[21] M. Milanese, J. Norton, H. Piet-Lahanier, and E. Walter, *Bounding Approaches to System Identification*. New York: Plenum, 1996.

[22] D. K. Montemerlo, S. Thrun, and B. Wegbreit, “Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2003, pp. 1151–1156.

[23] N. Nedialkov, K. Jackson, and G. Corliss, “Validated solutions of initial value problems for ordinary differential equations,” *Appl. Math. Comput.*, vol. 105, no. 1, pp. 21–68 1999.

[24] P. Newman, J. Leonard, J. Tardós, and J. Neira, “Explore and return: Experimental validation of real-time concurrent mapping and localization,” in *Proc. Int. Conf. Robot. Autom. 2002*, Washington, DC, pp. 1802–1809.

[25] R.Smith, M. Self and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Autonomous Robot Vehicles*. vol. 8, New York: Springer, 1990, pp. 167–193.

[26] D. Sam-Haroud, “Constraint consistency techniques for continuous domains” Ph.D. dissertation, Swiss Federal Inst. Technol., Lausanne, Switzerland, 1995.

[27] S. Thrun, W. Bugard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[28] M. van Emden, “Algorithmic power from declarative use of redundant constraints,” *Constraints*, vol. 4, no. 4, pp. 363–381, 1999.

[29] P. van Hentenryck, L. Michel, and Y. Deville, *Numerica—A Modelling Language for Global Optimization*. Cambridge, MA: MIT Press, 1997.

Online Trajectory Generation: Straight-Line Trajectories

Torsten Kröger, *Member, IEEE*

***Abstract*—A concept of online trajectory generation for robot motion control systems that enables instantaneous reactions to unforeseen sensor events was introduced in a former publication. This concept is now extended with the important feature of homothety. Homothetic trajectories are 1-D straight lines in a multidimensional space and are relevant for all straight-line motion operations in robotics. This paper clarifies 1) how online concepts can be used to generate homothetic trajectories and 2) how we can instantaneously react to (sensor) events with homothetic trajectories. To underline the practical relevance, real-world experimental results with a seven-degree-of-freedom (DOF) robot arm are shown.**

***Index Terms*—Homothety, online trajectory generation (OTG), robot motion control, sensor integration.**

I. Introduction and Problem Formulation

Homothetic trajectories belong to the most common ones in commercially available robotic manipulator control. They represent a motion along a 1-D straight line in a multidimensional space (Euclidian space, Euler space, spherical coordinates, joint space, etc.). If we consider a mechanical system with multiple degrees of freedom (DOFs) that is equipped with one or more sensors delivering digital and/or analog sensor signals, it is an essential feature to *instantaneously* react to *unforseen* sensor signals and events. In [1], a framework for the online generation of time-synchronized robot motion trajectories was introduced, which generates trajectories from *arbitrary* initial states of motion. This paper now extends the framework of [1] by enabling phase-synchronized (homothetic) trajectories that are generated within low-level control cycle (typically, 1 ms or less).

Manuscript received December 23, 2010; revised April 13, 2011; accepted May 18, 2011. Date of publication June 16, 2011; date of current version October 6, 2011. This paper was recommended for publication by Associate Editor A. Albu-Schäffer and Editor J.-P. Laumond upon evaluation of the reviewers' comments. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation).

The author is with the Artificial Intelligence Laboratory, Stanford University, Stanford, CA 94305-9010 USA (e-mail: tkr@stanford.edu).

Digital Object Identifier 10.1109/TRO.2011.2158021

Let us define a trajectory $\mathcal{M}_i(t)$, which is calculated at a discrete-time instant T_i, as

$$\mathcal{M}_i(t) = \left\{ \left({}^{1}\mathbf{m}_i(t),\, {}^{1}\mathcal{V}_i\right),\, \ldots,\, \left({}^{l}\mathbf{m}_i(t),\, {}^{l}\mathcal{V}_i\right) \right. \\ \left. \ldots,\, \left({}^{L}\mathbf{m}_i(t),\, {}^{L}\mathcal{V}_i\right) \right\} \quad (1)$$

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

$$ {}^{l}\mathbf{m}_i(t) = \left({}^{l}\vec{p}_i(t),\, {}^{l}\vec{v}_i(t),\, {}^{l}\vec{a}_i(t),\, {}^{l}\vec{j}_i(t)\right) \\ = \left({}^{l}_{1}\vec{m}_i(t),\, \ldots,\, {}^{l}_{k}\vec{m}_i(t),\, \ldots,\, {}^{l}_{k}\vec{m}_i(t)\right)^T . \quad (2)$$

Here, K is the total number of DOFs, and a trajectory segment l of a single DOF k is described by the motion polynomials

$$ {}^{l}_{k}\vec{m}_i(t) = \left({}^{l}_{k}p_i(t),\, {}^{l}_{k}v_i(t),\, {}^{l}_{k}a_i(t),\, {}^{l}_{k}j_i(t)\right) \quad (3)$$

where ${}^{l}_{k}p_i(t)$ represents the position progression, ${}^{l}_{k}v_i(t)$ represents the velocity progression, ${}^{l}_{k}a_i(t)$ represents the acceleration progression, and ${}^{l}_{k}j_i(t)$ represents the jerk progression. According to (1), a complete trajectory is described by L segments, and each segment l is accompanied by a set of time intervals

$$ {}^{l}\mathcal{V}_i = \left\{{}^{l}_{1}\vartheta_i,\, \ldots, {}^{l}_{k}\vartheta_i,\, \ldots,\, {}^{l}_{k}\vartheta_i\right\} \quad \text{where} \quad {}^{l}_{k}\vartheta_i = \left[{}^{l-1}_{k}t_i,\, {}^{l}_{k}t_i\right] \quad (4)$$

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

In [2], a good introduction about homothety is given, and in [3], it is applied to robot trajectory generation. To generate homothetic trajectories in the K-dimensional space, we can take an arbitrary DOF, i.e., $\kappa \in \{1, \ldots, K\}$, as the reference DOF and design the trajectory parameters, such that the condition

$$\forall\,(k,l) \in \{1,\ldots,K\} \times \{1,\ldots,L\} \\ {}^{l}_{k}v_i(t) = {}_{k}\varrho_i \cdot {}^{l}_{\kappa}v_i(t) \quad \text{with} \quad t \in {}^{l}_{k}\vartheta_i \quad (5)$$

is fulfilled. This naturally also implies that

$$\forall\,(k,l) \in \{1,\ldots,K\} \times \{1,\ldots,L\} \\ \left.\begin{array}{l} {}^{l}_{k}a_i(t) = {}_{k}\varrho_i \cdot {}^{l}_{\kappa}a_i(t) \\ {}^{l}_{k}j_i(t) = {}_{k}\varrho_i \cdot {}^{l}_{\kappa}j_i(t) \\ {}^{l}_{k}d_i(t) = {}_{k}\varrho_i \cdot {}^{l}_{\kappa}d_i(t) \end{array}\right\} \quad \text{with} \quad t \in {}^{l}_{k}\vartheta_i \quad (6)$$

are fulfilled. The constant vector

$$\vec{\varrho}_i = \left({}_{1}\varrho_i,\, \ldots,\, {}_{k}\varrho_i,\, \ldots,\, {}_{k}\varrho_i\right)^T \quad \text{with} \quad {}_{\kappa}\varrho_i = 1 \quad (7)$$

defines the ratios between the reference DOF κ and all other DOFs $\{1, \ldots, K\}\backslash\{\kappa\}$. Usually, homothetic trajectories are generated as described by (5)–(7): A scalar function specifies the velocity progression for one DOF, which is referred to as a reference DOF, and the motion of the other DOFs is calculated by the use of $\vec{\varrho}_i$ [3].

Fig. 1 illustrates the path of a simple 2-DOF point-to-point motion with zero velocities in $\vec{P}_0$ and $\vec{P}_0^{\,\mathrm{trgt}}$. One can clearly recognize the differences between the phase-synchronized (homothetic), time-synchronized (cf., [1]), and nonsynchronized trajectories. In particular, the elements of the kinematic motion constraints

$$\mathbf{B}_i = \left(\vec{V}_i^{\max},\, \vec{A}_i^{\max},\, \vec{J}_i^{\max},\, \vec{D}_i^{\max}\right) \quad (8)$$

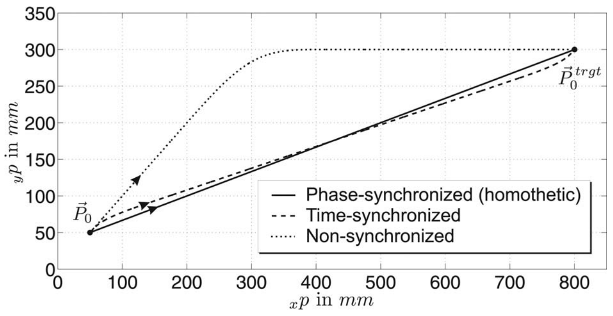


Fig. 1. *XY*-plot of a plain 2-DOF point-to-point motion: phase synchronized, time synchronized, and without synchronization.

are the same for both DOFs, such that the *nonsynchronized* motion starts with an angle of 45° with respect to the reference frame. The *time-synchronized* one starts in $\vec{P}_0$ and ends in $\vec{P}_0^{\text{trgt}}$ with an angle of 45°. In (8), $\vec{D}_i^{\max}$ contains the maximum values of the derivative of jerk; holonomic constraints are not part of $\mathbf{B}_i$ and are not considered in this paper.

The problem that is addressed in this paper is the *online* generation of homothetic trajectories, i.e., a new motion state is calculated at each single low-level control cycle in order to be capable to *instantaneously* react to *unforeseen* events that are triggered by sensor signals. In [1], an algorithm that addresses this problem for time-synchronized trajectories was suggested, which will be used as the basis for this paper. After a discussion of related works, Section III briefly repeats and introduces the basic concept of [1], Section IV introduces the extended algorithm for straight-line motions, and Section V presents real-world experimental results of a 6-DOF robot task.

II. Related Work

The works that are most related to this paper are [4]–[10], all of which belong to the fields of robot motion control [11] and trajectory generation [12], [13] in robotic systems. Macfarlane *et al.* [4] 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 [5], Cao *et al.* use rectangular jerk pulses to compute trajectories, but accelerations that are different from zero cannot be applied. Compared with the multi-DOF approach that is presented here, the latter method has been developed for *1-D problems* only. Broquère *et al.* [6] 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 [7] and is based on the classic seven-segment acceleration profile [14]. With regard to [15], it is a Type V online trajectory generation (OTG) approach that is designed to handle several DOFs individually. Another very recent concept was proposed by Haddadin *et al.* [10]. Instead of generating motion trajectories, virtual springs and damping elements are the setups that are used as input values for a Cartesian impedance controller of the robot.

A disadvantage of [4], [5], and [7] is that they cannot cope with initial acceleration values that are unequal to zero. A further, recent work of Haschke *et al.* [8] presents an online trajectory planner in the very same sense as [1] 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.* [9] proposed a work for the online calculation of 1-D motion trajectories for any given state of motion and

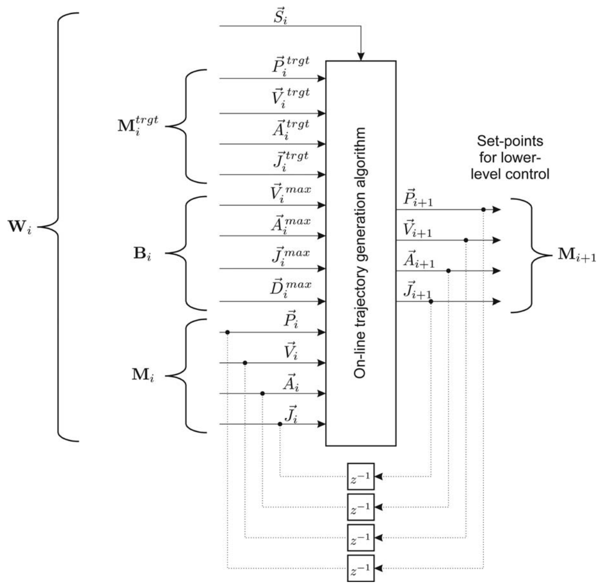


Fig. 2. Input and output values of the Type IX OTG algorithm [1].

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. The major drawback of this work is that no kinematic motion constraints, such as maximum velocity, acceleration, and jerk values, can be specified.

III. Basic Concept

For a better understanding of the approach that is presented in this paper, the basic algorithm for OTG [1], [15] is briefly summarized in this section.

Fig. 2 shows the input and output values of the OTG algorithm in a generic manner (cf., [1]). It is the task of the algorithm to time-optimally transfer an arbitrary current state of motion

$$\mathbf{M}_i = \left(\vec{P}_i, \vec{V}_i, \vec{A}_i, \vec{J}_i\right) \tag{9}$$

into the desired target state of motion

$$\mathbf{M}_i^{\text{trgt}} = \left(\vec{P}_i^{\text{trgt}}, \vec{V}_i^{\text{trgt}}, \vec{A}_i^{\text{trgt}}, \vec{J}_i^{\text{trgt}}\right) \tag{10}$$

under consideration of the kinematic motion constraints $\mathbf{B}_i$ [see (8)]. The algorithm works memoryless and calculates only the next state of motion $\mathbf{M}_{i+1}$, which is used as the input value for lower level motion controllers. The resulting trajectories are time optimal and synchronized, such that all selected DOFs simultaneously reach their target state of motion. The selection vector $\vec{S}_i$ contains Boolean values to mask single DOFs, for which no output values are calculated. All types and variant of the algorithm consist of three steps, which are introduced in the following.

A. Step 1: Calculate the Synchronization Time t_i^{sync}

Although only one single scalar value is calculated in this step, it the the most complex one. First, the minimum execution times $_k t_i^{\min}$ are calculated for each selected DOF $k \in \{1, \ldots, K\}$. The value of the minimum synchronization time t_i^{sync} must be equal to or greater than the maximum value of all minimum execution times. To transfer the motion state of one DOF to another, a finite set of motion profiles $\mathcal{P}_{\text{Step1}}$

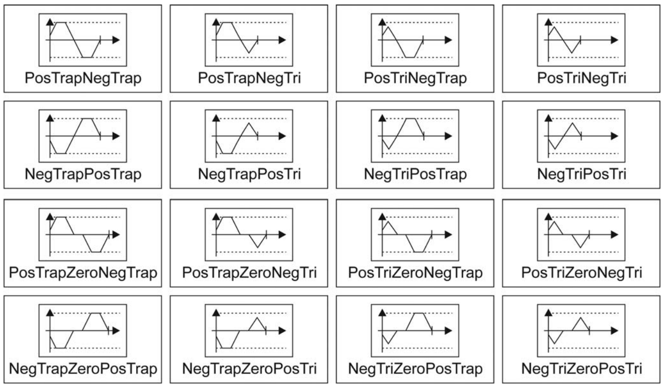


Fig. 3. Subset of the acceleration profile set $\mathcal{P}_{\mathrm{Step1}}$ of Type IV [1]. The dotted horizontal line indicates the maximum acceleration values. These profiles are required to calculate ${}_k t_i^{\min}$, as well as all limits of existing inoperative time intervals ${}_k\mathcal{Z}_i$, for each single selected DOF k.

is considered, and a decision tree selects a motion profile ${}_k\Psi_i^{\mathrm{Step1}} \in \mathcal{P}_{\mathrm{Step1}}$ for each selected DOF $k \in \{1, \ldots, K\}$. To calculate ${}_k t_i^{\min}$, a system of nonlinear equations is set up, and the solution contains the desired value. Fig. 3 exemplarily shows a subset of the set of acceleration profiles for the Type IV OTG algorithm.

Depending on the type of the algorithm (I – IX), it may occur that up to $Z = 3$, time intervals are existent, within which the target state of motion cannot be reached. Such inoperative time intervals may occur if the vectors $\vec{V}_i^{\mathrm{trgt}}$, $\vec{A}_i^{\mathrm{trgt}}$, and $\vec{J}_i^{\mathrm{trgt}}$ are different from zero [cf., [1] and (10)]. Furthermore, decision trees are used to calculate all limits of these inoperative time intervals ${}_k\mathcal{Z}_i$, whose elements are denoted by

$$ {}_k^z\zeta_i = \left[{}_k^z t_i^{\mathrm{begin}},\, {}_k^z t_i^{\mathrm{end}}\right] \quad \text{with} \quad z \in \{1, \ldots, Z\}. \tag{11}$$

Finally, the minimum time not being within any inoperative time interval

$$ {}_k^z\zeta_i \;\forall\, (z, k) \in \{1, \ldots, Z\} \times \{1, \ldots, K\} \tag{12}$$

is selected as the value for the synchronization time t_i^{sync}.

B. Step 2: Synchronization of All Selected Degrees of Freedom

All selected DOFs that did not determine t_i^{sync} have to be synchronized to this time value. In principle, an infinite number of solutions can be found to parameterize a trajectory that transfers such a DOF from ${}_k\vec{M}_i$ to ${}_k\vec{M}_i^{\mathrm{trgt}}$ in t_i^{sync}. In order to achieve a deterministic framework, an optimization criterion must be used; in [1], the most simple variant for time synchronization was suggested, which leads to rectangular jerk signals that only switch between $-{}_kJ_i^{\max}$, zero, and $+{}_kJ_i^{\max}$. This way, another decision tree can be used to select a motion profile ${}_k\Psi_i^{\mathrm{Step2}}$ from another set of motion profiles $\mathcal{P}_{\mathrm{Step2}}$. Again, a system of nonlinear equations can be setup and solved. The solution contains all required trajectory parameters of the synchronized trajectory of DOF k.

C. Step 3: Calculate Output Values

This last step is trivial: The resulting trajectory parameters of Step 2 are used to calculate a new state of motion $\mathbf{M}_{i+1}$, which is used as a set point for lower level controllers at T_{i+1}.

Fig. 4 summarizes the generic version of the OTG algorithm.

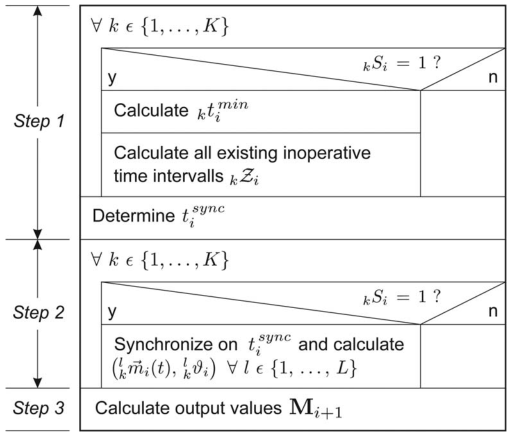

Step 1
$\forall\, k \in \{1, \ldots, K\}$
${}_kS_i = 1$?
y
n
Calculate ${}_k t_i^{min}$
Calculate all existing inoperative time intervalls ${}_k\mathcal{Z}_i$
Determine t_i^{sync}
Step 2
$\forall\, k \in \{1, \ldots, K\}$
${}_kS_i = 1$?
y
n
Synchronize on t_i^{sync} and calculate $\left({}_k^l\vec{m}_i(t),\, {}_k^l\vartheta_i\right)\ \forall\, l \in \{1, \ldots, L\}$
Step 3
Calculate output values $\mathbf{M}_{i+1}$

Fig. 4. Nassi–Shneiderman structogram of the basic OTG algorithm [1].

IV. Algorithm

This paper now extends the proposed concept to be applicable for homothetic (i.e., phase synchronized) robot-motion trajectories.

To generate *phase-synchronized* (homothetic) trajectories online, we have to calculate the internal parameter vector $\vec{\varrho}_i$ online [see (5)–(7)] and adapt the parameters of the kinematic motion constraints $\mathbf{B}_i$ to a matrix $\mathbf{B}_i'$, as will be shown in this paper.

Compared with the general case that is described in [1], the desired homothetic trajectories cannot be generated from arbitrary states of motions. The basic requirement for homothety is that the vectors $\vec{V}_i$, $\vec{A}_i$, $\vec{J}_i$, $\vec{V}_i^{\mathrm{trgt}}$, $\vec{A}_i^{\mathrm{trgt}}$, $\vec{J}_i^{\mathrm{trgt}}$, and $\left(\vec{P}_i^{\mathrm{trgt}} - \vec{P}_i\right)$ are collinear.[1] This is the basic requirement in order to be able to establish a relation to (5)–(7) and to determine the internal parameter vector $\vec{\varrho}_i$.

Condition 1:

$$\exists\, \vec{\gamma} = (\gamma_1, \ldots, \gamma_6) \in \mathbb{R}^6, \quad \text{such that}$$
$$\left(\vec{P}_i^{\mathrm{trgt}} - \vec{P}_i\right) = \gamma_1 \cdot \vec{V}_i = \gamma_2 \cdot \vec{V}_i^{\mathrm{trgt}} = \gamma_3 \cdot \vec{A}_i$$
$$= \gamma_4 \cdot \vec{A}_i^{\mathrm{trgt}} = \gamma_5 \cdot \vec{J}_i = \gamma_6 \cdot \vec{J}_i^{\mathrm{trgt}}. \tag{13}$$

If the input values comply with the condition of (13), we have to select a reference DOF κ, which we use in the following to adapt the respective elements of the matrix $\mathbf{B}_i$. To sustain the feature of kinematic time optimality, we first calculate the execution times of all the selected DOFs

$$\vec{t}_i^{\min} = \left({}_1t_i^{\min}, \ldots, {}_kt_i^{\min}, \ldots, {}_Kt_i^{\min}\right)^T \tag{14}$$

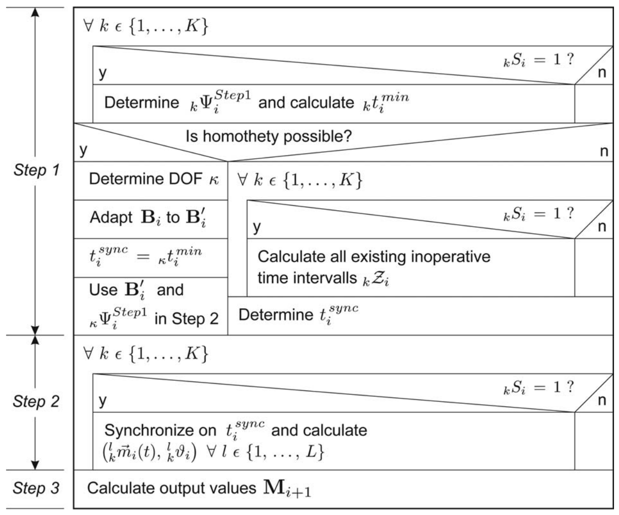
and the respective motion profiles

$$\vec{\Psi}_i^{\mathrm{Step1}} = \left({}_1\Psi_i^{\mathrm{Step1}}, \ldots, {}_k\Psi_i^{\mathrm{Step1}}, \ldots, {}_K\Psi_i^{\mathrm{Step1}}\right)^T \tag{15}$$

$$\text{with} \quad {}_k\Psi_i^{\mathrm{Step1}} \in \mathcal{P}_{\mathrm{Step1}} \quad \forall\, k \in \{1, \ldots, K\} \quad \text{and}$$
$$\mathcal{P}_{\mathrm{Step1}} = \left\{{}^1\Psi^{\mathrm{Step1}}, \ldots, {}^r\Psi^{\mathrm{Step1}}, \ldots, {}^R\Psi^{\mathrm{Step1}}\right\}. \tag{16}$$

The previous equation is taken from [1]; the finite set $\mathcal{P}_{\mathrm{Step1}}$ contains all R possible motion profiles for Step 1 of the OTG algorithm. Fig. 5 shows the generic Nassi–Shneiderman structogram for the algorithms that are presented here. Based on (14), we can determine the maximum element of $\vec{t}_i^{\min}$, which we define as ${}_\kappa t_i^{\min}$, the minimum possible

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



$\forall\, k \,\epsilon\, \{1,\dots,K\}$
$_kS_i = 1\,?$
y
n
Determine $_\kappa\Psi_i^{Step1}$ and calculate $_kt_i^{min}$
Is homothety possible?
Step 1
Determine DOF κ
Adapt $\mathbf{B}_i$ to $\mathbf{B}_i'$
$t_i^{sync} = {}_\kappa t_i^{min}$
Use $\mathbf{B}_i'$ and $_\kappa\Psi_i^{Step1}$ in Step 2
Calculate all existing inoperative time intervals $_k\mathcal{Z}_i$
Determine t_i^{sync}
Step 2
Synchronize on t_i^{sync} and calculate $\left({}_k^l\vec{m}_i(t),\, {}_k^l\vartheta_i\right)\ \forall\, l \in \{1,\dots,L\}$
Step 3
Calculate output values $\mathbf{M}_{i+1}$

Fig. 5. Generic Nassi–Shneiderman structogram for the OTG algorithm for homothetic trajectories (see Fig. 4).

synchronization time for all DOFs. If phase synchronization is possible, κ will become the reference DOF for the current trajectory [cf., (5) – (7)], and the synchronization time $t_i^{\rm sync}$ will be set to ${}_\kappa t_i^{\min}$.

Along with the first requirement of (13), the single vectors of $\mathbf{B}_i$ also have to be collinear to the vectors of (13)

$$\begin{aligned}\exists\, \vec{\gamma} = (\gamma_1, \dots, \gamma_4) \in \mathbb{R}^4, \quad &\text{such that}\\ \left(\vec{P}_i^{\rm trgt} - \vec{P}_i\right) &= \gamma_1 \cdot \vec{V}_i^{\max} = \gamma_2 \cdot \vec{A}_i^{\max}\\ &= \gamma_3 \cdot \vec{J}_i^{\max} = \gamma_4 \cdot \vec{D}_i^{\max}. \end{aligned} \tag{17}$$

Since this is commonly not given, we have to adapt $\mathbf{B}_i$ to $\mathbf{B}_i'$: more precisely, all elements of $\mathbf{B}_i$, except the ones of the reference DOF κ. This is done in two steps. First, we calculate the vector

$$\vec{\varrho}_i = \frac{\vec{P}_i^{\rm trgt} - \vec{P}_i}{\left|{}_\kappa P_i^{\rm trgt} - {}_\kappa P_i\right|} \tag{18}$$

whose κth element ${}_\kappa\varrho_i$ is 1. Subsequently, the adapted elements of $\mathbf{B}_i'$ can be calculated by

$$\begin{aligned}\forall\, k \in \{1,\dots,K\}:\ &{}_kV_i^{\max\,\prime} = {}_k\varrho_i \cdot {}_kV_i^{\max}\ \ {}_kJ_i^{\max\,\prime}\\ &= {}_k\varrho_i \cdot {}_kJ_i^{\max}\ \ {}_kA_i^{\max\,\prime} = {}_k\varrho_i \cdot {}_kA_i^{\max}\ \ {}_kD_i^{\max\,\prime} = {}_k\varrho_i \cdot {}_kD_i^{\max}.\end{aligned} \tag{19}$$

With the matrix

$$\mathbf{B}_i' = \left(\vec{V}_i^{\max\,\prime},\ \vec{A}_i^{\max\,\prime},\ \vec{J}_i^{\max\,\prime},\ \vec{D}_i^{\max\,\prime}\right) \tag{20}$$

the requirement of (17) is fulfilled. In order not to breach the kinematic motion constraint values $\mathbf{B}_i$, the elements of $\mathbf{B}_i'$ must be less than or equal to the original ones of $\mathbf{B}_i$.

Condition 2:

$$\begin{aligned}\forall\, k \in \{1,\dots,K\}:\ &{}_kV_i^{\max\,\prime} \le {}_kV_i^{\max} \wedge {}_kJ_i^{\max\,\prime} \le {}_kJ_i^{\max}\\ &\wedge\ {}_kA_i^{\max\,\prime} \le {}_kA_i^{\max} \wedge {}_kD_i^{\max\,\prime} \le {}_kD_i^{\max}.\end{aligned} \tag{21}$$

If this condition is also fulfilled, one further condition has to be satisfied to be able to generate a homothetic trajectory. All selected DOFs have to be executed with the same motion profile as used for the reference DOF ${}_\kappa\Psi_i^{\rm Step1}$ to comply with (5) and (6). As shown in [1], we can use ${}_\kappa\Psi_i^{\rm Step1}$ to set up a system of equations for each selected DOF. But instead of $\mathbf{B}_i$, $\mathbf{B}_i'$ becomes applicable here together with $\mathbf{M}_i$, $\mathbf{M}_i^{\rm trgt}$, and $\vec{S}_i$. The solution of a system of equations contains the execution time ${}_kt_i^{\min\,\prime}$ for a selected DOF k. For all DOFs except κ, the value differs from ${}_kt_i^{\min}$

$${}_kt_i^{\min\,\prime} \ge {}_kt_i^{\min} \quad \forall\, k \in \{1, \dots, K\}\backslash\{\kappa\} \tag{22}$$

$${}_\kappa t_i^{\min\,\prime} = {}_\kappa\, t_i^{\min} \tag{23}$$

because $\mathbf{B}_i'$ and ${}_\kappa\Psi_i^{\rm Step1}$ are applied, and it may happen that no valid solution can be found for one or more DOFs. In such a case, the profile ${}_\kappa\Psi_i^{\rm Step1}$ cannot transfer $\mathbf{M}_i$ to $\mathbf{M}_i^{\rm trgt}$, and homothety is not possible. To generate a homothetic trajectory, all selected DOFs have to reach their target state of motion synchronously.

Condition 3:

$${}_kt_i^{\min\,\prime} = {}_\kappa t_i^{\min}\ \ \forall\, k \in \{1,\dots,K\} \tag{24}$$

must be fulfilled as a final condition. Equation (24) has to be regarded as theoretical. Because of numerical inaccuracies

$$\left|{}_kt_i^{\min\,\prime} - {}_\kappa t_i^{\min}\right| \le T^{\rm cycle}\ \ \forall\, k \in \{1,\dots,K\} \tag{25}$$

has to be applied in practice. $T^{\rm cycle}$ is the cycle time, i.e., the time interval, in which the OTG algorithm is periodically executed.

As a result, Conditions 1–3 [see (13), (21), and (25)] must be fulfilled to generate a homothetic trajectory. In Fig. 5, this is done in the block “Is homothety possible?”, i.e., if possible, the OTG algorithm generates a homothetic trajectory (left branch); otherwise, only a time-synchronized one is generated (right branch). Here, alternative branches may be possible, as will be discussed in Section VI. In the case of homothety, the profile ${}_\kappa\Psi_i^{\rm Step1}$ and the adapted kinematic motion constraints $\mathbf{B}_i'$ are applied in Step 2 to all selected DOFs in order to calculate homothetic motion parameters for $\mathcal{M}_i(t)$ (cf., (1) and [1]).

This section presented an adaptation of the general OTG algorithm of Section III in order to provide the possibility to generate trajectories along a 1-D straight line in a multidimensional space during runtime, i.e., the possibility to *instantaneously* react to *unforeseen* and *abrupt* set-point switchings is also sustained for these kinds of trajectories. The key feature again is that all proposed algorithms work *online* and can be executed in every low-level control cycle.

V. Experimental Results

Subsequent to the algorithm description, let us now discuss an example of a homothetic and a time-synchronous trajectory that have been achieved with a real-world setup. A *KUKA light-weight robot IV* [17], [18] was set up and controlled through the *fast research interface* at a rate of 1 KHz. The library and the OTG algorithm were executed under real-time conditions on a *QNX* [19] PC node; the translational DOFs x and y, as well as all rotational DOFs, ⓧ, ⓨ, and ⓩ, were controlled by the OTG algorithm, while DOF z was controlled by a force controller (i.e., ${}_zS_i = 0\, \forall\, i \in \mathbb{Z}$). Fig. 6 shows both trajectories, and Fig. 7 shows the corresponding paths in the *XY*-plane. The sensor-guided DOF z is shown in gray, the other two translational DOFs are shown by solid lines, and the rotational DOFs are shown by dashed lines. The beginnings and endings of both motions feature zero velocity and zero acceleration, and the simple task is to move from the

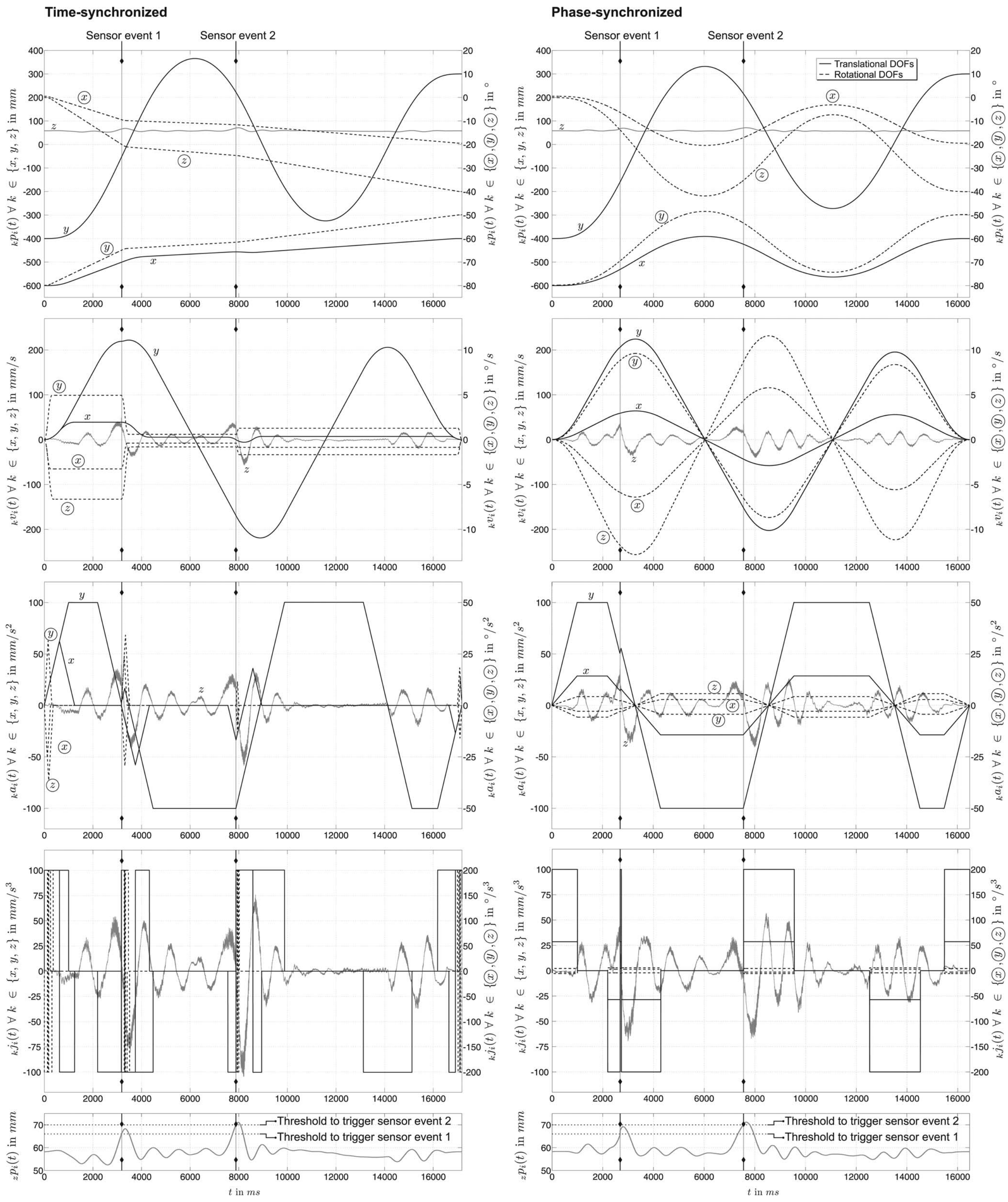


Fig. 6. Position, velocity, acceleration, and jerk progressions of the sample 2-DOF paths that are shown in Fig. 7. The left diagrams depict the time-synchronized *trajectory* (left diagram of Fig. 7), and the right ones depict the homothetically generated *trajectory* (right diagram of Fig. 7). The two sensor-dependent switchings that occur in both cases are triggered by the position signal of the sensor-guided DOF z (gray lines), which are enlarged in the bottom two diagrams. The thresholds (dotted lines) to trigger the switchings are in both cases at 66 mm for sensor event 1 and at 70 mm for sensor event 2. In the time synchronized case, the switchings occur at $t = 3.168$ s and at $t = 7.887$ s and in the phase-synchronized case at $t = 2.686$ s and at $t = 7.554$ s. Although the switchings occur at arbitrary and unforeseen instants, the right trajectory always remains homothetic, which can, in particular, be recognized by means of the jerk progressions and is an important feature for many industrial robot applications. The used Type IV OTG algorithms are part of the *Reflexxes Motion Libraries* [16].

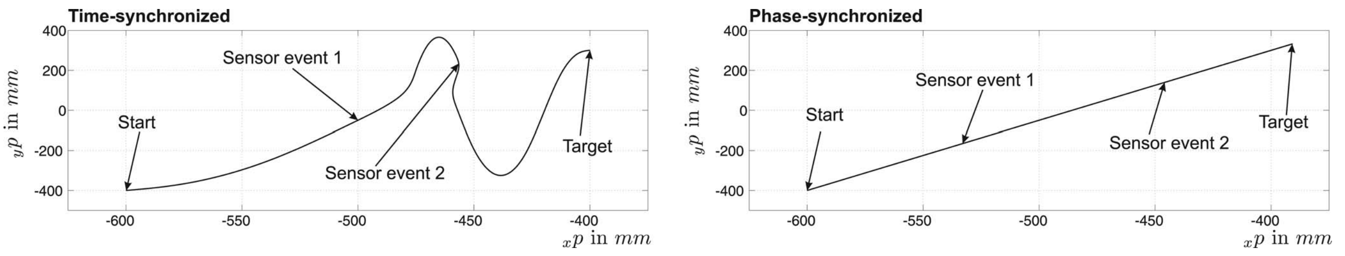


Fig. 7. *XY*-plot of the trajectory that is depicted in Fig. 6. The left part shows the *path* of the time-synchronized trajectory, and the right one shows the *path* of the phase-synchronized (homothetic) trajectory. A straight line results because of the property of homothety.

initial pose

$$\vec{P}_0 = \begin{pmatrix} _xP_0 \\ _yP_0 \\ _zP_0 \\ _{\textcircled{x}}P_0 \\ _{\textcircled{y}}P_0 \\ _{\textcircled{z}}P_0 \end{pmatrix} = \begin{pmatrix} -600\,\text{mm} \\ -400\,\text{mm} \\ 58\,\text{mm} \\ 0^\circ \\ -80^\circ \\ 0^\circ \end{pmatrix}$$

to the target pose

$$\vec{P}_0^{\text{trgt}} = \begin{pmatrix} _xP_0^{\text{trgt}} \\ _yP_0^{\text{trgt}} \\ _zP_0^{\text{trgt}} \\ _{\textcircled{x}}P_0^{\text{trgt}} \\ _{\textcircled{y}}P_0^{\text{trgt}} \\ _{\textcircled{z}}P_0^{\text{trgt}} \end{pmatrix} = \begin{pmatrix} -400\,\text{mm} \\ 300\,\text{mm} \\ 58\,\text{mm} \\ -20^\circ \\ -50^\circ \\ -40^\circ \end{pmatrix}$$

whereas the orientation is represented by roll-pitch-yaw angles.[2] The third DOF z is not considered by the OTG algorithm. The kinematic motion constraints $\mathbf{B}_i$ remain constant and are adapted to $\mathbf{B}'_i$ in the homothetic case [cf., (19) and (20)]. Two sensor-dependent switching points were defined: If the position signal $_zp_i(t)$ exceeds 66 mm (sensor event 1), a target state of motion of

$$\begin{aligned} \vec{P}_i^{\text{trgt}} &= \frac{\vec{P}_0^{\text{trgt}} - \vec{P}_0}{2} \\ \vec{V}_i^{\text{trgt}} &= -\vec{V}_i \\ \vec{A}_i^{\text{trgt}} &= \vec{0} \end{aligned}$$

is set up, and if $_zp_i(t)$ exceeds 70 mm (sensor event 2), the original target state of motion is applied again, i.e., $\vec{P}_0^{\text{trgt}}$, at zero velocity and zero acceleration. In the time synchronized case, the switchings occur at $t = 3.168$ s and $t = 7.887$ s and in the phase-synchronized case at $t = 2.686$ s and $t = 7.554$ s. To illustrate the trigger procedure, the position signal $_zp_i(t)$ was enlarged at the bottom of Fig. 6, and the thresholds were marked by dotted lines. By means of Fig. 7, one can clearly see the difference between both methods of synchronization: the phase-synchronized case results independently from the switching events in a straight line, while the path of the time-synchronized case depends on the switching instants and trajectory parameters. Nevertheless, a continuous jerk-limited trajectory results in both cases.

The used Type IV OTG algorithms are provided as part of the *Reflexxes Motion Libraries* [16].

In the time-synchronized case, the computations for this 6-DOF system requires an average execution time of 135 μs on a single-core machine[3] for the Type IV OTG algorithm; the worst case execution time is 540 μs [1]. The software library was compiled by the use of the QNX Compile Command *QCC* with optimization enabled.[4] In the phase-synchronized case, the worst-case execution time is less than 90 μs on the same hardware. This value is lower, because no inoperative time intervals have to be calculated [cf., (11)], and for Step 2, only one system of nonlinear equations has to be solved. This solution can be used for all selected DOFs by simply multiplying with the scaling vector $\vec{\varrho}_i$ [cf., (7)]. In all cases, the worst case execution times scale linearly with the number of selected DOFs, such that a complexity of $O(n)$ results, where n is the number of DOFs K.

VI. Discussion

One important question that has to be addressed in this context of phase synchronization the following: What happens if the user or the application require a homothetic motion, but one of the three Conditions 1–3 is not fulfilled? In Fig. 5, it is suggested to utilize a time-synchronized trajectory, but this must not be the only alternative. The reason, why one of the conditions is not fulfilled can have several causes, and depending on the current task, one of the following three cases may be considered.

1) $(\vec{P}_i^{\text{trgt}} - \vec{P}_i)$, $\vec{V}_i$, $\vec{A}_i$, and $\vec{J}_i$ are not collinear. In this case, phase synchronization is impossible.
2) $(\vec{P}_i^{\text{trgt}} - \vec{P}_i)$, $\vec{V}_i$, $\vec{A}_i$, and $\vec{J}_i$ are collinear, and $\vec{V}_i^{\text{trgt}}$, $\vec{A}_i^{\text{trgt}}$, and $\vec{J}_i^{\text{trgt}}$ are not collinear to the former vectors. In this case, the desired target state of motion cannot be reached homothetically, but it would be possible to calculate a state of motion that continues the straight-line motion. Another option would be to use the time-synchronized trajectory (see Fig. 5). Which option is selected depends on the current task.
3) $(\vec{P}_i^{\text{trgt}} - \vec{P}_i)$, $\vec{V}_i$, $\vec{A}_i$, $\vec{J}_i$, $\vec{V}_i^{\text{trgt}}$, $\vec{A}_i^{\text{trgt}}$, and $\vec{J}_i^{\text{trgt}}$ are collinear, but the elements of $\mathbf{B}'_i$ exceed the corresponding values of $\mathbf{B}_i$. In this case, we can either accept the exceeding of $\mathbf{B}_i$ (which may lead to a motion that is beyond the kinematic capability of the robot), or we calculate a matrix $\mathbf{B}''_i$, whose columns are collinear to $\vec{\varrho}_i$, and which do not exceed the original motion constraints of $\mathbf{B}_i$. A third option is the utilization of a time-synchronized trajectory (see Fig. 5).

VII. Conclusion

Based on the concept of OTG that is introduced in [1], an extended version has been presented in this paper. This version allows—if possible—the generation of straight-line trajectories in multidimensional space, i.e., homothetic trajectories. This class of trajectories is

[2]Other orientation representations work as well.

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

[4]QNX Neutrino Version 6.5.0; QCC is based on the *GNU Compiler Collection* Version 4.4.2 [20] and optimization option *O* enabled.

very important for many real-world applications, and because of the online motion generation, the robot controllers become able to *instantaneously* react to *unforeseen* (sensor) events and/or set-point switchings, while keeping the motion phase synchronized. This feature plays a very important role in all fields of robotics requiring (multi)sensor integration. Besides the algorithmic concept, real-world experimental results with a 7-DOF robot arm of an online-generated Type IV trajectory were shown in order to support a clear understanding of this new approach.

Acknowledgment

The author would like to thank the anonymous reviewers for their invaluable comments that have helped to improve the quality of this paper. This work was conducted at the Institut für Robotik und Prozessinformatik, Technische Universitaet Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany, which is headed by Prof. F. M. Wahl, to whom the author would like to express his sincere gratitude. Furthermore, the author would like to express his appreciation to Prof. O. Khatib, who is currently hosting him at the Stanford Artificial Intelligence Laboratory, Stanford University, Stanford, CA. The author would also like to appreciate the works of his former diploma students M. Hanisch, C. Hurnaus, and A. Tomiczek, who worked hard on the first ideas and implementations of this concept. Finally, the author would like to thank *QNX Software Systems* and *Wind River Systems* for providing free software licenses.

References

[1] T. Kröger and F. M. Wahl, “On-line trajectory generation: Basic concepts for instantaneous reactions to unforeseen events,” *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 94–111, Feb. 2010.
[2] A. I. Kostrikin, Y. I. Manin, and M. E. Alferieff, “Projective groups and projections,” in *Linear Algebra and Geometry*, 1st ed. New York: Taylor & Francis, 1997, ch. 3.
[3] W. Khalil and E. Dombre, “Trajectory generation,” in *Modeling, Identification and Control of Robots*, 1st ed. London, U.K.: Hermes Penton, 2002, ch. 13, pp. 313–345.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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, pp. 123–128.
[10] 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. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Taipei, Taiwan, Oct. 2010, pp. 3109–3116.
[11] W. Chung, L.-C. Fu, and S.-H. Hsu, “Motion control,” in *Springer Handbook of Robotics,* Ist ed., B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008, ch. 6, pp. 133–159.
[12] L. Biagiotti and C. Melchiorri, “Composition of elementary trajectories,” in *Trajectory Planning for Automatic Machines and Robots,* Ist ed. Berlin, Germany: Springer-Verlag, 2008, ch. 3, pp. 59–150.
[13] L. E. Kavraki and S. M. LaValle, “Motion planning,” in *Springer Handbook of Robotics* Ist ed., B. Siciliano and O. Khatib, Eds. Berlin, Germany: Springer-Verlag, 2008, ch. 5, pp. 109–131.
[14] 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.
[15] T. Kröger, *On-Line Trajectory Generation in Robotic Systems* (Springer Tracts in Advanced Robotics Series 58), 1st ed. Berlin, Heidelberg, Germany: Springer-Verlag, Jan. 2010.
[16] Reflexxes GmbH. (2011, May 18). [Online]. Available: http://www.reflexxes.com
[17] KUKA Roboter GmbH . (2011, Mar. 20). [Online]. Available: http://www.kuka.com/en/company/group
[18] R. Bischoff, J. Kurth, G. Schreiber, R. Köppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, and G. Hirzinger, “The KUKA-DLR lightweight robot arm—A new reference platform for robotics research and manufacturing,” presented at the Joint Conf. ISR 2010 (41st Int. Symp. Robot.), Munich, Germany, Jun. 2010. VDE Verlag.
[19] QNX Software Systems. (2011, Mar. 10). [Online]. Available: http://www.qnx.com
[20] GNU Compiler Collection. (Apr. 13, 2011). Hompepage. [Online]. Available: http://gcc.gnu.org

Path Following for Unicycle Robots With an Arbitrary Path Curvature

Angelo Morro, Antonio Sgorbissa, and Renato Zaccaria

***Abstract*—A new feedback control model is provided that allows a wheeled vehicle to follow a prescribed path. Differently from all other methods in the literature, the method that is proposed neither requires the computation of a projection of the robot position on the path, nor does it need to consider a moving virtual target to be tracked. Nevertheless, it guarantees asymptotic convergence to a generic 2-D curve which can be represented through its implicit equation in the form $f(x,y) = 0$, and it puts no bounds on the initial position of the vehicle, provided that $\nabla f \neq 0$.**

***Index Terms*—Mobile robots, path following, wheeled vehicles.**

I. Introduction

This paper provides a new path following model for a wheeled unicycle vehicle that moves in the 2-D Cartesian space.

Path following has been deeply investigated in the literature. The assumption made in early works [1], [2] is that the vehicle forward speed conforms to a prescribed speed profile, while the controller acts on the vehicle orientation to steer it to the path. To achieve this, the orthogonal projection of the robot position on the path is computed, and the distance and angular error are consequently defined. Next, an asymptotically stable control law is proposed to minimize both errors by controlling the vehicle orientation.

Manuscript received December 9, 2010; accepted April 22, 2011. Date of publication June 9, 2011; date of current version October 6, 2011. This paper was recommended for publication by Associate Editor M. Vendittelli and Editor J.-P. Laumond upon evaluation of the reviewers’ comments.

A. Morro is with the Department of Biophysical and Electronic Engineering, University of Genova, 16145 Genova, Italy (e-mail: angelo.morro@unige.it).

A. Sgorbissa and R. Zaccaria are with Department of Communication, Computer, and System Sciences, University of Genova, 16145 Genova, Italy (e-mail: antonio.sgorbissa@unige.it; renato.zaccaria@unige.it).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2011.2148250