Quaternion-based Smooth Trajectory Generator for Via Poses in $SE(3)$ Considering Kinematic Limits in Cartesian Space

Reinhard M. Grassmann and Jessica Burgner-Kahrs, *Senior Member, IEEE*

***Abstract*—Smooth position and orientation interpolation has a great effect on the performance of robot manipulators. Interpolation between several via positions can be done in a straightforward manner, which is well covered in the literature. However, generating a suitable trajectory between several orientations is still an open problem. In this paper, we introduce a novel trajectory generator capable of respecting kinematic limits. We address the problem of generating a singularity-free trajectory for multiple via poses in $SE(3)$, while complying with the requirement of $\mathcal{C}^4$ continuity. To achieve this, a smooth trapezoidal-like velocity profile and unit quaternions are used. A simulation platform in V-REP based on a 7-DOF (degree of freedom) lightweight robot including inverse kinematics and dynamics is used to demonstrate the effectiveness of our trajectory generator.**

***Index Terms*—Flexible Robots, Formal Methods in Robotics and Automation, Industrial Robots, Motion and Path Planning**

I. INTRODUCTION

REGARDING human-robot-collaboration, generated trajectories of a robot manipulators should be intuitive and predictable for human co-workers. Therefore, the geometric path of the position and orientation is planned in Cartesian space. Furthermore, it is important to guarantee accurate motion to perform a task in the presence of human co-workers and for general application. Hence, kinematic limits have to be considered. Smooth position interpolation between several via points is straightforward. Surprisingly, smooth orientation interpolation in $SO(3)$ between several orientations, while complying with the requirement of being smooth and respecting kinematic limits, is still an open problem. A curve constructed in $\mathbb{R}^3$ may not turn out smoothly in $SO(3)$ because many common geometric approaches do not properly account for the geometry of $SO(3)$. Consequently, no satisfactory solution to pose interpolation in $SE(3)$ combining position and orientation has been presented yet.

A. Related Work

The generation of trajectories is well covered by standard robotics textbooks, such as [4] and [23]. Moreover, it is extensively studied in [3] and [15]. However, most of the trajectories in literature are designed for industrial robots and show at most $\mathcal{C}^2$ smoothness or less. As stated in [7], for manipulators with mechanical flexibility, such as common lightweight robots, e.g. [11] and [1], the minimum requirement for the exact reproducibility of the desired trajectory is that it admits a continuously differentiable jerk, i.e. it is at least $\mathcal{C}^4$ smooth. A smooth trajectory is an essential requirement for avoiding structural oscillations, [3], [25], improving accuracy in tracking of the end effector [7], and reducing energy consumption [14], [20]. Smooth trajectories are also perceived more natural [25], which is beneficial in the context of human-robot-collaboration. Further, trajectories in Cartesian space are mostly designed for position neglecting orientation.

Interpolation of orientation is only rarely treated. According to [3], interpolation is often based on a set of three angles, i.e. Euler angles. Euler angles are a widely used $SO(3)$ representation. However, the interpolation of each of the twelve sets of Euler angles can result in a singularity, i.e. Gimbal Lock. Furthermore, it leads to undesirable trajectories which are neither intuitive nor predictable for human co-workers. Regarding kinematic limits, Euler angles cannot express kinematic limits in algebraic form which makes it an undesirable orientation parameterization [16].

It is indeed well-known and accepted that quaternions are best suited for representing [24], [5] as well as interpolating [9] orientations. Therefore, SLERP (spherical linear interpolation) [22], the gold-standard for interpolation between two orientations, has been extended to SQUAD (spherical and quadrangle), which is applied to an industrial robot in [17]. However, SQUAD generates $\mathcal{C}^1$ smooth orientation trajectories [5]. Further, it is unclear how to consider kinematic limits. Other adaptation of SLERP can be found in [6] and in [26]. In [18] a method based on the exponential mapping of quaternions where its argument is evaluated by B-splines is proposed. In [18], however, only path planning is discussed and a quadratic optimization problem has to be formulated. In [21] an algorithm to generate $\mathcal{C}^2$ smooth interpolations based on quaternionic polynomials is proposed. The component-wise quaternion interpolation treats quaternions as a vector in Euclidean space leading to a wrong treatment of the special orthogonal group $SO(3)$ and enforcing a re-normalization at each execution. In computer graphics, several interpolation methods, e.g. [2], [13], [19], can be found. However, none of these approaches are suitable for robotic application.

Generally, in order to design the desired path, geometric paths for the translation and orientation components are defined separately and synchronized afterwards. Without a separate definition, an undesired screw motion could be generated and its translation component is generally not a straight path [25]. Hence, approaches in $SE(3)$ without separation into $E(3)$ and $SO(3)$ are neglected in this paper.

Manuscript received: February 24, 2019; Revised May 26, 2019; Accepted June 27, 2019.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments.

Both authors are with Continuum Robotics Laboratory, Department of Computer Science, University of Toronto, Toronto, Canada reinhard.grassmann@utoronto.ca

This work has been conducted, when all authors were still with Laboratory for Continuum Robotics at Gottfried Wilhelm Leibniz Universität Hannover, Hanover 30167, Germany.

Digital Object Identifier (DOI): see top of this page.

To the best of our knowledge, no pose interpolator satisfying the requirement of smooth interpolation across multiple orientations and positions exists. Furthermore, no orientation interpolation, which can consider a priori kinematic limits, is available.

B. Problem Statement

This paper assumes that a sequence of via poses in Cartesian space is available. The studied problem is how to generate a trajectory such that the robot can pass through the given via poses, while complying with the requirement of being $\mathcal{C}^4$ smooth and respecting kinematic limits.

C. Contribution

In the present paper, we introduce an approach for pose interpolation, where a sequence of via poses in $SE(3)$ is given. We propose a blending method leading to a trajectory generator respecting defined kinematic limits. In particular, the following contributions are made:

- A formalism extending the trajectory generator in [10] to via poses and to asymmetric kinematic limits w.r.t. acceleration and deceleration is provided.
- A blending method bridging two overlapping segments for a $\mathcal{C}^4$ smooth trapezoidal-like velocity profile with three segments is developed.
- A pose interpolator taking into account kinematic limits such as maximum velocity, acceleration, and deceleration is proposed.
- Generation of a geometric path on a unit-hypersphere in $SO(3)$ without requiring re-normalization and numeric optimization being suitable for real time applications.
- A framework, which is straightforward to adapt to joint space interpolation and to varied smoothness requirements.

II. Methods

First, we define and extend the point-to-point (p2p) trajectory for single DOF (degree of freedom) based on [10]. Second, the quaternion interpolation is recapped. Third, we show how to synchronize arbitrary DOF. Afterwards, the blending method is described in order to interpolate between intermediate via poses. This section ends with assembling the $SE(3)$ trajectory generator.

A. Motion Law with $\mathcal{C}^4$ smoothness

In contrast to [10], the present motion law is asymmetric w.r.t. acceleration and deceleration. Moreover, we introduce variables, i.e. $C_{\mathrm{a},n}$ and $C_{\mathrm{j},n}$, in order to generalize our approach to $\mathcal{C}^n$ smoothness, where $n \in \mathbb{N} \setminus 0$.

The following motion law is based on a trapezoidal-like velocity profile with three segments. They are denoted as *lift-off*, *cruise*, and *set-down*. As index we use the abbreviation *lo*, *cr*, and *sd*, respectively.

In order to achieve a C^n smooth trajectory, the velocity profile is designed with C^{n-1} polynomial functions for the first and third segment and a constant second segment with durations T_{lo}, T_{sd} and T_{cr}, respectively. Furthermore, $T \coloneqq T_{\mathrm{lo}} + T_{\mathrm{cr}} + T_{\mathrm{sd}}$ is the total trajectory duration. The coefficients a_i of the normalized polynomial $v_{\mathrm{N},n}(\tau) \in [0,1]$ depend on the chosen smoothness and fulfill the null boundary condition, meaning that any order of time derivation of $v_{\mathrm{N},n}(\tau)$ is zero at $\tau = 0$ and $\tau = 1$. The time primitive $\tau \in [0,1]$ is defined piece-wise by

$$\tau(t) = \begin{cases} \dfrac{t}{T_{\mathrm{lo}}} & \text{for } 0 \le t < T_{\mathrm{lo}} \\ \dfrac{t - T_{\mathrm{lo}}}{T_{\mathrm{cr}}} & \text{for } T_{\mathrm{lo}} \le t < T_{\mathrm{lo}} + T_{\mathrm{cr}} \\ \dfrac{t - T_{\mathrm{lo}} - T_{\mathrm{cr}}}{T_{\mathrm{sd}}} & \text{for } T_{\mathrm{lo}} + T_{\mathrm{cr}} \le t \le T \end{cases}, \tag{1}$$

where the durations of the respective segments T_{lo}, T_{cr}, and T_{sd} are properly defined in (12), (15), and (13), respectively.

In order to achieve a C^4 smooth trajectory, we provide $\mathcal{C}^3$ smooth *lift-off* and *set-down* velocity segments including

$$v_{\mathrm{N},4}(\tau) = -20\tau^7 + 70\tau^6 - 84\tau^5 + 35\tau^4. \tag{2}$$

The coefficients of $v_{\mathrm{N},n}$ up to $n = 11$ are summarized in [3]. Now, the three segments of the velocity profile $v(t)$ can be defined. Considering (2), $v(t)$ is piece-wise designed as

$$v(t) = \begin{cases} 0 & \text{for } t < 0 \\ \operatorname{sign}(L)\, v_{\mathrm{lo}}(t) & \text{for } 0 \le t < T_{\mathrm{lo}} \\ \operatorname{sign}(L)\, v_{\mathrm{cr}}(t) & \text{for } T_{\mathrm{lo}} \le t < T_{\mathrm{lo}} + T_{\mathrm{cr}} \\ \operatorname{sign}(L)\, v_{\mathrm{sd}}(t) & \text{for } T_{\mathrm{lo}} + T_{\mathrm{cr}} \le t \le T \\ 0 & \text{for } T < t, \end{cases} \tag{3}$$

where $\operatorname{sign}(\cdot)$ gives the sign of its argument and L is the total displacement of the trajectory. The velocities are defined by

$$v_{\mathrm{lo}}(t) = \lambda v_{\max} v_{\mathrm{N},4}(\tau(t)), \tag{4}$$

$$v_{\mathrm{cr}}(t) = \lambda v_{\max}, \text{ and} \tag{5}$$

$$v_{\mathrm{sd}}(t) = \lambda v_{\max} v_{\mathrm{N},4}(1 - \tau(t)), \tag{6}$$

where $v_{\max} > 0$ is the maximum velocity and λ is a scaling factor defined in (16). Note that the velocity v_{cr} is constant over the interval $[T_{\mathrm{lo}}, T_{\mathrm{lo}} + T_{\mathrm{cr}}]$.

The position profile is obtained by integrating (3) with (4), (5), and (6), leading to the piece-wise motion law

$$s(t) = \begin{cases} 0 & \text{for } t < 0 \\ \operatorname{sign}(L)\, s_{\mathrm{lo}}(t) & \text{for } 0 \le t < T_{\mathrm{lo}} \\ \operatorname{sign}(L)\, s_{\mathrm{cr}}(t) & \text{for } T_{\mathrm{lo}} \le t < T_{\mathrm{lo}} + T_{\mathrm{cr}} \\ \operatorname{sign}(L)\, s_{\mathrm{sd}}(t) & \text{for } T_{\mathrm{lo}} + T_{\mathrm{cr}} \le t \le T \\ L & \text{for } T < t, \end{cases} \tag{7}$$

with

$$s_{\mathrm{lo}}(t) = \lambda v_{\max} T_{\mathrm{lo}} V_{\mathrm{N},4}(\tau(t)), \tag{8}$$

$$s_{\mathrm{cr}}(t) = \lambda v_{\max}(T_{\mathrm{cr}}\tau(t) + T_{\mathrm{lo}}/2), \text{ and} \tag{9}$$

$$s_{\mathrm{sd}}(t) = |L| - \lambda v_{\max} T_{\mathrm{sd}} V_{\mathrm{N},4}(1 - \tau(t)), \tag{10}$$

where $|L|$ is the absolute value of the total displacement L and $V_{\mathrm{N},4}$ obtained by integrating (2) w.r.t. τ is defined as

$$V_{\mathrm{N},4}(\tau) = -2.5\tau^8 + 10\tau^7 - 14\tau^6 + 7\tau^5. \tag{11}$$

Note that $V_{\mathrm{N},4} \in [0, 0.5]$ is used on an interval $[0, 1]$.

The durations of the respective segments are dependent on the $\mathcal{C}^n$ smoothness and used $V_{\mathrm{N},n}$. Considering a $\mathcal{C}^4$ smooth trajectory, the durations are determined as follows. Computing the maximum derivative of (4), which is an acceleration and can be set equal to $\lambda a_{\max}$, leads to

$$T_{\mathrm{lo}} = C_{\mathrm{a},4} \frac{v_{\max}}{a_{\max}}, \tag{12}$$

where $a_{\max} > 0$ is the maximum acceleration. Analogously,

$$T_{\mathrm{sd}} = C_{\mathrm{a},4} \frac{v_{\max}}{d_{\max}} \tag{13}$$

determines the duration for the set-down segment, where $d_{\max} > 0$ is the maximum deceleration. The scaling factor λ cancels out for the lift-off and set-down segment, leading to a constant duration w.r.t. λ. In (12) and (13) a special case of $C_{\mathrm{a},n}$ (cf. Tab. I) is used, which is defined as

$$C_{\mathrm{a},4} = \frac{35}{16}. \tag{14}$$

The duration T_{cr} for the constant velocity part is given by

$$T_{\mathrm{cr}} = \frac{|L|}{\lambda v_{\max}} - \frac{1}{2}\left(\frac{1}{a_{\max}} + \frac{1}{d_{\max}}\right) v_{\max} C_{\mathrm{a},4}. \tag{15}$$

Equation (15) can be found by combining and reordering the sum of the lengths, which can be obtained from (8)-(10).

In case L is too short such that the maximum velocity $v_{\max}$ cannot be reached, $v_{\max}$ is automatically adapted by the scaling factor λ, which is defined as

$$\lambda = \begin{cases} 1 & \text{for } T_{\mathrm{cr}}(\lambda = 1) \geq 0 \\ \dfrac{2 a_{\max} d_{\max} |L|}{(a_{\max} + d_{\max})\, v_{\max}^2 C_{\mathrm{a},4}} & \text{otherwise.} \end{cases} \tag{16}$$

Therefore, if $v_{\max}$, $a_{\max}$, and $d_{\max}$ can be reached, the cruise segment with constant velocity, i.e. $v_{\max}$, is executed, otherwise the velocity profile is bell-shaped. Note that the executed velocity is scaled directly (cf. (3)), while the executed acceleration and deceleration is scaled indirectly (compare with the derivation of (12) and (13)).

In order to compute the displacements of each segment, we may evaluate the integral of (8) and (11) as well as subsequently substitute (12)-(15). Consequently, the displacements

$$L_{\mathrm{lo}} = \lambda C_{\mathrm{a},4} {v_{\max}}^2 \left(2 a_{\max}\right)^{-1}, \tag{17}$$

$$L_{\mathrm{sd}} = \lambda C_{\mathrm{a},4} {v_{\max}}^2 \left(2 d_{\max}\right)^{-1}, \text{and} \tag{18}$$

$$L_{\mathrm{cr}} = |L| - L_{\mathrm{lo}} - L_{\mathrm{sd}} \tag{19}$$

can be computed accordingly. Now, we can compute all necessary durations and displacements of each segment for given kinematic limits and total displacement defined by the start and goal state.

In Fig 1, trajectories generated by the motion law $s(t)$ are depicted. Note that $s(t)$ is not a path primitive since it is not generally defined for $[0, 1]$ but for $[0, |L|]$ instead, as it is not independent of L.

In order to consider the maximum jerk $j_{\max}$, we compute the required times $T_{\mathrm{lo,jerk}}$ and $T_{\mathrm{sd,jerk}}$. For the sake of simplicity, we assume the same absolute value of the maximum jerk for the lift-off and set-down segment being $j_{\max}$. In the following,

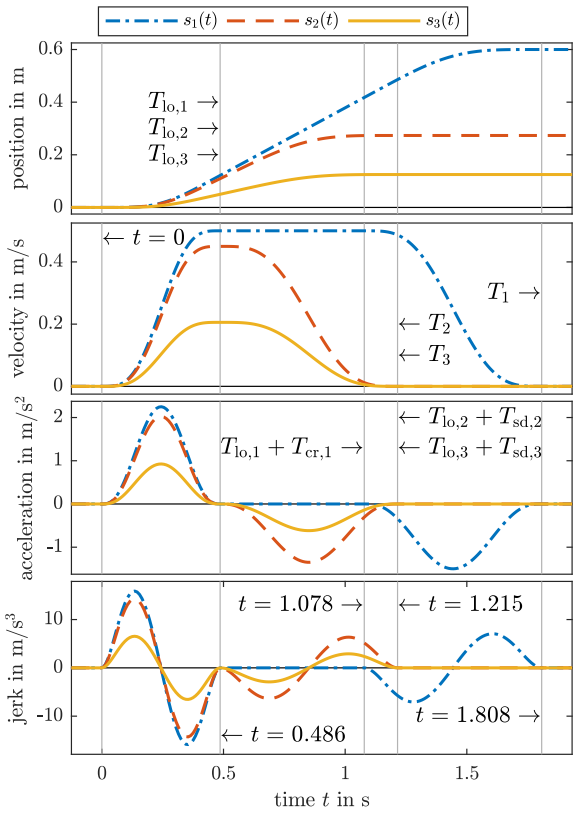

Fig. 1. Course of position, velocity, acceleration, and jerk of $s_i(t)$ with $\mathcal{C}^4$ smoothness and different total displacements $L_1 = 0.6\,\mathrm{m}$, $L_2 = 0.2734\,\mathrm{m}$ and $L_3 = 0.125\,\mathrm{m}$. The first trajectory $s_1(t)$ has a constant velocity segment, while the second $s_2(t)$ does not. The third trajectory $s_3(t)$ is even shorter and cannot reach $v_{\max} = 0.5\,\mathrm{m/s}$ without violating $a_{\max} = 2.25\,\mathrm{m/s^2}$ and $d_{\max} = 1.5\,\mathrm{m/s^2}$. Therefore, $v_{\max}$ is scaled by λ. The longer duration guarantees that the kinematic limits $a_{\max}$ and $d_{\max}$ are not violated. As can be seen, all three trajectories have a smooth jerk profile.

we denote (12) and (13) by $T_{\mathrm{lo,acc}}$ and $T_{\mathrm{sd,dec}}$, respectively. Analogous to (12) and to (13) we may calculate

$$T_{\mathrm{lo,jerk}} = C_{\mathrm{j},4} \sqrt{\frac{v_{\max}}{j_{\max}}} \text{ and} \tag{20}$$

$$T_{\mathrm{sd,jerk}} = C_{\mathrm{j},4} \sqrt{\frac{v_{\max}}{j_{\max}}}, \tag{21}$$

where $C_{\mathrm{j},4}$ is a special case of $C_{\mathrm{j},n}$ (cf. Tab. I), which is

$$C_{\mathrm{j},4} = \sqrt{\frac{84}{5\sqrt{5}}}. \tag{22}$$

The new duration of segment *lift-off* and *set-down* are then

$$T_{\mathrm{lo}} = \max\left(T_{\mathrm{lo,jerk}}, T_{\mathrm{lo,acc}}\right) \text{ and} \tag{23}$$

$$T_{\mathrm{sd}} = \max\left(T_{\mathrm{sd,jerk}}, T_{\mathrm{sd,dec}}\right), \tag{24}$$

respectively. Note that only either $a_{\max}$ or $j_{\max}$ can be reached and analogously for $d_{\max}$.

In case that (20) or (21) is larger than (12) and (13),

TABLE I
DIMENSIONLESS CONSTANTS $C_{\mathrm{a},n}$ AND $C_{\mathrm{j},n}$ FOR VARIED $\mathcal{C}^n$ SMOOTHNESS DEPENDING ON $v_{\mathrm{N},n}$.

	n									
	2	3	4	5	6	7	8	9	10	11
$C_{\mathrm{a},n}$	$\frac{3}{2}$	$\frac{15}{8}$	$\frac{35}{16}$	$\frac{315}{128}$	$\frac{693}{256}$	$\frac{3003}{1024}$	$\frac{6435}{2048}$	$\frac{109395}{32768}$	$\frac{230945}{65536}$	$\frac{969969}{262144}$
$C_{\mathrm{j},n}$	$\sqrt{6}$	$\sqrt{\frac{10}{\sqrt{3}}}$	$\sqrt{\frac{84}{5\sqrt{5}}}$	$\sqrt{\frac{1215}{49\sqrt{7}}}$	$\sqrt{\frac{24640}{2187}}$	$\sqrt{\frac{2559375}{58564\sqrt{11}}}$	$\sqrt{\frac{20207880}{371293\sqrt{13}}}$	$\sqrt{\frac{2002033033}{30375000\sqrt{15}}}$	$\sqrt{\frac{32051036160}{410338673\sqrt{17}}}$	$\sqrt{\frac{98891016919695}{1086948034624\sqrt{19}}}$

respectively, $v_{\max}$, $a_{\max}$, and $d_{\max}$, are recalculated by

$$v_{\max} = \frac{|L|}{\lambda\left(T_{\mathrm{cr}} + 0.5T_{\mathrm{lo}} + 0.5T_{\mathrm{sd}}\right)}, \tag{25}$$

$$a_{\max} = \frac{|L|}{\lambda\left(T_{\mathrm{cr}} + 0.5T_{\mathrm{lo}} + 0.5T_{\mathrm{sd}}\right)} \frac{C_{\mathrm{a},4}}{T_{\mathrm{lo}}}, \text{ and} \tag{26}$$

$$d_{\max} = \frac{|L|}{\lambda\left(T_{\mathrm{cr}} + 0.5T_{\mathrm{lo}} + 0.5T_{\mathrm{sd}}\right)} \frac{C_{\mathrm{a},4}}{T_{\mathrm{sd}}}, \text{ respectively,} \tag{27}$$

which can be derived from (12), (13), and (15), respectively.

B. Quaternion-based Trajectory Generator

We provide a self-contained description of orientation interpolator which was first derived in our previous work [10] including the relation to SLERP [22].

A unit quaternion is a hypercomplex number defined by

$$\xi = \eta + \epsilon_1 \imath + \epsilon_2 \jmath + \epsilon_3 k \tag{28}$$

with property $\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 = 1$, where $\imath$, $\jmath$ and k follow Hamilton's rule $\imath^2 = \jmath^2 = k^2 = \imath\jmath k = -1$ [12]. Hence, it is a linear combination of the real unit 1 and the three quaternionic units $\imath$, $\jmath$ and k with associated real coefficients. A quaternionic conjugate of ξ is defined as

$$\xi^* = \eta - \epsilon_1 \imath - \epsilon_2 \jmath - \epsilon_3 k, \tag{29}$$

which is also the inverse ξ^{-1} of a unit quaternion. The associative and non-commutative product expands to

$$\begin{aligned} \xi\xi' = &\left(\eta + \epsilon_1 \imath + \epsilon_2 \jmath + \epsilon_3 k\right)\left(\eta' + \epsilon_1' \imath + \epsilon_2' \jmath + \epsilon_3' k\right) \\ = &\quad \left(\eta\eta' - \epsilon_1\epsilon_1' - \epsilon_2\epsilon_2' - \epsilon_3\epsilon_3'\right) \\ &+ \left(\eta\epsilon_1' + \eta'\epsilon_1 + \epsilon_2\epsilon_3' - \epsilon_2'\epsilon_3\right)\imath \\ &+ \left(\eta\epsilon_2' + \eta'\epsilon_2 + \epsilon_3\epsilon_1' - \epsilon_3'\epsilon_1\right)\jmath \\ &+ \left(\eta\epsilon_3' + \eta'\epsilon_3 + \epsilon_1\epsilon_2' - \epsilon_1'\epsilon_2\right)k, \end{aligned} \tag{30}$$

where ξ and ξ' are two arbitrary quaternions.

Unit quaternions can be used to represent orientation, while quaternion multiplication defined by (30) can be used to retrieve the result of a rotation. In the context of orientation representation, unit quaternions denoted by

$$\xi(\theta) = \cos(\theta/2) + (n_x\imath + n_y\jmath + n_z k)\sin(\theta/2) \tag{31}$$

are often used, where $\boldsymbol{n} = (n_x, n_y, n_z)^T$ with $\|\boldsymbol{n}\|_2 = 1$ is a fixed axis and θ is the angle between the two given orientations. In the following (31) is used as unit quaternion.

Let ${}^{\text{base}}\xi_{\text{start}}$ and ${}^{\text{base}}\xi_{\text{goal}}$ denote the unit quaternion expressing the orientation of the two frames w.r.t. the base frame. The superscript denotes the frame in which a unit quaternion (31) is expressed, while the subscript denotes the frame to which it refers. The result of the quaternion product

$${}^{\text{base}}\xi_{\mathrm{d}}(\theta) = {}^{\text{base}}\xi_{\text{start}}\,{}^{\text{start}}\xi_{\mathrm{d}}(\theta) \tag{32}$$

describes an interpolation between start orientation ${}^{\text{base}}\xi_{\text{start}}$ and the goal orientation ${}^{\text{base}}\xi_{\text{goal}}$. The transition between them is described by the quaternion ${}^{\text{start}}\xi_{\mathrm{d}}(\theta)$. The quaternion ${}^{\text{start}}\xi_{\mathrm{d}}$ complies with ${}^{\text{start}}\xi_{\mathrm{d}}(\theta = 0) = 1$ and ${}^{\text{start}}\xi_{\mathrm{d}}(\theta = \theta_L) = {}^{\text{start}}\xi_{\text{goal}}$. The product ${}^{\text{start}}\xi_{\text{base}}\,{}^{\text{base}}\xi_{\text{goal}}$ defined by (30) yields the axis $\boldsymbol{n}$ and the goal angle θ_L in (31), where ${}^{\text{start}}\xi_{\text{base}}$ is the inverse to ${}^{\text{base}}\xi_{\text{start}}$, i.e. $\left({}^{\text{base}}\xi_{\text{start}}\right)^{-1} = \left({}^{\text{base}}\xi_{\text{start}}\right)^* = {}^{\text{start}}\xi_{\text{base}}$.

Applying the proposed motion law (7) to the geometric path defined by (32) and (31) leads to

$${}^{\text{base}}\xi_{\mathrm{d}}(\theta(t)) = {}^{\text{base}}\xi_{\text{start}}\,{}^{\text{start}}\xi_{\mathrm{d}}(s(t)) \tag{33}$$

a C^4 continuous p2p trajectory in $SO(3)$ with $L = \theta_L$, $v_{\max} = \omega_{\max}$, $a_{\max} = \alpha_{\max}$, and $d_{\max} = \delta_{\max}$, where $\omega_{\max}$, $\alpha_{\max}$ and $\delta_{\max}$ are the orientation kinematic limits.

Orientation kinematic limits can easily be taken into account because the rotation axis $\boldsymbol{n}$ is fixed [14]. The angular velocity $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$, angular acceleration $\boldsymbol{\alpha} = (\alpha_x, \alpha_y, \alpha_z)^T$, and angular deceleration $\boldsymbol{\delta} = (\delta_x, \delta_y, \delta_z)^T$ are decoupled and linear w.r.t. the base frame:

$$\omega_x = n_x\dot{\theta}(t), \quad \omega_y = n_y\dot{\theta}(t), \quad \omega_z = n_z\dot{\theta}(t), \tag{34}$$

$$\alpha_x = n_x\ddot{\theta}(t), \quad \alpha_y = n_y\ddot{\theta}(t), \quad \alpha_z = n_z\ddot{\theta}(t), \tag{35}$$

$$-\delta_x = n_x\ddot{\theta}(t), \quad -\delta_y = n_y\ddot{\theta}(t), \quad -\delta_z = n_z\ddot{\theta}(t). \tag{36}$$

Note that $\delta_x, \delta_y, \delta_z > 0$ and, therefore, the minus signs in (36). As mentioned in Sec. I, none of the twelve sets of Euler angle are able to take into account the orientation kinematic limits efficiently due to its non-algebraic form [16].

C. Synchronization

Synchronization leads to the important requirement of a straight line in the respective domain that all n DOF have to reach their target pose simultaneously at zero velocity, zero acceleration, and so forth. This is achieved with common time evolution. Therefore, the constraints for all n DOF are adapted for synchronization. We utilize (25)-(27) which can be used to synchronize n DOF by rewriting them as

$$v_{\max,i} = \frac{|L_i|}{\left(T_{\mathrm{cr,max}} + 0.5T_{\mathrm{lo,max}} + 0.5T_{\mathrm{sd,max}}\right)}, \tag{37}$$

$$a_{\max,i} = v_{\max,i}\frac{C_{\mathrm{a},4}}{T_{\mathrm{lo}}}, \text{ and} \tag{38}$$

$$d_{\max,i} = v_{\max,i}\frac{C_{\mathrm{a},4}}{T_{\mathrm{sd}}}, \tag{39}$$

where $T_{\mathrm{cr,max}} = \max_i (0, T_{\mathrm{cr},i})$, $T_{\mathrm{lo,max}} = \max_i T_{\mathrm{lo},i}$, and $T_{\mathrm{sd,max}} = \max_i T_{\mathrm{sd},i}$. Note that $T_i = T_{\mathrm{lo},i} + T_{\mathrm{cr},i} + T_{\mathrm{sd},i}$ The subscription $i = 1, \ldots, n$ displays the respective DOF, e.g. Cartesian space, and the *max* suffix denotes the maximum duration of the respective segment considering all n DOF.

D. Blending

The geometric path can be constructed directly or indirectly. The idea is to construct directly the entire geometric path first. Afterwards the blending part is constructed indirectly, which concatenates the adjacent trajectories.

For the following we need the properties of $v_{\mathrm{N},4}$ which is

$$v_{\mathrm{N},4}(1-\tau) = 1 - v_{\mathrm{N},4}(\tau)\,. \tag{40}$$

By rearranging (40), we obtain

$$v_{\mathrm{N},4}(1-\tau) + v_{\mathrm{N},4}(\tau) = 1 = \textit{const.} \tag{41}$$

Note that $v_{\mathrm{N},4}$ has all properties of a path primitive which are summarized in [3].

For three adjacent points, i.e. p_{k-1}, p_k, p_{k+1}, with given kinematic limits, the blending part is defined by v_{sd}^k and v_{lo}^{k+1}, see (6) and (4), respectively. The superscript denotes the appropriate p2p trajectory and the three adjacent points were chosen in order to have positive velocities. The velocity of the blending part with $\lambda^k = \lambda^{k+1} = 1$ is defined by

$$\begin{aligned} v\,(t) &= v_{\mathrm{sd}}^k\left(1-\tau^k\right) + v_{\mathrm{lo}}^{k+1}\left(\tau^{k+1}\right) \\ &= v_{\mathrm{max}}^k v_{\mathrm{N},4}\left(1-\tau^k\right) + v_{\mathrm{max}}^{k+1} v_{\mathrm{N},4}\left(\tau^{k+1}\right) \end{aligned}$$

followed by substituting (40) and $\tau^k = \tau^{k+1}$ leading to

$$\begin{aligned} v\,(t) &= v_{\mathrm{max}}^k\left(1 - v_{\mathrm{N},4}\left(\tau^k\right)\right) + v_{\mathrm{max}}^{k+1} v_{\mathrm{N},4}\left(\tau^k\right) \\ &= v_{\mathrm{max}}^k + \left(v_{\mathrm{max}}^{k+1} - v_{\mathrm{max}}^k\right) v_{\mathrm{N},4}\left(\tau^k\right), \end{aligned} \tag{42}$$

which is a smooth velocity profile leading to a $\mathcal{C}^4$ trajectory. However, (42) may lead to high accelerations and, therefore, potentially violate the kinematic limits. In this case, we redefine the maximum deceleration d_{max}^k of k^{th} trajectory and the maximum acceleration a_{max}^{k+1} of the $(k+1)^{\text{th}}$ trajectory. The deceleration and acceleration are rewritten as

$$d_{\mathrm{ble}}^k = \frac{v_{\mathrm{max}}^k C_{\mathrm{a},4}}{T_{\mathrm{ble}}} \text{ and} \tag{43}$$

$$a_{\mathrm{ble}}^{k+1} = \frac{v_{\mathrm{max}}^{k+1} C_{\mathrm{a},4}}{T_{\mathrm{ble}}} \text{ , respectively,} \tag{44}$$

where T_{ble} is the blending duration and it is specified by

$$T_{\mathrm{ble}} = C_{\mathrm{a},4} \max\left\{\frac{|v^*|}{a_{\mathrm{max}}^{k+1}}, \frac{|v^*|}{d_{\mathrm{max}}^k}, \frac{v_{\mathrm{max}}^{k+1}}{a_{\mathrm{max}}^{k+1}}, \frac{v_{\mathrm{max}}^k}{d_{\mathrm{max}}^k}\right\} \tag{45}$$

with $v^* = \operatorname{sign}\left(L^{k+1}\right) v_{\mathrm{max}}^{k+1} - \operatorname{sign}\left(L^k\right) v_{\mathrm{max}}^k$ respecting negative and positive velocities. In order to derive (43)-(45), we may compute the duration of (42). The computation is similar to (12) and requires the maximum derivative of (42), which is equal to the smallest value of d_{max}^k and a_{max}^{k+1}. The new duration of (42) is the maximum between the computed duration, T_{sd}^k, and T_{lo}^{k+1}. The derivation is completed after simplifying the maximum formulation. Note that after utilizing (43) and (44) the inequalities $d_{\mathrm{ble}}^k \leq d_{\mathrm{max}}^k$ and $a_{\mathrm{ble}}^{k+1} \leq a_{\mathrm{max}}^k$ holds and, therefore, no kinematic limits will be violated. Further, note that the subsequent adaptations of duration T_{sd}^k and T_{lo}^{k+1} are now equal due to (45). Thus, the blending (cf. (42)) and, therefore, the overall trajectory is $\mathcal{C}^4$ smooth.

In Fig 2, example 1-DOF trajectories are depicted. The blending is smooth and all kinematic limits are met. Note that the next trajectory starts if the previous one starts with its *set-down* segment. The switching time T_{enable} indicates the start of the respective trajectory.

Lets now consider two p2p trajectories in $SO(3)$ given by (33) and the proposed motion law in Sec. II-A. The concatenation via (30) of both p2p trajectories leads to one overall trajectory. Each orientation interpolation is described by its time-variant angle $\theta(t) = s(t)$ and its fixed axis of rotation $\boldsymbol{n}$. Therefore, during the blending the overall axis of rotation changes over the interpolation due to the concatenation of two time-variant quaternions, for instance ${}^{\text{base}}\xi_{k-1}{}^{k-1}\xi_k\left(s\left(t\right)\right)^k\xi_{k+1}\left(s\left(t\right)\right)$, while during the cruise segment with constant velocity the overall axis of rotation is constant, for instance ${}^{\text{base}}\xi_{k-1}{}^{k-1}\xi_k{}^k\xi_{k+1}\left(s\left(t\right)\right)$. Note that ${}^{\text{base}}\xi_{k-1}$ and ${}^{\text{base}}\xi_{k-1}{}^{k-1}\xi_k$ are constant and, therefore, the multiplication by these constants does not affect the angular velocity and its time derivatives regarding their magnitude. Furthermore, both p2p trajectories concatenated without blending lead to two straight lines being the shortest great arc on the surface of the unit quaternion sphere in the four-dimensional space between two adjacent orientations. Note the relation to SLERP [22] which is shown in [10]. During the blending both straight lines in $SO(3)$ are smoothly blended similar to the example depicted in Fig 2 but on the surface of the unit quaternion sphere. An important fact is the decoupled linear form of the angular velocity (34), angular acceleration (35), and angular deceleration (36) in order to provide a smooth blending. The orientation kinematic limits are preserved thanks to the smooth blending and the eventual redefinition of the orientation kinematics limits of the respective p2p trajectory by appropriate adaptation of (42)-(45). A similar blending approach can be found in [6]. However, an additional interpolation parameter during the blending is mandatory, the overall trajectory is only $\mathcal{C}^1$ continuous, and kinematic limits cannot be considered.

E. Assembling the $SE(3)$ trajectory planner

Now that the motion law, quaternionic path, the synchronization for several DOF, and the necessary blending approach have been defined, the components can be assembled into one framework. For the sake of simplification, the translational kinematic limits are assigned component-wise and are equal among Cartesian axes. Further, the rotational kinematic limits are designed as magnitude constraints, i.e. $\omega_{\mathrm{max}} = \max\left\{||\boldsymbol{\omega}||_2\right\}$, $\alpha_{\mathrm{max}} = \max\left\{||\boldsymbol{\alpha}||_2\right\}$, and $\delta_{\mathrm{max}} = \max\left\{||\boldsymbol{\delta}||_2\right\}$ with $||\cdot||_2$ representing the Euclidean norm.

The framework requires m p2p trajectories with kinematic limits. We introduce a p2p trajectory state $\mathcal{T}$ which include all necessary information without time law indicating all

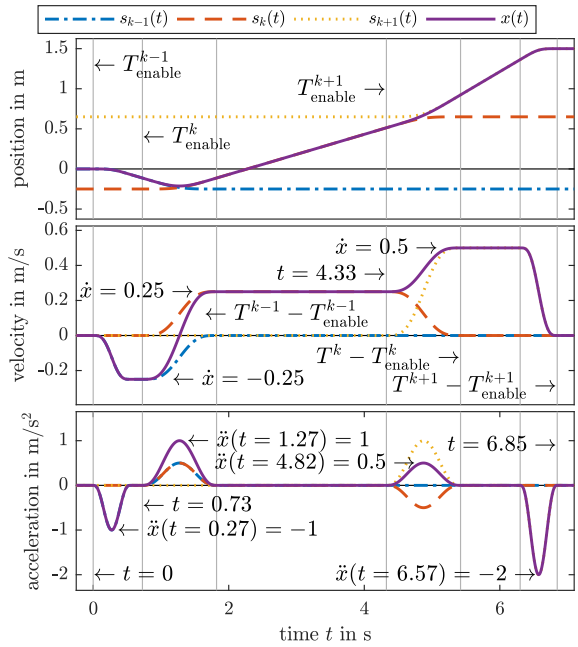

Fig. 2. Course of position, velocity, and acceleration of $s_i(t)$ with $\mathcal{C}^4$ smoothness, different total displacements, different kinematic limits, and switching time T_{enable} defines the overall trajectory for the position $x\,(t)$. The blending method guaranties that the kinematic limits $a_{\max}$ and $d_{\max}$ are not violated in the respective segment. As can be seen in the course of position, $x\,(t)$ passes the via points in the vicinity, whereas the start and goal position are exactly reached. Moreover, $x\,(t)$ has a continuously differentiable velocity, acceleration, and jerk. The course of jerk is omitted.

boundaries of a trajectory, such as start and goal pose and kinematic limits. The i^{th} state $\mathcal{T}^i$ is defined as

$$\begin{aligned}\mathcal{T}^i = \big(\; & x^i_{\text{s}},\; x^i_{\text{g}},\; y^i_{\text{s}},\; y^i_{\text{g}},\; z^i_{\text{s}},\; z^i_{\text{g}},\; \xi^i_{\text{s}},\; \xi^i_{\text{g}},\\ & v^i_{x,\max},\; a^i_{x,\max},\; d^i_{x,\max},\; v^i_{y,\max},\; a^i_{y,\max},\; d^i_{y,\max},\\ & v^i_{z,\max},\; a^i_{z,\max},\; d^i_{z,\max},\; \omega^i_{\max},\; \alpha^i_{\max},\; \delta^i_{\max}\big) \end{aligned} \tag{46}$$

with $\mathcal{T}^0 = (0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0)$, where x, y and z are the respective Cartesian axes and ξ represented an orientation as unit quaternion (31). The suffix s denotes the start, while suffix g stands for goal. For the sake of readability, the suffix for the base frame is omitted. Note that $l^k_{\text{g}} = l^{k+1}_{\text{s}}$ for $l = x, y, z$ and $\xi^k_{\text{g}} = \xi^{k+1}_{\text{s}}$ holds if $0 \leq k < m$. The following five steps are needed for preparation:

- 1^{st} compute $L^i_j = l^i_{\text{g}} - l^i_{\text{s}}$ for $l = x, y, z$ and compute θ^i_L and $\boldsymbol{n}^i$ from the product $\xi^i_{\text{s}} \left(\xi^i_{\text{g}}\right)^*$ defined by (29)-(30).
- 2^{nd} adapting the kinematic limits of $\mathcal{T}^i$ by considering $j_{\max}$ via (25)-(27). Note that this step is optional.
- 3^{rd} synchronization of all DOF by (37)-(39).
- 4^{th} ensure kinematic limits and smoothness during blending via (43)-(44).
- 5^{th} synchronization of all DOF by (37)-(39). This step is necessary if 4^{th} step adapts the kinematic limits.
- 6^{th} compute switching time $T^i_{\text{enable}} = \sum_{k=0}^{i-1} T^k_{\text{lo}} + T^k_{\text{cr}}$ with $T^0_{\text{lo}} = T^0_{\text{cr}} = 0\,\text{s}$.

Afterwards, we rewrite the i^{th} p2p trajectory state $\mathcal{T}^i$ as

$$\begin{aligned}\mathcal{T}^i = \big(\; & L^i_x,\; L^i_y,\; L^i_z,\; \theta^i_L,\; \boldsymbol{n}^i,\; T^i_{\text{enable}},\\ & v^i_{x,\max},\; a^i_{x,\max},\; d^i_{x,\max},\; v^i_{y,\max},\; a^i_{y,\max},\; d^i_{y,\max},\\ & v^i_{z,\max},\; a^i_{z,\max},\; d^i_{z,\max},\; \omega^i_{\max},\; \alpha^i_{\max},\; \delta^i_{\max}\big), \end{aligned} \tag{47}$$

where the kinematic limits may have been adapted according to the previous preparation steps.

The trajectories of each DOF are concatenated at certain switching time T^i_{enable} to compose the overall trajectory in $SE(3)$. For the translational part, they are simply the sums

$$x\,(t) = x^1_{\text{s}} + \sum_{i=1}^{m} s\left(L^i_x,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \tag{48}$$

$$y\,(t) = y^1_{\text{s}} + \sum_{i=1}^{m} s\left(L^i_y,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \text{ and} \tag{49}$$

$$z\,(t) = z^1_{\text{s}} + \sum_{i=1}^{m} s\left(L^i_z,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \tag{50}$$

where $t^i = t - \sum_{k=0}^{i} T^k_{\text{enable}}$ acts as switching time and (7) is applied. The velocity profiles of the Cartesian axes are

$$\dot{x}\,(t) = \sum_{i=1}^{m} v\left(L^i_x,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \tag{51}$$

$$\dot{y}\,(t) = \sum_{i=1}^{m} v\left(L^i_y,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \text{ and} \tag{52}$$

$$\dot{z}\,(t) = \sum_{i=1}^{m} v\left(L^i_z,\; v^i_{\max},\; a^i_{\max},\; d^i_{\max},\; t^i\right), \tag{53}$$

where (3) is used. The rotational part needs to be treated differently, which leads to the product

$$\xi\,(t) = \xi_1 \prod_{i=1}^{m} {}^i\xi_{\text{d}}\left(s\left(\theta^i_L, \boldsymbol{n}^i, \omega^i_{\max}, \alpha^i_{\max}, \delta^i_{\max}, t^i\right)\right), \tag{54}$$

where (30) and (31) are utilized. Note the similarities between (33) and (54). By making use of the property $||\xi\xi'||_2 = ||\xi||_2\,||\xi'||_2 = 1$ for two given arbitrary unit quaternion ξ and ξ', it can be guaranteed that (54) is an unit quaternion in every time step and its generated rotational path is a path on the unit hypersphere in $SO(3)$. Therefore, no re-normalization is needed which may cause distortions.

III. Simulation

For the following validation we use a DLR LWR-III [1] in simulation. The nine dots puzzle is applied in order to show the realized implementation of the proposed trajectory generator. The task is to connect all nine given dots which were arranged in a square with only four straight lines. The accompanying video provides additional visual aid.

TABLE II
Planned poses for the validation.

	Translation (unit in meters)			Rotation (dimensionless quantity)			
i	x	y	z	η	ϵ_1	ϵ_2	ϵ_3
1	0.75	0.0	0.59	0.708	0	0.707	0
2	0.55	0.15	0.4	0.866	0	0.5	0
3	0.55	−0.15	0.7	0.845	0.191	0.462	−0.191
4	0.55	0.3	0.7	0.845	−0.191	0.462	0.191
5	0.55	−0.15	0.25	0.854	0.354	0.354	0.146
6	0.55	−0.15	0.7	0.845	0.191	0.462	−0.191
7	0.75	0.0	0.59	0.708	0	0.707	0

a) Setup: In the validation, a trajectory is planned between seven poses specified by poses reported in Table II. For all poses the applied kinematic limits $v_{\max} = 0.25\,\mathrm{m/s}$, $a_{\max} = d_{\max} = 5.5\,\mathrm{m/s^2}$, $\omega_{\max} = 3.14\,\mathrm{rad/s}$, $\alpha_{\max} = \delta_{\max} = 62.83\,\mathrm{rad/s^2}$ are equal. The sampling time is set to $10\,\mathrm{ms}$. The robot simulator V-REP [8] is used. All parameters of the simulated robot are designed to behave realistically and similarly to its real robot counterpart. However, several simplifications and assumptions are made, e.g. negligible friction and un-modeled motor dynamics. The inverse kinematic module of V-REP allowing kinematic calculation for any type of mechanism is used and the damped least-squares method is utilized. The proposed trajectory planner is implemented in MATLAB R2017b connected with V-REP 3.6.0, running on a 64-bit Linux operating system, on a computer with a Xeon $3.60\,\mathrm{GHz} \times 8$ processor.

b) Results: The robot moves on a predefined trajectory in Cartesian space. Figure 3 depicts sequences of the accompanying video. The course of the desired trajectory is shown in Fig. 4, in Fig. 5, and in Fig. 6.

c) Discussion: From the image sequence (cf. Fig. 3) and video, it can be observed that the robot moves in straight lines except during blending. Therefore, it emphasizes how we fulfill a key requirements of path planning algorithms: maintaining near-linear trajectories in $SE(3)$ between successive via poses. Our approach can guarantee geodesic line movements in $SE(3)$ during the constant velocity segment. Regarding the translational part, it is a geodesic (straight) line in Cartesian space because the Cartesian axes are synchronized, cf. Fig. 4. Regarding the rotational part, it is a geodesic (shortest great arc) line in $SO(3)$ since our planner utilizes the proposed quaternion interpolator in [10], which inherits the property of being the shortest great arc between the two quaternions on the unit quaternion sphere from SLERP [22], see [10] for reference. Note that the variation in the geometric path during blending is crucial in order to comply with the kinematic limits. Further note that the higher the acceleration and deceleration, the closer the generated trajectory gets to the given via pose. If the maximum velocity, acceleration, and deceleration of the generated overall trajectory are not adapted, it may violate the kinematic limits. Recomputing the duration of the blending part T_{ble} leads to a longer duration in the acceleration and deceleration phase. Surprisingly, the overall duration of the trajectory does not change after adaptation because the longer the generated path in the blending part is, the shorter its duration is, and vice versa. Therefore, it can be concluded that the generated trajectory is optimal in the sense of minimum duration and distance travelled under kinematic limits in $SE(3)$ and required $\mathcal{C}^4$ smoothness.

Regarding the implicit generation of the geodesic path during the blending part, the following should be noted. First, many commonplace approaches in geometry cannot obtain and preserve $\mathcal{C}^2$ smoothness, e.g. path constructed from linear and circular segments [3] and cubic B-spline quaternion curves constructed with the de Casteljau algorithm [13], respectively. Second, with minimizing the acceleration it is possible to reduce energy consumption significantly [20].

To the best of our knowledge, this is the first trajectory generator, which can a priori consider kinematic limits on $SE(3)$. In comparison to recent published works on quaternion-based orientation interpolation in [18] and in [21], no numerical optimization and no re-normalization is required. Utilizing numerical optimization is usually a bottleneck when hard real-time capabilities is needed. Numerical optimization and re-normalization may incurs distortions leading to unpredictable motion in $SO(3)$. Furthermore, our quaternion-based orientation interpolator generates a trapezoidal-like velocity profile which should result in a significantly shorter duration while respecting angular kinematic limits throughout the motion.

IV. CONCLUSIONS

In this paper, we present a solution for trajectory planning in $SE(3)$ including singularity-free orientation interpolation. The proposed trajectory planner can handle multiple via poses and kinematic limits. It is expandable to multiple DOF and different smoothness requirement, making it suitable for a broad range of applications such as computer animation, holonomic mobile robots, spacecrafts, and robot manipulators. The effectiveness is shown for a 7-DOF robot manipulator in simulation.

The key idea is to blend the overlapping segment of two trapezoidal-like velocity profiles, i.e. the deceleration segment of the previous one and the acceleration segment of the following one. This implicitly generates a geometric path which considers the kinematic limits and satisfies the required smoothness for the overall trajectory in $SE(3)$. The segment with constant velocity explicitly generates a geodesic path in $SE(3)$ due to the synchronized Cartesian axes and quaternion-based orientation interpolation.

The merit of our approach lies in the fact that all necessary values can be computed in advance, all kinematic limits are considered, and all degrees of freedom in Cartesian space are taken into account without requiring a numerical optimization. Hence, it is suitable for hard real-time application. Due to the fact that the velocity of the end-effector can be computed in advance, which can be interpreted as kinetic energies on velocity level, the velocity can be directly constrained. Due to the capability of computing the desired acceleration in advance, our approach is suitable for acceleration-based control and inverse dynamics problems.

For future improvements, the abolition of the null-boundary condition can extend our approach to an online trajectory generator with instantaneously reaction capabilities to unforeseen events. Further, we will conduct real robot-experiments with applications in human-robot-collaboration.

ACKNOWLEDGMENT

We thank Sami Haddadin for discussion and sharing his knowledge on related topics that have greatly inspired the course of this research.

REFERENCES

[1] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger. The DLR lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: an international journal*, 2007.

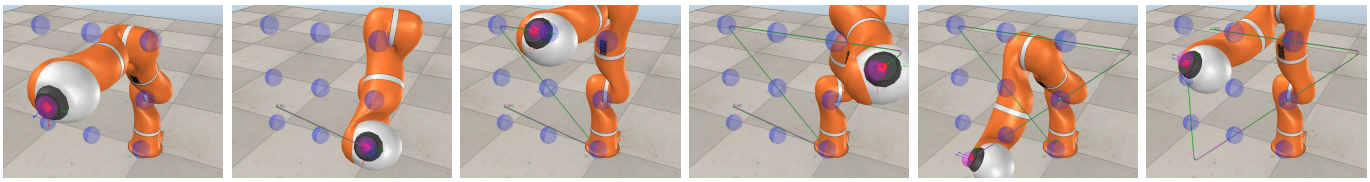

Fig. 3. Image sequence of accompanying video. The desired pose generated by the proposed trajectory planner is shown by a red trace, while the executed trajectories executed by the robot are shown by a green trace.

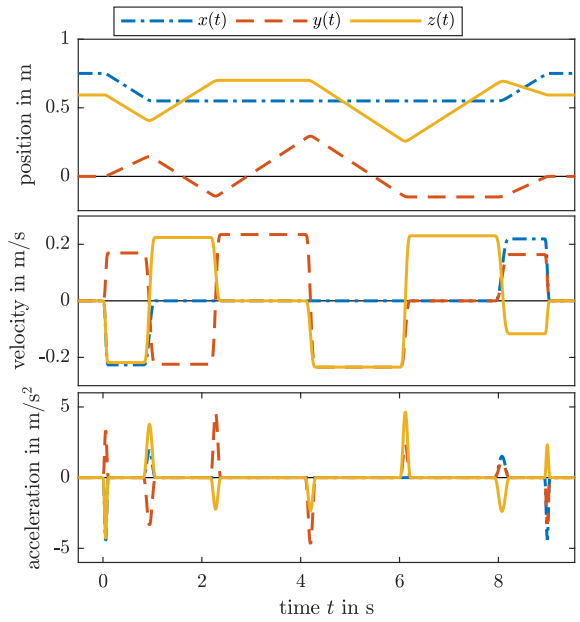

Fig. 4. Course of the desired position, velocity, and acceleration.

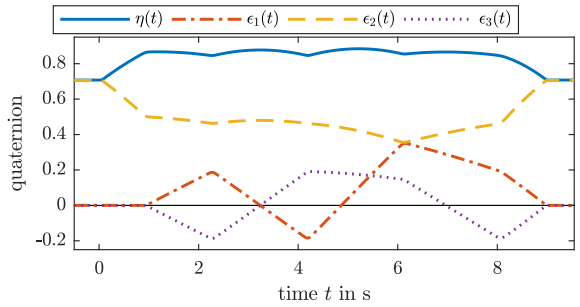

Fig. 5. Desired quaternions generated by the proposed trajectory planner.

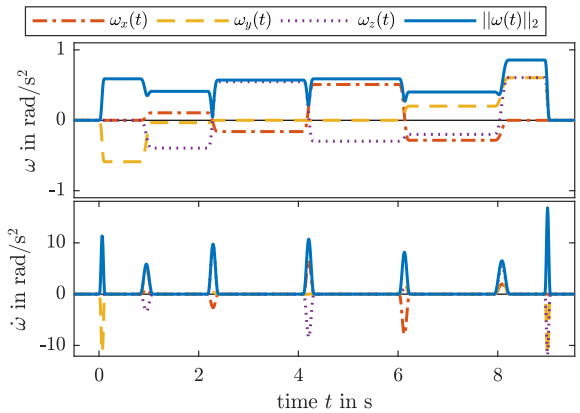

Fig. 6. Course of the angular velocity ω and acceleration $\dot{\omega}$ which are computed numerically.

[2] A. H. Barr, B. Currin, S. Gabriel, and J. F. Hughes. Smooth interpolation of orientations with angular velocity constraints using quaternions. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 313–320, 1992.

[3] L. Biagiotti and C. Melchiorri. *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[4] J. J. Craig. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River, 2005.

[5] E. B. Dam, M. Koch, and M. Lillholm. *Quaternions, interpolation and animation*, volume 2. Datalogisk Institut, Københavns Universitet, 1998.

[6] N. Dantam and M. Stilman. Spherical parabolic blends for robot workspace trajectories. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3624–3629, 2014.

[7] A. De Luca and W. Book. Robots with flexible elements. In *Springer Handbook of Robotics*, pages 287–319. Springer, 2008.

[8] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 51–62, 2010.

[9] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of graphics tools*, 3(3):29–48, 1998.

[10] R. Grassmann, L. Johannsmeier, and S. Haddadin. Smooth point-to-point trajectory planning in se(3) with self-collision and joint constraints avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 8352 – 8359, 2018.

[11] S. Haddadin, S. Haddadin, and S. Parusel. Franka emika panda. Accessed: Feb. 12, 2019. [Online]. Available: www.franka.de.

[12] I. L. Kantor and A. S. Solodovnikov. *Hyperkomplexe Zahlen*. Teubner, 1978.

[13] M.-J. Kim, M.-S. Kim, and S. Y. Shin. A c/sup 2/-continuous b-spline quaternion curve interpolating a given sequence of solid orientations. In *Computer Animation'95., Proceedings.*, pages 72–81, 1995.

[14] M.-X. Kong, C. Ji, Z.-S. Chen, and R.-f. Li. Application of orientation interpolation of robot using unit quaternion. In *IEEE International Conference on Information and Automation*, pages 384–389, 2013.

[15] T. Kröger and F. M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, 26(1):94–111, 2010.

[16] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 1991.

[17] S.-g. Liu, S. Zhu, X. Wang, and H.-f. WANG. Smooth orientation planner for manipulators based on quaternion and b-spline. *Journal of Zhejiang University (Engineering Science)*, 43(7):1192–1196, 2009.

[18] M. Neubauer and A. Müller. Smooth orientation path planning with quaternions using b-splines. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2087–2092, 2015.

[19] T. Popiel and L. Noakes. C2 spherical bézier splines. *Computer aided geometric design*, 23(3):261–275, 2006.

[20] S. Riazi, K. Bengtsson, O. Wigstrom, E. Vidarsson, and B. Lennartson. Energy optimization of multi-robot systems. In *IEEE International Conference on Automation Science and Engineering*, pages 1345–1350, 2015.

[21] M. Shahbazi, N. Kashiri, D. Caldwell, and N. Tsagarakis. On the orientation planning with constrained angular velocity and acceleration at endpoints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7033–7038, 2018.

[22] K. Shoemake. Animating rotation with quaternion curves. In *ACM SIGGRAPH computer graphics*, volume 19, pages 245–254, 1985.

[23] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[24] J. Stuelpnagel. On the parametrization of the three-dimensional rotation group. *Society for Industrial and Applied Mathematics Review*, 6(4):422–430, 1964.

[25] M. Žefran, V. Kumar, and C. B. Croke. On the generation of smooth three-dimensional rigid body motions. *Transactions on Robotics and Automation*, 14(4):579–589, 1998.

[26] R. Weitschat, A. Dietrich, and J. Vogel. Online motion generation for mirroring human arm motion. In *IEEE International Conference on Robotics and Automation*, pages 4245–4250, 2016.