2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Towards Online Trajectory Generation Considering Robot Dynamics and Torque Limits

Robert Katzschmann, Torsten Kröger, Tamim Asfour, and Oussama Khatib

***Abstract*— Generating robot motion trajectories instantaneously in the moment unforeseen sensor events happen is very essential for many real-world robot applications. Using a previous work on online trajectory generation as a basis, this paper proposes an alternative approach that also considers dynamic models. The former class of algorithms does not take into account dynamically changing acceleration capabilities based on maximum actuator forces/torques. This paper extends target velocity-based algorithms of the previous approach by taking into consideration the entire system dynamics when generating trajectories online within one control cycle (typically 1 ms or less). The extension includes the acceleration capabilities of a robot at every discrete time step assuming constant values for the maximum actuator forces/torques, thus allowing the generation of adaptive trajectory profiles during the motion of the robot. Several real-world experimental results using a seven-degree-of-freedom lightweight robot arm underline the relevance of this extension.**

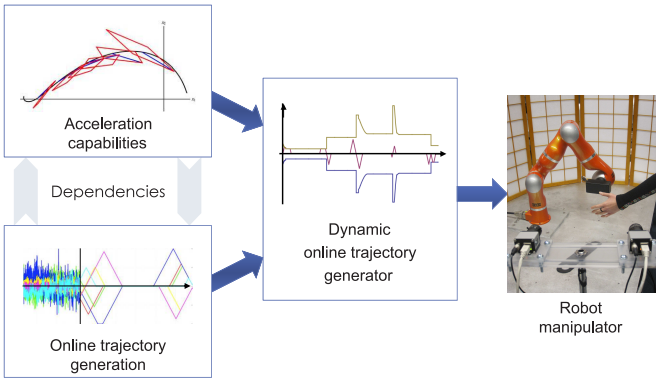

Fig. 1: Overview: online trajectory generation algorithms [5]–[7] are combined with offline methods [8]–[10] to take into account time-varying acceleration capabilities.

I. Introduction

Integrating sensors in robot motion control systems and using sensor signals at several levels — for perception, for high-level task and motion planning, for discrete or continuous task transitions, and for low-level trajectory generation and feedback control — is essential for all domains of robotics. A major obstacle that prevents robots from accomplishing real-world tasks is their inability to physically interact with, and effectively manipulate, the most common objects generally found in human environments. Robots generally employ precision to perform a manipulation task. Humans, in contrast, employ compliance through tactile and force feedback to overcome their imprecision, allowing them to resolve uncertainties associated with the task. The lack of compliance and force control has been indeed a major limiting factor in the ability of robots to interact and manipulate in human environments.

Considering robots with compliant motion control capabilities (e.g., [1], [2]), different — task-dependent — reaction behaviors [3] are desired in the moment a collision or prospective collision is detected. In the moment a potential collision is detected, high control gains are desirable in order to let the robot move away from the objects that it is about to collide with [4]. To achieve the best immediate reaction behaviors for avoiding collisions, controllers have to be switched and trajectories have to be generated instantaneously in order to allow smooth and continuous motions during discrete switching events.

In this paper, we consider a robot with multiple degrees of freedom (DOFs) with constant maximum joint forces/torques. The kinematic and dynamic model of the system is known, and we take the concepts of [5]–[7] as a basis, so that trajectories can be generated online from arbitrary states of motion. While the existing algorithms [5]–[7] assume constant acceleration values over entire trajectories, we propose a new concept that

1) **generates robot motion trajectories from arbitrary initial states of motion within one low-level control cycle (≈ 1 ms) and**
2) **takes into account jerk-limits and dynamics not only locally but globally over the entire trajectory.**

The acceleration capabilities based on the dynamics have dependent boundaries between each single axis, while the acceleration constraints used for the online trajectory generation have to be linearly independent. The new concept takes this into account and converts the dependent capabilities into linearly independent constraints. Trajectories that will bring at least one of the actuators into force/torque saturation are computed instantaneously in the moment unforeseen sensor signals or events happen. As a result, robots can employ their dynamic capabilities immediately and react instantaneously (cf. Fig. 1). This method can be applied to purely position-controlled mechanisms and to variable-stiffness systems.

II. Related Work

In [11]–[14], overviews on robot motion planning algorithms are given. Most related to this work are the online

R. Katzschmann, T. Kröger, O. Khatib are with the Artificial Intelligence Laboratory at Stanford University, Stanford, CA 94305-9010, USA, {katzschmann, tkr, ok}@cs.stanford.edu.

T. Asfour is with the High Performance Humanoid Technologies Lab, Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Adenauerring 2, D-76131 Karlsruhe, Germany, asfour@kit.edu.

trajectory generation (OTG) concepts of Broquère *et al.* [5], Haschke *et al.* [6], and Kröger [7]. All three approaches allow generating robot motion trajectories from arbitrary initial states of motion, however, they are not capable of using dynamic models for their planning procedures, so that varying acceleration capabilities are not taken into account.

Based on the approach of Hollerbach [15], Bobrow *et al.* [8], Shin *et al.* [10], and Pfeiffer *et al.* [9] independently developed a technique for time-optimal trajectory planning for arbitrarily specified paths.

Here, robot dynamics are described in dependence on a parametric path representation, so that a maximum acceleration is calculated for each point of the trajectory.

Dahl *et al.* [16] suggested an online trajectory adaptation method to improve the path accuracy of pre-planned robot motions. Using the basic algorithm of Bobrow/Pfeiffer/Shin [8]–[10], their method adapts the acceleration along the path, and furthermore the parameters of the trajectory-following controller are adapted online, so that underlying trajectory-following controllers become adapted depending on the current motion state.

An offline numerical approach has been shown in Wu [17], which assumes a given path and also limits the jerk. The method does not allow start and target velocities and accelerations to be unequal to zero.

The real-time adaptive motion planning framework by Vannoy *et al.* [18] and elastic strips framework by Brock *et al.* [19] are key to implement reactive motion control behaviors in robot control systems. These works use splines to represent calculated trajectories; it is the task to calculate respective sets of spline knots and trajectory parameters during runtime.

Instead of generating motion trajectories, a concept proposed by Haddadin *et al.* [20] uses virtual springs and damping elements as input values for a Cartesian impedance controller. Other recent online trajectory generation approaches were published by Guarino Lo Bianco *et al.* [21]. Based on the work of De Luca *et al.* [4], a time-scaling method is proposed that takes into account dynamic models of robot systems. Formulated as a control problem, the works of Khansari-Zadeh *et al.* [22] illustrates a new trajectory generation method allowing immediate reactions to unforeseen sensor signals. Zanchettin *et al.* [23] show a very similar concept and applies it to human-robot interaction scenarios.

III. Online Trajectory Generation (OTG)

This section briefly summarizes the functionality and nomenclature used for the target velocity-based OTG, short VOTG. For more details about this algorithm, please refer to [7]. The new target velocity-based dynamic OTG, short VDOTG, will be described based on this.

Figure 2 illustrates the input and output values: At a discrete instant t_i, the algorithm receives the command to calculate a synchronized motion trajectory, which transfers the motion state from $\boldsymbol{\Xi}_i$ to $\boldsymbol{\Xi}_{i,trgt}$ within the shortest possible time while not exceeding the given motion constraints $\boldsymbol{B}_i$.

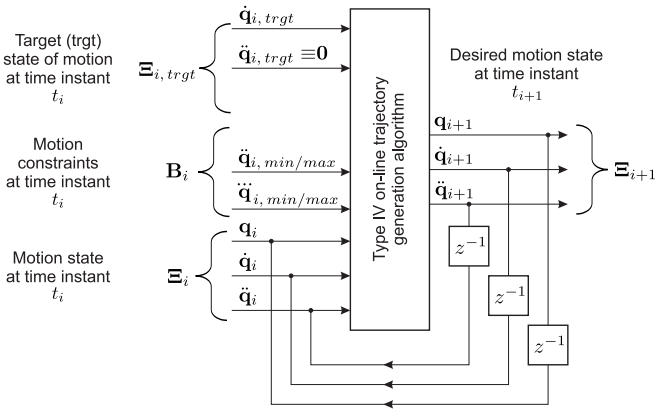

Fig. 2: Input and output values of the target velocity-based Type IV OTG algorithm (cf. [7]).

We assume a time-discrete system with $t_i = t_{i-1} + t_{cycle}$ and $i \in \{1, \ldots, N\}$ where t_{cycle} represents the cycle time. Time-discrete values have an additional subscript i compared to time-continuous values. The position of a robotic system with K DOFs at time t_i is $\boldsymbol{q}_i = ({}_1q_i, \ldots, {}_kq_i, \ldots, {}_Kq_i)^T$, the time derivations are indicated by $\dot{\boldsymbol{q}}_i$ and $\ddot{\boldsymbol{q}}_i$, respectively. A complete motion state at time t_i is described by the matrix $\boldsymbol{\Xi}_i = (\boldsymbol{q}_i, \dot{\boldsymbol{q}}_i, \ddot{\boldsymbol{q}}_i) = ({}_1\boldsymbol{\xi}_i, ..., {}_k\boldsymbol{\xi}_i, ..., {}_K\boldsymbol{\xi}_i)^T$. ${}_k\boldsymbol{\xi}_i$ stands for the k-th row of the motion state matrix $\boldsymbol{\Xi}_i$. The desired target motion state at time instant t_i is contained in $\boldsymbol{\Xi}_{i,trgt} = (\dot{\boldsymbol{q}}_{i,trgt}, \ddot{\boldsymbol{q}}_{i,trgt} \equiv 0)$. A target position cannot be defined, but is determined by the algorithm. The target acceleration has to be zero.

The kinematic motion constraints at a time t_i are denoted as $\boldsymbol{B}_i = \left(\{\ddot{\boldsymbol{q}}_{i,min}, \ddot{\boldsymbol{q}}_{i,max}\}, \{\dddot{\boldsymbol{q}}_{i,min}, \dddot{\boldsymbol{q}}_{i,max}\}\right)$ and constrain only the acceleration and the jerk of the motion state $\boldsymbol{\Xi}_i$ $\forall\, i \in \{1, \ldots, N\}$. The velocities are not constrained by this algorithm. All variables combined are the input parameters of the OTG algorithm

$$\boldsymbol{W}_i = (\boldsymbol{\Xi}_i, \boldsymbol{\Xi}_{i,trgt}, \boldsymbol{B}_i)\ . \tag{1}$$

The VOTG algorithm computes $\boldsymbol{\Xi}_{i+1}$, which is the motion state on the *kinematically time-optimal* trajectory that will be used as a command value for underlying controllers at t_{i+1}. z^{-1} indicates, that $\boldsymbol{\Xi}_{i+1}$ is fed back as the new $\boldsymbol{\Xi}_i$ of the next cycle. All DOFs will reach the desired velocities $\boldsymbol{\Xi}_{i,trgt}$ simultaneously.

The algorithm assumes that the kinematic motion constraints $\boldsymbol{B}_i$ are constant at every instant t_i. This way, the dynamic capabilities of a robot system cannot be deployed, and conservative values for $\boldsymbol{B}_i$ have to be chosen to compute executable trajectories. In order to fully deploy the time-varying maximum acceleration capabilities, the OTG algorithm has to take into account the dynamic model. Our goal in this paper is generating *dynamically time-optimal* trajectories that are provided within one control cycle, that is, within t_{cycle}.

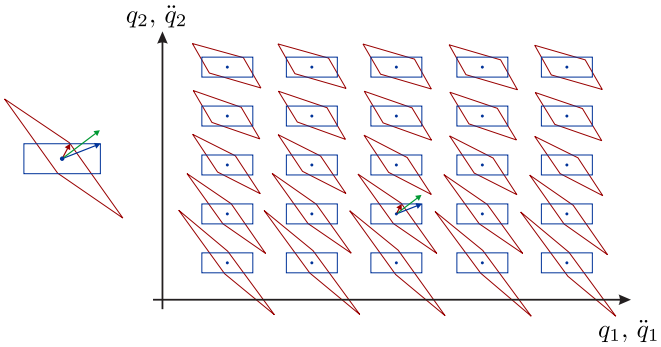

Fig. 3: Maximum torques as blue rectangles are mapped to the maximum accelerations as red parallelograms for a whole range of joint space configurations. For one configuration the torque combination $({}_1\tau_{max},\ {}_2\tau_{max})^T$ is shown as a blue arrow pointing to one corner of the blue rectangle. The blue arrow maps to the red arrow, which is the equivalent acceleration corner of the parallelogram. The green arrow indicates the direction of the current joint velocity.

IV. Dynamics and Problem Description

The standard form of the equations of motion in joint space is given in Eqn. (2.37) of [24]:

$$\boldsymbol{M}(\boldsymbol{q})\,\ddot{\boldsymbol{q}} \;+\; \boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} \;+\; \boldsymbol{\tau}_g(\boldsymbol{q}) \;=\; \boldsymbol{\tau} \tag{2}$$

the vector $\boldsymbol{q}$ contains the joint positions, $\boldsymbol{M}(\boldsymbol{q})$ is the mass matrix, $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})$ are the Coriolis/centrifugal torques, $\boldsymbol{\tau}_g(\boldsymbol{q})$ are the gravity torques and $\boldsymbol{\tau}$ are the actuator torque. Using Eqn. (2) leads to the inverse dynamic model

$$\ddot{\boldsymbol{q}}_{extr} = \boldsymbol{M}(\boldsymbol{q})^{-1}\,(\boldsymbol{\tau}_{extr} - \;\boldsymbol{\tau}_g(\boldsymbol{q}) - c(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}}) \tag{3}$$

where the extreme actuator torques $\boldsymbol{\tau} = \boldsymbol{\tau}_{extr}$ are mapped to the extreme joint accelerations $\ddot{\boldsymbol{q}} = \ddot{\boldsymbol{q}}_{extr}$. The corner values $\ddot{\boldsymbol{q}}_{extr}$ can be found through combination of the extreme torque values ${}_k\tau_{extr}$ of each individual actuator:

$$\forall\ k\ \in\ \{1,\ldots,K\}:\quad {}_k\tau_{extr} \in \{{}_k\tau_{min},\ {}_k\tau_{max}\} \tag{4}$$

$$\text{with}\qquad {}_k\tau_{min} \le {}_k\tau \le {}_k\tau_{max} \tag{5}$$

We assume that elements of $\boldsymbol{\tau}_{min}$ and $\boldsymbol{\tau}_{max}$ are constant and do not depend on $\dot{\boldsymbol{q}}$. In order to illustrate the joint acceleration capabilities, Fig. 3 considers a simple two-DOF planar robot with two revolute joints.

A. Acceleration Parallelotopes

The mass matrix maps actuator forces to accelerations, while the Coriolis/centrifugal and the gravity torques cause an offset. Figure 4 visualizes this: For 1D, a *line segment* is mapped to another one. for 2D, the torque rectangles are mapped to a *parallelogram* with a shifted origin compared to the torque-box origin. For 3D, the shape of the acceleration capabilities is called *parallelepiped*. For the general case of n-dimensions, the shape is called *parallelotope (PT)*.

If the origin of the acceleration coordinate system is inside the PT, the manipulator can theoretically accelerate in all directions with values anywhere between zero and the extremal values described by the PT bounds. Any chosen acceleration direction can also be extended to its negative direction. If

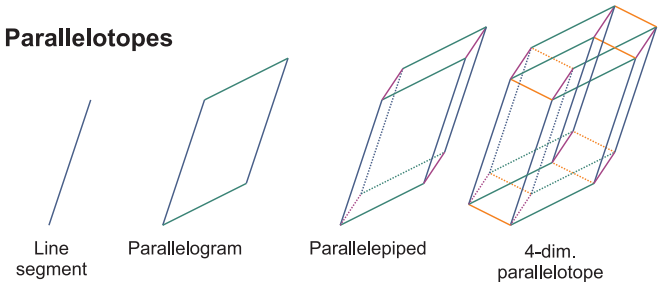

Fig. 4: Visualization of parallelotopes.

the sum of Coriolis/centrifugal and gravity torques surpasses the available actuator torques, the origin moves outside the PT. In this case, the robot can only accelerate in one of the directions lying inside a truncated cone. The minimal and maximal acceleration values of a direction vector inside the cone have the same sign. The apex point of the cone is the origin of the acceleration frame, the cap point is the first intersection of the central ray with the PT, the base point is the second intersection with the PT. In the further discussions we assume, that actuator torques are always high enough, so that the origin of the acceleration frame is always inside the PT.

V. Target Velocity-based Dynamic OTG (VDOTG)

A. One-Dimensional Path Representation

The VOTG [7] provides an initial motion trajectory, which is used when simplifying the analysis of the acceleration capabilities of a multi-DOF robot to a one-dimensional path problem. According to [8], a multidimensional joint position trajectory $\boldsymbol{q}(t)$ can also be described in dependency to the one-dimensional path variable $s(t)$:

$$\boldsymbol{q}(t) := \boldsymbol{r}(s(t)) \tag{6}$$

The path describes the arc length of a continuous position progression and is defined by:

$$s(t) = \int_{t_0}^{t} ||\dot{\boldsymbol{q}}(t)||\ dt \tag{7}$$

The VOTG computes a second-order, piecewise defined velocity progression $\dot{\boldsymbol{q}}(t)$, which does not allow to solve the integral in Eqn. (7) in an analytical way. Only the derivations of $s(t)$ are needed, as will be shown later. The dependency to time is dropped in the following to improve legibility. Differentiating Eqn. 6 with respect to time yields

$$\dot{\boldsymbol{q}} = \frac{\partial \boldsymbol{r}(s)}{\partial s}\ \dot{s} = \boldsymbol{r}_s(s)\ \dot{s} \tag{8}$$

$\boldsymbol{r}_s(s)$ describes the first partial derivation of $\boldsymbol{r}(s)$ by s. The second derivation leads to

$$\begin{aligned}\ddot{\boldsymbol{q}} \;&=\; \boldsymbol{r}_s(s)\ \ddot{s} + \frac{\partial \boldsymbol{r}_s(s)}{\partial s}\ \frac{\partial s}{\partial t}\ \dot{s} \\ &=\; \underbrace{\boldsymbol{r}_s(s)\ \ddot{s}}_{tang.acc.} + \underbrace{\boldsymbol{r}_{ss}(s)\ \dot{s}^2}_{norm.acc.}\end{aligned} \tag{9}$$

where the acceleration consists of a tangential component along the path and a normal component perpendicular to the

path. The expression $\boldsymbol{r}_{ss}(s)$ is the second partial derivation of $\boldsymbol{r}(s)$ by s. To prove this statement, we differentiate Eqn. (7) by time:

$$\dot{s} = ||\dot{\boldsymbol{q}}|| \tag{10}$$

Combining Eqn. (10) with Eqn. (8) leads to:

$$\dot{\boldsymbol{q}} = \frac{\dot{\boldsymbol{q}}}{||\dot{\boldsymbol{q}}||}\ ||\dot{\boldsymbol{q}}|| = \boldsymbol{r}_s(s)\ \dot{s} \tag{11}$$

This shows that the velocity consist of a unit vector tangential to the path:

$$\boldsymbol{r}_s(s) = \frac{\dot{\boldsymbol{q}}}{||\dot{\boldsymbol{q}}||} = \frac{\dot{\boldsymbol{q}}}{\dot{s}}. \tag{12}$$

$\boldsymbol{r}_s(s)$ times $\ddot{s}$ results in the tangential acceleration of Eqn. (9) and the vector $\boldsymbol{r}_{ss}(s)$ describes the change along the path s and is therefore always perpendicular to $\boldsymbol{r}_s(s)$. $\boldsymbol{r}_{ss}(s)$ is a vector normal to the path and calculated using Eqn. (9):

$$\boldsymbol{r}_{ss}(s) \quad = \quad \frac{\ddot{\boldsymbol{q}}\ -\ \boldsymbol{r}_s(s)\ \ddot{s}}{\dot{s}^2} \tag{13}$$

For the second derivation of s follows

$$\begin{aligned} \ddot{s} \quad &= \quad \frac{\partial \dot{s}}{\partial t} = \frac{\partial ||\dot{\boldsymbol{q}}||}{\partial t} = \frac{\partial \sqrt{\sum_{i=1}^{K}\ {}_i\dot{q}^2}}{\partial t} \\ &= \quad \frac{\sum_{i=1}^{K}\ {}_i\dot{q}_i\ {}_i\ddot{q}}{\sqrt{\sum_{i=1}^{K}\ {}_i\dot{q}^2}} = \frac{\dot{\boldsymbol{q}} \cdot \ddot{\boldsymbol{q}}}{||\dot{\boldsymbol{q}}||} \end{aligned} \tag{14}$$

The denominators of $\ddot{s}$, $\boldsymbol{r}_s(s)$, and $\boldsymbol{r}_{ss}(s)$ contain $\dot{s}$. In the special case, that $\dot{\boldsymbol{q}} = 0$ and therefore $\dot{s} = 0$, we have to consider in each case L'Hospital's Rule and find after rearranging the following feasible solutions:

a) Case 1: $\dot{\boldsymbol{q}} \to \mathbf{0}$ and $\ddot{\boldsymbol{q}} \to \mathbf{0}$:

$$\lim_{\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \to 0} \ddot{s} \quad = \quad 0 \tag{15}$$

$$\lim_{\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \to 0} \boldsymbol{r}_s(s) \quad \overset{0 <_k \ddot{q}}{=} \quad \frac{\dddot{\boldsymbol{q}}(\boldsymbol{q})}{||\ddot{\boldsymbol{q}}(\boldsymbol{q})||}\ \forall\, k \in \{1, \ldots, K\} \tag{16}$$

$$\lim_{\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}} \to 0} \boldsymbol{r}_{ss}(s) \quad = \quad \mathbf{0} \tag{17}$$

b) Case 2: $\dot{\boldsymbol{q}} \to \mathbf{0}$ and $\ddot{\boldsymbol{q}} \neq \mathbf{0}$:

$$\lim_{\dot{\boldsymbol{q}} \to 0} \ddot{s} \quad = \quad ||\ddot{\boldsymbol{q}}|| \tag{18}$$

$$\lim_{\dot{\boldsymbol{q}} \to 0} \boldsymbol{r}_s(s) \quad \overset{0 <_k \ddot{q}}{=} \quad \frac{\ddot{\boldsymbol{q}}}{||\ddot{\boldsymbol{q}}||}\ \forall\, k \in \{1, \ldots, K\} \tag{19}$$

$$\lim_{\dot{\boldsymbol{q}} \to 0} \boldsymbol{r}_{ss}(s) \quad = \quad \mathbf{0} \tag{20}$$

B. Combining Path Representation with Dynamics

We combine the one-dimensional path-representation with the inverse dynamics model to derive the extremal accelerations for a path described by the variable $\ddot{s}$. From this point on we shorten $\boldsymbol{r}_s(s)$ to $\boldsymbol{r}_s$ and $\boldsymbol{r}_{ss}(s)$ to $\boldsymbol{r}_{ss}$. We substitute $\ddot{\boldsymbol{q}}$ from Eqn. (9) into Eqn. (2).

$$\boldsymbol{M}(\boldsymbol{q})\,(\boldsymbol{r}_s\,\ddot{s} + \boldsymbol{r}_{ss}\,\dot{s}^2) = \boldsymbol{\tau} - \boldsymbol{\tau}_g(\boldsymbol{q}) - c(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} \tag{21}$$

Using $\boldsymbol{\alpha} := \boldsymbol{M}(\boldsymbol{q})\,\boldsymbol{r}_s$ and $\boldsymbol{\beta} := \boldsymbol{\tau}_g(\boldsymbol{q}) + c(\boldsymbol{q}, \dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{M}(\boldsymbol{q})\,\boldsymbol{r}_{ss}\ \dot{s}^2$ abbreviates Eqn. (21) to

$$\boldsymbol{\alpha}\ \ddot{s} \quad = \quad \boldsymbol{\tau}\ -\ \boldsymbol{\beta} \tag{22}$$

The three variables $\boldsymbol{\tau}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ only depend on $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, $\dot{s}$, $\boldsymbol{r}_s$, and $\boldsymbol{r}_{ss}$. $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ can be acquired from sensor inputs in the first instant the acceleration capabilities are calculated and after that, it can be calculated in every time step using the analytical descriptions of the polynomials generated by the OTG algorithm. $\dot{s}$, $\boldsymbol{r}_s$, and $\boldsymbol{r}_{ss}$ are calculated in every time step using the values of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$. We write the k-th row of Eqn. (22), which stands for the k-th DOF:

$${}_k\alpha\ \ddot{s} \quad = \quad {}_k\tau\ -\ {}_k\beta \tag{23}$$

In a next step, we take into account the torque capabilities, which leads to K individual inequalities, one for each DOF:

$${}_k\tau_{min} -\ {}_k\beta \leq\ {}_k\alpha\ \ddot{s} \leq\ {}_k\tau_{max} -\ {}_k\beta \tag{24}$$

Note that ${}_k\alpha \neq 0$, because the mass matrix $\boldsymbol{M}(\boldsymbol{q})$ is always positive definite and $\boldsymbol{r}_s$ is the non-zero path vector. Therefore, Eqn. (24) can be written as:

$$\begin{aligned} {}_kl \leq \ddot{s} \leq {}_kh\ , \quad {}_kl &= \begin{cases} \frac{{}_k\tau_{min} -\ {}_k\beta}{{}_k\alpha} & \text{for } {}_k\alpha > 0 \\ \frac{{}_k\tau_{max} -\ {}_k\beta}{{}_k\alpha} & \text{for } {}_k\alpha < 0 \end{cases} \\ {}_kh &= \begin{cases} \frac{{}_k\tau_{max} -\ {}_k\beta}{{}_k\alpha} & \text{for } {}_k\alpha > 0 \\ \frac{{}_k\tau_{min} -\ {}_k\beta}{{}_k\alpha} & \text{for } {}_k\alpha < 0 \end{cases} \end{aligned} \tag{25}$$

Equation (25) describes the range of joint accelerations $\ddot{s}$, for which a single actuator can hold the manipulator on its joint path without violating the k-th constraint. To fulfill all the constraints,

$$\ddot{s} \in [{}_kl, {}_kh]\ \forall\, k\ \in\ \{1, \ldots, K\} \tag{26}$$

has to be fulfilled. Only if the absolute velocity is too high, the intervals $[{}_kl, {}_kh]$ will have no intersection $\forall\ k\ \in\ \{1, \ldots, K\}$. Otherwise there will be an intersection of all the intervals. It follows that an admissible acceleration is any tangential acceleration $\ddot{s}$ that does not violate Eqn. (24) $\forall\ k\ \in\ \{1, \ldots, K\}$, that is:

$$\ddot{s}_{min} \leq \ddot{s} \leq \ddot{s}_{max} \quad \text{with} \quad \begin{aligned} \ddot{s}_{min} &= \max_k\ {}_kl \\ \ddot{s}_{max} &= \min_k\ {}_kh \end{aligned} \tag{27}$$

We transform $\ddot{s}_{min}$ and $\ddot{s}_{max}$ back into the n-dimensional space to get the two intersections with the PT:

$$\ddot{\boldsymbol{q}}_{min/max,path} = \ddot{s}_{min/max}\,\boldsymbol{r}_s + \boldsymbol{r}_{ss}\,\dot{s}^2 \tag{28}$$

For a given motion state $\boldsymbol{\Xi}$, these two values express how much the robot could accelerate/decelerate on the current path while still staying on it. We transform these values in absolute minimal and absolute maximal acceleration values $\forall\ k\ \in\ \{1, \ldots, K\}$:

$${}_k\ddot{q}_{min,abs} \quad = \quad \min({}_k\ddot{q}_{min,path}, {}_k\ddot{q}_{max,path}) \tag{29}$$

$${}_k\ddot{q}_{max,abs} \quad = \quad \max({}_k\ddot{q}_{min,path}, {}_k\ddot{q}_{max,path}) \tag{30}$$

C. Merging Algorithm

In order to plan trajectories for every axis from start to end, there has to be a period of acceleration and a period of deceleration. ${}_k\ddot{q}_{min/max,abs}$ represent only the limitations of the current motion state, but do not work well for the planning of a complete motion trajectory. A 2D example in Fig. 5 illustrates the problem. Zero acceleration is at the origin $\ddot{\boldsymbol{q}}_{1/2}$. The tangential acceleration is prolonged in both directions till the line intersects with the boundary. The intersection is $\ddot{\boldsymbol{q}}_{min/max,path}$ (cf. Eqn. (28)). In a 3D or higher dimensional case, the prolonged (hyper) line would intersect with two of the (hyper) planes of the PT. The blue-dashed rectangle shows the absolute acceleration limits $\ddot{\boldsymbol{q}}_{min/max,abs}$. This rectangle does not include the origin, because all coordinates of $\ddot{\boldsymbol{q}}_{min/max,abs}$ are positive. These values are not useful for trajectory planning, since they would need to be extended in the negative direction in order to include the origin. This is shown as a black-dashed rectangle. The discrepancy between what is given and what is at least needed is shown as a light grey area. Even though the origin would now be included, only *acceleration*, but almost no *deceleration* capabilities are given. A very small deceleration limit renders the execution time far too long and is therefore not optimal at all.

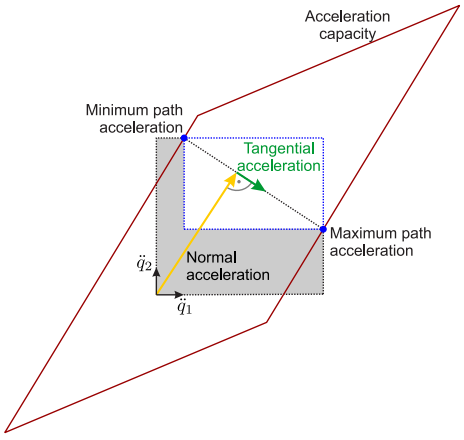

Fig. 5: Acceleration capabilities for a given motion state.

The "potentially" available acceleration/deceleration capabilities of the PT need to be considered, even though most of it is not needed for the current motion state. This motivates the merging algorithm, which is outlined in the following:

1) Using the current motion state $\boldsymbol{\Xi}_i$, calculate the path derivations $\dot{s}$, $\ddot{s}$, $\boldsymbol{r}_s$, and $\boldsymbol{r}_{ss}$.
2) Extend each coordinate axis in negative and positive direction until they intersect with the boundaries. To do this, we substitute the acceleration vector $\ddot{\boldsymbol{q}}$ in Eqn. (21) with the axis vector $\boldsymbol{\nu}_l$. This vector is computed as $\boldsymbol{\nu}_l = \boldsymbol{e}_l\,\nu_l$, where $\boldsymbol{e}_l$ is a unit vector along each axis with $l \in \{1,\ldots,K\}$. ν_l describes the scalar value of the vector along axis l. The rest of the calculation is analogous to Sec. V-B. It results in the acceleration constraints, $\ddot{\boldsymbol{q}}_{min/max,axes}$.
3) If $\dot{\boldsymbol{q}} = \boldsymbol{0} \wedge \ddot{\boldsymbol{q}} = \boldsymbol{0}$: $\ddot{\boldsymbol{q}}_{min/max} = \ddot{\boldsymbol{q}}_{min/max,axes}$
4) Else: Calculate the extremal path accelerations $\ddot{\boldsymbol{q}}_{min/max,path}$ according to Sec. V-B and find for both their absolute largest coordinate.
5) If both accelerations have their largest absolute value on the same coordinate axis with the same sign, we only store the larger of both as a constraint for that axis. We take a further look into the other one. We call the larger acceleration *A* and the smaller acceleration *B*:
   a) For *B*, we look at its second largest coordinate and compare it with the same coordinate axis of *A*.
   b) If both points have on that axis a value with the same sign and the coordinate of *A* is larger than the one of *B*, we look at the third largest coordinate of *B* in the same way again.
   c) We repeat this until they neither have the same sign or they have the same sign and the coordinate of *B* is larger than the one of *A*. In either case we use the found coordinate value of *B* as the second constraint.
   d) If the coordinates of *B* were always smaller than the ones of *A*, *B* is not used in this step.
6) The found scalar values are taken as fixed constraints on those coordinates.
7) The remaining coordinates of $\ddot{\boldsymbol{q}}_{min/max,path}$ are compared coordinate by coordinate with the earlier calculated $\ddot{\boldsymbol{q}}_{min/max,axes}$. The larger of both values is taken as constraint on that axis. This is done to ensure, that the merged boundaries always encompass $\ddot{\boldsymbol{q}}_{min/max,path}$. The merged result is $\ddot{\boldsymbol{q}}_{min/max}$.
8) The optimized acceleration constraints $\ddot{\boldsymbol{q}}_{min/max}$ are used as acceleration constraints for the VOTG.

Figure 6 takes up the example from before to demonstrate how the Merging Algorithm works. Extending the axes starting from the origin with the dotted grey lines to the PT results in two intersections per axis, which are the "axes acceleration constraints" $\ddot{\boldsymbol{q}}_{min/max,axes}$. We compare the tangential acceleration values with the axes acceleration limits and generate the finally used acceleration constraints, shown by the dotted black box. *The key principle used herefore is the fact that we only need to limit* ${}_k\ddot{q}_{min/max,path}$ *each by its largest coordinate value. For the other coordinate values, the axes acceleration limits can be used.* This principle works for any number of DOFs. The gain in potentially usable acceleration capability is shown by the light grey area.

D. Dynamics, Merging Algorithm, and OTG Combined

The acceleration constraints $\ddot{\boldsymbol{q}}_{i,min/max}$ used by the VOTG were in the past only roughly estimated. A possible approach to improve this would be to calculate at the moment a new target motion state $\boldsymbol{\Xi}_{i,trgt}$ is chosen the acceleration constraints $\ddot{\boldsymbol{q}}_{min/max}$ for the current motion state $\boldsymbol{\Xi}_i$ according to Sec. V-C. We then choose only a small fraction of these constraint values as constant input to the

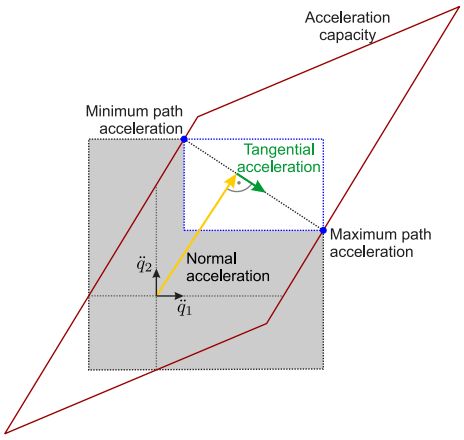

Fig. 6: Application example for the merging algorithm.

VOTG to ensure that the generated acceleration trajectory will always be executable without tracking-errors, even if the motion goes through a robot configuration, which has only relatively small acceleration capabilities. We do not settle with this rather pragmatic approach, instead provide a new algorithm that takes into accout the dynamics throughout the motion. In the following, we will show how the procedure of Sec. V-C is applied to the calculation of acceleration constraints and combine them with the VOTG. We provide two approaches for the new *target velocity-based dynamic OTG, short VDOTG*.

The general use case for a trajectory generator is the following: An unforeseen event happens at t_i, and the system is in a motion state $\boldsymbol{\Xi}_i$, from which we have to generate a new trajectory in order to reach a user-given value $\boldsymbol{\Xi}_{i,trgt} = \{\dot{\boldsymbol{q}}_{i,trgt}, \ddot{\boldsymbol{q}}_{i,trgt} \equiv \boldsymbol{0}\}$ in the shortest possible time. The VOTG (cf. Sec. III) will bring the current velocity as fast as possible to the target velocity by choosing the shortest possible motion trajectory. This version is useful if the system has to do an emergency stop ($\dot{\boldsymbol{q}}_{i,trgt} = \boldsymbol{0}$) or if it has to get to a specific velocity ($\dot{\boldsymbol{q}}_{i,trgt} \neq \boldsymbol{0}$) as fast as possible.

1) First Approach: We calculate the acceleration constraints for the start motion state and based on that generate an initial motion trajectory using the VOTG. The DOF κ determines the execution time of the complete trajectory, all other DOFs are adjusted accordingly. We read out from the initial motion trajectory the next motion state after one cycle time and calculate the new acceleration constraints for it. We compute the ratios between the current acceleration constraints and the initial acceleration constraints for every DOF. Using the smallest ratio value we adjust the acceleration constraint of the DOF κ at the current step. We then call the VOTG only for the DOF κ. We take from this trajectory the next motion state and only use the position value to look up the time η when the original trajectory would have reached this position for the DOF κ. For all other DOFs, we read out the new position values from the original trajectory using the time η. Using these new position values and the system cycle time, we calculate the new velocities and accelerations. We compute the acceleration constraints for this new motion state and based on that calculate the ratios again. We repeat this until the target velocity is reached. This approach is not stable, because the acceleration profiles grow exponentially; Fig. 7 in Sec. VI shows this behavior.

2) Second Approach: A second approach is an algorithm that optimizes an initially calculated motion trajectory by adjusting it at every time step, if needed. The optimization is based on the acceleration constraints, which are calculated at every motion state. The used path is computed by an initial run of the VOTG. This algorithm will not find the global optimum, but it will optimize in real-time the initially given trajectory to fully account for the dynamic acceleration constraints of the system. The steps are detailed in the following:

1) Define $i = 0$
2) For the values $\boldsymbol{q}_i$ and $\dot{\boldsymbol{q}}_i$ of the current motion state $\boldsymbol{\Xi}_i$, calculate the joint space dynamic parameters of the system as described in Sec. IV. The "Rigid Body Dynamics Algorithm" of [24] is used.
3) For the current motion state $\boldsymbol{\Xi}_i$, calculate new values for the minimum and maximum accelerations $\ddot{\boldsymbol{q}}_{i,min}$, $\ddot{\boldsymbol{q}}_{i,max}$ as described in Sec. V-C.
4) Call the Type IV VOTG algorithm. This will provide a trajectory (incl. path) for all K DOFs to reach $\dot{\boldsymbol{q}}_{i,trgt}$ and $\ddot{\boldsymbol{q}}_{i,trgt} \equiv \boldsymbol{0}$ at the same time instant.
   - Input values:
     - motion state: $\boldsymbol{\Xi}_i$
     - current acceleration constraints: $\ddot{\boldsymbol{q}}_{i,min}$, $\ddot{\boldsymbol{q}}_{i,max}$
     - constant jerk constraints: $\dddot{\boldsymbol{q}}_{0,min}$, $\dddot{\boldsymbol{q}}_{0,max}$
     - constant target motion state: $\dot{\boldsymbol{q}}_{0,trgt}$
   - Output values: new motion state
     - next motion state: $\boldsymbol{\Xi}_{i+1}$
5) If the trajectory is executed on a robot system, $\boldsymbol{\Xi}_{i+1}$ will be send to the controllers.
6) Increment i
7) Check if $\dot{\boldsymbol{q}}_{0,trgt}$ is reached. If not, go back to step 2.
8) $\boldsymbol{\Xi}_{i+1}$ describes the motion state, at which $\dot{\boldsymbol{q}}_{i,trgt}$ is reached.

E. Summary

The robot dynamics are described along a one-dimensional motion path. Based on this, the acceleration is split into a tangential and normal component, which again is used for calculating the path dependent acceleration capabilities. The planning principle of the VOTG does not allow to directly use the acceleration capabilities gained from the path dependent method as acceleration constraints. A merging algorithm is proposed, which takes the results of the acceleration capabilities along a path into proper acceleration constraints that can always be used with the VOTG.

The first VDOTG approach calculates in a first step for all DOF an initial trajectory and then optimizes it at every time step using the VOTG only for one DOF, the one that determines the execution time. The input constraints of the VOTG are calculated taking into account the acceleration

constraints for every DOF. The one-dimensional output result of the VOTG is feed to an inverse lookup of the original trajectory in order to determine the new motion state of the remaining DOFs. The second VDOTG approach uses for all DOFs the current acceleration constraints in every time step as input for the VOTG.

VI. Experimental Results

A. Simulation

The first VDOTG approach of Sec. V-D.1 showed unstable behavior when simulating it in Mathematica [25]. Only after a few time steps, the algorithm suddenly destabilizes and the values go astray. Fig. 7 shows this behavior for three DOFs. The first row of the figure shows the VOTG run for 73 time steps and the second row shows what the first approach of the VDOTG does in the same time frame. After 60 steps, axes 1 and 4 start to go astray from the VOTG trajectory. This would not necessarily be a problem as long as the target velocity would finally be reached. But the VDOTG breaks up after 73 steps when the allowable maximum acceleration is breached for axis 4. The VOTG trajectory reaches the target velocity after 175 steps without exceeding the constraints. This behavior also happens when varying the input values randomly, it destabilizes before reaching the target velocity.

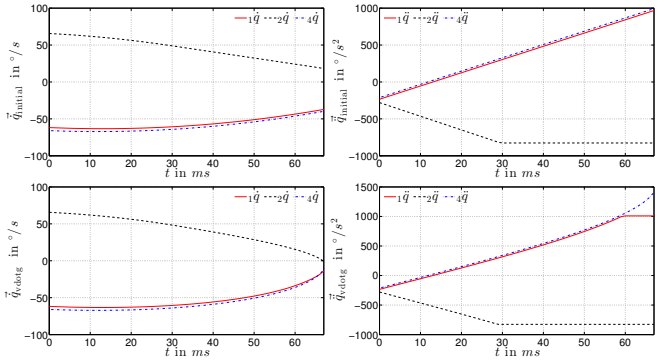

Fig. 7: Comparison of the initial VOTG and the VDOTG first approach trajectory: the acceleration profile of the VDOTG becomes unstable.

That is why we chose to work on a second approach for the VDOTG. As can be seen in Fig. 8, the initial VOTG trajectory shown in the first row uses the acceleration constraints calculated at start, while the VDOTG adjusts the trajectory during the whole motion. The VOTG would breach the acceleration constraints after about 50 steps while the VDOTG stays within the upper and lower acceleration constraints. These acceleration constraints are explicitly shown in Fig. 9.

B. Robot Experiments

We were able to show in simulation that the second approach of the VDOTG stays within the limits and runs stable. We then implemented the algorithm in C++ to run it at a rate of $1\,kHz$ on a KUKA Lightweight Robot [1]. To demonstrate a realistic scenario, the robot's end-effector was equipped with a payload of $4.2\,kg$. In order to demonstrate

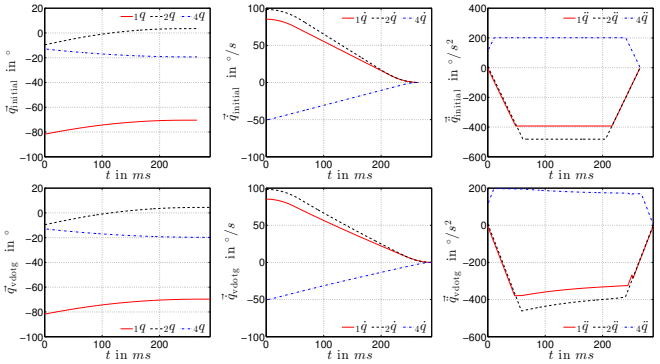

Fig. 8: Comparison of the initial VOTG and the VDOTG second approach trajectory: only the VDOTG respects the variable acceleration constraints.

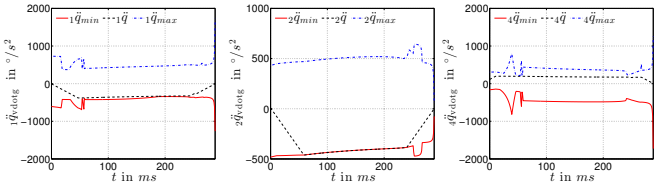

Fig. 9: VDOTG second approach: VDOTG values within minimal and maximal acceleration constraints.

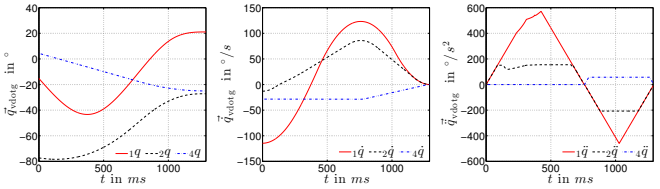

Fig. 10: Experimental results of the VDOTG on a KUKA Lightweight Robot: position, velocity, and acceleration trajectories of axis 1, 2, and 4.

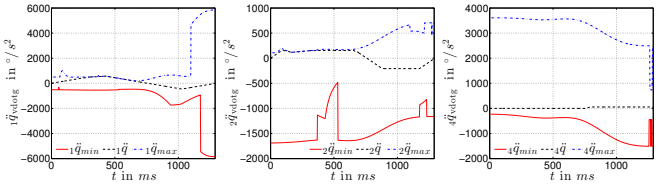

Fig. 11: Experimental results of VDOTG on a KUKA Lightweight Robot: acceleration profiles including lower and upper constraints for axis 1, 2, and 4.

the capability of the second approach of the VDOTG, we moved the robot from one point to another and then activate at a random instant the VDOTG algorithm through an unexpected user input. The robot arm first accelerates to $\dot{\boldsymbol{q}}_{trgt} = (123,\ 86,\ 11,\ -28,\ 6,\ 29,\ 52)^T\ ^\circ/s$, and then the VDOTG brings all seven joints *time-optimally* back to zero-velocity, whereas the computation of this trajectory is done within one control cycle. The VDOTG profiles are shown for axes 1, 2, and 4 in Fig. 10 and Fig. 11.

VII. Conclusion

This paper combines target velocity-based online trajectory generation, short VOTG, with robot acceleration capa-

bilities. An acceleration can be split along its motion path into a normal and a tangential vector component. Extending the tangential acceleration vector so that it intersects with the acceleration capabilities described geometrically as a parallelotope results in the acceleration limits for any given motion state. In order to obtain suitable acceleration constraints for online trajectory generation, the actual acceleration capabilities were converted to useful linearly independent values. The combination of the acceleration constraints with the existing VOTG resulted in a target velocity-based dynamic online trajectory generation, short VDOTG. This combined approach was successfully implemented and tested with a seven-degree-of-freedom lightweight robot. Motion trajectories were computed within one single control cycle ($1\,ms$) taking into account global kinematic and dynamic constraints.

ACKNOWLEDGEMENTS

The fruitful discussions with and the permanent support by Samir Menon and Francois Conti are highly appreciated. Research funding was provided by the Dr.-Willy-Höfler-Stiftung, Förderverein Kurt Fordan für herausragende Begabungen e.V., and the Friedrich-Naumann-Stiftung für die Freiheit.

REFERENCES

[1] R. Bischoff, J. Kurth, G. Schreiber, R. Köppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR lightweight robot arm — A new reference platform for robotics research and manufacturing. In *Proc. of the Joint Conference of ISR 2010 (41st Internationel Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, Munich, Germany, June 2010. VDE Verlag.

[2] Meka Robotics LLC, 1240 Pennsylvania Ave, San Francisco, CA 94107, USA. Homepage. `http://www.meka.com` (accessed: Mar. 22, 2013). Internet, 2013.

[3] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3356–3363, Nice, France, September 2008.

[4] A. De Luca and R. Farina. Dynamic scaling of trajectories for robots with elastic joints. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2436–2442, Washington, D.C., USA, May 2002.

[5] X. Broquère, D. Sidobre, and I. Herrera-Aguilar. Soft motion trajectory planner for service manipulator robot. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2808–2813, Nice, France, September 2008.

[6] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3248–3253, Nice, France, September 2008.

[7] T. Kröger. *On-Line Trajectory Generation in Robotic Systems*, volume 58 of *Springer Tracts in Advanced Robotics*. Springer, Berlin, Heidelberg, Germany, first edition, January 2010.

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

[9] F. Pfeiffer and R. Johanni. A concept for manipulator trajectory planning. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 3, pages 1399–1405, San Francisco, CA, USA, April 1986.

[10] K. G. Shin and N. D. McKay. Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Trans. on Automatic Control*, 30(5):531–541, June 1985.

[11] L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[12] W. Chung, L.-C. Fu, and S.-H. Hsu. Motion control. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 6, pages 133–159. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[13] O. Brock, J. Kuffner, and J. Xiao. Manipulation for robot tasks. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 26, pages 615–645. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[14] L. E. Kavraki and S. M. LaValle. Motion planning. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 5, pages 109–131. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[15] J. M. Hollerbach. Dynamic scaling of manipulator trajectories. *ASME Journal on Dynamic Systems, Measurement, and Control*, 106(1):102–106, 1984.

[16] O. Dahl and L. Nielsen. Torque limited path following by on-line trajectory time scaling. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1122–1128, Scottsdale, AZ, USA, May 1989.

[17] C.-J. Wu. A numerical approach for time-optimal path-planning of kinematically redundant manipulators. *Robotica*, 12(5):401–410, September 1994.

[18] J. Vannoy and J. Xiao. Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Trans. on Robotics*, 24(5):1199–1212, October 2008.

[19] O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, December 2002.

[20] S. Haddadin, H. Urbanek, S. Parusel, D. Burschka, J. Roßmann, A. Albu-Schäffer, and G. Hirzinger. Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3109–3116, Taipei, Taiwan, October 2010.

[21] C. Guarino Lo Bianco and F. Ghilardelli. Third order system for the generation of minimum-time trajectories with asymmetric bounds on velocity, acceleration, and jerk. In *Workshop on Robot Motion Planning: Online, Reactive, and in Real-time at the IEEE International Conference on Intelligent Robots and Systems*, pages 137–143, Vilamoura, Portugal, October 2012.

[22] S.-M. Khansari-Zadeh and A. Billard. A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, 32(4):433–454, May 2012.

[23] A. M. Zanchettin and B. Lacevic. Sensor-based trajectory generation for safe human-robot cooperation. In *Workshop on Robot Motion Planning: Online, Reactive, and in Real-time at the IEEE International Conference on Intelligent Robots and Systems*, pages 62–66, Vilamoura, Portugal, October 2012.

[24] R. Featherstone and D. E. Orin. Dynamics. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 2, pages 35–65. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[25] Mathematica. *Version 8.0*. Wolfram Research Inc., Champaign, Illinois, 2012.