Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events

Torsten Kröger, *Member, IEEE*, and Friedrich M. Wahl, *Member, IEEE*

***Abstract*—This paper introduces a new method for motion-trajectory generation of mechanical systems with multiple degrees of freedom (DOFs). The key feature of this new concept is that motion trajectories are generated online, i.e., within every control cycle, typically every millisecond. This enables systems to react instantaneously to unforeseen and unpredictable (sensor) events at any time instant and in any state of motion. As a consequence, (multi)sensor integration in robotics, in particular the development of control systems enabling sensor-guided and sensor-guarded motions, becomes greatly simplified. We introduce a class of online trajectory-generation algorithms and present the mathematical basics of this new approach. The algorithms presented here consist of three steps: calculation of the minimum synchronization time for all DOFs, synchronization of all DOFs, and calculation of output values. The theory is followed by real-world experimental results indicating new possibilities in robot-motion control.**

***Index Terms*—Hybrid switched systems, multisensor integration, robot-motion control, trajectory generation.**

I. Introduction

THIS PAPER focuses on sensor integration in robotics, in particular in robotic-manipulation control systems. We consider a mechanical system with multiple degrees of freedom (DOFs) equipped with one or more sensors delivering digital and/or analog sensor signals. It is—of course—no matter of question that sensor integration and sensor-based control belong to the very basics in robotics. However, there is still one important question that has yet to be completely answered: If we consider a robot in an arbitrary state of motion, how can we calculate a trajectory if we want the robot to *react instantaneously to unforeseen sensor events*? This is comparable to many human everyday life scenarios: If a little child accidentally touches a stove with its hand, he or she knee-jerkily reacts by pulling its hand away from the hot surface. Another scenario could be the fight of two swordsmen: Depending on the motion of the opponent, the fighters react *immediately* by adapting their own body motions and moves.

Before coming up with details, we would like to make some basic clarifications: A robotic system is considered as a mechanical and/or mobile system with multiple DOFs. Assuming such a system to be equipped with a number of (different) sensors, we distinguish between

Manuscript received April 15, 2009; revised October 6, 2009 and October 19, 2009. First published December 8, 2009; current version published February 9, 2010. This paper was recommended for publication by Associate Editor A. Albu-Schäffer and Editor J.-P. Laumond upon evaluation of the reviewers' comments.

The authors are with the Institut für Robotik und Prozessinformatik, Technische Universität Carolo-Wilhelmina zu Braunschweig, Mühlenpfordtstraße 23, D-38106 Braunschweig, Germany (e-mail: t.kroeger@tu-bs.de; f.wahl@tu-bs.de).

Digital Object Identifier 10.1109/TRO.2009.2035744

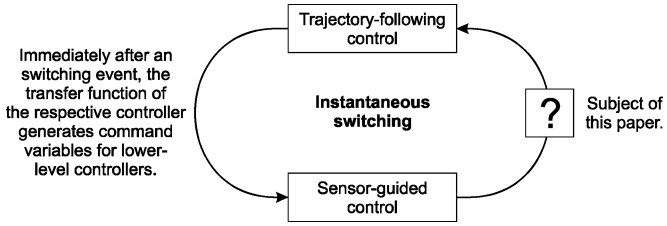

Fig. 1. Subject of this paper: Instantaneous switchings between trajectory-following motions and sensor-guided motions.

1) trajectory-following control;
2) sensor-guided motion control;
3) sensor-guarded motion control.

Sensor-guided motion control employs the system's actuators to be part of a control loop, whose control variables are based on sensor signals (cf. Fig. 1). We consider a *sensor* as a generic device that delivers a signal depending on the overall system state, i.e., the robotic system with its complete environment. Clear and concrete examples are force/torque control (e.g., [1], just to name one out of many) and visual servo control (e.g., [2]). Here, the robot motion of one single control cycle depends on the sensor signal(s) of the current control cycle.

In contrast to this, *sensor-guarded motions* are considered as trajectory-following motions and/or sensor-guided motions (cf. Fig. 1), whose motion parameters may change abruptly, depending on *sensor events*. These parameter changes may include set-point switchings for closed-loop controllers (e.g., new force/torque control set-points) as well as new target positions for trajectory-following control. Simple examples here are the transition from free space to contact, i.e., the transition from pose to force/torque control. In the moment of transition, the robot controller switches the controllers for all DOFs being in contact from trajectory-following control to force/torque control. A second obvious example is the instantaneous reaction of a robotic system to any predictable or unpredictable event: The manipulator of [3] plays the parlor game Jenga [4] and has to interrupt any motion as soon as the game tower starts to topple.

In the development of such systems, it becomes evident that we need to switch discretely between several (open- and/or closed-loop) continuous controllers at any time. If we arbitrarily switch one or more DOFs from trajectory-following control to sensor-guided control, this problem is usually solved. By using the transfer function of a desired controller, command variables for lower level control can be generated at any time instant (cf. Fig. 1). However, how can we switch *one, some, or all* DOFs of a robotic system from sensor-guided control

to trajectory-following control? If we consider a robot in an arbitrary state of motion, how can we calculate a trajectory if we want the robot to *react instantaneously to unforeseen (sensor) events*?

This is the central question of this paper, whose central novelty is an open-loop control module that is able to generate time-optimal and time-synchronized motion trajectories for mechanical systems with multiple DOFs during runtime, i.e., online during *every* control cycle. This opens completely new advances for multisensor integration in robotic systems, and hybrid switched-control systems become suited for a wide range of robotic applications.

Let us now give a short overview of this paper: After related works have been presented in Section II, Section III introduces some basics, a dedicated notation for this paper, and some special conventions used throughout the paper. During the development of this paper, nine different types of online trajectory generators (OTG) were elaborated and are classified in Section IV. The basic algorithm applied to any type of online trajectory generation is introduced in Section V, whereas Section VI describes the most relevant type of OTG, i.e., Type IV. After the relation of this new concept to higher-level (sensor-based) motion-planning system is discussed in Section VII, Section VIII presents simulation and real-world experimental results, which highlight the practical relevance of this concept.

II. Related Work

Off-Line Trajectory-Generation Concepts: Kahn and Roth [5] belong to the pioneers in the field of time-optimal trajectory planning. They used methods of optimal, linear control theory and achieved a near-time-optimal solution for linearized manipulators. The work of Brady [6] introduces several techniques of trajectory planning. In later works, the manipulator dynamics were taken into account [7], and jerk-limited trajectories were applied [8]. The concept of Lambrechts *et al.* [9] produces very-smooth fourth-order trajectories but is also limited to a known initial state of motion and to one DOF.

Real-Time Adaptive-Motion Planning: In recent times, works on real-time adaptive-motion planning have been published [10]–[13]. Here, the field of online trajectory planning is addressed and the *path* of a robotic manipulator is adapted that depends on its state and environment. This proposal focuses on online trajectory generation, which is more related to the field of robot *control*. It acts as an important interface to such higher-level planning methods of [10]–[13] (cf. Section VII).

Online Trajectory-Generation Concepts: The works mostly related to this paper are [14]–[19]. Macfarlane and Croft [14] present a jerk-bounded, near-time-optimal trajectory planner that uses quintic splines, which are also computed online but only for *1-DOF systems*. In [15], Cao *et al.* use rectangular jerk pulses to compute trajectories, but accelerations different from zero cannot be applied. Compared with the multi-DOF approach presented here, the latter method has been developed for *1-D problems* only. Broquère *et al.* [16] published a work that uses an online trajectory generator for an arbitrary number of independently acting DOFs. The approach is very similar to the one of Liu [17] and is based on the classic seven-segment acceleration profile [20], but the approach is unfortunately also incomplete and can only perform reactions if the current acceleration value of a DOF is zero. Briefly summarized, a major disadvantage of [14]–[17] is that they cannot cope with initial acceleration values unequal to zero. A further, very recent work of Haschke *et al.* [18] presents an online trajectory planner in the very same sense as this paper does. The proposed algorithm generates jerk-limited trajectories from arbitrary states of motion, but it suffers from numerical stability problems, i.e., it may happen that no jerk-limited trajectory can be calculated. In such a case, a second-order trajectory with infinite jerks is calculated. Furthermore, the algorithm only allows target velocities of zero. Ahn *et al.* [19] proposed a work for the online calculation of 1-D trajectories for any given state of motion *and* with arbitrary target states of motion, i.e., with target velocities and target accelerations unequal to zero. Sixth-order polynomials are used to represent the trajectory, which is called arbitrary-states polynomial-like trajectory (ASPOT). The major drawback of this paper is that no kinematic-motion constraints, such as maximum velocity, acceleration, and jerk values, can be specified.

Concepts for Instantaneous Reactions to Collisions and Interactions: Heinzmann and Zelinsky [21] introduced a method to limit impact forces in the case of undesired collisions, such that damage and possible human injuries are prevented. Based on the skeleton algorithm [22], De Santis and Siciliano [23] present a reactive method for collision avoidance in which proprioceptive sensor data are used to calculate repulsion forces and joint torques to "flee" from the position in which a potential collision is expected. Based on these works, Haddadin *et al.* [24] present a very impressive work on the detection of unforeseen collisions and respective reaction concepts. Five different collision (= sensor event) reaction strategies are investigated: 1) no reaction; 2) immediate stopping; 3) switching from position control to zero-gravity torque control [25]; 4) switching to torque control with gravity compensation, but, in contrast, to; 3) using joint torque feedback and the signal of the estimated external torque, which is used as a collision signal, to scale down both the motor inertia as well as the link inertia, thus obtaining an even "lighter" robot; and 5) using the estimated external torque to implement an admittance controller. The strategies 3–5 contain switchings from trajectory-following control to sensor-guided robot motion control; this paper considers the opposite way of switching: from sensor-guided motion back to trajectory-following control (cf. Fig. 1) or abrupt switching of trajectory parameters (e.g., target state of motion and/or kinematic-motion constraints) to react to unforeseen events such as the detection of potential collisions.

The proposed class of algorithms of this paper closes the important loop of Fig. 1. Summarizing this section briefly, all mentioned approaches are not able to cope with *arbitrary initial states of motion*, i.e., arbitrary position, velocity, and acceleration values, in a robust way—neither for the one- nor for the multidimensional case. This paper extends all mentioned works, such that *instantaneous* reactions to unforeseen (sensor) events become feasible for multi-DOF robotic systems (cf. Fig. 1).

III. Basics, Notations, and Conventions

This section briefly introduces the nomenclature used throughout this paper, defines some important terms, and gives a first impression how the regarded trajectories are mathematically represented.

As we consider PC- or microcontroller-based systems for robot-motion control, we assume a time-discrete overall system with a set of time instants

$$\mathcal{T} = \{T_0, \ldots, T_i, \ldots, T_N\} \quad \text{with } T_i = T_{i-1} + T^{\text{cycle}} \text{ and } i \in \{1, \ldots, N\} \tag{1}$$

where T^{cycle} represents the cycle time of the system. Time-discrete values are represented by capital letters, time-continuous values by lowercase letters. The position of the robotic system at time T_i is $\vec{P}_i = ({}_1P_i, \ldots, {}_kP_i, \ldots, {}_KP_i)^T$, where K is the number of DOFs. Velocities, accelerations, and jerks are analogously represented by $\vec{V}_i$, $\vec{A}_i$, and $\vec{J}_i$. A complete state of motion at time T_i is described by the matrix

$$\begin{aligned}\mathbf{M}_i &= (\vec{P}_i, \vec{V}_i, \vec{A}_i, \vec{J}_i) \\ &= ({}_1\vec{M}_i, \ldots, {}_k\vec{M}_i, \ldots, {}_K\vec{M}_i)^T.\end{aligned} \tag{2}$$

Kinematic-motion constraints are analogously denoted as

$$\mathbf{B}_i = (\vec{V}_i^{\max}, \vec{A}_i^{\max}, \vec{J}_i^{\max}, \vec{D}_i^{\max}) \tag{3}$$

where $\vec{D}_i^{\max}$ is the maximum value for the derivative of jerk at time T_i. A target state of motion is denoted by

$$\mathbf{M}_i^{\text{trgt}} = (\vec{P}_i^{\text{trgt}}, \vec{V}_i^{\text{trgt}}, \vec{A}_i^{\text{trgt}}, \vec{J}_i^{\text{trgt}}). \tag{4}$$

T_N is the time instant at which $\mathbf{M}_i^{\text{trgt}}$ will be reached. As we will explain later, time-continuous polynomials are needed for the internal representation of trajectories. Here

$$\begin{aligned}{}_k^l p_i(t) &= a_4(t - \Delta t)^4 + a_3(t - \Delta t)^3 \\ &\quad + a_2(t - \Delta t)^2 + a_1(t - \Delta t) + a_0\end{aligned} \tag{5}$$

constitutes a fourth-order polynomial describing the position progression for DOF k, time-shifted by Δt, and calculated at time instant T_i. The index l is described later in this section. Polynomials for velocity, acceleration, and jerk progressions are analogously denoted by ${}^l\vec{v}_i(t)$, ${}^l\vec{a}_i(t)$, and ${}^l\vec{j}_i(t)$. In summary, we obtain a matrix of polynomials

$$\begin{aligned}{}^l\mathbf{m}_i(t) &= ({}^l\vec{p}_i(t), {}^l\vec{v}_i(t), {}^l\vec{a}_i(t), {}^l\vec{j}_i(t)) \\ &= ({}_1^l\vec{m}_i(t), \ldots, {}_k^l\vec{m}_i(t), \ldots, {}_K^l\vec{m}_i(t))^T\end{aligned} \tag{6}$$

where a single row, i.e., the polynomials of one single DOF k, is denoted by

$${}_k^l\vec{m}_i(t) = \left({}_k^l p_i(t), {}_k^l v_i(t), {}_k^l a_i(t), {}_k^l j_i(t)\right). \tag{7}$$

Each set of motion polynomials ${}^l\mathbf{m}_i(t)$ for all K DOFs is accompanied by a set of time intervals

$${}^l\mathcal{V}_i = \left\{{}_1^l\vartheta_i, \ldots, {}_k^l\vartheta_i, \ldots, {}_K^l\vartheta_i\right\}, \text{ where } {}_k^l\vartheta_i = \left[{}_k^l t_i,\ {}_k^{(l+1)} t_i\right] \tag{8}$$

in which a set of polynomials ${}_k^l\vec{m}_i(t)$ for one single DOF k is valid. A complete motion trajectory $\mathcal{M}_i(t)$ is finally composed of a set of motion polynomials with according time intervals

$$\begin{aligned}\mathcal{M}_i(t) = \{&({}^1\mathbf{m}_i(t), {}^1\mathcal{V}_i), \ldots, ({}^l\mathbf{m}_i(t), {}^l\mathcal{V}_i), \\ &\ldots, ({}^L\mathbf{m}_i(t), {}^L\mathcal{V}_i)\}.\end{aligned} \tag{9}$$

Depending on the type and variant of OTG (cf. Section IV), the initial state of motion $\mathbf{M}_0$, and the target state of motion $\mathbf{M}_0^{\text{trgt}}$, the value L determines the required number of polynomials (i.e., the number of single trajectory segments) to describe the complete trajectory from $\mathbf{M}_0$ to $\mathbf{M}_0^{\text{trgt}}$.

One important property of OTG is that all DOFs, which are selected for trajectory-following control, have to reach their target state of motion $\mathbf{M}_i^{\text{trgt}}$ at the same time instant, namely at t_i^{sync} in order to achieve *time synchronization*. As a consequence of this requirement, we can already state that

$$T_N - t_i^{\text{sync}} \leq T^{\text{cycle}}. \tag{10}$$

Just to give an impression of time dimensions, T^{cycle} lies in the range of one millisecond or less, i.e., the resulting OTG algorithms are designed to be applicable on a very low control level.

In the following, the term *time optimality* is supposed to be defined; we distinguish between the following two different kinds:

1) *Kinematic time-optimality:* A system is transferred from an initial state of motion $\mathbf{M}_i$ at instant T_i into a desired target state of motion $\mathbf{M}_i^{\text{trgt}}$ within the shortest possible time without any consideration of couplings between individual DOFs.
2) *Dynamic time-optimality:* A system is transferred from an initial state of motion $\mathbf{M}_i$ at instant T_i into a desired target state of motion $\mathbf{M}_i^{\text{trgt}}$ within the shortest possible time with consideration of the whole system dynamics.

In the context of this paper, *kinematic* time optimality is considered. The important consequence of kinematic time optimality is, that all K DOFs can be considered to be linearly independent. When calculating $\mathcal{M}_i(t)$ at T_i, the following requirements have to be fulfilled to achieve time optimality:

$$\begin{aligned}&\forall\, k \in \{1, \ldots, K\}: \\ &{}_k^1 t_i = T_i \\ &{}_k^1\vec{m}_i(T_i) = {}_k\vec{M}_i \\ &{}_k^l\vec{m}_i({}_k^l t_i) = {}_k^{l-1}\vec{m}_i({}_k^l t_i) \text{ with } l \in \{2, \ldots, L\} \\ &{}_k^L\vec{m}_i(t_i^{\text{sync}}) = {}_k\vec{M}_i^{\text{trgt}}\end{aligned} \tag{11}$$

and

$$\left.\begin{aligned}&\forall\, (k, l) \in \{1, \ldots, K\} \times \{1, \ldots, L\}: \\ &{}_k^l v_i(t) \leq {}_kV_i^{\max} \\ &{}_k^l a_i(t) \leq {}_kA_i^{\max} \\ &{}_k^l j_i(t) \leq {}_kJ_i^{\max} \\ &{}_k^l d_i(t) \leq {}_kD_i^{\max}\end{aligned}\right\}, \text{ with } t \in \left[{}_k^l t_i, {}_k^{l+1} t_i\right] \tag{12}$$

such that

$$t_i^{\text{sync}} \longrightarrow \min. \tag{13}$$

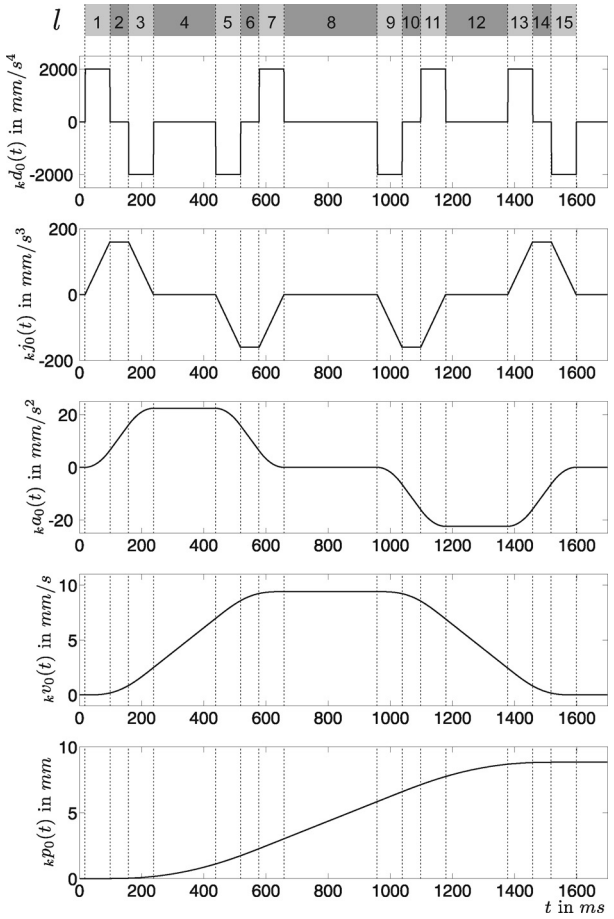

Fig. 2. Example of a very simple case of a fourth-order motion trajectory for one DOF k consisting of $L = 15$ matrices of polynomials ${}^l\mathbf{m}_i(t)$ [cf. (6)]. The input parameters are ${}_k\vec{M}_0 = \vec{0}$, ${}_k\vec{B}_0 = (9.4\text{ mm/s}, 22.4\text{ mm/s}^2, 160\text{ mm/s}^3, 2000\text{ mm/s}^4)$, and ${}_k\vec{M}_0^{\text{trgt}} = (8.8\text{ mm}, 0\text{ mm/s}, 0\text{ mm/s}^2, \text{and } 0\text{ mm/s}^3)$ (cf. [9]).

Equation (11) describes the single-motion states at the beginning or at the end of a trajectory segment l, respectively. Due to (12), it is guaranteed that the motion variables are kept within their kinematic constraints.

Now, we know how to describe a complete motion trajectory $\mathcal{M}_i(t)$ at a time instant T_i. It constitutes the minimum set of parameters to handle sets of trajectories for OTG. For a better understanding and to clarify these introductory equations, Fig. 2 depicts a simple (translational) motion trajectory for one single DOF k.

As last part of this section, we explain the accordance of these introductory description to the input and output values of the OTG algorithms. Fig. 3 shows all input parameters

$$\mathbf{W}_i = (\mathbf{M}_i, \mathbf{M}_i^{\text{trgt}}, \mathbf{B}_i, \vec{S}_i) \tag{14}$$

and all output values $\mathbf{M}_{i+1}$ of the OTG algorithm, where $\mathbf{W}_i$ is an arbitrary $(K \times 13)$ matrix. The selection vector $\vec{S}_i$ is a Boolean vector and determines which of the K DOFs have to be controlled by the OTG. The DOFs, which are controlled by other open- or closed-loop controllers, are not considered by the algorithm.

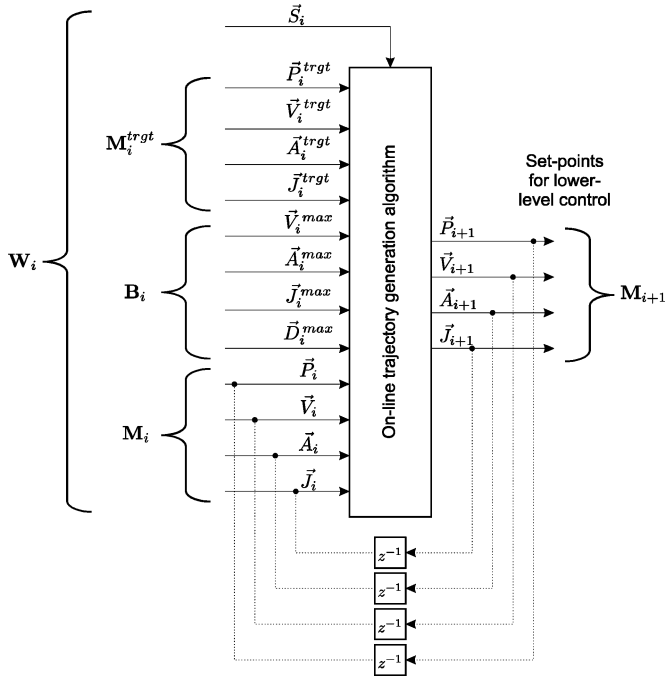

Fig. 3. Input and output values the OTG (z^{-1} represents a hold element). The dotted part indicates how the output values of the OTG are usually fed back.

We assume arbitrary values for $\mathbf{W}_i$ as granted; the only (trivial) kinematic constraint is given by (15)

$$\forall\, k \in \{1, \ldots, K\}:$$
$${}_kV_i^{\text{trgt}} \leq {}_kV_i^{\max} \wedge {}_kA_i^{\text{trgt}} \leq {}_kA_i^{\max} \wedge {}_kJ_i^{\text{trgt}} \leq {}_kJ_i^{\max}. \tag{15}$$

As already stated in (13), the challenge is to find a motion trajectory $\mathcal{M}_i(t)$ that transfers the state of motion from $\mathbf{M}_i$ to $\mathbf{M}_i^{\text{trgt}}$ within the shortest possible time. For the current control cycle at T_i, only $\mathbf{M}_{i+1}$ is needed, because in the next control cycle, we might have completely new input values $\mathbf{W}_i$ due to an unforeseen event or switching action. Hence, only $\mathbf{M}_{i+1}$ is forwarded to the output. These values are then used as input values for lower level control. This leads to the interesting requirement that if the input values $\mathbf{M}_i^{\text{trgt}}$, $\mathbf{B}_i$, and $\vec{S}_i$ remain constant for $i \in \{0, \ldots, N\}$, and if the output values $\mathbf{M}_{i+1}$ are directly fed back as input values for the following control cycle (cf. dotted part of Fig. 3), then we have the following.

1) The value of the synchronization time must remain constant during the whole trajectory execution, i.e., $t_i^{\text{sync}} = \text{const}\ \forall\, i \in \{0, \ldots, N\}$. This fact is relevant for time synchronization such that $\vec{V}_i^{\text{trgt}}$, $\vec{A}_i^{\text{trgt}}$, and $\vec{J}_i^{\text{trgt}}$ are coinstantaneously reached in $\vec{P}_i^{\text{trgt}}$ at t_i^{sync}.
2) All trajectories $\mathcal{M}_u(t)$ with $u \in \{1, \ldots, N\}$ must exactly fit into the one of $\mathcal{M}_0(t)$.
3) Furthermore, any trajectory $\mathcal{M}_u(t)$ with $u \in \{1, \ldots, N\}$ must exactly fit into all previously-calculated motion trajectories $\mathcal{M}_v(t)$ with $v \in \{0, \ldots, u-1\}$.

The general algorithm for the calculation of $\mathcal{M}_i(t)$ is described in Section V, but prior to this, we classify different types and variants of OTG in the next section.

TABLE I
DIFFERENT TYPES OF OTG

$\forall\,(i,k)\,\in\,\mathbb{Z}\times\{1,\ldots,K\}$	${}_kV_i^{trgt} = 0\ \wedge$ ${}_kA_i^{trgt} = 0\ \wedge$ ${}_kJ_i^{trgt} = 0$	${}_kV_i^{trgt} \in \mathbb{R}\ \wedge$ ${}_kA_i^{trgt} = 0\ \wedge$ ${}_kJ_i^{trgt} = 0$	${}_kV_i^{trgt} \in \mathbb{R}\ \wedge$ ${}_kA_i^{trgt} \in \mathbb{R}\ \wedge$ ${}_kJ_i^{trgt} = 0$	${}_kV_i^{trgt} \in \mathbb{R}\ \wedge$ ${}_kA_i^{trgt} \in \mathbb{R}\ \wedge$ ${}_kJ_i^{trgt} \in \mathbb{R}$
${}_kA_i^{max} \in \mathbb{R}\ \wedge$ ${}_kJ_i^{max} = \infty\ \wedge$ ${}_kD_i^{max} = \infty$	**Type I** $\alpha = 5,\ \beta = 2$	**Type II** $\alpha = 6,\ \beta = 2$	—	—
${}_kA_i^{max} \in \mathbb{R}\ \wedge$ ${}_kJ_i^{max} \in \mathbb{R}\ \wedge$ ${}_kD_i^{max} = \infty$	**Type III** $\alpha = 7,\ \beta = 3$	**Type IV** $\alpha = 8,\ \beta = 3$	**Type V** $\alpha = 9,\ \beta = 3$	—
${}_kA_i^{max} \in \mathbb{R}\ \wedge$ ${}_kJ_i^{max} \in \mathbb{R}\ \wedge$ ${}_kD_i^{max} \in \mathbb{R}$	**Type VI** $\alpha = 9,\ \beta = 4$	**Type VII** $\alpha = 10,\ \beta = 4$	**Type VIII** $\alpha = 11,\ \beta = 4$	**Type IX** $\alpha = 12,\ \beta = 4$

The type no. define the set of available input parameters $\mathbf{B}_i$ and $\mathbf{M}_i^{\text{trgt}}$ [cf. (3), (4), and (16)].

To summarize this section briefly, we introduced a dedicated notation and described the task of OTG very generally without offering solutions. Besides this, the important terms of *time optimality* and *time synchronization* were clarified for the context of this paper.

IV. TYPES AND VARIANTS OF ONLINE TRAJECTORY GENERATION

This section defines different types and variants of OTG algorithms, i.e., the class of algorithms introduced by this paper. Depending on the type, the algorithmic complexity and the practical relevance differ strongly.

A. Types of OTG

In general, we can denote that the output values of an OTG algorithm are a function f of the input parameters, the OTG algorithm itself is *memoryless* (it has no memory)

$$\mathbf{M}_{i+1} = f(\mathbf{W}_i),\ \text{where}$$
$$f\colon \mathbb{R}^{\alpha K} \times \mathbb{B}^K \longrightarrow \mathbb{R}^{\beta K} \tag{16}$$

and $\mathbb{B} = \{0, 1\}$ is responsible for the selection vector $\vec{S}_i$. α and β are type-dependent integer values. Table I shows a summary of types of OTG. The OTG block of Fig. 3—if fully connected to all input parameters and output values—corresponds to Type IX. Depending on the type, not all input and output magnitudes are in use. Type VIII, for example, does not offer to specify $\vec{J}_i^{\,\text{trgt}}$. Fig. 2, for example, shows a Type VI trajectory. $\beta = 4$ means that the position progression is described by polynomials of up to fourth order, i.e., even the derivative of the jerk is limited ($\vec{D}_i^{\,\max} \in \mathbb{R}^K$). All further types are defined analogously. To finalize Table I, one may denote the trivial and irrelevant case with rectangular velocity profiles as Type 0 (with $\alpha = 3, \beta = 1$). Of course, it would also be possible to extend Table I by higher order trajectories (Type X, XI, etc.), but the complexity strongly increases with increasing type numbers, such that the development of these algorithms is hardly possible. For clarification, this paper presents a whole class of algorithms that is applicable to all types of OTG; the higher order types (Types VI and higher) are of theoretical character, but the Types I–V are highly relevant for sensor-based robot-motion control.

A very specific version of Type I was already suggested in [26] and works as well as Type II with unlimited jerks. These two types may offer sensor-integration possibilities for experimental purposes, for example, in research institutions, but due to the nonsmooth bang–bang or trapezoidal trajectory characteristics [27], they are not relevant for industrial practice, as has already been stated in [5]. To achieve long lifetimes for mechanical systems, a jerk limitation is required (Types III–V). Compared with Type III, Types IV and V additionally provide the possibility of specifying target-velocity vectors and/or target-acceleration vectors in space, which is important for the consideration of system dynamics. As a result, Type IV is the first type of OTG, which is relevant for professional usage.

B. Variants of OTG

Regarding different variants of OTG, we distinguish between constant and nonconstant kinematic constraint values and define two variants A and B as

A) $\mathbf{B}_i = \mathbf{const}\ \forall\, i \in \mathbb{Z}$
B) $\mathbf{B}_i \neq \mathbf{const}$.

The second, slightly more advanced variant with time-variant values of kinematic motion constraints will be introduced in a follow-up publication and is important for the integration of robot dynamics in order to consider state-dependent and time-variant values of $\vec{A}_i^{\max}$.

C. Positional Limits

The consideration of positional limits $({}_k\vec{P}_i^{\min}, {}_k\vec{P}_i^{\max})$ (e.g., joint limits or Cartesian work space limits) cannot be embedded in the OTG algorithm. In order to clarify this, Fig. 4 shows the cuboidal motion constraint space for the OTG Types III–V [cf. Table I and (3) and (12)] as well as the positional constraints ${}_kP_i^{\max}$ and ${}_kP_i^{\min}$ for one single DOF k at instant T_i. If we then consider the current state of motion for this DOF k as

$${}_k\vec{M}_i = ({}_kP_i, +\,{}_kV_i^{\max}, 0, 0) \tag{17}$$

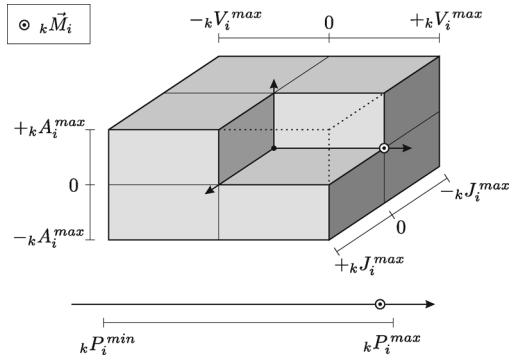

Fig. 4. (Top) Kinematic motion constraints ${}_k\vec{B}_i$ of 1-DOF k at instant T_i for the OTG Types III–V [cf. Table I and (3) and (12)], 3-D. (Bottom) Positional limits, 1-D. The current state of motion for DOF k, ${}_k\vec{M}_i$, is marked by the white circle with the black dot in both diagrams [cf. (17)].

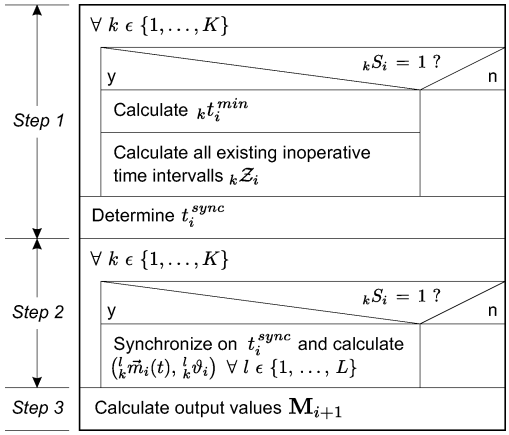

Fig. 5. Nassi–Shneiderman structogram of the general OTG algorithm.

where ${}_kP_i$ is very close to ${}_kP_i^{\max}$ (cf. Fig. 4), we have the choice to

1) either overshoot ${}_kP_i^{\max}$ and keep ${}_k\vec{M}_i$ within ${}_k\vec{B}_i$, or
2) exceed ${}_k\vec{B}_i$ and keep ${}_kP_i^{\max}$.

However, we cannot guarantee both (which is natural in this situation). The requirement we have for practical implementations is that we have to react to sensor events at unforeseen instants and in arbitrary states of motion. Hence, the responsibility of keeping ${}_kP_i$ within $[{}_kP_i^{\min}, {}_kP_i^{\max}]$ for all $k \in \{1,\ldots,K\}$ is transferred to the system (or user) *above* the layer of the OTG algorithm (e.g., a task or high-level motion planner). As also described in [14]–[18], it is the task of the OTG layer to move one, some, or all DOFs of a robotic system from an initial position $\vec{P}_i$ to a target position $\vec{P}_i^{\text{trgt}}$, and the system above is responsible for the (trivial) fact that

$$ {}_kP_i^{\text{trgt}} \in \left[{}_kP_i^{\min}, {}_kP_i^{\max}\right]\ \forall\ k \in \{1,\ldots,K\} \tag{18} $$

for the OTG Types I, III, and VI [cf. (15) and Table I). For other OTG types, (18) becomes extended, because case differentiations have to be made. This topic will be investigated in Section VII.

V. Online Trajectory Generator Algorithm—General Version

Sections III and IV introduced basics, requirements, and types of OTG. This section applies these foundations and describes the central OTG algorithm generally, such that it is applicable to all types and variants of OTG. To clarify its usage, Section VI details the algorithm concretely by means of Type IV and presents practical results.

Assuming the algorithm is called at a discrete time instant T_i; all types of OTG require the same three algorithmic steps:

Step 1: calculation of the minimum possible t_i^{sync};
Step 2: synchronization of all selected DOFs to t_i^{sync} and calculation of $\mathcal{M}_i$;
Step 3: calculation of $\mathbf{M}_{i+1}$ based on $\mathcal{M}_i$.

These steps are depicted in Fig. 5, which shows the overall structogram of the OTG algorithm. The following three sections describe each step in detail.

A. Step 1

This step is the most complex one, although it only computes the synchronization time t_i^{sync}, i.e., one scalar value. It is a function

$$ f:\ \mathbb{R}^{\alpha K} \times \mathbb{B}^K\ \longrightarrow\ \mathbb{R}. \tag{19} $$

As can be seen in Fig. 5, *Step 1* can be subdivided into three parts: the individual calculation of the minimum execution time ${}_kt_i^{\min}$ for each selected DOF k; the calculation of possibly existing inoperative time intervals ${}_k\mathcal{Z}_i$, in which a selected DOF cannot be synchronized; and finally, the determination of t_i^{sync}.

1) Minimum Execution Time: One of the key ideas of this algorithm is that there is a *finite* set of possible motion profiles of which one profile transfers one single selected DOF k from the initial state of motion ${}_k\vec{M}_i$ to its target state of motion ${}_k\vec{M}_i^{\text{trgt}}$ within the shortest possible time ${}_kt_i^{\min}$ (time optimally). This finite set for Step 1 is denoted by

$$ \mathcal{P}_{\text{Step1}} = \left\{{}^1\Psi^{\text{Step1}}, \ldots, {}^r\Psi^{\text{Step1}}, \ldots, {}^R\Psi^{\text{Step1}}\right\} \tag{20} $$

where R is the number of elements in $\mathcal{P}_{\text{Step1}}$ and depends on the type of OTG. A concrete profile is denoted by ${}^r\Psi^{\text{Step1}}$. What kinds of motion profiles are considered also depends on the used type of OTG:

Types I–II	:	Velocity profiles	$(\beta = 2)$
Types III–V	:	Acceleration profiles	$(\beta = 3)$
Types V–IX	:	Jerk profiles	$(\beta = 4)$.

It is the task of this substep to execute a function $f:\ \mathbb{R}^{\alpha} \longrightarrow \mathcal{P}_{\text{Step1}}$ for every single selected DOF in order to select the motion profile which leads to *the* time-optimal trajectory and which simultaneously determines ${}_kt_i^{\min}$ for each DOF k. Once *the* time-optimal profile ${}_k\Psi_i^{\text{Step1}}$ has been selected for 1-DOF k, a system of nonlinear equations can be set up and solved to calculate ${}_kt_i^{\min}$. The selection of ${}_k\Psi_i^{\text{Step1}}$ can be realized by decision trees, which are exemplarily presented in Section VI.[1]

[1]Alternatively, one could set up R systems of nonlinear equations, calculate all solutions, take all valid solutions for ${}_kt_i^{\min}$, and choose the minimum one. However, this procedure is computationally too expensive, especially if low values for T^{cycle} are desired.

Each motion profile ${}^{r}\Psi^{\mathrm{Step1}}$ with $r \in \{1, \ldots, R\}$ leads to a system of nonlinear equations, and each system is solvable for a certain domain ${}^{r}\mathcal{D}_{\mathrm{Step1}}$ with

$$ {}^{r}\mathcal{D}_{\mathrm{Step1}} \subset \mathbb{R}^{\alpha}. \tag{21}$$

For the decision tree implementing (19), it is *absolutely essential* that

$$ \bigcup_{r=1}^{R} {}^{r}\mathcal{D}_{\mathrm{Step1}} \equiv \mathbb{R}^{\alpha} \tag{22}$$

holds. If this is not the case, the tree is erroneous, and the algorithm will not work for certain input parameters, which would be unacceptable for its practical application. In the work of Broquère *et al.* [16] as well as in the contribution of Liu [17], which suggest similar approaches for 1-DOF systems, (22) does not hold, and hence, these approaches are not applicable in general, but only for some (practically hardly relevant) special cases.

Not even the theoretic literature provides a concept to let only 1-DOF react instantaneously to unforeseen events with jerk-limited trajectories. In this paper, we implicitly introduce both, the 1-D case as well as the multidimensional case. If only 1-DOF is considered, the solution of the system of equations for *the* time-optimal motion profile ${}_{k}\Psi_{i}^{\mathrm{Step1}}$ contains all required trajectory parameters to set up $\mathcal{M}_i$. The multidimensional case, which also requires synchronization, is introduced in the following.

2) Inoperative Time Intervals: After the minimum execution time ${}_{k}t_{i}^{\min}$ has been calculated, we have to check whether it is possible to execute the trajectory for this DOF k within *any* time $t >_k t_i^{\min}$. If this is the case, no inoperative time intervals ${}_{k}\mathcal{Z}_i = \{\}$ are existent; depending on the type, ${}_{k}\mathcal{Z}_i$ may contain up to $Z = 3$ time intervals in which a selected DOF k cannot be synchronized. Referring to Table I, this can be expressed by

$$ Z = \alpha - 2\beta - 1 \tag{23}$$

such that the following values of Z appear:

$$\begin{aligned} \textit{Types I, III, VI} &: Z = 0 \\ \textit{Types II, IV, VII} &: Z = 1 \\ \textit{Types V, VIII} &: Z = 2 \\ \textit{Type IX} &: Z = 3. \end{aligned}$$

If only a target position vector $\vec{P}_i^{\mathrm{trgt}}$ is given (Types I, III, and VI), the target state can, of course, be reached at any time $t \geq t_i^{\min}$. For every further target-motion state vector, one inoperative time interval may occur, i.e., if an additional target velocity vector $\vec{V}_i^{\mathrm{trgt}}$ is specified (Types II, IV, and VII) one inoperative time interval may be present (cf. Fig. 6); a target acceleration vector $\vec{A}_i^{\mathrm{trgt}}$ (Types V and VIII) leads to up to two inoperative time intervals, etc. The inoperative time intervals of one single DOF may overlap such that two or more intervals affiliate to one interval.

The single elements of ${}_{k}\mathcal{Z}_i$ are then denoted by

$$ {}_{k}^{z}\zeta_i = \left[{}_{k}^{z}t_i^{\mathrm{begin}}, {}_{k}^{z}t_i^{\mathrm{end}}\right], \text{ with } z \in \{1, \ldots, Z\}. \tag{24}$$

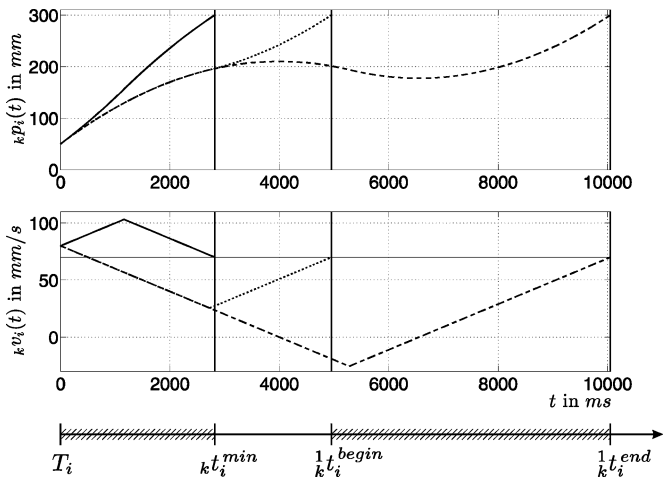

Fig. 6. Example of an inoperative time interval for one translational DOF k calculated by a Type II OTG. Assuming ${}_{k}A_i^{\max} = 20\ \mathrm{mm/s^2}$, ${}_{k}\vec{M}_i = (50\ \mathrm{mm}, 80\ \mathrm{mm/s}, 0, 0)$, and ${}_{k}\vec{M}_i^{\mathrm{trgt}} = (300\ \mathrm{mm}, 70\ \mathrm{mm/s}, 0, 0)$, *the* time-optimal case would take ${}_{k}t_i^{\min} = 2820$ ms. If other DOFs require more time, this DOF k can also reach ${}_{k}\vec{M}_i^{\mathrm{trgt}}$ after ${}_{k}^{1}t_i^{\mathrm{begin}} = 4950$ ms, but it is not possible to transfer DOF k to ${}_{k}\vec{M}_i^{\mathrm{trgt}}$ in a time ${}_{k}^{1}t_i^{\mathrm{begin}} < t <_k^1 t_i^{\mathrm{end}}$ with ${}_{k}^{1}t_i^{\mathrm{end}} = 10,050$ ms. For all times $t \geq_k^1 t_i^{\mathrm{end}}$, the synchronized execution of DOF k is possible again. As a result for DOF k, there is an inoperative time interval of ${}_{k}^{1}\zeta_i = [4950\ \mathrm{ms}, 10,050\ \mathrm{ms}]$.

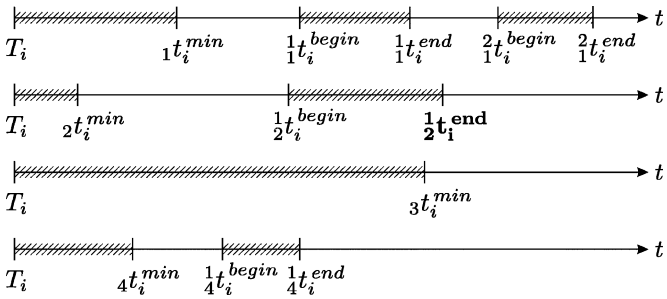

Fig. 7. Example of the determination of t_i^{sync} for $K = 4$ DOFs. Here, $t_i^{\mathrm{sync}} = {}_{2}^{1}t_i^{\mathrm{end}}$.

To explain the origin of these inoperative time intervals, Fig. 6 illustrates a simple Type II trajectory with simple bang–bang characteristics and one inoperative time interval ${}_{k}^{1}\zeta_i$. It is absolutely essential that the OTG algorithm be able to provide a solution for *any* set of input values $\mathbf{W}_i$. If this could not be guaranteed, the concept would not be complete, would be unsafe, and, thus, would be practically irrelevant. As a result, $2\,Z + 1 = 2\,\alpha - 4\,\beta - 1$ decision trees are required for Step 1, i.e., one tree for the calculation of t_i^{sync} and two Z trees for the calculation of ${}_{k}^{z}\zeta_i \ \forall\ z \in \{1, \ldots, Z\}$.

3) Determining t_i^{sync}: After the minimum execution times for all selected DOFs and all existing inoperative time intervals have been calculated, t_i^{sync} can be determined easily. First, the greatest time of all minimum execution times is taken. t_i^{sync} cannot be less than this value. As second step, we have to regard that t_i^{sync} must not be part of any inoperative time interval ${}_{k}^{z}\zeta_i$ with $(z, k) \in \{1, \ldots, Z\} \times \{1, \ldots, K\}$. Fig. 7 illustrates an example with four DOFs, where t_i^{sync} is determined.

B. Step 2

The purpose of this step is to set up all parameters of the trajectory $\mathcal{M}_i$ calculated at T_i [cf. (6)–(9)]. $\mathbf{W}_i$ and t_i^{sync} act as input parameters for this second step. Due to Step 1, we already know that all selected DOFs are able to reach their desired target state of motion $\mathbf{M}_i^{\mathrm{trgt}}$ exactly at t_i^{sync} without exceeding the kinematic-motion constraints $\mathbf{B}_i$. Here, a similar procedure as in the first part of Step 1 is applied. The key idea that there is a *finite* set of motion profiles to set up the final motion trajectories is employed again. This set $\mathcal{P}_{\mathrm{Step2}}$ differs from the one of Step 1: $\mathcal{P}_{\mathrm{Step1}}$

$$\mathcal{P}_{\mathrm{Step2}} = \left\{ {}^{1}\Psi^{\mathrm{Step2}}, \ldots, {}^{s}\Psi^{\mathrm{Step2}}, \ldots, {}^{S}\Psi^{\mathrm{Step2}} \right\} \tag{25}$$

where S denotes the number of elements in $\mathcal{P}_{\mathrm{Step2}}$ and depends on the type of OTG. According to Table I, a function $f: \mathbb{R}^{\alpha+1} \longrightarrow \mathcal{P}_{\mathrm{Step2}}$ is required, which determines a motion profile for each selected DOF k. This function is again represented by a decision tree, which will be briefly described in the next section. After the correct motion profile ${}_k\Psi_i^{\mathrm{Step2}}$ is chosen for a DOF k, another system of nonlinear equations can be set up.

According to Table I, a decision tree that acts as function

$$f: \left((\mathbb{R}^{\alpha+1}) \backslash \mathcal{H} \right) \longrightarrow \mathcal{P}_{\mathrm{Step2}} \tag{26}$$

is required, which determines a motion profile ${}^{s}\Psi^{\mathrm{Step2}}$ for each selected DOF k. The meaning of $\mathcal{H}$, i.e., holes in the domain space, will be described next. Compared with Step 1, Step 2 always requires only one decision tree, which is executed once for every selected DOF and control cycle. However, compared with Step 1, there are two significant differences.

1) While in Step 1, more than one system of equations may achieve a valid solution, and in Step 2, only one element of $\mathcal{P}_{\mathrm{Step2}}$ leads to a solvable system of equations. This fact is rooted in the nature of the Step 2 motion profiles and leads to a deterministic behavior, i.e., there can always be only one possible trajectory, which transfers a selected DOF k from ${}_k\vec{M}_i$ to ${}_k\vec{M}_i^{\mathrm{trgt}}$ in t_i^{sync}.
2) If we denote the input domain of the system of equations that corresponds to the motion profile ${}^{s}\Psi^{\mathrm{Step2}}$ as

$${}^{s}\mathcal{D}_{\mathrm{Step2}} \subset \mathbb{R}^{\alpha+1} \tag{27}$$

we do *not* have such an equivalence as described by (22) for Step 1

$$\bigcup_{s=1}^{S} {}^{s}\mathcal{D}_{\mathrm{Step2}} \not\equiv \mathbb{R}^{\alpha+1}. \tag{28}$$

Due to the inoperative time intervals ${}_k\mathcal{Z}_i$ for each selected DOF k, the $(\alpha+1)$-dimensional space contains *holes*, in which none of the systems of equations that correspond to the motion profiles of $\mathcal{P}_{\mathrm{Step2}}$ is solvable and in which the decision tree of (26) does not deliver a solution. These holes in the $(\alpha+1)$-dimensional space are represented by the set

$$\mathcal{H} \subset \mathbb{R}^{\alpha+1}. \tag{29}$$

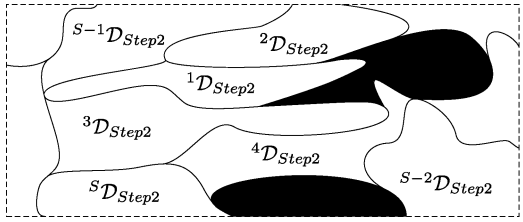

Fig. 8. Two-dimensional illustration of (30). Union of all input domains ${}^{s}\mathcal{D}_{\mathrm{Step2}}$ of the systems of equations for the motion profiles of Step 2: $\mathcal{P}_{\mathrm{Step2}}$. The black areas represent $\mathcal{H}$.

As a result, the union of all input domains of the systems of equations for the Step 2 motion profiles $\mathcal{P}_{\mathrm{Step2}}$ can be described in (30); this is illustrated by Fig. 8. We can also logically conclude here that the set of holes $\mathcal{H}$ is disjunctive with the union of all input domains

$$\bigcup_{s=1}^{S} {}^{s}\mathcal{D}_{\mathrm{Step2}} = (\mathbb{R}^{\alpha+1}) \backslash \mathcal{H}. \tag{30}$$

Furthermore, it is an important property that the union of pairwise intersections of all Step 2 input domains

$$\mathcal{S} = \bigcup_{s=1}^{S} \left\{ {}^{s}\mathcal{D}_{\mathrm{Step2}} \cap {}^{u}\mathcal{D}_{\mathrm{Step2}} \; \forall \, u \in \{1, \ldots, S\} \, \middle| \, u \neq s \right\} \tag{31}$$

constitutes α-dimensional hyperplanes in the $(\alpha+1)$-dimensional space. These hyperplanes are described in detail by the decision tree of (26). The fact that $\mathcal{S}$ represents only hyperplanes in the input space of Step 2 is essential for the deterministic behavior of the OTG algorithm, because this way there is only one possible motion profile for any set of input values. If the $(\alpha+1)$ input values for a selected DOF k are an element of $\mathcal{S}$, two or even more motion profiles will lead to a solvable system of equations, but the solution will be *exactly* the same for all profiles. This fact emphasizes the *overall consistency* of the proposed concept.

Let us now answer the following question: How can we describe the set of holes $\mathcal{H}$? This is an important question because we have to prevent from entering such a hole. Entering a hole would lead to an unsolvable problem, and no output values $\mathbf{M}_{i+1}$ could be calculated.

The $(\alpha+1)$-dimensional space for one DOF k is spanned by the first α elements of ${}_k\vec{W}_i$ and by the synchronization time t_i^{sync}. The values of ${}_k\vec{W}_i$ can be arbitrary [cf. (14)]. Hence, the only parameter that can be responsible for $\mathcal{H}$ is t_i^{sync}, which was determined in Step 1. As described there, we need $2\,Z$ decision trees to determine the limits of all inoperative time intervals ${}_k^z\zeta_i$ with $(z,k) \in \{1, \ldots, Z\} \times \{1, \ldots, K\}$. These $2\,Z$ decision trees exactly describe the set of holes $\mathcal{H}$ because we mask the codomain of t_i^{sync} and, thus, the $(\alpha+1)$-dimensional input space.

An important insight at this point is that all $(2\,Z+1)$ decision trees of Step 1 and the decision tree of Step 2 must *exactly* fit

each other. It is comparable with a multidimensional puzzle game whose parts have to fit each other exactly, and if this were not the case, the game would not come to an end. The same holds for the OTG algorithm: If the $(2\,Z+1)$ Step 1 decision trees do not fit each other, the algorithm will be erroneous, and there will exist input values $\mathbf{W}_i$, to which no output values can be calculated.

The resulting solution of the nonlinear system of equations enables us to set up all L trajectory segments ${}_k^l\vec{m}_i\ \forall\, l \in \{1,\ldots,L\}$ as well as all corresponding time intervals ${}_k^l\vartheta_i\ \forall\, l \in \{1,\ldots,L\}$ [cf. (5), (6), and (8)]. If this is done for all selected DOFs, $\mathcal{M}_i$ is calculated for this time step T_i [cf. (9)].

C. Step 3

This third step of the OTG algorithm is trivial. After the complete trajectory for one time step T_i, $\mathcal{M}_i$, has been calculated in Step 2, we only have to find the valid time interval ${}_k^{\hat{l}}\vartheta_i$ with $\hat{l} \in \{1,\ldots,L\}$ such that

$$ {}_k^{\hat{l}-1}t_i \leq (T_i + T^{\mathrm{cycle}}) \leq {}_k^{\hat{l}}t_i \tag{32}$$

is satisfied [cf. (8) and (9)]. The output values $\mathbf{M}_{i+1}$can finally be calculated by

$$ \mathbf{M}_{i+1} = {}^{\hat{l}}\mathbf{m}_i(T_i + T^{\mathrm{cycle}}). \tag{33}$$

D. Final Remarks on the General OTG Algorithm

This section introduced the general OTG algorithm for all types and variants of OTG. Its complexity strongly depends on the implemented type; the higher the roman number of the OTG type, the higher its algorithmic complexity. The measure of complexity is influenced, in particular, by two properties: 1) the size of the decision trees in Steps 1 and 2 and 2) the solvability of the systems of equations, which are generated from the elements in $\mathcal{P}_{\mathrm{Step1}}$ and $\mathcal{P}_{\mathrm{Step2}}$. Over the past few years, implementations of Types I–IV were realized by the authors, where Type IV is the first one, which is relevant for practical usage. This type includes Types I–III (cf. Table I) and will be concretized in the next section.

VI. Type IV Online Trajectory Generation

Compared to the general OTG algorithm in Section V, this section exemplarily introduces one concrete type of OTG: Type IV-A. This leads us to the following general parameters for the algorithm: $\alpha = 8$, $\beta = 3$, $\vec{A}_i^{\mathrm{trgt}} = \vec{0}\ \wedge\ \vec{J}_i^{\mathrm{trgt}} = \vec{0}\ \forall\ i \in \mathbb{Z}$; in addition, $\vec{D}_i^{\max}$ may contain infinite values for any $i \in \mathbb{Z}$. In simple words, this type achieves kinematically time-optimal and time-synchronized trajectories, whose velocity, acceleration, and jerk values are limited, and the specification of target velocity vectors $\vec{V}_i^{\mathrm{trgt}}$, which are reached in $\vec{P}_i^{\mathrm{trgt}}$, is possible. Furthermore, variant A leads to constant values for $\mathbf{B}_i$, i.e., $\vec{V}_i^{\max} = \mathrm{const}\ \wedge\ \vec{A}_i^{\max} = \mathrm{const}\ \wedge\ \vec{J}_i^{\max} = \mathrm{const}\ \forall\ i \in \mathbb{Z}$.

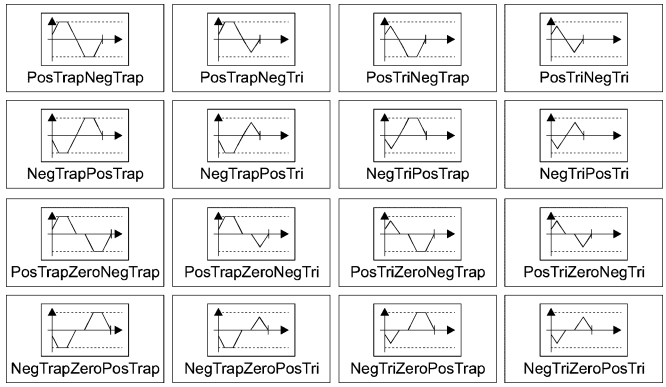

Fig. 9. Subset of the acceleration profile set $\mathcal{P}_{\mathrm{Step1}}$ of Type IV-A. The dotted horizontal line indicates the maximum acceleration values. These profiles are required to calculate ${}_k t_i^{\min}$ as well as all limits of existing inoperative time intervals ${}_k\mathcal{Z}_i$ for each single selected DOF k.

A. Step 1

The goal of this step is to calculate the synchronization time t_i^{sync}.

1) Minimum Execution Time: We first calculate the minimum possible execution times for each individual DOF k, ${}_k t_i^{\min}$. As described in the previous section, we need to specify an acceleration profile, which enables us to set up a system of equations and to calculate the desired time value. Fig. 9 shows a subset of all possible acceleration profiles $\mathcal{P}_{\mathrm{Step1}}$. Compared with triangle (Tri) profiles, trapezoid (Trap) profiles always reach the maximum acceleration value for the respective DOF ${}_k A_i^{\max}$. All profiles that do not contain a *zero* phase do not reach the maximum velocity for the respective DOF ${}_k V_i^{\max}$.

In the following substep, we have to answer the question: How can we determine *the* time-optimal acceleration profile ${}_k\Psi_i^{\mathrm{Step1}}$, i.e., the element of $\mathcal{P}_{\mathrm{Step1}}$ that leads to ${}_k t_i^{\min}$ for DOF k? Due to lack of space, we can only show a small cutout of the Step 1 decision tree in Fig. 10, which exemplarily explains the functionality of decision trees. The tree actually acts as function $f\colon \mathbb{R}^8 \longrightarrow \mathcal{P}_{\mathrm{Step1}}$ and determines *the* time-optimal acceleration profile for 1-DOF k. Decision *1A.001* checks whether the current acceleration value ${}_k A_i$ is positive or negative. The left branch is designed for positive values of ${}_k A_i$. Decision *1A.002* calculates the velocity value if we would bring ${}_k A_i$ down to zero (by applying ${}_k J_i^{\max}$). If this value is less than ${}_k V_i^{\mathrm{trgt}}$, the left branch will be taken. Hence, we already know that $-{}_k V_i^{\max} \leq {}_k V_i \leq {}_k V_i^{\mathrm{trgt}}$. It is our aim to reach ${}_k V_i^{\mathrm{trgt}}$, and Decision *1A.003* checks whether a triangle or a trapezoid profile is required for this. If a trapezoid profile is required, Decision *1A.004* will check whether the position would be less or greater than the target position ${}_k P_i^{\mathrm{trgt}}$. If we are still in front of ${}_k P_i^{\mathrm{trgt}}$, we will need to increase the trapezoid acceleration profile, i.e., we will definitely need a *PosTrap...Neg...* profile. Decision *1A.005* then checks the velocity value that would be achieved if we would accelerate to $+{}_k V_i^{\max}$ by applying a trapezoid profile and subsequently decelerate with a negative triangular profile, which exactly reaches $-{}_k A_i^{\max}$. If the achieved velocity value is less than ${}_k V_i^{\mathrm{trgt}}$, we will definitely need a negative triangle profile as second part of the composed acceleration profile. Now, we only

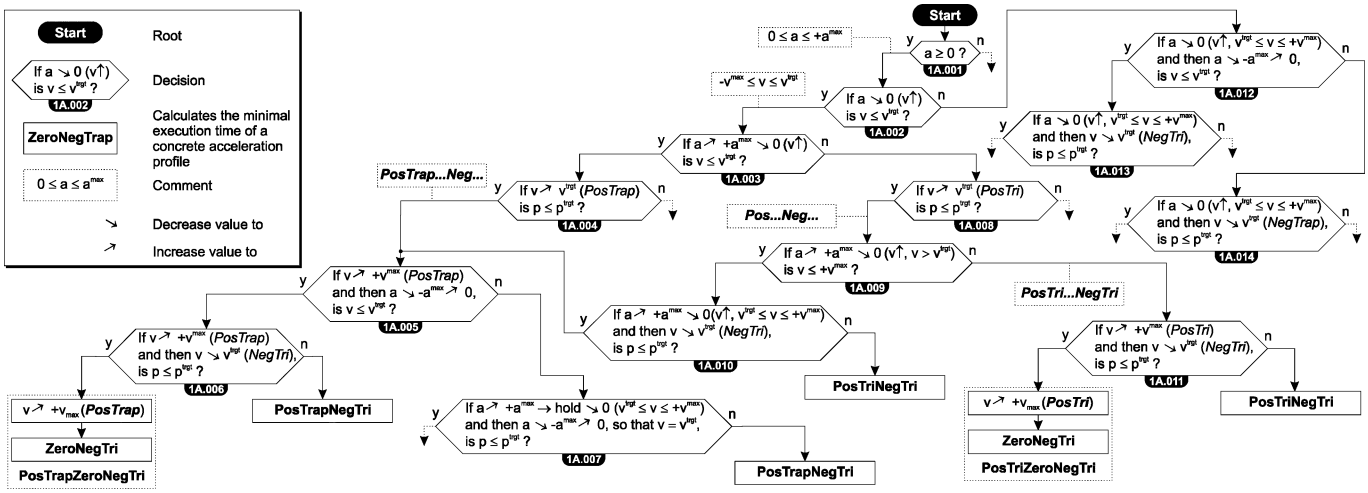

Fig. 10. Cutout of the Type IV-A decision tree for Step 1 (Part A, calculate ${}_k t_i^{\min}$) to determine an acceleration profile ${}_k \Psi_i^{\mathrm{Step1}}$ that leads to the minimum-time solution for one DOF k.

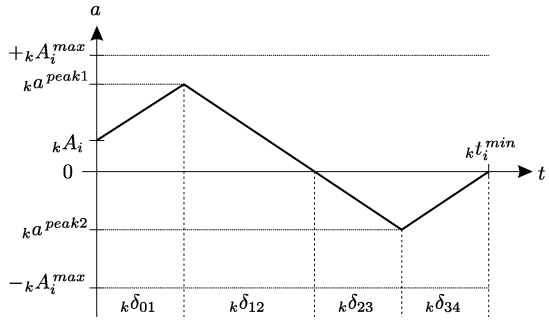

Fig. 11. *PosTriNegTri* profile with all relevant variables such that a system of equations can be set up and solved in order to calculate ${}_k t_i^{\min}$ or any beginning or end of an inoperative time interval.

have to check whether ${}_k V_i^{\max}$ can be reached in order to obtain ${}_k V_i^{\mathrm{trgt}}$ in ${}_k P_i^{\mathrm{trgt}}$ (Decision *1A.006*). The further decisions work analogously to the described ones. For the development of this kind of decision trees, it is essential that a tree covers the whole input domain (here: $\mathbb{R}^8$) such that the algorithm can work with arbitrary input values $\mathbf{W}_i$.

Once we know the correct acceleration profile ${}_k \Psi_i^{\mathrm{Step1}}$ for 1-DOF k, we can set up a system of equations that corresponds to this profile. Let us exemplarily set up the system for the *PosTriNegTri* profile (see the top right of Fig. 9). Fig. 11 shows this profile and all relevant variables such that we can set up a system of equations to calculate ${}_k t_i^{\min}$. The symbol δ indicates time differences, ν velocity differences, and ξ position differences. The system of equations for 1-DOF k can be directly derived from Fig. 11 and is given by (34)–(50). In this full-length way, we achieve 17 equations with 17 unknown variables ${}_k t_i^{\min}$, ${}_k\delta_{01}$, ${}_k\delta_{12}$, ${}_k\delta_{23}$, ${}_k\delta_{34}$, ${}_k a^{\mathrm{peak1}}$, ${}_k a^{\mathrm{peak2}}$, ${}_k\nu$, ${}_k\nu_{01}$, ${}_k\nu_{12}$, ${}_k\nu_{23}$, ${}_k\nu_{34}$, ${}_k\xi$, ${}_k\xi_{01}$, ${}_k\xi_{12}$, ${}_k\xi_{23}$, and ${}_k\xi_{34}$:

$$ {}_k t_i^{\min} = {}_k\delta_{01} + {}_k\delta_{12} + {}_k\delta_{23} + {}_k\delta_{34} \tag{34} $$

$$ {}_k\delta_{01} = \frac{({}_k a^{\mathrm{peak1}} - {}_k A_i)}{{}_k J_i^{\max}} \tag{35} $$

$$ {}_k\delta_{12} = \frac{{}_k a^{\mathrm{peak1}}}{{}_k J_i^{\max}} \tag{36} $$

$$ {}_k\delta_{23} = -\frac{{}_k a^{\mathrm{peak2}}}{{}_k J_i^{\max}} \tag{37} $$

$$ {}_k\delta_{34} = -\frac{{}_k a^{\mathrm{peak2}}}{{}_k J_i^{\max}} \tag{38} $$

$$ {}_k\nu = {}_k V_i^{\mathrm{trgt}} - {}_k V_i \tag{39} $$

$$ {}_k\nu = {}_k\nu_{01} + {}_k\nu_{12} + {}_k\nu_{23} + {}_k\nu_{34} \tag{40} $$

$$ {}_k\nu_{01} = {}_k\delta_{01}\ {}_k A_i + \frac{1}{2}\ {}_k\delta_{01}({}_k a^{\mathrm{peak1}} - {}_k A_i) \tag{41} $$

$$ {}_k\nu_{12} = \frac{1}{2}\ {}_k\delta_{12}\ {}_k a^{\mathrm{peak1}} \tag{42} $$

$$ {}_k\nu_{23} = \frac{1}{2}\ {}_k\delta_{23}\ {}_k a^{\mathrm{peak2}} \tag{43} $$

$$ {}_k\nu_{34} = \frac{1}{2}\ {}_k\delta_{34}\ {}_k a^{\mathrm{peak2}} \tag{44} $$

$$ {}_k\xi = {}_k P_i^{\mathrm{trgt}} - {}_k P_i \tag{45} $$

$$ {}_k\xi = {}_k\xi_{01} + {}_k\xi_{12} + {}_k\xi_{23} + {}_k\xi_{34} \tag{46} $$

$$ {}_k\xi_{01} = {}_k V_i\ {}_k\delta_{01} + \frac{1}{2}\ {}_k A_i({}_k\delta_{01})^2 + \frac{1}{6}\ {}_k J_i^{\max}({}_k\delta_{01})^3 \tag{47} $$

$$ {}_k\xi_{12} = ({}_k V_i + {}_k\nu_{01})\ {}_k\delta_{12} + \frac{1}{2}\ {}_k a^{\mathrm{peak1}}({}_k\delta_{12})^2 - \frac{1}{6}\ {}_k J_i^{\max}\ ({}_k\delta_{12})^3 \tag{48} $$

$$ {}_k\xi_{23} = ({}_k V_i + {}_k\nu_{01} + {}_k\nu_{12})\ {}_k\delta_{23} - \frac{1}{6}\ {}_k J_i^{\max}({}_k\delta_{23})^3 \tag{49} $$

$$ {}_k\xi_{34} = \left({}_k V_i^{\mathrm{trgt}} - {}_k\nu_{34}\right)\ {}_k\delta_{34} + \frac{1}{2}\ {}_k a^{\mathrm{peak2}}({}_k\delta_{34})^2 + \frac{1}{6}\ {}_k J_i^{\max}({}_k\delta_{34})^3. \tag{50} $$

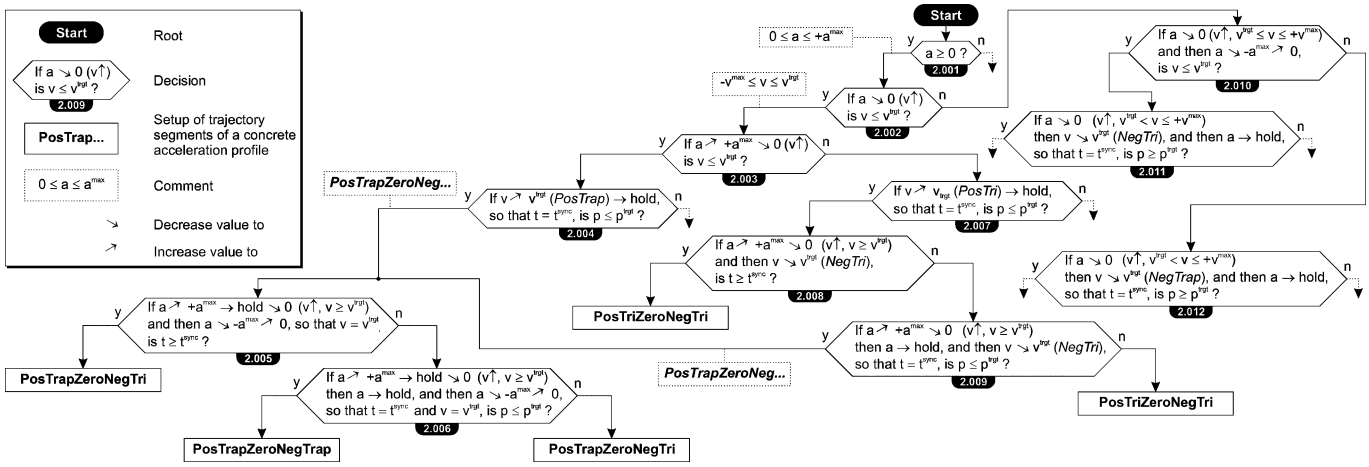

Fig. 12. Cutout of the Type IV-A decision tree for Step 2 to determine an acceleration profile ${}_k\Psi_i^{\mathrm{Step2}}$ for 1-DOF k.

The domains of the systems of equations for the elements of $\mathcal{P}_{\mathrm{Step1}}$ are disjunctive. Although these systems of equations are nonlinear for all elements of the set $\mathcal{P}_{\mathrm{Step1}}$, we are able to find analytical solutions by employing computer algebra programs; this would be the straightforward way. However, these expressions become very large, and problems with numerical stability appear. To bypass these problems, we can use efficient and real-time capable numerical methods as, for example, described in [28]. In particular, the methods of Anderson and Björck [29] and King [30], as well as simple bisection methods are applied here to calculate ${}_k t_i^{\min}$.

For higher types of OTG algorithms (Type V and higher), the number of equations increases, but their numerical solution can be done in the same way [28]. Of course, the complexity increases too, and thus, the numerical solution of these equations becomes computationally more expensive.

2) Inoperative Time Intervals: Corresponding to Section V-A.2, we have to look for up to one inoperative time interval. If such an interval exists for 1-DOF k, we have to calculate ${}_k^1 t_i^{\mathrm{begin}}$ and ${}_k^1 t_i^{\mathrm{end}}$ to determine ${}_k^1\zeta_i$, the only possible element of ${}_k\mathcal{Z}_i$ [cf. (24)]. For this purpose, we need $2\,Z = 2\,\alpha - 4\,\beta = 2$ further decision trees, $1B$ and $1C$, that work very similar to the one of Fig. 10. Instead of trying to reach ${}_k P_i^{\mathrm{trgt}}$ and ${}_k V_i^{\mathrm{trgt}}$ as fast as possible, the first decision tree ($1B$) tries to reach the target state of motion as slow as possible and without any turning back on the way to ${}_k P_i^{\mathrm{trgt}}$ (cf. Fig. 6). As a result, we obtain a further acceleration profile ${}_k\hat{\Psi}_i^{\mathrm{Step1}} \in \mathcal{P}_{\mathrm{Step1}}$, on whose base we can set up a further system of equations in the same way we did before (34)–(50), but here, we calculate not ${}_k t_i^{\min}$ but ${}_k^1 t_i^{\mathrm{begin}}$. If there are two possible solutions for ${}_k^1 t_i^{\mathrm{begin}}$, we always take the minimum one, because the maximum would already be ${}_k^1 t_i^{\mathrm{end}}$.

${}_k^1 t_i^{\mathrm{end}}$ is calculated in a very similar way. The third decision tree of Step 1 ($1C$) selects the acceleration profile ${}_k\tilde{\Psi}_i^{\mathrm{Step1}} \in \mathcal{P}_{\mathrm{Step1}}$, which determines the minimum possible time after a turning back on the way to ${}_k P_i^{\mathrm{trgt}}$. Similar to the calculation of ${}_k t_i^{\min}$ and ${}_k^1 t_i^{\mathrm{begin}}$, we set up a third system of equations, whose solution leads to ${}_k^1 t_i^{\mathrm{end}}$. In the case of two valid solutions, we take the maximum one (the other one would lead to ${}_k^1 t_i^{\mathrm{begin}}$). Of course, ${}_k t_i^{\min} \leq {}_k^1 t_i^{\mathrm{begin}} \leq {}_k^1 t_i^{\mathrm{end}}$ always holds for any DOF k (cf. Figs. 6 and 7).

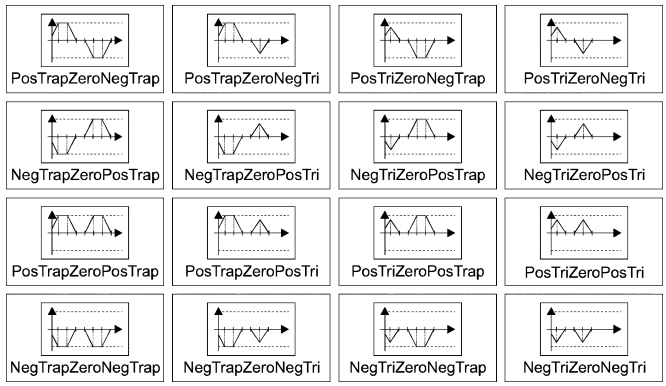

Fig. 13. Subset of the acceleration profile set $\mathcal{P}_{\mathrm{Step2}}$ of Type IV-A. Step 2 sets up all polynomial coefficients of ${}^l\mathbf{m}_i(t)$ as well the time intervals ${}^l\mathcal{V}_i$ with $l \in \{1, \ldots, L\}$, in which the respective polynomials are valid [cf. (8)].

3) Determining t_i^{sync}: The determination of minimum possible value of t_i^{sync} works in the same way as generally described in Section V-A3.

B. Step 2

In accordance with Section V-B, Step 2 calculates the coefficients of all sets of polynomials ${}^l\mathbf{m}_i(t)$ [cf. (6)]. That means, we need an acceleration profile ${}_k\Psi_i^{\mathrm{Step2}}$ for each single selected DOF k that enables the desired time synchronism, i.e., all selected DOFs reach their ${}_k P_i^{\mathrm{trgt}}$ and ${}_k V_i^{\mathrm{trgt}}$ at t_i^{sync}. Besides the eight input values of $\mathbf{W}_i$ for each DOF k, t_i^{sync} is the ninth input value for a further decision tree (see Fig. 12) that acts as function $f\colon \mathbb{R}^9 \longrightarrow \mathcal{P}_{\mathrm{Step2}}$. Fig. 13 shows a subset of $\mathcal{P}_{\mathrm{Step2}}$, which can be parameterized in order to calculate the final trajectory for a selected DOF k.

Fig. 12 shows a small cutout of the Step 2 decision tree and was drawn analogously to Fig. 10. As Decision *1A.001*

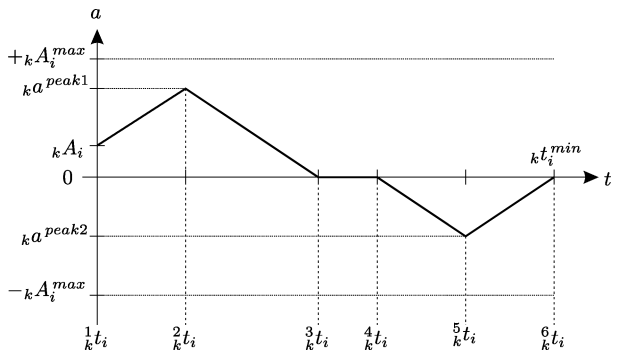

Fig. 14. *PosTriZeroNegTri* profile with all relevant variables such that a system of equations can be set up and solved in order to calculate ${}_k\mathcal{M}_i$ for 1-DOF k.

does, Decision *2.001* checks whether the current acceleration value ${}_kA_i$ is positive or negative. We assume to take the left branch, and let Decision *2.002* check whether the velocity value would be greater or less then ${}_kv_i^{\mathrm{trgt}}$ after $-{}_kJ_i^{\max}$ has been applied to decrease the acceleration value to zero. Decision *2.003* checks whether ${}_kA_i^{\max}$ must be applied to reach ${}_kv_i^{\mathrm{trgt}}$, i.e., we find out whether a positive triangle or trapezoidal acceleration profile would lead us to ${}_kv_i^{\mathrm{trgt}}$. We assume that a trapezoidal profile is required. Decision *2.004* inspects whether the resulting position value of DOF k at t_i^{sync} is greater or less than ${}_kp_i^{\mathrm{trgt}}$ if ${}_kv_i^{\mathrm{trgt}}$ would be reached as soon as possible by applying a trapezoidal acceleration profile (*PosTrap*). If the resulting value is less, we know that we will have to increase the hold time of the trapezoidal acceleration profile, i.e., a *PosTrapZeroNeg*... profile will be required. Before we decide whether ${}_k\Psi_i^{\mathrm{Step2}} = \textit{PosTrapZeroNegTri}$ or ${}_k\Psi_i^{\mathrm{Step2}} = \textit{PosTrapZeroNegTrap}$ is correct, Decision *2.005* verifies whether we would have enough time to apply a positive trapezoidal acceleration profile directly followed by a negative triangle-shaped profile that exactly touches $-{}_kA_i^{\max}$ such that ${}_kv_i^{\mathrm{trgt}}$ is finally reached. If we do not have enough time for this acceleration progression, ${}_k\Psi_i^{\mathrm{Step2}} = \textit{PosTrapZeroNegTri}$ will be the solution. If there is enough time for the acceleration behavior checked by Decision *2.005*, Decision *2.006* will check the boundary case for the profiles *PosTrapZeroNegTrap* and *PosTrapZeroNegTri* and finally determine the acceleration profile. The decisions from *2.007* to *2.012* work analogously.

Analogous to the system of equations given by (34)–(50), we can set up a further system of equations for each ${}_k\Psi_i^{\mathrm{Step2}} \in \mathcal{P}_{\mathrm{Step2}}$ (cf. Fig. 13), whose solution delivers all required parameters for all L trajectory segments ${}^l\mathbf{m}_i \;\forall\; l \in \{1,\ldots,L\}$ [cf. (7)] as well as all time intervals ${}^l\mathcal{V}_i$ [cf. (8)]. This procedure will be exemplarily explained by means of the profile ${}^s\Psi^{\mathrm{Step2}} = \textit{PosTriZeroNegTri}$, as shown in Fig. 14 (cf. top right element of Fig. 13). In a straightforward way, we can set up a system of 14 equations for each selected DOF k at time instant T_i

$$ {}^2_kt_i - T_i = \frac{({}_ka^{\mathrm{peak1}} - {}_kA_i)}{{}_kJ_i^{\max}} \tag{51} $$

$$ {}^3_kt_i - {}^2_kt_i = \frac{{}_ka^{\mathrm{peak1}}}{{}_kJ_i^{\max}} \tag{52} $$

$$ {}^5_kt_i - {}^4_kt_i = -\frac{{}_ka^{\mathrm{peak2}}}{{}_kJ_i^{\max}} \tag{53} $$

$$ t_i^{\mathrm{sync}} - {}^5_kt_i = -\frac{{}_ka^{\mathrm{peak2}}}{{}_kJ_i^{\max}} \tag{54} $$

$$ {}^2_kv_i - V_i = \frac{1}{2}\left({}^2_kt_i - T_i\right)\left({}_kA_i + {}_ka^{\mathrm{peak1}}\right) \tag{55} $$

$$ {}^3_kv_i - {}^2_kv_i = \frac{1}{2}\left({}^3_kt_i - {}^2_kt_i\right){}_ka^{\mathrm{peak1}} \tag{56} $$

$$ {}^4_kv_i - {}^3_kv_i = 0 \tag{57} $$

$$ {}^5_kv_i - {}^4_kv_i = \frac{1}{2}\left({}^5_kt_i - {}^4_kt_i\right){}_ka^{\mathrm{peak2}} \tag{58} $$

$$ {}_kV_i^{\mathrm{trgt}} - {}^5_kv_i = \frac{1}{2}\left(t_i^{\mathrm{sync}} - {}^5_kt_i\right){}_ka^{\mathrm{peak2}} \tag{59} $$

$$ {}^2_kp_i - {}_kP_i = {}_kV_i\left({}^2_kt_i - T_i\right) + \frac{1}{2}\,{}_kA_i\left({}^2_kt_i - T_i\right)^2 + \frac{1}{6}\,{}_kJ_i^{\max}\left({}^2_kt_i - T_i\right)^3 \tag{60} $$

$$ {}^3_kp_i - {}^2_kp_i = {}^2_kv_i\left({}^3_kt_i - {}^2_kt_i\right) + \frac{1}{2}\,{}_ka_i^{\mathrm{peak1}}\left({}^3_kt_i - {}^2_kt_i\right)^2 - \frac{1}{6\,{}_kJ_i^{\max}}\left({}^3_kt_i - {}^2_kt_i\right)^3 \tag{61} $$

$$ {}^4_kp_i - {}^3_kp_i = {}^3_kv_i\left({}^4_kt_i - {}^3_kt_i\right) \tag{62} $$

$$ {}^5_kp_i - {}^4_kp_i = {}^4_kv_i\left({}^5_kt_i - {}^4_kt_i\right) - \frac{1}{6}\,{}_kJ_i^{\max}\left({}^5_kt_i - {}^4_kt_i\right)^3 \tag{63} $$

$$ {}_kP_i^{\mathrm{trgt}} - {}^5_kp_i = {}^5_kv_i\left(t_i^{\mathrm{sync}} - {}^5_kt_i\right) + \frac{1}{2}\,{}_ka_i^{\mathrm{peak2}}\left(t_i^{\mathrm{sync}} - {}^5_kt_i\right)^2 + \frac{1}{6}\,{}_kJ_i^{\max}\left(t_i^{\mathrm{sync}} - {}^5_kt_i\right)^3. \tag{64} $$

There is a significant difference between the systems of equations for Step 1 (34)–(50) and Step 2 (51)–(64). In Step 1, there is a number of systems of equations (elements of $\mathcal{P}_{\mathrm{Step1}}$), which lead to a valid solution, but in Step 2, exactly one system of equations ${}^s\Psi^{\mathrm{Step2}}$ leads to the desired solution. This system has to be determined by the decision tree of Fig. 12.

The solution of (51)–(64) contains ${}^2_kt_i$, ${}^3_kt_i$, ${}^4_kt_i$, ${}^5_kt_i$, ${}^2_kv_i$, ${}^3_kv_i$, ${}^4_kv_i$, ${}^5_kv_i$, ${}^2_kp_i$, ${}^3_kp_i$, ${}^4_kp_i$, ${}^5_kp_i$, ${}_ka_i^{\mathrm{peak1}}$, and ${}_ka_i^{\mathrm{peak2}}$. The first four values of this list together with T_i and t_i^{sync} compose ${}^l_k\vartheta_i \;\forall\; l \in \{1,\ldots,5\}$ [cf. (8)]. The latter eight values are used to calculate the motion polynomials at time instant T_i ${}^l_k\vec{m}_i \;\forall\; l \in \{1,\ldots,5\}$ such that the trajectory for DOF k ${}_k\mathcal{M}_i$ is completely described.

C. Step 3

Step 3 calculates the output values $\vec{P}_{i+1}$, $\vec{V}_{i+1}$, and $\vec{A}_{i+1}$ and works according to Section V-C.

D. Final Remarks on the Type IV-A OTG Algorithm

Because these decision trees would take too much space, they cannot be depicted completely in a book, thesis, or paper. The following gives an impression of the complexity: The tree $1B$—written in font size of 10 pt, prepared in a minimized version,

TABLE II
NUMBER OF NODES PER DECISION TREE FOR THE TYPE IV, VARIANT A OTG ALGORITHM

Decision tree	Calculated variable	Representation format complete	minimized
1A	${}_k t_i^{min}$	149	40
1B	${}_k^1 t_i^{begin}$	3885	61
1C	${}_k^1 t_i^{end}$	167	13
2	${}_k \mathcal{M}_i$	559	88

and with all nodes tightly arranged—can just be plotted on a poster of DIN A0 size. Even the description would fill a book of several hundred pages such that here only an impression shall be imparted, and only the basic conceptual ideas are explained. The cutouts of the two Type IV decision trees presented in Figs. 10 and 12 can only be considered as small samples. The total numbers of nodes per tree for the Type IV-A OTG algorithm are given by Table II. *Minimized* means that the number of nodes was minimized, i.e., subtrees were used multiple times (e.g., in the tree of Fig. 12, the decisions *2.005* and *2.006* are used twice in order to save space; although this kind of representation looks like a graph, the actual structure is a tree, i.e., there are no loops existent).

The challenge during the development of decision trees is the guarantee that (22) and (30) hold. When starting to develop a decision tree, the motion profile sets $\mathcal{P}_{\mathrm{Step1}}$ and $\mathcal{P}_{\mathrm{Step2}}$ are unknown [cf. (20) and (25)]. These two sets have to fit exactly to each other such that the two equations mentioned above hold. The design of these sets can only take place during the development of the $(2\,\alpha\,-4\,\beta)$ (cf. Table I) decision trees, because we cannot know how these profiles look like until we know all possible cases. Summarizing this

1) if we knew the motion profile sets $\mathcal{P}_{\mathrm{Step1}}$ and $\mathcal{P}_{\mathrm{Step2}}$ (and thus, all input domains ${}^r\mathcal{D}_{\mathrm{Step1}}\ \forall\ r \in \{1,\ldots,R\}$ and ${}^s\mathcal{D}_{\mathrm{Step2}}\ \forall\ s \in \{1,\ldots,S\}$) for a concrete type of OTG, it *might* be possible to generate the decision trees automatically;
2) if we knew all $(2\alpha - 4\beta)$ decision trees for a concrete type of OTG, it would be possible to determine the motion profile sets $\mathcal{P}_{\mathrm{Step1}}$ and $\mathcal{P}_{\mathrm{Step2}}$.

However, since we neither know $\mathcal{P}_{\mathrm{Step1}}$ or $\mathcal{P}_{\mathrm{Step2}}$ of a concrete type nor the decision trees beforehand, it cannot be possible to generate an OTG algorithm automatically. For the validation of (20), (22), (25), and (30), random input values $\mathbf{W}_i$ are generated until every single edge of the trees (and thus, also every single profile of the sets $\mathcal{P}_{\mathrm{Step1}}$ and $\mathcal{P}_{\mathrm{Step2}}$, cf. Table II) were used at least once.

VII. RELATION TO HIGH-LEVEL MOTION-PLANNING SYSTEMS

Robot motion planning and, in particular, path planning belong to the classic and fundamental areas of robotics. Here, we regard a special field of this area: real-time adaptive motion planning (RAMP, [10]). We assume robots that have to act in a dynamic and/or unknown environment and which are equipped with sensor systems to react to (unknown) static or dynamic obstacles, events, or abrupt changes of task parameters. References [11] and [31] give general overviews about the field of motion planning, and [10], [32], and [33] focus on real-time capable methods for (multi-)robot motion planning.

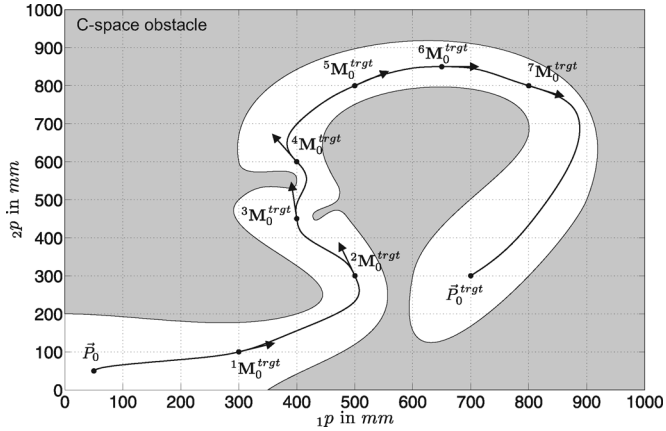

Fig. 15. High-level motion planning system may calculate intermediate motion states ${}^h\mathbf{M}_0^{\mathrm{trgt}}\ \forall\ h \in \{1,\ldots,7\}$ in configuration space, which are passed through by the online-generated trajectories from $\vec{P}_0$ to $\vec{P}_0^{\mathrm{trgt}}$.

As in [10], the generation of splines is one commonly used method to represent calculated trajectories; it is the task of a motion-planning algorithm to calculate respective knots. In [33], collision-free vertices (*"milestones"*) and edges on a roadmap, which is another kind of representation, are used to represent currently planned trajectories. These knots or milestones are generated from an overall view onto the robotic system and its environment—it is, in particular, a kind of motion planning from a global point of view. The output values of such higher level motion-planning systems, i.e., the knots and milestones, can be ideally used as input values for the OTG algorithm such that the combination of these systems leads to a very good symbiosis. Based on this idea, we can realize robotic systems which can move according to global and task-dependent motion planning and do not loose the ability to instantaneously react to low-level sensor events.

This is also comparable to human scenarios: If we unexpectedly touch a very spiky object and suddenly perceive pain, we can immediately pull our hand away in a reactive manner (without thinking, i.e., without global motion planning). As consequence, the OTG would be responsible for providing a kind of *robot reflex*, as was discussed in Section II.

In the following, we explain this idea by means of a concrete, simple, and static example. For illustration, Fig. 15 depicts a configuration space obstacle in 2-D-dimensional space. The task of the robot is to move from $\vec{P}_0 = (50, 50)\,\mathrm{mm}$ to $\vec{P}_0^{\mathrm{trgt}} = (700, 300)\,\mathrm{mm}$. For this purpose, the high-level motion planning system may calculate intermediate motion states

$$ {}^h\mathbf{M}_0^{\mathrm{trgt}} = \left({}^h\vec{P}_0^{\mathrm{trgt}},\ {}^h\vec{V}_0^{\mathrm{trgt}},\ {}^h\vec{A}_0^{\mathrm{trgt}}\right),\ \text{with}\ h \in \{1,\ldots,H\} \tag{65} $$

which have to be passed through by the trajectories of the OTG algorithm. H is the number of calculated motion states, which correspond to the knots [10] or milestones [33], i.e., the motion states of (65) constitute the interface to the high-level

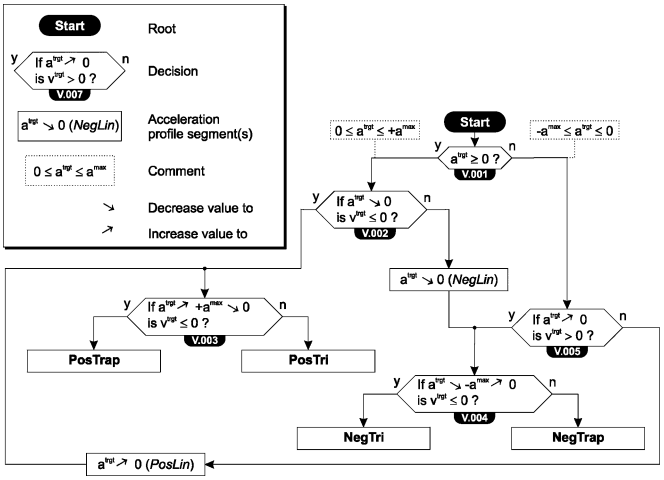

Fig. 16. Complete decision tree to determine the acceleration profile for transferring the target state of motion ${}^{h}\mathbf{M}_0^{\mathrm{trgt}}$ to zero velocity and zero acceleration in a time-optimal way as it would be required for the OTG algorithms of Types IV and V.

motion-planning system. Of course, ${}^{h}\mathbf{B}_i$ [cf. (3)] can also be adapted for the motions in-between two knots. In the case of dynamic environments, all motion states, which have not been passed by the robot yet, can, of course, be furthermore adapted at any future time.

The result of Fig. 15 was achieved with the Type IV-B OTG algorithm, i.e., ${}^{h}\vec{A}_0^{\mathrm{trgt}} = \vec{0}\ \forall\ h \in \{1,\ldots,7\}$ such that velocity vectors were simply put into the given configuration space

$$
\begin{aligned}
{}^{1}\vec{P}_0^{\mathrm{trgt}} &= (300,100)\,\mathrm{mm} & {}^{1}\vec{V}_0^{\mathrm{trgt}} &= (80,30)\,\mathrm{mm/s}\\
{}^{2}\vec{P}_0^{\mathrm{trgt}} &= (500,300)\,\mathrm{mm} & {}^{2}\vec{V}_0^{\mathrm{trgt}} &= (-30,100)\,\mathrm{mm/s}\\
{}^{3}\vec{P}_0^{\mathrm{trgt}} &= (400,450)\,\mathrm{mm} & {}^{3}\vec{V}_0^{\mathrm{trgt}} &= (-10,100)\,\mathrm{mm/s}\\
{}^{4}\vec{P}_0^{\mathrm{trgt}} &= (400,600)\,\mathrm{mm} & {}^{4}\vec{V}_0^{\mathrm{trgt}} &= (-50,80)\,\mathrm{mm/s}\\
{}^{5}\vec{P}_0^{\mathrm{trgt}} &= (500,800)\,\mathrm{mm} & {}^{5}\vec{V}_0^{\mathrm{trgt}} &= (60,40)\,\mathrm{mm/s}\\
{}^{6}\vec{P}_0^{\mathrm{trgt}} &= (650,850)\,\mathrm{mm} & {}^{6}\vec{V}_0^{\mathrm{trgt}} &= (120,0)\,\mathrm{mm/s}\\
{}^{7}\vec{P}_0^{\mathrm{trgt}} &= (800,800)\,\mathrm{mm} & {}^{7}\vec{V}_0^{\mathrm{trgt}} &= (150,-70)\,\mathrm{mm/s}.
\end{aligned}
$$

Finally, $\vec{P}_0^{\mathrm{trgt}} = (700,300)\,\mathrm{mm}$ is achieved with zero velocity.

Depending on the type of OTG, several restrictions may hold for the higher level planning algorithms. As discussed in Section IV-C, (18) only holds for the Types I, III, and VI. To keep the position of the system within its positional limits $({}_k\vec{P}_i^{\min}, {}_k\vec{P}_i^{\max})$, we have to assure the target states of motion ${}^{h}\mathbf{M}_i^{\mathrm{trgt}}\ \forall\ h \in \{1,\ldots,H\}$ are bounded. Analogous to the decision trees of Figs. 10 and 12, Fig. 16 depicts the complete decision tree used for the calculation of the the actual values of ${}^{h}\vec{P}_i^{\mathrm{trgt}}$ in ${}^{h}\mathbf{M}_i^{\mathrm{trgt}}$ for the OTG Types IV and V. First, one out of four acceleration profiles is determined to transfer ${}^{h}\mathbf{M}_i^{\mathrm{trgt}}$ to zero velocity and zero acceleration in a time-optimal way.

In the same way the systems of equations in (34)–(50) and (51)–(64) were set up based on the acceleration profiles of Figs. 11 and 14, the value for the required position difference (i.e., the minimum distance between ${}_k P_i^{\min}$ or ${}_k P_i^{\max}$ and ${}^{h}_{k}P_i^{\mathrm{trgt}}$) can be calculated in a closed form for each of the four acceleration profiles of Fig. 16.

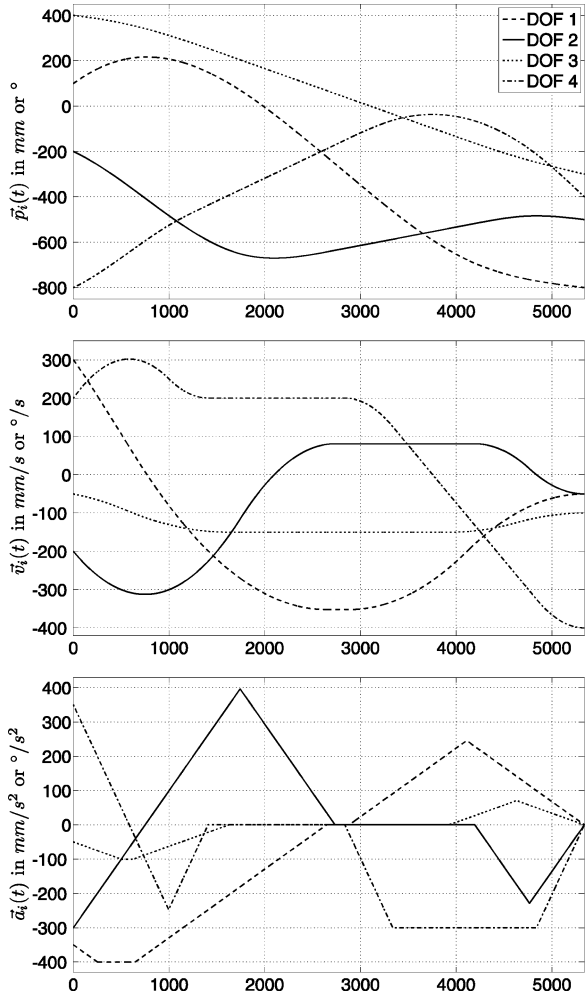

Fig. 17. Sample of a Type IV trajectory for $K = 4$ DOFs.

This section introduced how an OTG algorithm can act as an interface to higher level motion planning systems. The major symbiotic effect of this use case is that a robotic system, which is guided by a higher level planning system, can obtain the ability of performing immediate reflex motions as instantaneous reactions to unforeseen events.

VIII. Results of Type IV Online Trajectory Generation

A. Handling Arbitrary States of Motion

Earlier, we have only discussed the mathematical basics for Type IV OTG. To clarify the idea of this paper, Fig. 17 exemplarily illustrates one concrete result trajectory for $K = 4$ DOFs, i.e., the Type IV OTG acts as function $f\colon \mathbb{R}^{32} \times \mathbb{B}^4 \longrightarrow \mathbb{R}^{12}$, which is computed every control cycle. Our implemented robot motion controller works at a frequency of 1 KHz such that the algorithm is called once per $T^{\mathrm{cycle}} = 1$ ms. The given arbitrary

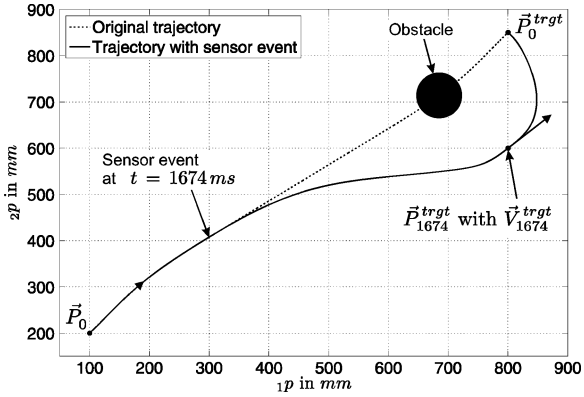

Fig. 18. XY plot of the exemplary geometric of a 2-DOF *path* according the trajectory in Fig. 19 with and without sensor event.

motion parameters are (normalized values, without units):

$$\left.\begin{array}{l}\left.\begin{array}{l}\vec{P}_0 = (100, -200, 400, -800)^T \\ \vec{V}_0 = (300, -200, -50, 200)^T \\ \vec{A}_0 = (-350, -300, -50, 350)^T\end{array}\right\}\mathbf{M}_0 \\ \left.\begin{array}{l}\vec{P}_0^{\,\mathrm{trgt}} = (-800, -500, -300, -400)^T \\ \vec{V}_0^{\,\mathrm{trgt}} = (-50, -50, -100, -400)^T\end{array}\right\}\mathbf{M}_0^{\mathrm{trgt}} \\ \left.\begin{array}{l}\vec{V}_0^{\,\max} = (800, 750, 150, 600)^T \\ \vec{A}_0^{\,\max} = (400, 400, 100, 300)^T \\ \vec{J}_0^{\,\max} = (200, 400, 100, 600)^T\end{array}\right\}\mathbf{B}_0 \\ \vec{S}_0 = (1, 1, 1, 1)^T\end{array}\right\}\mathbf{W}_0.$$

In Fig. 17, one can clearly recognize that all four DOFs reach their desired state of motion $\mathbf{M}_i^{\mathrm{trgt}}$ at the same time instant $t_i^{\mathrm{sync}} = 5340\,\mathrm{ms}\ \forall\ i \in \{0, \ldots, 5340\}$ ($N = 5340$ cycles, i.e., the OTG algorithm was executed 5340 times). t_i^{sync} was determined by ${}_3t_i^{\min}$. The selected acceleration profiles are

$$\begin{aligned}{}_1\Psi_i^{\mathrm{Step2}} &= NegTrapZeroPosTri \\ {}_2\Psi_i^{\mathrm{Step2}} &= PosTriZeroNegTri \\ {}_3\Psi_i^{\mathrm{Step2}} &= NegTrapZeroPosTri \\ {}_4\Psi_i^{\mathrm{Step2}} &= NegTriZeroNegTrap.\end{aligned}$$

For simplicity, the input parameters $\mathbf{M}_i^{\mathrm{trgt}}$, $\mathbf{B}_i$, and $\vec{S}_i$ remained constant during the whole execution time from T_0 to T_N, i.e., $\mathbf{M}_i^{\mathrm{trgt}} = \mathbf{M}_0^{\mathrm{trgt}} \wedge \mathbf{B}_i = \mathbf{B}_0 \wedge \vec{S}_i = \vec{S}_0\ \forall\ i \in \{0, \ldots, 5340\}$ [cf. (1) and (10)].

B. Reaction to Sensor Events

This section explains, how the OTG is applied to the simplest case of *sensor-guarded* motion control. For a simple and clear demonstration, we consider only a 2-DOF Cartesian robot. Of course, the OTG concept works for any number of DOFs K.

Fig. 18 depicts the *geometric path* of a trivial point-to-point motion (dotted line). It is the robot's task to move from an initial position $\vec{P}_0$ to a target position $\vec{P}_0^{\,\mathrm{trgt}}$ under the kinematic

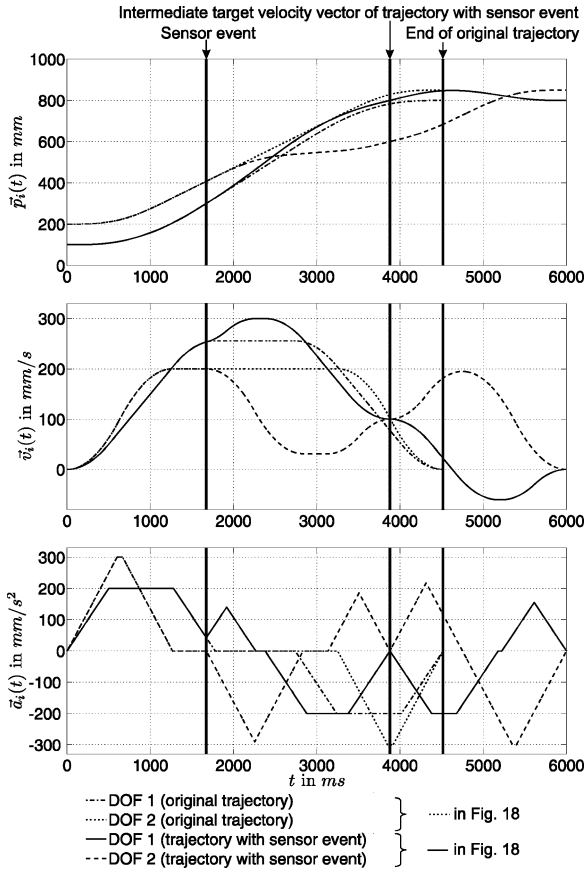

Fig. 19. Position, velocity, and acceleration progressions of the 2-DOF Type IV-A *trajectory* that corresponds to the path of Fig. 18.

constraints of $\vec{V}_0^{\,\max}$, $\vec{A}_0^{\,\max}$, and $\vec{J}_0^{\,\max}$

$$\begin{aligned}\vec{P}_0 &= (100, 200)^T\ \mathrm{mm} \\ \vec{P}_0^{\,\mathrm{trgt}} &= (800, 850)^T\ \mathrm{mm} \\ \vec{V}_0^{\,\max} &= (300, 200)^T\ \mathrm{mm/s} \\ \vec{A}_0^{\,\max} &= (200, 300)^T\ \mathrm{mm/s^2} \\ \vec{J}_0^{\,\max} &= (400, 500)^T\ \mathrm{mm/s^3}.\end{aligned}$$

In the following, we analyze this for two cases.

1) Without Obstacle: If we would not have to react to sensor events, off-line methods, which belong to the most classic ones in the field of robot-motion control, could be applied. An overview of trajectory generation methods is given in [27]. The resulting path is marked by the dotted line. The position, velocity, and acceleration progressions of the original trajectory are also depicted in Fig. 19 (dash-dotted and dotted lines). As can be seen in the bottom diagram of Fig. 19, both DOFs are transferred into their target positions by trapezoidal acceleration profiles such that both reach their target state at $t = 4518\,\mathrm{ms}$ with symmetrical velocity profiles.

2) With Obstacle and Reaction to Sensor Event: If we consider unknown objects/obstacles in our workspace, we have to react right after their detection. The results of this procedure are illustrated by the solid line in Fig. 18 and the solid and

dashed lines of the diagrams in Fig. 19. Type IV-A OTG is applied in this example. Due to some sensor, the system detects the unforeseen obstacle at $t = 1674\,\mathrm{ms}$ (${}_1P_{1674} = 300\,\mathrm{mm}$, cf. Figs. 18 and 19). A simple solution could then be the abrupt change of the target position values in the moment of obstacle detection, for example, to $\vec{P}_{1674}^{\,\mathrm{trgt}} = (800, 600)^T$ mm, such that we would prevent the collision with the object. After reaching $\vec{P}_{1674}^{\,\mathrm{trgt}}$, the old target position $\vec{P}_0^{\,\mathrm{trgt}}$ could then be reused to reach the originally desired position.

See the velocity and acceleration progressions in Fig. 19. Compared to the earlier idea, we present an advanced and more dynamic version in which the system puts an *intermediate target-velocity vector* $\vec{V}_{1674}^{\,\mathrm{trgt}} = (100, 100)^T$ mm/s into the position $\vec{P}_{1674}^{\,\mathrm{trgt}} = (800, 600)^T$ mm in order to bypass the obstacle dynamically. Right after $\vec{P}_{1674}^{\,\mathrm{trgt}}$ and $\vec{V}_{1674}^{\,\mathrm{trgt}}$ have been time-optimally reached (at $t = 3879\,\mathrm{ms}$, cf. Fig. 19), a second (abrupt) switching of input parameters occurs, and the system sets up the original parameters again ($\vec{P}_{3879}^{\,\mathrm{trgt}} = \vec{P}_0^{\,\mathrm{trgt}}$ and $\vec{V}_{3879}^{\,\mathrm{trgt}} = \vec{0}$). These are finally reached at $t = 6$ s (cf. Fig. 19).

Final Remarks : The OTG algorithm is executed every millisecond, and the output values of the algorithm lead to continuous trajectories, which reach each desired state of motion *time optimally* and *time-synchronized*. Furthermore, all kinematic motion constraints, $\vec{V}_0^{\max}$, $\vec{A}_0^{\max}$, and $\vec{J}_0^{\max}$ are kept during the whole trajectory. How the intermediate positions and velocity vectors as $\vec{P}_{1674}^{\,\mathrm{trgt}}$ and $\vec{V}_{1674}^{\,\mathrm{trgt}}$ are calculated depends on the system layer above the OTG (cf. Section VII).

C. Switching From Sensor-Guided Motion Control to Trajectory-Following Control

This paper contains theory, simulation results, and real-world experimental results. The latter demonstrates the functionality of the OTG concept in practice and is presented in this section. For the experiments, the following hardware setup has been used: The original controller of a Stäubli RX60 industrial manipulator [34] was replaced, and the frequency inverters were directly interfaced. Three PCs running with QNX [35] as real-time operating system perform a control rate of 10 kHz for the joint controllers; a hybrid switched-system controller is used for Cartesian space control and runs at a frequency of 1 kHz.

To explain the behavior of instantaneous switchings from sensor-guided control to trajectory-following control, Fig. 20 now closes the loop of the introductory chapter (cf. Fig. 1). At $T_0 = 0\,\mathrm{ms}$, a sensor-guided robot motion command was executed w.r.t. the hand frame of the manipulator. All six Cartesian DOFs are controlled by a simple zero-force/torque controller (proportional-integral differential), which uses unfiltered force/torque values of a JR3 force/torque sensor [36].[2] This was done intentionally in order to show the response of the overall system (including the OTG algorithm) on strongly noised sensor data. At $t = 586\,\mathrm{ms}$, a (sensor) event happens, and the system instantaneously switches from sensor-guided robot-motion control to trajectory-following control. The new trajectory is calculated instantaneously (within the control cycle after the event) and the manipulator performs a smooth, continuous motion. Here, this is done for all six DOFs, i.e., the selection vector switches from $\vec{S}_{585} = \vec{0}$ to $\vec{S}_{586} = \vec{1}$. Of course, it would also be possible that only some DOFs were switched from one controller to another. To describe the relevance for industrial practice, imagine that a sensor (e.g., a force/torque sensor or a vision system) fails during a sensor-guided robot motion, and then, the OTG algorithm can *always* take over control in any state of motion and at any time such that a smooth, continuous motion results. Furthermore, if a desired (force/torque or vision) set-point cannot be achieved for some reason, the current motion can be interrupted at any instant such that the OTG algorithm guides respective DOFs to a safe state.

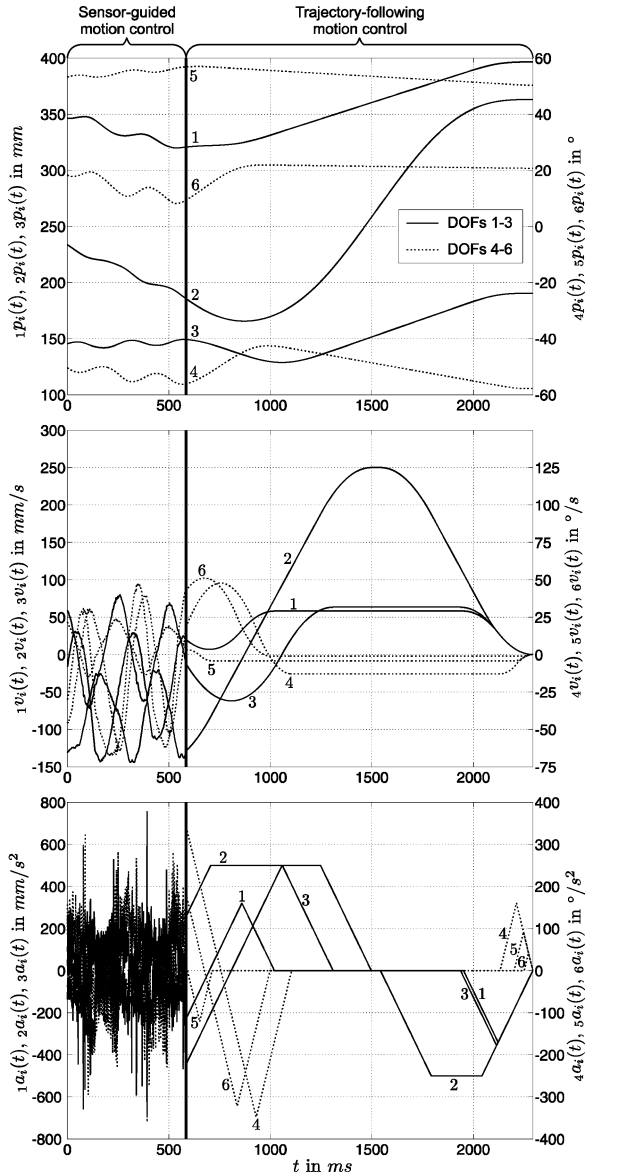

Fig. 20. Position, velocity, and acceleration progressions in Cartesian space of a sample motion of Stäubli RX60 industrial manipulator. First, all six DOFs are controlled by a feedback controller using force/torque sensor signals. At $t = 584\,\mathrm{ms}$, an event happens, and the (open-loop) Type IV OTG algorithm takes over control.

[2]Sensor model *85M35A-40 200N12*, receiver board running at 8 kHz.

The computations for this 6-DOF system requires an average execution time of $135\,\mu$s on a single-core machine.[3] The worst-case execution time is $540\,\mu$s. This difference is due to the calculation of inoperative time intervals. In most cases, inoperative time intervals are not existent, but for the worst case, they are considered for all DOFs.

IX. Conclusion and Outlook

This work introduced a new concept for motion generation in robotic systems during runtime. The presented online trajectory-generation algorithm is executed in parallel to low-level motion controllers such that systems using it are able to react *instantaneously* to unforeseen (sensor) events. In particular, the algorithm closes a significant gap: Switching from sensor-guided machine motions to trajectory-following motions becomes possible at any time and in any state of motion. As a consequence, (multi)sensor integration becomes substantially simplified, and robot-motion control systems are enabled to execute trajectory-following motions, sensor-guided motions, and *sensor-guarded motions* within one framework. The proposed online trajectory-generation algorithm acts as an open-loop controller and can take over control at any time instant such that safe and continuous motions can be guaranteed—even if sensors fail.

The algorithm was developed with the aim to advance motion control systems for many robotic applications in various fields, such as service robotics, manipulation control systems, mobile robotics and manipulation, or robotic surgery—in short, all fields in which sensor integration plays a fundamental role. The algorithm can be regarded as an intermediate control layer, thus constituting one element of the important bridge between low-level robot motion control and higher level (sensor-based) motion planning.

Besides the general theory of online trajectory generation, this contribution introduced the A-variants of OTG, i.e., the kinematic-motion constraint values for velocity, acceleration, jerk, and derivative of jerk remain constant. In a follow-up publication, we will present the procedures and results of the B-variants (i.e., the matrix $\mathbf{B}_i$ becomes time-variant), which are required for the embedding of robot dynamics, for unforeseen control-space switchings, and for unforeseen reference-frame switchings. Furthermore, an adaptation of the algorithm will be described that enables the generation of homothetic trajectories (i.e., straight-line trajectories).

Acknowledgment

The authors wish to thank the anonymous reviewers for their invaluable comments, which helped us to improve the quality of this paper. The works of our former diploma students M. Hanisch, C. Hurnaus, and A. Tomiczek, who worked hard on the first ideas and implementations of this concept, are highly appreciated. Furthermore, the authors would like to thank *QNX Software Systems* for providing free software licenses for experimental setups.

[3]Used hardware: AMD Athlon64 3700+ (2.2 GHz, 1024 KB L2 Cache), 2-GB DDR-400, gigabyte GA-K8NF9 ultra F5 mainboard.

References

[1] B. Siciliano and L. Villani, *Robot Force Control*. Norwell, MA: Kluwer, 1999.

[2] F. Chaumentte and S. A. Hutchinson, "Visual servoing and visual tracking," in *Springer Handbook of Robotics*, 1st ed., B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008 ch. 24, pp. 563–583.

[3] T. Kröger, B. Finkemeyer, S. Winkelbach, S. Molkenstruck, L.-O. Eble, and F. M. Wahl, "A manipulator plays Jenga," *IEEE Robot. Autom. Mag.*, vol. 15, no. 3, pp. 79–84, Sep. 2008.

[4] Hasbro, Inc. (2008). Jenga homepage. [Online]. Pawtucket, RI. Available: http://www.jenga.com

[5] M. E. Kahn and B. Roth, "The near-minimum-time control of open-loop articulated kinematic chains," *ASME J. Dyn. Syst., Meas., Control*, vol. 93, pp. 164–172, Sep. 1971.

[6] M. Brady, "Trajectory planning," in *Robot Motion: Planning and Control*, M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Pérez, and M. T. Mason, Eds. Cambridge, MA: MIT Press, 1982, ch. 4, pp. 221–243.

[7] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, Fall 1985.

[8] K. J. Kyriakopoulos and G. N. Sridis, "Minimum jerk path generation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Philadelphia, PA, Apr. 1988, vol. 1, pp. 364–369.

[9] P. Lambrechts, M. Boerlage, and M. Steinbuch, "Trajectory planning and feedforward design for high performance motion systems," in *Proc. Amer. Control Conf.*, Boston, MA, Jun. 2004, vol. 5, pp. 4637–4642.

[10] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1199–1212, Oct. 2008.

[11] S. Lindemann and S. LaValle, "Current issues in sampling-based motion planning," in *Proc. 8th Int. Symp. Robot. Res.*, P. Dario and R. Chatila, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 36–54.

[12] O. Brock and O. Khatib, "Elastic strips: A framework for motion generation in human environments," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 1031–1052, Dec. 2002.

[13] O. Brock, J. Kuffner, and J. Xiao, "Manipulation for robot tasks," in *Springer Handbook of Robotics*, 1st ed., B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008, ch. 26, pp. 615–645.

[14] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[15] B. Cao, G. I. Dodds, and G. W. Irwin, "A practical approach to near time-optimal inspection-task-sequence planning for two cooperative industrial robot arms," *Int. J. Robot. Res.*, vol. 17, no. 8, pp. 858–867, Aug. 1998.

[16] X. Broquère, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 2808–2813.

[17] S. Liu, "An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators," in *Proc. 7th Int. Workshop Adv. Motion Control*, Maribor, Slovenia, Jul. 2002, pp. 365–370.

[18] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 3248–3253.

[19] K. Ahn, W. K. Chung, and Y. Yourn, "Arbitrary states polynomial-like trajectory (ASPOT) generation," in *Proc. 30th Annu. Conf. IEEE Ind. Electron. Soc.*, Busan, Korea, Nov. 2004, vol. 1, pp. 123–128.

[20] R. H. Castain and R. P. Paul, "An on-line dynamic trajectory generator," *Int. J. Robot. Res.*, vol. 3, no. 1, pp. 68–72, Mar. 1984.

[21] J. Heinzmann and A. Zelinsky, "Quantitative safety guarantees for physical human–robot interaction," *Int. J. Robot. Res.*, vol. 22, no. nos. 7/8, pp. 479–504, Jul. 2003.

[22] A. D. Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, and G. Hirzinger, "The skeleton algorithm for self-collision avoidance of a humanoid manipulator," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatron.*, Zürich, Switzerland, Sep. 2007, pp. 1–6.

[23] A. D. Santis and B. Siciliano, "Reactive collision avoidance for safer human–robot interaction," presented at the IARP IEEE/RAS-EURON Workshop Tech. Challenges Dependable Robots Hum. Environ., Roma, Italy, Apr. 2007.

[24] S. Haddadin, A. Albu-Schäffer, A. D. Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human–robot interaction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 3356–3363.

[25] A. Albu-Schäffer and C.Ott, G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *Int. J. Robot. Res.*, vol. 26, no. 1, pp. 23–39, Jan. 2007.
[26] T. Kröger, A. Tomiczek, and F. M. Wahl, "Towards on-line trajectory computation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 2006, pp. 736–741.
[27] W. Khalil and E. Dombre, *Modeling, Identification and Control of Robots*, Trajectory Generation, 1st ed. London, U.K.: Hermes Penton, 2002, ch. 13, pp. 313–345.
[28] G. Engeln-Müllges and F. Uhlig, *Numerical Algorithms with C*. New York: Springer-Verlag, 1996.
[29] N. Anderson and Å. Björck, "A new high order method of regula falsi type for computing a root of an equation," *BIT Numer. Math.*, vol. 13, no. 3, pp. 253–264, Sep. 1973.
[30] R. King, "An improved pegasus method for root finding," *BIT Numer. Math.*, vol. 13, no. 4, pp. 423–427, Dec. 1973.
[31] H. H. González-Baños, D. Hsu, and J.-C. Latombe, "Motion planning: Recent developments," in *Autonomous Mobile Robots: Sensing, Control, Decision-Making and Applications*, S. S. Ge and F. L. Lewis, Eds. Boca Raton, FL: CRC, 2006, ch. 10.
[32] O. Brock and L. E. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 2001, pp. 1469–1474.
[33] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," presented at the Robot.: Sci. Syst. Conf., Philadelphia, PA, Aug. 2006.
[34] Stäubli Faverges SCA. (2008), Faverges (Annecy), France. [Online]. Available: http://www.staubli.com/en/robotics
[35] QNX Software Systems. (2008). Ottawa, ON, Canada. [Online]. Available: http://www.qnx.com
[36] JR3, Inc. (2008). Woodland, CA. [Online]. Available: http://www.jr3.com

Torsten Kröger (M'02) received the Ph.D. degree from the Institut für Robotik und Prozessinformatik, Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany, in 2009.

In 2001, he attended an industrial internship at Lenze Corporation, Atlanta, GA. He finished his studies in electrical engineering at the Technische Universität Carolo-Wilhelmina zu Braunschweig in 2002. Since 2003, he has been a Research Fellow with the Institut für Robotik und Prozessinformatik, Technische Universität Carolo-Wilhelmina zu Braunschweig. Since 2006, he has been a Technical Consulter for several companies dealing with robot technologies. His current research interests include online trajectory generation, hybrid switched-motion control, multisensor integration, and new robot-programming paradigms.

Friedrich M. Wahl (M'82) studied electrical engineering/cybernetics at the Technical University of Munich, Munich, Germany, from which he received the Diploma, Ph.D., and "venia legendi" degrees in digital signal and image processing in 1974, 1980, and 1984, respectively.

From 1974 to 1981, he conducted research with the Institute of Communication Engineering, Technical University of Munich, in the fields of pattern recognition and signal and image processing. From 1981 to 1986, he was with the IBM Research Laboratories, San Jose, CA, and Zurich, Switzerland, where he was enagaged in research in the areas of document analysis, industrial image analysis, and machine vision. Since 1986, he has been a Professor of computer science with the Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany, where he set up the Institut für Robotik und Prozessinformatik. He became an Advisory Professor with Shanghai University, Shanghai, China, in 1992 and Chairman of the Collaborative Research Centre on Robotic Systems for handling and assembly with the Technische Universität Carolo-Wilhelmina zu Braunschweig in 2000. His current research interests include robotics, computer vision, and algorithmic aspects of computer science.