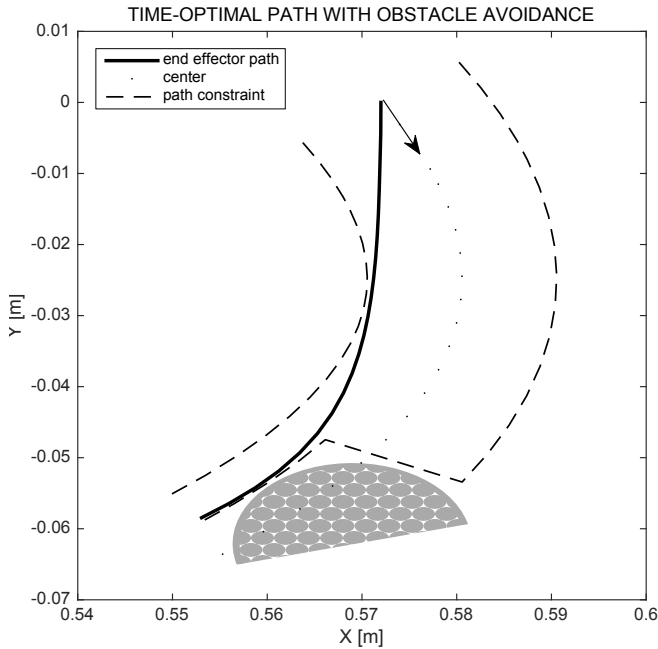
(a) Top view of the time-optimal path. Near the end of the path, there is a static obstacle narrowing the available space of the robot end-effector.
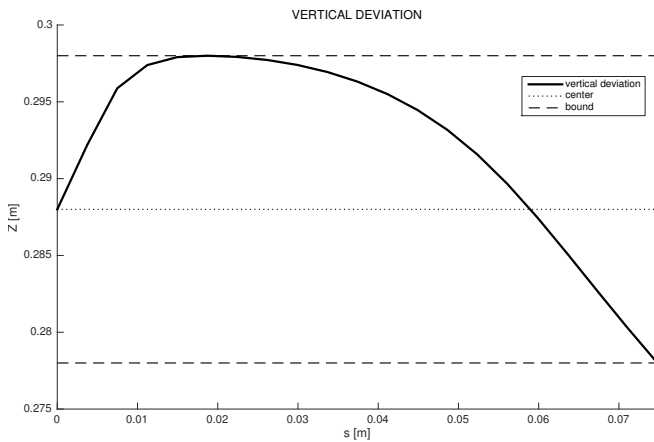
(b) Vertical deviation from the $z = 0.288\,\mathrm{m}$ horizontal plane.

Fig. 5: Time-optimal optimal control problem with obstacle avoidance around an arbitrary central path with piecewise constant curvature.

and control trajectories can be computed offline, in a similar fashion as to compute the simulation results showed. To this end, an effective way of taking into account Coulomb and viscous friction is needed.

Another possible advancement lies in the application of the presented techniques in a model predictive framework, both in simulation and on a real-life robot setup. More specifically, the application of this method in real-time seems a computational challenge, although the first steps in this direction have been taken. Lastly, a setup running entirely online, where path information is not specified in advance but, for instance, supplied by human interaction via a manual control device could exploit all the computational advantages proposed above to a greater extent.

REFERENCES

[1] Sundials. SUite of Nonlinear and DIfferential/ALgebraic equation Solvers. https://computation.llnl.gov/casc/sundials, 2009.

[2] Hsl. A collection of fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk, 2011.

[3] CasADi. http://casadi.org, 2013.

[4] ABB. IRB120 technical data. http://new.abb.com/products/robotics, 2015. [Online; accessed 22 September 2015].

[5] J. Andersson, J. Åkesson, and M. Diehl. CasADi – a symbolic package for automatic differentiation and optimal control. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, Lecture Notes in Computational Science and Engineering, pages 297–307, Berlin, 2012. Springer.

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

[7] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the IFAC World Congress*, pages 242–247. Pergamon Press, 1984.

[8] O. Dahl and L. Nielsen. Torque-limited path following by online trajectory time scaling. *IEEE Transactions on Robotics and Automation*, 1990.

[9] F. Debrouwere, W. Van Loock, G. Pipeleers, and J. Swevers. Time-optimal tube following for robotic manipulators. In *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*, 2014.

[10] T. Faulwasser, B. Kern, and R. Findeisen. Model predictive path-following for constrained nonlinear systems. In *Joint 48th IEEE conference on Decision and Control and 28th Chinese Control Conference*, 2009.

[11] J. V. Frasch, A. J. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl. An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. In *Proceedings of the European Control Conference (ECC)*, pages 4136–4141, 2013.

[12] Y. Gao, A. Gray, J. V. Frasch, T. Lin, H. E. Tseng, J. Hedrick, and F. Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th International Symposium on Advanced Vehicle Control*, 2012.

[13] H. Guggenheimer. *Differential Geometry*. Dover, 1977.

[14] D. Lam, C. Manzie, and M. Good. Model Predictive Contouring Control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2010.

[15] B. Olofsson. *Topics in Machining with Industrial Robot Manipulators and Optimal Motion Control*. PhD thesis, Lund University, Department of Automatic Control, 2015.

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

[17] C. D. Sousa. Sympybotics v1.0, Aug. 2014.

[18] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control*. Wiley, 2006.

[19] N. van Duijkeren, T. Keviczky, P. Nilsson, and L. Laine. Real-time nmpc for semi-automated highway driving of long heavy vehicle combinations. In *Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control 2015 (NMPC'15)*, 2015.

[20] R. Verschueren, S. D. Bruyne, M. Zanon, J. V. Frasch, and M. Diehl. Towards time-optimal race car driving using nonlinear MPC in real-time. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 2505–2510, 2014.

[21] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.