2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

On-Line Trajectory Generation: Nonconstant Motion Constraints

Torsten Kröger

***Abstract*— A concept of on-line trajectory generation for robot motion control systems enabling instantaneous reactions to unforeseen sensor events was introduced in a former publication. This previously proposed class of algorithms requires constant kinematic motion constraints, and this paper extends the approach by the usage of time-variant motion constraints, such that low-level trajectory parameters can now abruptly be changed, and the system can react instantaneously within the same control cycle (typically one millisecond or less). This feature is important for instantaneous switchings between state spaces and reference frames at sensor-dependent instants of time, and for the usage of the algorithm as a control submodule in a hybrid switched robot motion control system. Real-world experimental results of two sample use-cases highlight the practical relevance of this extension.**

I. Introduction

Sensor integration in the feedback loops of low-level motion controllers belongs to key technologies for the future advancement of robot arm controllers. It is important to enable robot motion controllers to *instantaneously* switch from sensor-guided motion control (e.g., force/torque control [1] or visual servo control [2]) to trajectory-following motion control (and vice versa) at *unforeseen* instants. This way, new *event-based* robot programming methodologies can be realized as robots become enabled to react instantaneously in the moment the event is detected. In recent works [3], [4], a concept of on-line trajectory generation (OTG) was proposed. The resulting algorithms run in parallel to low-level motion controllers and are able to compute a trajectory from arbitrary states of motion within the same control cycle that the unforeseen switching occurs. The major limitation of the algorithms described in [3] is that only constant kinematic motion constraints can be applied to them, that is,

$$\mathbf{B}_i = \left(\vec{V}_i^{\,max}, \vec{A}_i^{\,max}, \vec{J}_i^{\,max}, \vec{D}_i^{\,max}\right) = \textbf{const}\ \forall\, i \in \mathbb{Z}\,, \tag{1}$$

where the columns of the matrix $\mathbf{B}_i$ contain the maximum velocity vector $\vec{V}_i^{\,max}$ at a discrete instant T_i, the maximum acceleration vector $\vec{A}_i^{\,max}$, the maximum jerk vector $\vec{J}_i^{\,max}$, and/or even a vector for the maximum derivatives of jerk $\vec{D}_i^{\,max}$. This paper extends the algorithms of [3], such that time-variant values for all elements of $\mathbf{B}_i$ can be applied. This way, the algorithm can generate a trajectory even if one or more elements of the current state of motion

$$\mathbf{M}_i = \left(\vec{P}_i, \vec{V}_i, \vec{A}_i, \vec{J}_i\right) = \left({}_1\vec{M}_i, \ldots, {}_k\vec{M}_i, \ldots, {}_K\vec{M}_i\right)^T \tag{2}$$

T. Kröger is with the Artificial Intelligence Laboratory at Stanford University, Stanford, CA 94305-9010, USA, tkr@stanford.edu.

exceed the values of $\mathbf{B}_i$. In eqn. (2), K represents the number of degrees of freedom (DOF) of the robotic system; $\vec{P}_i$ contains the position, $\vec{V}_i$ the velocity, $\vec{A}_i$ the acceleration, and $\vec{J}_i$ the jerk at the discrete instant T_i. The major benefits of this extension will be:

- The values of the kinematic motion constraints can be abruptly increased or decreased, such that motion trajectory parameters can be adapted on-line and the system reacts to the change immediately within *one* control cycle (commonly, one millisecond or less).
- The algorithm can be used as a control submodule in a *hybrid switched robot motion control system*, which is available even if sensors fail.
- *Instantaneous* switchings between state spaces and reference frames at *unforeseen* instants become possible.
- One of the prerequisites for the embedding of *robot dynamics* to the OTG concept is set up (future work).

The next section introduces related works, Sec. III describes the extension of the OTG algorithm, and Sec. IV discusses real-world experimental results of this extended class of OTG algorithms.

II. Related Work

The works most related to this paper are [5]–[11], all of which belong to the fields of robot motion control [12] and trajectory generation [13], [14] in robotic systems. Macfarlane *et al.* [5] present a jerk-bounded, near-time-optimal trajectory planner that uses quintic splines, which are also computed on-line but only for *one-DOF systems*. In [6], Cao *et al.* use rectangular jerk pulses to compute trajectories, but initial accelerations different from zero cannot be applied. Compared to the multi-DOF approach presented here, the latter method has been developed for *one-dimensional problems* only. Broquère *et al.* [7] published a work that uses an on-line trajectory generator for an arbitrary number of independently acting DOFs. The approach is very similar to the one of Liu [8] and is based on the classic seven-segment acceleration profile [15]. With regard to [4], it is a Type V on-line trajectory generation approach designed for handling several DOFs individually. Another very recent concept was proposed by Haddadin *et al.* [11]; Instead of generating motion trajectories, virtual springs and damping elements are setup up used as input values for a Cartesian impedance controller of the robot.

A disadvantage of [5], [6], [8] is that they cannot cope with initial acceleration values unequal to zero. A further, recent work of Haschke *et al.* [9] presents an on-line trajectory planner in the very same sense as [3] does. The

978-1-4673-1404-6/12/$31.00 ©2012 IEEE

proposed algorithm generates jerk-limited trajectories from arbitrary states of motion, but it suffers from numerical stability problems, that is, 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.* [10] proposed a work for the on-line calculation of one-dimensional motion trajectories for any given state of motion *and* with arbitrary target states of motion, that is, 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 work is that no kinematic motion constraints, such as maximum velocity, acceleration, and jerk values, can be specified.

III. The Extended Algorithm

Let us first describe the extension of the OTG algorithm in a generic manner and afterwards concretely by means of the extension of the OTG Types III – V.[1]

A. Formal Description

This nomenclature used here is inherited from [3], [4]. Let us define a trajectory $\mathcal{M}_i(t)$, which is calculated at a discrete time instant T_i, as

$$\begin{aligned}\mathcal{M}_i(t) &= \Big\{ \left({}^1\mathbf{m}_i(t),\, {}^1\mathcal{V}_i\right), \ldots, \left({}^l\mathbf{m}_i(t),\, {}^l\mathcal{V}_i\right), \\ &\quad \ldots, \left({}^L\mathbf{m}_i(t),\, {}^L\mathcal{V}_i\right) \Big\}\,,\end{aligned} \tag{3}$$

where the elements ${}^l\mathbf{m}_i(t)$ are matrices of motion polynomials

$$\begin{aligned}{}^l\mathbf{m}_i(t) &= \left({}^l\vec{p}_i(t),\, {}^l\vec{v}_i(t),\, {}^l\vec{a}_i(t),\, {}^l\vec{j}_i(t)\right) \\ &= \left({}^l_1\vec{m}_i(t), \ldots, {}^l_k\vec{m}_i(t), \ldots, {}^l_K\vec{m}_i(t)\right)^T\,.\end{aligned} \tag{4}$$

Time-discrete values are represented by capital letters, time-continuous values by lower case letters. A trajectory segment l of a single DOF k is described by the motion polynomials

$${}^l_k\vec{m}_i(t) = \left({}^l_k p_i(t),\, {}^l_k v_i(t),\, {}^l_k a_i(t),\, {}^l_k j_i(t)\right)\,, \tag{5}$$

where ${}^l_k p_i(t)$ represents the position progression, ${}^l_k v_i(t)$ the velocity progression, ${}^l_k a_i(t)$ the acceleration progression, and ${}^l_k j_i(t)$ the jerk progression. According to eqn. (3), a complete trajectory is described by L segments, and each segment l is accompanied by a set of time intervals

$${}^l\mathcal{V}_i = \left\{{}^l_1\vartheta_i, \ldots, {}^l_k\vartheta_i, \ldots, {}^l_K\vartheta_i\right\}, \text{ where } {}^l_k\vartheta_i = \left[{}^{l-1}_k t_i,\, {}^l_k t_i\right]\,, \tag{6}$$

such that a single set of motion polynomials ${}^l_k\vec{m}_i(t)$ is only valid within the interval ${}^l_k\vartheta_i$.

Fig. 1 shows the input and output values of the OTG algorithm in a generic manner (cf. [3]). It is the task of the algorithm to time-optimally transfer an arbitrary current state of motion $\mathbf{M}_i$ into the desired target state of motion

$$\mathbf{M}_i^{trgt} = \left(\vec{P}_i^{\,trgt},\, \vec{V}_i^{\,trgt},\, \vec{A}_i^{\,trgt},\, \vec{J}_i^{\,trgt}\right) \tag{7}$$

under consideration of the kinematic motion constraints $\mathbf{B}_i$. The algorithm works memoryless and calculates only the state of motion of the next control cycle, $\mathbf{M}_{i+1}$ (because an unforeseen sensor event may occur until T_{i+1}). The cycle time commonly is in the range of one millisecond or less to be able to react *instantaneously*, that is, in the same control cycle an unforeseen switching event occurs.

The algorithms described in [3] require constant values of $\mathbf{B}_i$ (cf. eqn. (1)) and are extended now, such that the elements of $\mathbf{B}_i$ can be time-variant. The extension works in the same way as the basic algorithms of [3] do, but here an additional decision tree is connected upstream of each basic decision tree. The case in which one or more elements of the initial motion state values ${}_k\vec{M}_i$ exceed their corresponding constraints of ${}_k\vec{B}_i$ has to be considered in this extension, and it may happen at any discrete time instant T_i with $i \in \mathbb{Z}$ and $k \in \{1, \ldots, K\}$, in which

$$\begin{aligned}&|{}_kV_i| > {}_kV_i^{max} \quad \text{and/or} \quad |{}_kA_i| > {}_kA_i^{max} \quad \text{and/or} \\ &|{}_kJ_i| > {}_kJ_i^{max} \quad \text{and/or} \quad |{}_kD_i| > {}_kD_i^{max}\end{aligned} \tag{8}$$

is true. Furthermore, motion states ${}_k\vec{M}_i$ may occur, which are within their respective bounds ${}_k\vec{B}_i$ at instant T_i, but which will lead to an unavoidable future exceeding of ${}_k\vec{B}_i$ at a time instant T_{i+u}:

$$\begin{aligned}&|{}_kV_{i+u}| > {}_kV_i^{max} \quad \text{and/or} \quad |{}_kA_{i+u}| > {}_kA_i^{max} \quad \text{and/or} \\ &|{}_kJ_{i+u}| > {}_kJ_i^{max} \quad \text{and/or} \quad |{}_kD_{i+u}| > {}_kD_i^{max} \\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{with } u \in \mathbb{N}\backslash\{0\}\,.\end{aligned} \tag{9}$$

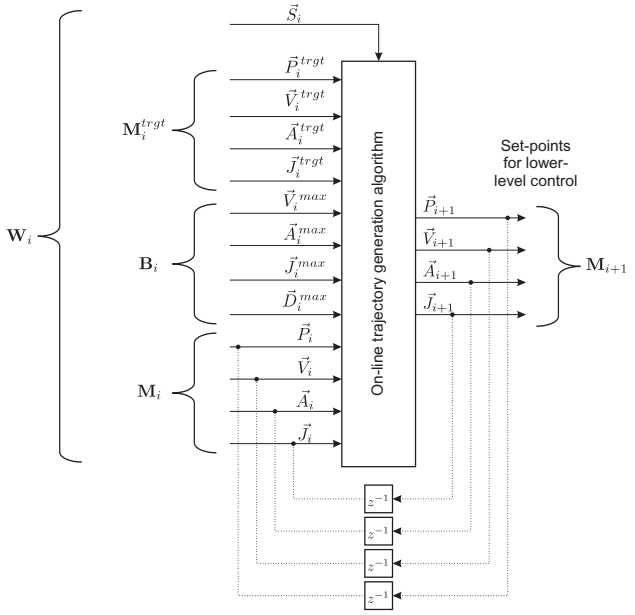

Fig. 1. Input and output values of the Type IX OTG algorithm (cf. [3]).

[1] For the OTG Types I – II, $\vec{J}_i^{\,max}$ and $\vec{D}_i^{\,max}$ are irrelevant; for the Types III – V, the values of $\vec{D}_i^{\,max}$ are not relevant, because they are not considered by these types of OTG algorithms (cf. [3]).

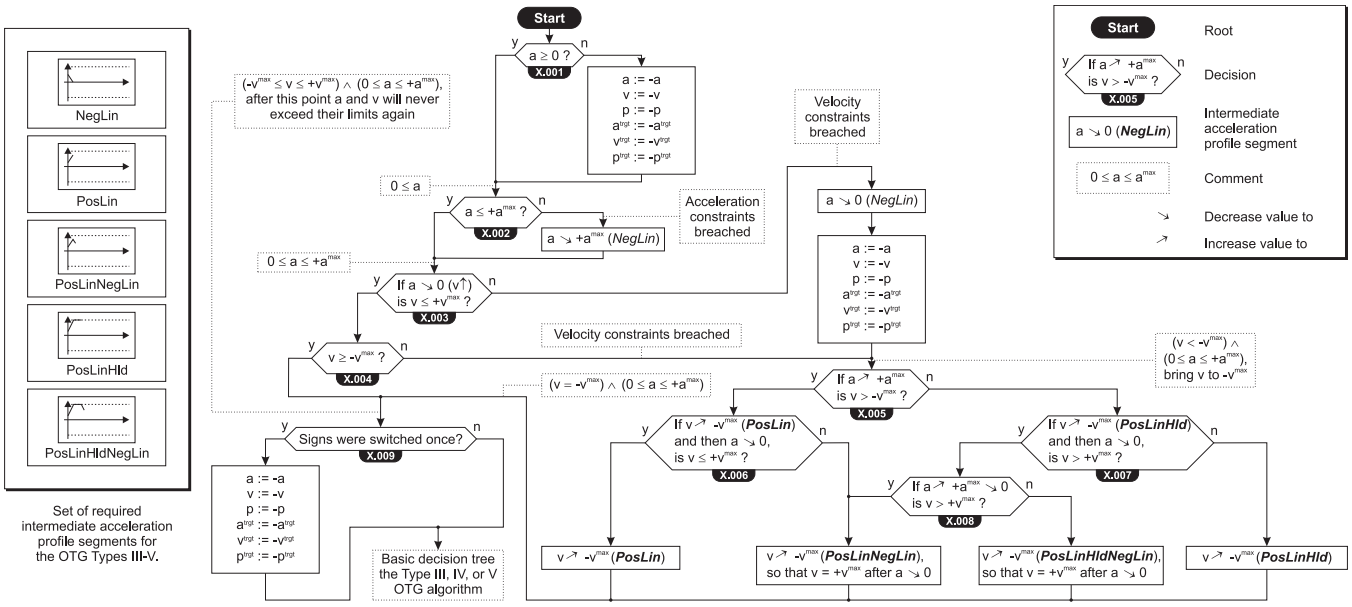

Fig. 2. This decision tree is applied prior to the decision trees of the Type III – V OTG algorithms described in [3]. Its task is to bring all motion state values $\mathbf{M}_i$ into their limits $\mathbf{B}_i$ and to guarantee that the elements of $\mathbf{M}_i$ can remain within in these bounds. For this purpose, one or more of the intermediate acceleration profiles, which are shown on the left, are connected upstream of the acceleration profiles selected by the main algorithm (cf. [3], [4]).

If one or more of the cases in eqns. (8) and (9) are true, the extended algorithm has to select and parameterize Λ intermediate trajectory segments ${}^{l}_{k}\vec{m}_i$ with $l \in \{1, \ldots, \Lambda\}$ with corresponding time intervals ${}^{l}_{k}\vartheta_i$ with $l \in \{1, \ldots, \Lambda\}$ (cf. eqns. (5) and (6)) in order to guide these values back into their bounds, and furthermore to bring the DOF k of the system into a state of motion after which the motion variables can be kept within the constraint values of ${}_{k}\vec{B}_i$. Therefore, this set of intermediate trajectory segments is determined and executed *prior* to the segments that are generated by the basic algorithm: ${}^{l}_{k}\vec{m}_i$ with $l \in \{\Lambda + 1, \ldots, L\}$ and ${}^{l}_{k}\vartheta_i$ with $l \in \{\Lambda + 1, \ldots, L\}$. Corresponding to the type of OTG, velocity profiles (Type I, II), acceleration profiles (Type III – V), or jerk profiles (Type VI – IX) are applied, respectively. As for the basic algorithms [3], [4], it is absolutely essential that the additional decision tree and the respective intermediate motion profiles cover the *complete input space* of the algorithm; otherwise there would be input values $\mathbf{W}_i$ (cf. Fig. 1) to which no trajectory, that is, no output signal $\mathbf{M}_{i+1}$, can be generated.

B. Extension by Means of Type III – V OTG Algorithms

Based on the OTG Types III – V, we will derive the required extension in this subsection; these types generate jerk-limited trajectories (cf. [3]). Fig. 2 shows the respective decision tree in a minimized version. The execution of the intermediate trajectory segments shown on the left of Fig. 2 is finished at ${}^{(\Lambda+1)}_{k}t_i$ (cf. eqns. (6) and (3)), and we have to assure, that

$$\begin{aligned} -{}_{k}V_i{}^{max} \;&\leq\; {}_{k}v_i\left({}^{(\Lambda+1)}_{k}t_i\right) \;\leq\; +{}_{k}V_i{}^{max} \quad \wedge \\ -{}_{k}A_i{}^{max} \;&\leq\; {}_{k}a_i\left({}^{(\Lambda+1)}_{k}t_i\right) \;\leq\; +{}_{k}A_i{}^{max} \end{aligned} \tag{10}$$

hold for all DOFs $k \in \{1, \ldots, K\}$. Furthermore, we have to ensure for each DOF k that if we bring ${}_{k}a_i\left({}^{(\Lambda+1)}_{k}t_i\right)$ to zero by applying the maximum possible jerk $\pm{}_{k}J_i^{max}$, the maximum velocity value of ${}_{k}V_i{}^{max}$ is not exceeded again (neither positively nor negatively). Therefore, the plain condition

$$\left| {}_{k}v_i\left({}^{(\Lambda+1)}_{k}t_i\right) \pm \frac{\left({}_{k}a_i\left({}^{(\Lambda+1)}_{k}t_i\right)\right)^2}{{}_{k}J_i^{max}} \right| \;\leq\; {}_{k}V_i{}^{max} \tag{11}$$

has to be fulfilled for all DOFs $k \in \{1, \ldots, K\}$. In the following, the extension is derived in order to let eqns. (10) and (11) be true and to subsequently apply the basic part of a Type III, IV, or V OTG algorithm (cf. [3]).

The minimized decision tree of Fig. 2 takes advantage of sign switchings. Since this tree is executed prior to all basic decision trees of OTG Types III – V and also prior to the further decision tree for the synchronization of multiple DOFs, the letter X has been chosen to replace the actual tree identifier (e.g., $1A$, $1B$, or 2). Decision *X.001* leads to a switching of signs for the initial and for the target state of motion if the current acceleration value ${}_{k}A_i$ is negative. Hence, ${}_{k}A_i$ is positive for decision *X.002*. This decision checks whether ${}_{k}A_i^{max}$ is currently exceeded. If it is exceeded, we set up a first intermediate acceleration profile segment (*NegLin*), which brings A_i down to ${}_{k}A_i^{max}$

by applying $-_k J_i^{max}$. The decisions *X.003* and *X.004* check whether $_k V_i^{max}$ is positively or negatively exceeded. Since our current acceleration value is positive, decision *X.003* calculates the velocity value that we would obtain if we were to bring the acceleration value to zero (which increases the velocity value). If the resulting velocity is then greater than $+_k V_i^{max}$, we decrease the acceleration to zero by applying $-_k J_i^{max}$ again (*NegLin*), perform a switching of signs, and let the decisions *X.005* to *X.008* bring the velocity value into its bounds. Decision *X.004* only checks whether $-_k V_i^{max}$ is exceeded. If this is the case, we continue at decision *X.005*. For this decision, we know that the velocity is less than $-_k V_i^{max}$, and the acceleration is positive (no matter if the branch of decision *X.003* or *X.004* has been taken). If we would now increase the acceleration to $+_k A_i^{max}$, decision *X.005* checks whether the resulting velocity value is greater or less than $-_k V_i^{max}$. If it is less, we know that a simple acceleration increase brings the velocity value back into its limits, but we have to make sure, that it can remain within these. For this purpose, decision *X.006* checks whether $+_k V_i^{max}$ would be exceeded if we subsequently decreased the acceleration value to zero. If this is not the case (left branch), a simple *PosLin* profile segment, which applies $+_k J_i^{max}$, complies with the requirements of eqns. (10) and (11). Otherwise (right branch), we would increase the acceleration value to a certain peak value, and subsequently decrease it again, such that we would reach $+_k V_i^{max}$ exactly after the full decrease to zero (profile segment *PosLin-NegLin*). The decisions *X.007* and *X.008* work analogously. In the last step, we have to re-switch the signs again if they have been switched before (decision *X.009*). Finally, we can assure that the conditions of eqns. (10) and (11) are fulfilled and will not be breached again, and we can continue with the basic decision trees of the Type III, IV, or V OTG algorithm as presented in [3].

Depending on the input values $\mathbf{W}_i$ (cf. Fig. 1) and in correspondence to the intermediate acceleration profiles of the decision tree shown in Fig. 2,

$$\Lambda \in \{0, \ldots, 5\} \tag{12}$$

holds for the Types III – V. Once the Λ intermediate trajectory segments are determined, they have to be parameterized. This is also done in the same way as for the acceleration profiles as described in [3], [4]. The resulting systems of equations are of trivial nature and can be solved in a straightforward way without any numerical problems.

IV. Results

This section discusses experimental results and the benefits achieved with the extended variant of the Type IV OTG algorithm. For comprehensiveness, we start with a simple one-DOF example, and subsequently we apply the proposed concept to a six-DOF robot manipulator.

A. Basic Example with a One-DOF System

For a better understanding, we illustrate the functionality of the extended Type IV OTG algorithm by means of a concrete example with one DOF k only. Let us assume some given input values $_k\vec{W}_0$ at instant $T_0 = 0\,ms$:

$$\begin{array}{rclrcl} {}_kP_0 & = & -100\,mm & {}_kP_0^{trgt} & = & 300\,mm \\ {}_kV_0 & = & -270\,mm/s & {}_kV_0^{trgt} & = & -100\,mm/s \\ {}_kA_0 & = & -450\,mm/s^2 & {}_kV_0^{max} & = & 300\,mm/s \\ {}_kA_0^{max} & = & 300\,mm/s^2 & {}_kJ_0^{max} & = & 900\,mm/s^3\ . \end{array} \tag{13}$$

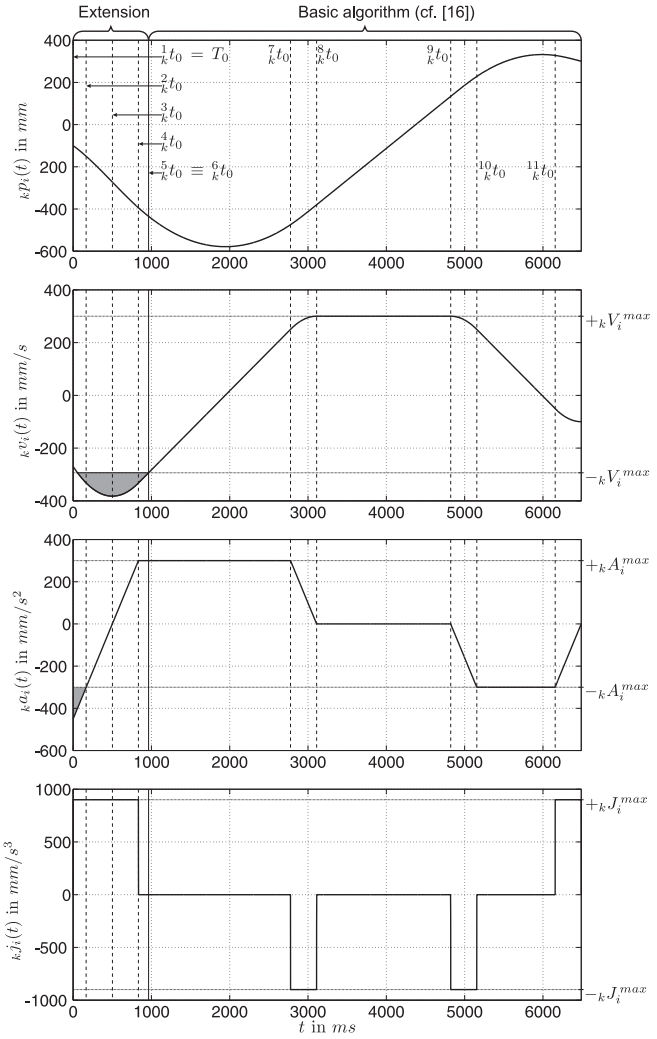

Fig. 3. Resulting Type IV trajectory at T_0 for one DOF k generated with the extended OTG algorithm presented in Sec. III for the given input values of eqn. (13). The vertical dashed lines indicate the bounds of the single trajectory segments, and the horizontal dotted lines indicate the kinematic motion constraints $_k\vec{B}_i$.

After the calculation at the control cycle of T_0, the trajectory of Fig. 3 results from the input values $_k\vec{W}_0$ of eqn. (13). In the first step, we select intermediate trajectory segments by applying the decision tree of Fig. 2. Here, we would take the following path:

$X.001 \rightarrow$ Change of signs $\rightarrow X.002 \rightarrow NegLin \rightarrow X.003 \rightarrow NegLin \rightarrow$ Change of signs $\rightarrow X.005 \rightarrow X.007 \rightarrow PosLinHld \rightarrow X.009 \rightarrow$ Basic decision tree of Type IV .

This example results in $\Lambda = 4$ intermediate trajectory

segments (cf. Fig. 2):

$$\begin{array}{rcll} NegLin & \Longrightarrow & PosLin & \text{One segment} & \left({}^{1}_{k}\vec{m}_0(t),\, {}^{1}_{k}\mathcal{V}_0\right) \\ NegLin & \Longrightarrow & PosLin & \text{One segment} & \left({}^{2}_{k}\vec{m}_0(t),\, {}^{2}_{k}\mathcal{V}_0\right) \\ & PosLinHld & & \text{Two segments} & \left({}^{3}_{k}\vec{m}_0(t),\, {}^{3}_{k}\mathcal{V}_0\right), \\ & & & & \left({}^{4}_{k}\vec{m}_0(t),\, {}^{4}_{k}\mathcal{V}_0\right) . \end{array}$$

At T_0, both conditions, eqns. (10) and (11), are not fulfilled. These $\Lambda = 4$ trajectory segments lead to a new state of motion ${}^{\Lambda}_{k}\vec{m}_0({}^{(\Lambda+1)}_{k}t_0)$, which satisfies eqns. (10) and (11), and we can execute the basic Type IV decision tree of [3]. The result of this tree is that $PosTrapZeroNegTrap$ acceleration profile is required for the time optimal solution; the corresponding nonlinear system of equations can be set up and solved to get all trajectory parameters ${}_{k}\mathcal{M}_i(t)$. Finally, we obtain $L = 4 + 7 = 11$ trajectory segments, whereas the fifth segment is actually not existent, because ${}^{5}_{k}a_0\left({}^{5}_{k}t_0\right) = {}_{k}A_i^{max}$, and, thus, ${}^{5}_{k}t_0 \equiv {}^{6}_{k}t_0$ holds (cf. Fig. 3).

B. Real-World Experimental Results

To highlight the practical relevance of the method proposed in this paper, we now discuss real-world experimental results and show, how the extended OTG algorithm can be applied in a hybrid switched-system for robot motion control. For the experiments, the same hardware setup as described in [3] has been used: The original controller of a six-joint Stäubli RX60 industrial manipulator [16] was replaced, and the frequency inverters were directly interfaced. Three PCs running with QNX [17] 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.

If we consider the actuator space control scheme as Lyapunov stable [18], we can focus on the task space control scheme, which contains the hybrid switched-system, and the on-line trajectory generation submodule. Especially, the works of Branicky [19], [20] and Liberzon [21], [22] provide elementary concepts to develop and analyze hybrid switched-system control techniques. In particular, the stability analysis is of fundamental interest here, because the stability of a switched-system cannot be assured by the stability of each single sub-controller. Proving the stability of hybrid switching systems can be extremely difficult and many researchers are working on analyzing such stability questions. Brockett [23] explains this subject for motion control systems. Žefran and Burdick [24], [25] suggest an approach, in which a system with changing dynamics is considered, and a hybrid controller is designed for handling the system in different "regimes" of dynamics. One essential benefit of the OTG submodule is that it can take over control at any time and in any state of motion to stabilize the system, if the underlying trajectory-tracking controller is stable.

Here, we consider a hybrid switched robot motion control system with six DOFs $\{\, x,\, y,\, z, \textcircled{x}, \textcircled{y}, \textcircled{z} \,\}$ [26]; the results are shown in Figs. 4 and 5. Starting with a sensor-guided motion of a simple PID zero-force controller [1], the

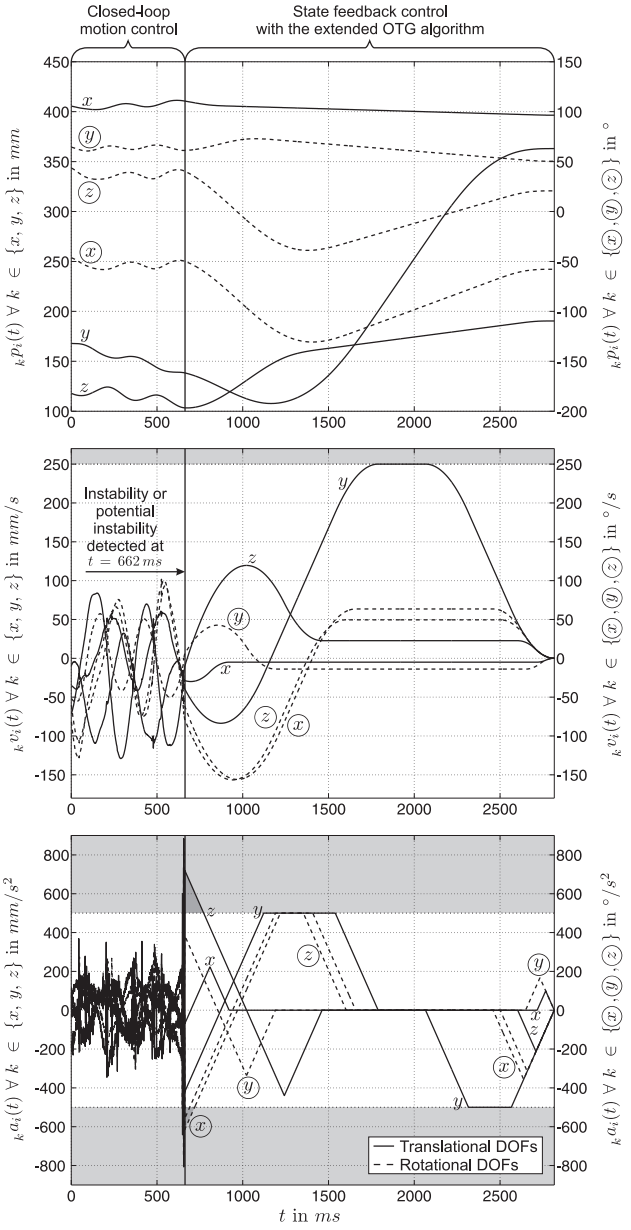

Fig. 4. Position, velocity, and acceleration progressions. The switching from sensor-guided robot motion control to trajectory-following control happened at $t = 662\,ms$ as the some acceleration values exceeded their maximum values (cf. eqn. (14)). All six trajectories coinstantaneously reach their desired target state of motion at $T_N = 2816\,ms$. Fig. 5 depicts these trajectories in the velocity-acceleration-plane of the state space (cf. [26]).

system detects that the acceleration amplitudes exceed

$$\begin{aligned} {}_{k}A_i^{max} &= 500\,mm/s^2 \;\; \forall\,(k,\,i) \;\in\; \{\,x,\,y,\,z\} \;\times\; \mathbb{Z} \\ {}_{k}A_i^{max} &= 500\,^\circ/s^2 \;\; \forall\,(k,\,i) \;\in\; \{\textcircled{x},\textcircled{y},\textcircled{z}\} \;\times\; \mathbb{Z} \end{aligned} \tag{14}$$

for $\left\{ z, \textcircled{x}, \textcircled{z} \right\}$ at $t = 662\,ms$. It is essential that appropriate input parameters $\mathbf{M}_i^{trgt}$ and $\mathbf{M}_i^{trgt}$ are setup in the moment of switching from sensor-guided motion control to trajectory-following control. $\vec{A}_i^{max}$ may be calculated from the local forward dynamics [27], $\vec{V}_i^{\,max}$ commonly is given through mechanical system properties, and $\vec{J}_i^{max}$ may be

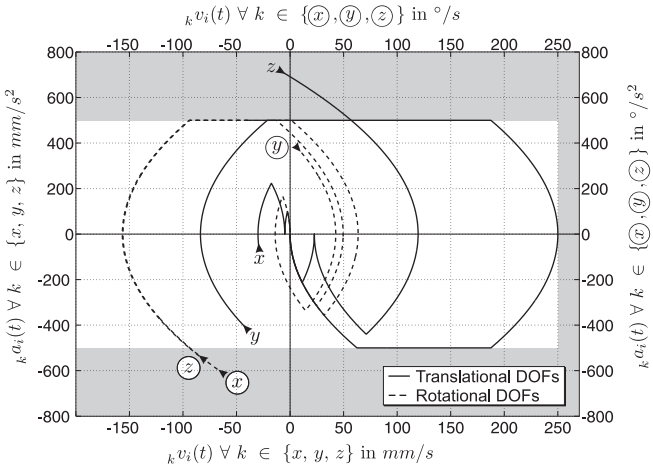

Fig. 5. Corresponding to Fig. 4, this diagram illustrates the six trajectories from the moment of switching on (i.e., in the interval $662\,ms \leq t \leq 2816\,ms$). All trajectories terminate in an equilibrium point of the underlying control loops (cf. [26]).

setup with regard to the current task again. The simplest way to setup $\mathbf{M}_i^{trgt}$ would be to choose

$$\vec{V}_i^{\,trgt} = \vec{0} \quad \text{and} \quad \vec{A}_i^{trgt} = \vec{0} \quad \forall \; i \; \in \; \mathbb{Z} \qquad (15)$$

again. The desired pose $\vec{P}_i^{\,trgt}$ should be set to a safe pose in workspace, such that no collisions and no singularities occur during the motion. Depending on the task, it can also be reasonable to specify a desired target velocity vector $\vec{V}_i^{\,trgt} \neq \vec{0}$ in space (e.g., in order to synchronize the system with cooperating one or to achieve a defined state of motion from which a safe motion can be continued).

Fig. 4 depicts the position, velocity, and acceleration progressions for all six DOFs $\{\, x, \, y, \, z, \textcircled{x}, \textcircled{y}, \textcircled{z} \,\}$, and Fig. 5 displays the corresponding trajectories the velocity-acceleration-plane of the state space from the moment of switching on. As one can clearly see in Fig. 4, the trajectories of all six DOFs are *continuous*, and they reach their desired target state of motion $\mathbf{M}_i^{trgt}$ coinstantaneously at $T_N = 2816\,ms$. Furthermore, Fig. 5 shows that all six trajectories terminate in an equilibrium point of the inner control loops as eqn. (15) was applied in this experiment (i.e., it is a Type III trajectory, cf. [3], [4]). Furthermore, it would also be possible to switch only a selection of DOFs instead of all DOFs.

The reason, why only results with decreasing motion constraints are shown, is that this case is more demanding, and increasing values can already be handled by the former algorithm of [3], [4]. For the reason of clarity, only these simple examples have been chosen for the simulation and real-world experimental results.

Due to the concept proposed in this paper, we are now able to *instantaneously* switch to state feedback control, that is, control performed by the extended OTG algorithm, in order to stabilize the system and continuously guide it to a safe pose. As proposed in [26], this concept may be used to stabilize hybrid switched-systems in a very general way, as the OTG algorithms can take over control from arbitrary states of motion at unforeseen instants.

V. Conclusion

The class of on-line trajectory generation algorithms described in [3] was extended, such that time-variant kinematic motion constraints can be applied to the algorithms. An additional decision tree is required to be applied upstream of the basic decision trees; if necessary, this tree selects intermediate motion profiles to be executed prior to the ones of the basic algorithm. As an example, this tree was derived for algorithm Types III – V, which generate jerk-limited motion trajectories.

Such an extended OTG algorithm enables users to on-line increase or decrease the values of kinematic motion constraints; motion trajectory parameters can be adapted on-line, and the system reacts to the change within *one* control cycle (commonly a millisecond or less). If used as a control submodule in a hybrid switched-system, the extended OTG algorithm can be used as a state feedback controller, which is available even if sensors fail. Real-world experimental results have shown how the extended algorithm can be used in a hybrid switched-system. Furthermore, *instantaneous* switchings between state spaces and reference frames at *unforeseen* instants become possible, and it is also a prerequisite for the embedding of robot dynamics, which is part of the future work to be done in this research direction.

Appendix

This extension of the OTG Framework has become part of the Reflexxes Motion Libraries [28], which can be downloaded from [29].

Acknowledgment

The research described in this paper was conducted at the *Institut für Robotik und Prozessinformatik* at the Technische Universitaet Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany, headed by *Professor Friedrich M. Wahl*, to whom I would like to express my sincere gratitude. Furthermore, I would like to express my appreciation to *Professor Oussama Khatib*, who is currently hosting me at the *Stanford Artificial Intelligence Laboratory* at Stanford University, Stanford, USA. The works of my former diploma students, *Michaela Hanisch*, *Christian Hurnaus*, and *Adam Tomiczek*, who worked hard on the first ideas and implementations of this concept, are highly appreciated. I am indebted to the *Deutsche Forschungsgemeinschaft* (DFG, German Research Foundation).

References

[1] L. Villani, , and J. De Schutter. Force control. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 7, pages 161–185. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[2] F. Chaumette and S. A. Hutchinson. Visual servoing and visual tracking. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, chapter 24, pages 563–583. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

[3] T. Kröger and F. M. Wahl. On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, February 2010.

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

[5] S. Macfarlane and E. A. Croft. Jerk-bounded manipulator trajectory planning: Design for real-time applications. *IEEE Trans. on Robotics and Automation*, 19(1):42–52, February 2003.

[6] 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. *The International Journal of Robotics Research*, 17(8):858–867, August 1998.

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

[8] S. Liu. An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. In *Proc. of the seventh International Workshop on Advanced Motion Control*, pages 365–370, Maribor, Slovenia, July 2002.

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

[10] K. Ahn, W. K. Chung, and Y. Yourn. Arbitrary states polynomial-like trajectory (ASPOT) generation. In *Proc. of the 30th Annual Conference of IEEE Industrial Electronics Society*, volume 1, pages 123–128, Busan, South Korea, November 2004.

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

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

[13] L. Biagiotti and C. Melchiorri. *Trajectory Planning for Automatic Machines and Robots*, chapter 3, Composition of Elementary Trajectories, pages 59–150. Springer, Berlin, Heidelberg, Germany, first edition, 2008.

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

[15] R. H. Castain and R. P. Paul. An on-line dynamic trajectory generator. *The International Journal of Robotics Research*, 3(1):68–72, March 1984.

[16] Stäubli Faverges SCA, Place Robert Stäubli BP 70, 74210 Faverges (Annecy), France. Homepage. `http://www.staubli.com/en/robotics` (accessed: Jan. 26, 2012). Internet, 2012.

[17] QNX Software Systems, 175 Terence Matthews Crescent, Ottawa, Ontario, Canada, K2M 1W8. Homepage. `http://www.qnx.com` (accessed: Mar. 10, 2011). Internet, 2011.

[18] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer, New York, NY, USA, 1999.

[19] M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Electrical Engineering and Computer Science Dept., Massachusetts Institute of Technology, `http://dora.cwru.edu/msb/pubs.html` (accessed: Mar. 10, 2011), 1995.

[20] M. S. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Trans. on Automatic Control*, 43(4):475–482, April 1998.

[21] D. Liberzon. *Switching in Systems and Control*. Systems and Control: Foundations and Applications. Birkhäuser, Boston, MA, USA, 2003.

[22] D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19(5):59–70, October 1999.

[23] R. W. Brockett. In H. L. Trentelman and J. C. Willems, editors, *Essays on Control: Perspectives in the Theory and its Applications*, chapter 2, pages 29–53. Birkhäuser, Boston, MA, USA, first edition, March 1993.

[24] M. Žefran and J. W. Burdick. Stabilization of systems with changing dynamics by means of switching. In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 2, pages 1090–1095, Leuven, Belgium, May 1998.

[25] M. Žefran and J. W. Burdick. Design of switching controllers for systems with changing dynamics. In *Proc. of the IEEE Conference on Decision and Control*, volume 2, pages 2113–2118, Tampa, FL, USA, December 1998.

[26] T. Kröger and F. M. Wahl. Stabilizing hybrid switched motion control systems with an on-line trajectory generator. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 4009–4015, Anchorage, AK, USA, May 2010.

[27] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer, New York, NY, USA, first edition, 2007.

[28] T. Kröger. Opening the door to new sensor-based robot applications — The Reflexxes Motion Libraries. In *Proc. of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[29] Reflexxes GmbH, Sandknöll 7, D-24805 Hamdorf, Germany. Hompepage. `http://www.reflexxes.com` (accessed: Feb. 6, 2012). Internet, 2012.