Robotica:page 1 of 22.
doi:10.1017/S0263574719001619

An Online Trajectory Generator on SE(3) for Human–Robot Collaboration

Gerold Huber* and Dirk Wollherr

Chair of Automatic Control Engineering (LSR), Department of Electrical and Computer Engineering, Technical University of Munich, Munich, Germany
E-mail: dw@tum.de

(Accepted October 19, 2019)

SUMMARY
With the increasing demand for humans and robots to collaborate in a joint workspace, it is essential that robots react and adapt instantaneously to unforeseen events to ensure safety. Constraining robot dynamics directly on SE(3), that is, the group of 3D translation and rotation, is essential to comply with the emerging Human–Robot Collaboration (HRC) safety standard ISO/TS 15066. We argue that limiting coordinate-independent magnitudes of physical dynamic quantities at the same time allows more intuitive constraint definitions. We present the first real-time capable online trajectory generator that constrains translational and rotational magnitude values of 3D translation and 3D rotation dynamics in a singularity-free formulation. Simulations as well as experiments on a hardware platform show the utility in HRC contexts.

KEYWORDS: Trajectory generation; Control of robotic systems; Motion planning; Human–Robot Collaboration; Human–Robot Interaction; Safety; SE(3); Orientation; Magnus expansion.

1. Introduction

Enabling humans and robots to physically work together on cooperative tasks in close distance is a very active area of current robotic research. In such HRC scenarios, a trajectory generator must not only assure human safety but at the same time respect human comfort, in order to increase acceptance of the robot by human co-workers. The former—safety—asks for a real-time capable trajectory generator that reacts instantaneously to sensor inputs or a change in constraints. This requires evaluation of new set points in every embedded low-level iteration cycle at high frequencies. While literature offers various solutions for online trajectory generators (OTGs) with axis-specific constraints on different derivative orders, the latter—comfort—favors constraining velocities and accelerations directly in the most intuitive context for the human, namely, 3D translation as well as 3D rotation in their geometric sense, that is, vectors with an orthonormal bases. Rather than limiting dynamics of the coordinate-wise components independently, human presence further suggests restricting end effector movement dynamics in their absolute values.

Restricting the magnitude of translational velocity is in particular essential for incorporating safety standards in HRC according to ISO/TS 15066[1]. Although this international standard is still under development, it serves as a guideline on how safety during collaborative operations must be provided. Possible methods listed in this standard are *safety-rated monitored stop*, *hand guiding*, *speed and separation monitoring* as well as *power and force-limited collaborative operation*. Especially for collaboration in close distance, only the latter is an appropriate option. Within this method of operation, contact events are categorized in quasi-static and transient contact. The former includes clamping or crushing situations and mainly addresses the robot controller. The latter treats dynamic impact hazards, which can be actively considered in OTGs. Safety is provided by limiting the transferred energy

* Corresponding author. E-mail: gerold.huber@tum.de

Table I. Comparison to the state-of-the-art w.r.t. the defined requirements.

Requirement		*R1*	*R2*	*R3*	*R4*	*R5*
Direct approaches [3–12]		✗	(✓)[a]	✓	✗	✓
Indirect approaches	FIR Filter based [13–15]	✗	✗	✓	✗	✓
	Sliding Mode based [16–18]	✗	✗	(✓)[b]	✓	✓
Optimization based [19–24]		✗	✓	✓	✓	✗
Our approach		✓	✓	✓	✓	✓

[a]Only true for ref. [12].
[b]Only true for ref. [18].

during impact, according to different human body regions. Given the effective mass of the two-body system, one can derive the maximal allowed relative speed between the robot and the human body region. Detailed calculation guidelines are outlined in.[1]

Considering these facts, we propose the following requirements for an OTG in HRC contexts:

R1: Translation and rotation dynamic quantities are constrained in their magnitudes.
R2: Rotation dynamics refer to geometric angular quantities.
R3: All degrees of freedom (DOF) are synchronized in their motions.
R4: Constraint-conform trajectories are directly forwarded without delay.
R5: Iteration cycle times below 1 ms must be guaranteed.

Fulfilling these requirements is not addressed in current state-of-the-art OTGs.

1.1. Related work

Current OTGs[2] are able to make robotic systems robust against infeasible trajectory inputs due to, for example, step inputs, improper path transitions caused by a potential higher level path planner, or possible communication delay fluctuations. Online capability also enables instantaneous reaction to sudden unforeseen events.[3] However, authors usually either consider trajectory generation in joint space while moving close to hardware limits to achieve time optimal solutions, or extend these approaches to end effector movements by treating the translational and rotational DOF as independent axes in $\mathbb{R}^n$. Due to the demand for instantaneous reaction to unforeseen events (*R5*), pure path planners without temporal information as well as pure path parametrization methods are not considered in this short review. We classify the relevant OTGs approaches into three major groups:

Direct approaches define a trajectory profile and precompute the whole trajectory to the target state. Most trajectory generators are based on piecewise polynomials as they can be easily designed for arbitrarily often differentiable trajectories. Many approaches, however, are restricted to zero dynamic start and/or end conditions, which entails a constant time delay when feeding the algorithm with already feasible trajectories.[4,5] Kröger et al.[3] provided a basic concept for OTGs and also introduced a classification scheme. Katzschmann et al.[6] extended the approach to regard the entire robot dynamics. Ezair et al.[7] proposed an iterative approach, that uses recursive S-curve polynomials to generate trajectories of arbitrary order with general initial and final conditions. Another iterative algorithm that considers a sampled input trajectory was proposed by Lange et al.[8] and later improved for fast trajectories in ref. [9]. Their work focuses on path-accurate OTG. These approaches are designed for application in joint space. Although the joint trajectories could be derived from an analogous Cartesian trajectory, only constraints in joint space are considered. Kröger expanded the ideas from[3] to straight motions in 3D Euclidean space in [10] by considering orientation in the minimal representation by Euler angles. However, the derivations of these values do not lead to geometric

angular velocities, but are merely of analytic nature as they represent concatenated velocities. This important difference will be further discussed in Section 5.2.

Rymansaib et al.[11] propose the usage of exponential functions rather than polynomials for trajectory generation. This allows consideration of hardware constraints as well as generation of trajectories with continuous jerk. They show that the exponential trajectory generation results in higher tracking accuracy than common S-curve velocity methods. Nevertheless, their approach is also designed for pre-planned point-to-point (PTP) motions with zero target velocities/acceleration, and therefore suffers from time delays as well, which conflicts with *R4*. Moreover, none of the above-mentioned direct methods can fulfill *R2*. The only to the authors known OTG that does formulate geometric angular velocity constraints on SO(3), that is, the group of 3D rotations, is an unpublished work by Lloyd.[12] He reduces the 3D problem to two 1D subproblems by transformation to a locally radial and thus easily solvable coordinate system. Due to that decomposition, however, the approach does not comply with *R1* and further does not allow desired velocity set points, necessary for *R4*.

Indirect approaches consider the trajectory generation as dynamic control system problem, that is, a chain of integrators. Filter-based techniques were adopted by Biagiotti et al.[13,14] and Besset et al.[15] The trajectory generators proposed by Besset et al. use finite impulse response (FIR) filters to generate jerk-limited profiles out of initially acceleration-limited trajectories. While a pregenerated trajectory is given in advance, it delivers a solution in less than 1 μs and thus is the fastest time optimal jerk-limited trajectory generator at this time, even for the multidimensional case. Biagiotti et al.[13] use a chain of FIR filters to smoothen trajectories to have continuous derivatives of arbitrary order. In [14], the strategy is generalized to produce piecewise exponential jerk profiles, to further reduce machine vibrations. Due to their filtering nature, these approaches again result in time delays and thus neither satisfy *R4*. Gerelli et al.[16] and Bianco et al.[17] proposed a discrete-time filter that incorporates constraints as sliding surfaces. These inspiring works generate time optimal trajectories under consideration of bounded velocity, acceleration, and jerk for single-DOF applications. In [18], Bianco proposes a strategy to synchronize multiple DOF. Besides fulfilling *R4*, the latter extends the approach to further comply with *R3*.

Optimization-based approaches rely on numerical solvers. Ardakani et al.[19] suggest a real-time joint trajectory generator based on model predictive control. Their formulation on $\mathbb{R}^n$ achieves an impressive 200 Hz sampling rate for a 6DOF robot kinematic despite the optimization procedure in every iteration. Dinh et al.[20] combine sequential action control[25] with indirect optimization. In numerical optimization frameworks, it is also possible to generate trajectories for hybrid dynamic systems such as the table tennis robot in Koç et al.[21] While solving the individual optimization problems with sequential quadratic programming requires over 1 s computation time, they precompute a lookup table from a fixed initial posture that can be used online. However, for a more general trajectory generation problem, this approach is infeasible. Gao et al.[22] use a rapidly exploring random graph method, for finding a collision-free trajectory for a quadrotor in complex environments. Note that formulating the kinematics in $\mathbb{R}^n$, which holds, for example, for joint trajectories as well as 3D translation, results in a linear chain of integrators. Optimization of 3D rotation trajectories on the other hand result in highly nonlinear and coupled dynamics. Le et al.[23] study the time optimal control problem for SO(3) in a general setting, but only include acceleration constraints. Another algorithm for trajectory generation directly on SE(3) is proposed by Watterson et al.[24] They use semi-definite programming techniques and also consider obstacles in the environment. Using numerical optimization techniques is usually a limiting factor when hard real-time capabilities as the one in *R5* are required, especially when nonlinear constraints and SO(3) dynamics (*R1*) are introduced. None of the above-mentioned work was designed for scenarios that impose requirements *R1* and *R2* while being real-time capable as required by *R5*.

Table I gives a concise summary of the referenced approaches w.r.t. the posed requirements.

1.2. Contribution

In this article, we significantly refine our method from ref. [26] with a rigorous treatment of the problem for 3D orientations in SO(3). We develop an algorithm that directly applies the special orthogonal rotation matrices. It is further adapted to unit quaternion representation. Special case

singularity treatment as well as consideration of multiple goal states are hence not necessary, in contrast to the often used Euler angle representations.

The main contributions of this work are:

1. This is the first OTG algorithm that allows constraining the norm of translational and rotational dynamic quantities, essential to comply with safety standards defined in ISO/TS 15066[1].
2. We introduce the Magnus expansion to OTG treatment. This allows calculating solution to the differential equations on SO(3) with high accuracy.

1.3. Outline

The remainder of the paper is organized as follows. The problem of a modular OTG under requirements *R1–R5* is formally formulated in Section 2. In Section 3, the approach is first introduced for the translational case and then adapted to the rotational DOF. Finally, it is shown how the two portions can be synchronized w.r.t. time. For a distinct highlighting of the potential of our algorithm in robotic applications, we outline constraint extensions for HRC scenarios in Section 4 and discuss its performance with a comparison w.r.t. to the state-of-the-art OTG algorithms in Section 5. We conclude the work and outline future directions of development in Section 6.

2. Problem Formulation

Regarding a modular robot architecture of a complex system, a clear distinction between trajectory generation and robot control is of advantage. Therefore, an interface is needed, that guarantees desired trajectories sent to an arbitrary robot platform stay within defined dynamic constraints. Especially in close distance HRC scenarios, it is essential for these constraints to be defined in an intuitive metric, that is, coordinate-independent magnitudes of translational and rotational speed and acceleration.

Let the end effector pose of a robot be given in $\mathrm{SE}(3) := \mathbb{R}^3 \times \mathrm{SO}(3)$, that consists of a 3D translational position $\boldsymbol{p} \in \mathbb{R}^3$ and a 3D orientation $\boldsymbol{R} \in \mathrm{SO}(3)$. The special orthogonal group SO(3) is defined as $\mathrm{SO}(3) := \{\boldsymbol{R} \in \mathbb{R}^{3\times3} | \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_{3\times3}, \det(\boldsymbol{R}) = 1\}$, with $\boldsymbol{I}_{3\times3}$ referring to the identity matrix in $\mathbb{R}^{3\times3}$. The system state vector $\boldsymbol{x}$ at time $t_k \in \mathbb{R}$ consists of the full 6D pose, together with translational velocity $\boldsymbol{v} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. It is denoted as the 4-tuple $\boldsymbol{x}(t_k) = \boldsymbol{x}_k := (\boldsymbol{p}_k, \boldsymbol{R}_k, \boldsymbol{v}_k, \boldsymbol{\omega}_k)$. Applying a constant acceleration tuple $\boldsymbol{\Lambda}_k := (\boldsymbol{a}_k, \boldsymbol{\alpha}_k)$ consisting of translational and rotational acceleration $\boldsymbol{a}_k \in \mathbb{R}^3$ and $\boldsymbol{\alpha}_k \in \mathbb{R}^3$, respectively, for a given time span $T \in \mathbb{R}$, the system state advances according to

$$\boldsymbol{x}(t_k + T) := \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\Lambda}_k, T), \tag{1}$$

where $\boldsymbol{f}$ denotes the OTG-dependent state propagation function. Assuming a discrete implementation of the OTG running at a sample time $T_\mathrm{s} \in \mathbb{R}$, we refer to the state at time $t_k + iT_\mathrm{s}$ with $i \in \mathbb{N}^+$ as $\boldsymbol{x}_{k+i}$.

Given a desired state $\boldsymbol{x}^\mathrm{des}$ and the current system state $\boldsymbol{x}_k$, the OTG must solve the problem

$$\underset{\boldsymbol{\Lambda}_k, \ldots, \boldsymbol{\Lambda}_{k+(N-1)}}{\text{minimize}} \; N \tag{2}$$

subject to the state progression

$$\boldsymbol{x}_{i+1} = \boldsymbol{f}\,(\boldsymbol{x}_i, \boldsymbol{\Lambda}_i, T_\mathrm{s}) \tag{3a}$$

$$\boldsymbol{x}_{k+N} = \boldsymbol{x}^\mathrm{des} \tag{3b}$$

and subject to the nonlinear user-defined dynamic magnitude constraints

$$\|\boldsymbol{v}_i\|_2 \leq v_\mathrm{max} \tag{4a}$$

$$\|\boldsymbol{a}_i\|_2 \leq a_\mathrm{max} \tag{4b}$$

$$\|\boldsymbol{\omega}_i\|_2 \leq \omega_\mathrm{max} \tag{4c}$$

$$\|\boldsymbol{\alpha}_i\|_2 \leq \alpha_\mathrm{max} \tag{4d}$$

for all $i \in [k, N-1]$ time steps. The operator $\|\cdot\|_2$ denotes the 2-norm and thus the magnitude of the vectors.

Note that Le et al.[23] solve a similar problem on SE(3) for a reduced set of dynamic constraints. Their numerical examples show computation times of over 90 s for an individual trajectory solution, and thus demonstrate the complexity of the problem at hand. Considering the core requirement of every OTG being fast computation, in order to guarantee safety at typical robot sampling frequencies of 1 kHz, the use of iterative numeric optimization algorithms is not appropriate. Accordingly, we relax the requirement of absolute time optimality and accept close-to-optimal solutions under real-time capable iteration cycles $\leq$1 ms instead.

3. Approach

Considering that the dynamic constraints (4) are related to translational and rotational quantities separately, we partition the state dynamics (1) into the individual mappings

$$\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\Lambda}_k, T) := \begin{cases} \boldsymbol{p}(t_k + T) & \leftarrow \boldsymbol{f}_p(\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{a}_k, T) \\ \boldsymbol{v}(t_k + T) & \leftarrow \boldsymbol{f}_v(\boldsymbol{v}_k, \boldsymbol{a}_k, T) \\ \boldsymbol{R}(t_k + T) & \leftarrow \boldsymbol{f}_R(\boldsymbol{R}_k, \boldsymbol{\omega}_k, \boldsymbol{\alpha}_k, T) \\ \boldsymbol{\omega}(t_k + T) & \leftarrow \boldsymbol{f}_\omega(\boldsymbol{\omega}_k, \boldsymbol{\alpha}_k, T) \end{cases} \tag{5}$$

according to the state tuple $\boldsymbol{x}_k = \left(\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{R}_k, \boldsymbol{\omega}\right)$. The translation dynamics are thus given with $\boldsymbol{f}_p$ and $\boldsymbol{f}_v$, whereas the rotation dynamics consist of $\boldsymbol{f}_R$ and $\boldsymbol{f}_\omega$.

Starting from the basic translational problem in Section 3.1, the approach is adapted for the rotational group SO(3) in Section 3.2. Based on these formulations, the synchronization over all 6 DOF w.r.t. to time is outlined in Section 3.3. The set of conditions listed in (4) from the problem formulation is extended to additional HRC motivated use cases in Section 4.

3.1. Translation

For 3D translation, the three components can be treated independently if they are defined w.r.t. an orthonormal basis. The desired behavior for the algorithm is a continuously differentiable position trajectory. This is achieved by calculating the next state via integration of the limited acceleration. The discrete system dynamics (3a) for the translational case are three decoupled double integrators, that is, for a given position $\boldsymbol{p}$ and velocity $\boldsymbol{v}$ at time t_k, applying a constant acceleration vector $\boldsymbol{a}_k$ for the time T advances according to the mappings

$$\boldsymbol{v}(t_k + T) = \boldsymbol{f}_v(\boldsymbol{v}_k, \boldsymbol{a}_k, T) := \boldsymbol{v}_k + \boldsymbol{a}_k T \tag{6a}$$

$$\boldsymbol{p}(t_k + T) = \boldsymbol{f}_p(\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{a}_k, T) := \boldsymbol{p}_k + \boldsymbol{v}_k T + \frac{1}{2}\boldsymbol{a}_k T^2. \tag{6b}$$

The aim of the algorithm is to compute the acceleration such that the position error $\boldsymbol{e}_{p,k} := \boldsymbol{p}_k^{\text{des}} - \boldsymbol{p}_k$ as well as the velocity error $\boldsymbol{e}_{v,k} := \boldsymbol{v}_k^{\text{des}} - \boldsymbol{v}_k$ both converge to zero. The minimum time needed to reach $\boldsymbol{v}_k^{\text{des}}$ is achieved, if the acceleration vector $\boldsymbol{a}_k$ in (6a) exploits the acceleration constraint (4b) and points in the same direction as the velocity error $\boldsymbol{e}_{v,k}$, respectively, $\boldsymbol{v}_k^{\text{des}} = \boldsymbol{f}_v\left(\boldsymbol{v}_k, a_{\max}\frac{\boldsymbol{e}_{v,k}}{\|\boldsymbol{e}_{v,k}\|_2}, T^*_{\text{min,tra}}\right)$ is solved for

$$T^*_{\text{min,tra}} := \frac{\|\boldsymbol{e}_{v,k}\|_2}{a_{\max}}. \tag{7}$$

As the discrete algorithm will run on a fixed sample time T_s, and acceleration is constant in between iterations, the minimal time needed is in fact the next multiple of T_s, denoted by

$$T_{\text{min,tra}} = \max\left\{\left\lceil \frac{T^*_{\text{min,tra}}}{T_\text{s}} \right\rceil, 1\right\} T_\text{s} \tag{8}$$

with operator $\lceil \cdot \rceil$ being the ceiling function that rounds up to the next full integer. The maximization operator in (8) ensures $T_{\text{min,tra}} \geq T_\text{s}$. Requiring to reach the desired velocity $\boldsymbol{v}_k^{\text{des}} = \boldsymbol{f}_v(\boldsymbol{v}_k, \hat{\boldsymbol{a}}_k, T_{\text{min,tra}})$ in the time span $T_{\text{min,tra}}$ is directly used to define the acceleration vector

$$\hat{\boldsymbol{a}}_k = \frac{\boldsymbol{e}_{v,k}}{T_{\text{min,tra}}}, \tag{9}$$

that would result in a synchronous convergence of $\boldsymbol{e}_v(t_k + T_{\text{min,tra}}) = \boldsymbol{0}$ while respecting the acceleration constraint (4b).

Applying $\hat{\boldsymbol{a}}_k$ to the position integration mapping $\boldsymbol{f}_p$ and imposing to reach the desired velocity $\boldsymbol{p}_k^{\text{des}} = \boldsymbol{f}_p\left(\boldsymbol{p}_k, \boldsymbol{v}_k^{\text{goal}}, \hat{\boldsymbol{a}}_k, T_{\text{min,tra}}\right)$ in $T_{\text{min,tra}}$ defines the goal velocity

$$\boldsymbol{v}_k^{\text{goal}} := \frac{\boldsymbol{e}_{p,k}}{T_{\text{min,tra}}} - \hat{\boldsymbol{a}}_k \frac{T_{\text{min,tra}}}{2}. \tag{10}$$

If this velocity is matched, the position error $\boldsymbol{e}_p$ converges to zero together with the velocity error $\boldsymbol{e}_v$ for $T \to T_{\text{min,tra}}$. Advancing $\boldsymbol{v}_k^{\text{goal}}$ for a single iteration leads to

$$\begin{aligned} \boldsymbol{v}_{k+1}^{\text{goal}} &:= \boldsymbol{f}_v\left(\boldsymbol{v}_k^{\text{goal}}, \hat{\boldsymbol{a}}_k, T_{\text{s}}\right) \\ &= \frac{\boldsymbol{e}_{p,k}}{T_{\text{min,tra}}} + \hat{\boldsymbol{a}}_k\left(T_{\text{s}} - \frac{T_{\text{min,tra}}}{2}\right) \end{aligned} \tag{11}$$

and is used as the goal velocity at time t_{k+1}.

The second phase of the algorithm handles given constraints by defining two sets of saturation factors $\mathcal{S}_v \subset \mathbb{R}^+$ and $\mathcal{S}_a \subset \mathbb{R}^+$ for velocity and acceleration, respectively.

Constraints on the trajectory generator output are incorporated by multiplication with the most restrictive element of the corresponding saturation factor set. The translational speed constraint (4a), for example, is met by the set definition

$$\mathcal{S}_v := \{1, \mu_v\} \quad \text{with} \quad \mu_v := \frac{v_{\max}}{\left\|\boldsymbol{v}_{k+1}^{\text{goal}}\right\|_2}. \tag{12}$$

Note that all factors in the set are normalized and saturation requires the factor 1 to be part of the set.[1] For the goal velocity $\boldsymbol{v}_{k+1}^{\text{goal}}$, this factorization is applied with

$$\boldsymbol{v}_{k+1}^{\text{sat}} := \boldsymbol{v}_{k+1}^{\text{goal}} \min\{\mathcal{S}_v\}, \tag{13}$$

where superscript $(\cdot)^{\text{sat}}$ denotes a saturated value.

The necessary acceleration vector to reach velocity constraint conform $\boldsymbol{v}_{k+1}^{\text{sat}}$ is found by solving $\boldsymbol{v}_{k+1}^{\text{sat}} = \boldsymbol{f}_v(\boldsymbol{v}_k, \boldsymbol{a}_k^{\text{goal}}, T_{\text{s}})$ for the goal acceleration

$$\boldsymbol{a}_k^{\text{goal}} := \frac{\boldsymbol{v}_{k+1}^{\text{sat}} - \boldsymbol{v}_k}{T_{\text{s}}}. \tag{14}$$

Eventually, a constraint-conform acceleration vector satisfying $\|\boldsymbol{a}_k\|_2 \leq a_{\max}$ is found with

$$\boldsymbol{a}_k^{\text{sat}} := \boldsymbol{a}_k^{\text{goal}} \min\{\mathcal{S}_a\} \tag{15}$$

using the set definition

$$\mathcal{S}_a := \{1, \mu_a\} \quad \text{with} \quad \mu_a := \frac{a_{\max}}{\left\|\boldsymbol{a}_k^{\text{goal}}\right\|_2} \tag{16}$$

analogously to the velocity saturation (12).

The output of the trajectory generator at hand is ultimately found by advancing the current state $\boldsymbol{x}_k$ for a sample period

$$\begin{aligned} \boldsymbol{v}_{k+1} &= \boldsymbol{f}_v(\boldsymbol{v}_k, \boldsymbol{a}_k^{\text{sat}}, T_{\text{s}}) \\ \boldsymbol{p}_{k+1} &= \boldsymbol{f}_p(\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{a}_k^{\text{sat}}, T_{\text{s}}). \end{aligned} \tag{17}$$

Note that the saturation steps for constraint handling do not conflict with the desired continuous differentiability of the pose trajectories, as velocity and position values are obtained by integration of

[1] A discussion on how to derive saturation factors is given in Section 4.

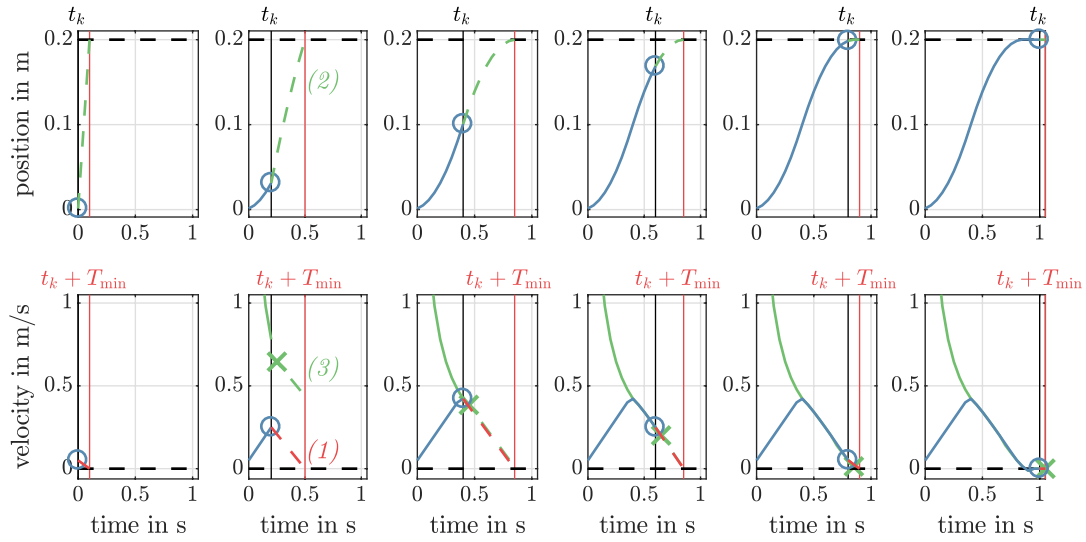

Fig. 1. Progression of the OTG with initial state $\boldsymbol{p}(0) = 0$, $\boldsymbol{v}(0) = 0$ and final state $\boldsymbol{p}^{\text{des}} = 0.2$, $\boldsymbol{v}^{\text{des}} = 0$ at $T_{\text{s}} = 0.05$.
The progress is given for six time instances $t_k = \{0, 0.2, 0.3, 0.5, 0.6, 0.8\}$. Algorithm: *(1)* find T_{min}, the time needed to reach final velocity $\boldsymbol{v}^{\text{des}}$ (dashed red line in velocity plot) *(2)* fit position profile that reaches final position $\boldsymbol{p}^{\text{des}}$ in T_{min} (dashed red line in position plot) *(3)* find $\boldsymbol{v}_{k+1}^{\text{goal}}$ of corresponding position profile (green mark in velocity plot) *(4)* after incorporating constraints, apply corresponding acceleration for T_{s} (not shown).

(6) under constant acceleration $\boldsymbol{a}_k^{\text{sat}}$ until the next iteration cycle. Figure 1 illustrates the mechanism of the approach for a PTP motion. A concise implementation including all necessary steps is outlined in Algorithm 1.

In Section 4, additional saturation factors relevant in HRC scenarios are developed. They are integrated by adding elements to the sets defined in (12) and (16).

Algorithm 1: OTG on $\mathbb{R}^3$ with magnitude constraints

Input : current state $(\boldsymbol{p}_k, \boldsymbol{v}_k)$, desired state $(\boldsymbol{p}_k^{\text{des}}, \boldsymbol{v}_k^{\text{des}})$, constraints $(v_{\text{max}}, a_{\text{max}})$, sampling time T_{s}

Output: next state $(\boldsymbol{p}_{k+1}, \boldsymbol{v}_{k+1})$

`/* ===== find new goal velocity ===== */`

$T_{\text{min,tra}} \leftarrow \max\left\{\left\lceil \frac{\|\boldsymbol{v}_k^{\text{des}} - \boldsymbol{v}_k\|_2}{a_{\text{max}}} \right\rceil, 1\right\} T_{\text{s}}$; `// discretized minimum Time (7)`

$\boldsymbol{v}_{k+1}^{\text{goal}} \leftarrow \frac{\boldsymbol{p}_k^{\text{des}} - \boldsymbol{p}_k}{T_{\text{min,tra}}} + \frac{\boldsymbol{v}_k^{\text{des}} - \boldsymbol{v}_k}{T_{\text{min,tra}}} \left(T_{\text{s}} - \frac{T_{\text{min,tra}}}{2}\right)$; `// goal velocity (11)`

$\mathcal{S}_v \leftarrow$ `collectVelocitySaturationFactors` $(\boldsymbol{v}_{k+1}^{goal}, v_{max})$;

$\boldsymbol{v}_{k+1}^{\text{sat}} \leftarrow \boldsymbol{v}_{k+1}^{\text{goal}} \min\{\mathcal{S}_v\}$; `// saturated goal velocity (13)`

`/* ===== find goal acceleration ===== */`

$\boldsymbol{a}_k^{\text{goal}} \leftarrow \frac{\boldsymbol{v}_{k+1}^{\text{sat}} - \boldsymbol{v}_k}{T_{\text{s}}}$; `// goal acceleration (14)`

$\mathcal{S}_a \leftarrow$ `collectAccelerationSaturationFactors` $(\boldsymbol{a}_{k+1}^{goal}, a_{max})$;

$\boldsymbol{a}_k^{\text{sat}} \leftarrow \boldsymbol{a}_{k+1}^{\text{goal}} \min\{\mathcal{S}_a\}$; `// saturated goal acceleration (15)`

`/* ===== advance current state (6) ===== */`

$\boldsymbol{v}_{k+1} \leftarrow \boldsymbol{f}_v(\boldsymbol{v}_k, \boldsymbol{a}_k^{\text{sat}}, T_{\text{s}})$;

$\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{f}_p(\boldsymbol{p}_k, \boldsymbol{v}_k, \boldsymbol{a}_k^{\text{sat}}, T_{\text{s}})$;

3.2. Rotation

In the rotational treatment of the OTG lies the main contribution of this work. Many different representation forms are known to represent 3D rotations on the special orthogonal group SO(3). In

our preliminary work,[26] we used Euler angles as they form a minimal description unlike axis angle, quaternions, or rotation matrix representation. This allowed a straightforward analogy between the translational and the rotational treatment. However, the well-known gimbal-lock singularities of this representation form require special treatment. Furthermore, there always exist two sets of angles to describe the same orientation. Exploiting this dual representation allows finding shorter paths in case of PTP motions.

Because angular velocity $\boldsymbol{\omega}$ and acceleration $\boldsymbol{\alpha}$ are both defined in $\mathbb{R}^3$ and thus are geometrically decoupled, the algorithm of the translational case can, to the most extend, directly be adapted for the rotational case, with the exception of $\boldsymbol{f}_p$ defined in (10) that incorporates the temporal evolution of the position vector. While position $\boldsymbol{p} \in \mathbb{R}^3$ allows elementary integration of the individual components, the same does not hold for the orientation.

Let $\boldsymbol{R} \in \mathrm{SO}(3)$ be a rotation matrix that describes the rotation of a orthonormal basis from a fixed inertial frame to the body-fixed frame. The angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ describes the angular velocity between the inertial frame and the body-fixed frame w.r.t. the inertial frame. To constrain the angular speed $\|\boldsymbol{\omega}\|_2$ in the same fashion as the translational case, we start from the relation between the rotation matrix $\boldsymbol{R}$ and angular velocity $\boldsymbol{\omega}$. It is given by the vector cross product

$$\dot{\boldsymbol{R}} = \boldsymbol{\omega} \times \boldsymbol{R} \tag{18}$$

and can be expressed in matrix form

$$\dot{\boldsymbol{R}} = \underbrace{\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}}_{=:[\boldsymbol{\omega}]_\times} \boldsymbol{R} \tag{19}$$

where $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ denotes the skew operator. Note that $[\boldsymbol{\omega}]_\times$ is also called the SO(3)-associated Lie algebra $\mathfrak{so}3$. We also use the inverse operator $[\cdot]_\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ that recovers the vector components of a skew symmetric matrix and define it as

$$\left[[\boldsymbol{\omega}]_\times\right]_\vee := \boldsymbol{\omega} \quad \text{where } \boldsymbol{\omega} \in \mathbb{R}^3,\ [\boldsymbol{\omega}]_\times \in \mathfrak{so}3. \tag{20}$$

While (19) suggests an exponential integration of the form

$$\boldsymbol{R}(t+T) = \exp\left(\int_t^{t+T} [\boldsymbol{\omega}(\tau)]_\times \, \mathrm{d}\tau\right) \boldsymbol{R}(t), \tag{21}$$

this only leads to correct orientations in the case of a fixed rotation axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\|_2$, due to the definition of the matrix exponential as a power series and the noncommutativity of the elements in SO(3). This leads to various iterative integration schemes, where the axis is assumed to be fixed only for a single small time step T_s. See ref. [27] for a detailed derivation and comparison of different schemes.

If (21) is solved for small time steps T, the error that comes from noncommutativity of SO(3) is negligible and thus often used for iterative schemes with small sampling times T_s. However, the crucial step of finding the new goal velocity $\boldsymbol{v}_{k+1}^{\mathrm{goal}}$ from (11) is accomplished by solving the time integration over a time span of $T_{\mathrm{min,tra}}$, which cannot be assumed to be small.

The chosen approach in this work is the use of the Magnus expansion as originally proposed in ref. [28]. This method provides an analytical framework for finding the solution of linear, time-variant matrix differential equations such as (19).

3.2.1. Magnus expansion. Opposed to the iterative integration methods mentioned above, the Magnus expansion gives an exact solution, important for longer time ranges. The reason that (21) is not correct for the general rotational case lies in the noncommutativity of matrices $[\boldsymbol{\omega}]_\times$. The nonvanishing portion of the commutation is denoted as the commutator

$$[A, B] := AB - BA, \quad A \in \mathfrak{g}, B \in \mathfrak{g} \tag{22}$$

and naturally arises in Lie structures such as the Lie algebra $\mathfrak{g}$. Magnus' treatment is given in a more general problem; however, our explanation refers to a real-valued time-variant matrix differential equation of the form (19).

Magnus' proposal is to find a solution for (19) in the form of a true matrix exponential

$$\boldsymbol{R}(t+T) = \exp\left([\boldsymbol{\Omega}(t,T)]_\times\right)\boldsymbol{R}(t) \tag{23}$$

with a series expansion

$$[\boldsymbol{\Omega}(t,T)]_\times := \sum_{i=1}^{\infty} [\boldsymbol{\Omega}_i(t,T)]_\times \tag{24}$$

that is referred to as Magnus expansion. It is an infinite power series, however as pointed out in ref. [29], it is usually sufficiently accurate for applications to truncate the series after the first three terms which are given as

$$[\boldsymbol{\Omega}_1(t,T)]_\times := \int_t^{t+T} [\boldsymbol{\omega}(t_1)]_\times \, \mathrm{d}t_1 \tag{25}$$

$$[\boldsymbol{\Omega}_2(t,T)]_\times := \frac{1}{2}\int_t^{t+T}\left[[\boldsymbol{\omega}(t_1)]_\times \, , \int_t^{t_1} [\boldsymbol{\omega}(t_2)]_\times \, \mathrm{d}t_2\right]\mathrm{d}t_1 \tag{26}$$

$$[\boldsymbol{\Omega}_3(t,T)]_\times := \frac{1}{6}\int_t^{t+T}\left[[\boldsymbol{\omega}(t_1)]_\times \, , \int_t^{t_1}\left[[\boldsymbol{\omega}(t_2)]_\times \, , \int_t^{t_2} [\boldsymbol{\omega}(t_3)]_\times \, \mathrm{d}t_3\right]\mathrm{d}t_2\right]\mathrm{d}t_1 \tag{27}$$

$$+ \frac{1}{6}\int_t^{t+T}\left[\left[[\boldsymbol{\omega}(t_1)]_\times \, , \int_t^{t_1} [\boldsymbol{\omega}(t_2)]_\times \, \mathrm{d}t_2\right] , \int_t^{t_2} [\boldsymbol{\omega}(t_3)]_\times \, \mathrm{d}t_3\right]\mathrm{d}t_1. \tag{28}$$

Note that all brackets used refer to the commutator defined in (22). For an explicit solution of the fourth-order term as well as a recursive scheme to calculate Magnus expansion terms of arbitrary order, we refer to ref. [29].

The first three terms of the Magnus expansion (24) can be calculated explicitly for our system (19) and read

$$\boldsymbol{\Omega}_1(t,T) := \boldsymbol{\omega}(t)T + \boldsymbol{\alpha}_k\frac{T^2}{2} \tag{29}$$

$$\boldsymbol{\Omega}_2(t,T) := [\boldsymbol{\alpha}_k]_\times \, \boldsymbol{\omega}_k\frac{T^3}{12} \tag{30}$$

$$\boldsymbol{\Omega}_3(t,T) := [\boldsymbol{\alpha}_k]_\times \, [\boldsymbol{\alpha}_k]_\times \, \boldsymbol{\omega}_k\frac{T^5}{240}. \tag{31}$$

The operator $[\cdot]_\times$ again denotes the skew symmetric matrix operator defined in (19). The Magnus expansion can thus be calculates as

$$\boldsymbol{\Omega}(T) := \boldsymbol{\Omega}_1(T) + \boldsymbol{\Omega}_2(T) + \boldsymbol{\Omega}_3(T) \tag{32a}$$

$$= \boldsymbol{M}(\boldsymbol{\alpha},T)\boldsymbol{\omega}_k + \boldsymbol{\alpha}_k\frac{T^2}{2} \tag{32b}$$

with

$$\boldsymbol{M}(\boldsymbol{\alpha},T) := \boldsymbol{I}_{3\times3}t + [\boldsymbol{\alpha}_k]_\times \, \frac{T^3}{12} + [\boldsymbol{\alpha}_k]_\times \, [\boldsymbol{\alpha}_k]_\times \, \frac{T^5}{240} \tag{33}$$

where $\boldsymbol{I}_{3\times3}$ is the identity matrix.

3.2.2. OTG for SO(3) using rotation matrices. The system dynamics of the rotational portion can now be stated as

$$\boldsymbol{\omega}(t_k + T) = \boldsymbol{f}_{\omega}(\boldsymbol{\omega}_k, \boldsymbol{\alpha}_k, T) := \boldsymbol{\omega}_k + \boldsymbol{\alpha}_k T \tag{34a}$$

$$\boldsymbol{R}(t_k + T) = \boldsymbol{f}_R(\boldsymbol{R}_k, \boldsymbol{\omega}_k, \boldsymbol{\alpha}_k, T) := \exp\left(\left[\boldsymbol{M}(\boldsymbol{\alpha}_k, T)\boldsymbol{\omega}_k + \boldsymbol{\alpha}_k \frac{T^2}{2}\right]_{\times}\right)\boldsymbol{R}_k \tag{34b}$$

where we see that unlike in the translation mapping $\boldsymbol{f}_p$, the orientation mapping $\boldsymbol{f}_R$ is nonlinear and strongly coupled.

Analog to the algorithm for translation, the minimum time needed to drive the rotational velocity error $\boldsymbol{e}_{\omega,k} := \boldsymbol{\omega}_k^{\text{des}} - \boldsymbol{\omega}_k$ to zero is

$$T^*_{\text{min,rot}} := \frac{\left\|\boldsymbol{e}_{\omega,k}\right\|_2}{\alpha_{\max}}. \tag{35}$$

Discretizing the value w.r.t. the given sampling time T_{s} reads

$$T_{\text{min,rot}} := \max\left\{\left\lceil \frac{T^*_{\text{min,rot}}}{T_{\text{s}}} \right\rceil, 1\right\} T_{\text{s}}. \tag{36}$$

The corresponding angular acceleration is

$$\hat{\boldsymbol{\alpha}}_k := \frac{\boldsymbol{e}_{\omega,k}}{T_{\text{min,rot}}} \tag{37}$$

and is analog to $\boldsymbol{v}_k^{\text{goal}}$ from (10) used to find the new goal velocity $\boldsymbol{\omega}_k^{\text{goal}}$ by solving $\boldsymbol{p}_k^{\text{des}} = \boldsymbol{f}_R(\boldsymbol{R}_k, \boldsymbol{\omega}_k, \hat{\boldsymbol{\alpha}}_k, T_{\text{min,rot}})$ for

$$\boldsymbol{\omega}_k^{\text{goal}} := \boldsymbol{M}(\hat{\boldsymbol{\alpha}}_k, T_{\text{min,rot}})^{-1}\left(\left[\log(\boldsymbol{R}^{\text{des}}\boldsymbol{R}_k^T)\right]_{\vee} - \hat{\boldsymbol{\alpha}}_k \frac{T^2_{\text{min,rot}}}{2}\right), \tag{38}$$

which in comparison to our treatment with Euler angle representation in ref. [26] is completely singularity-free and unambiguous. The proof that $\boldsymbol{M}$ is invertible for any $t \neq 0$ is provided in the appendix.

The goal velocity for the next time step is again found by advancing the previous equation for a single iteration

$$\boldsymbol{\omega}_{k+1}^{\text{goal}} := \boldsymbol{f}_{\omega}(\boldsymbol{\omega}_k^{\text{goal}}, \hat{\boldsymbol{\alpha}}_k, T_{\text{s}}) \tag{39}$$

$$= \boldsymbol{M}(\hat{\boldsymbol{\alpha}}_k, T_{\text{min,rot}})^{-1}\left(\left[\log(\boldsymbol{R}^{\text{des}}\boldsymbol{R}_k^T)\right]_{\vee} - \hat{\boldsymbol{\alpha}}_k \frac{T^2_{\text{min,rot}}}{2}\right) + \hat{\boldsymbol{\alpha}}_k T_{\text{s}}. \tag{40}$$

The constraints (4c) and (4d) are again incorporated by defining saturation factor sets $\mathcal{S}_{\omega} \subset \mathbb{R}^+$ and $\mathcal{S}_{\alpha} \subset \mathbb{R}^+$ for the angular velocity and acceleration, respectively. The saturated rotation velocity is thus found by

$$\boldsymbol{\omega}_{k+1}^{\text{sat}} := \boldsymbol{\omega}_{k+1}^{\text{goal}} \min\{\mathcal{S}_{\omega}\}, \tag{41}$$

using the factorization set definition

$$\mathcal{S}_{\omega} := \{1, \mu_{\omega}\} \quad \text{with} \quad \mu_{\omega} := \frac{\omega_{\max}}{\left\|\boldsymbol{\omega}_{k+1}^{\text{goal}}\right\|_2}. \tag{42}$$

This again leads to the find goal acceleration

$$\boldsymbol{\alpha}_k^{\text{goal}} := \frac{\boldsymbol{\omega}_{k+1}^{\text{sat}} - \boldsymbol{\omega}_k}{T_{\text{s}}}. \tag{43}$$

Similarly, defining the acceleration saturation set

$$\mathcal{S}_\alpha := \{1, \mu_\alpha\} \quad \text{with} \quad \mu_\alpha := \frac{\alpha_{\max}}{\left\| \boldsymbol{\alpha}_k^{\text{goal}} \right\|_2} \tag{44}$$

and applying it to the goal acceleration

$$\boldsymbol{\alpha}_k^{\text{sat}} := \boldsymbol{\alpha}_k^{\text{goal}} \min\{\mathcal{S}_\alpha\} \tag{45}$$

results in the saturated angular acceleration $\boldsymbol{\alpha}_k^{\text{sat}}$. This acceleration vector is eventually applied to the SO(3) system dynamics (34), to advance the current orientation states

$$\begin{aligned} \boldsymbol{\omega}_{k+1} &= \boldsymbol{f}_\omega(\boldsymbol{\omega}_k, \boldsymbol{\alpha}_k^{\text{sat}}, T_\text{s}) \\ \boldsymbol{R}_{k+1} &= \boldsymbol{f}_R(\boldsymbol{R}_k, \boldsymbol{\omega}_k, \boldsymbol{\alpha}_k^{\text{sat}}, T_\text{s}) \end{aligned} \tag{46}$$

which concludes the algorithm using rotation matrices.

3.2.3. OTG for SO(3) using unit quaternions. The above algorithm can be straightforward translated to unit quaternion representation of SO(3). This reduces computation time, as discussed in Section 5. Let $\mathcal{Q} \in \mathbb{H}$ describe the 3D orientation as a unit quaternion, that is, $\|\mathcal{Q}\|_2 = 1$. If we consider a quaternion $\mathcal{Q} := (\mathcal{Q}_w, \mathcal{Q}_v)$ consisting of a scalar part $\mathcal{Q}_w$ and a vector part $\mathcal{Q}_v$, the angular velocity $\boldsymbol{\omega}$ can be expressed as a pure quaternion $(0, \boldsymbol{\omega})$, that is, with zero scalar part. We introduce the mappings

$$[\boldsymbol{\omega}]_\mathcal{Q} \quad := (0, \boldsymbol{\omega}) \qquad \mathbb{R}^3 \to \mathbb{H} \tag{47}$$

$$[(0, \boldsymbol{\omega})]_{\mathcal{Q}^{-1}} := \boldsymbol{\omega} \qquad \mathbb{H} \to \mathbb{R}^3 \tag{48}$$

to specify notation. With this notation, the differential equation relating time derivatives of $\mathcal{Q}$ with the angular velocity $\boldsymbol{\omega}$ reads

$$\dot{\mathcal{Q}}(t) = \frac{1}{2} [\boldsymbol{\omega}(t)]_\mathcal{Q} \, \mathcal{Q}(t). \tag{49}$$

Magnus' idea to solve the differential equation results in solving (49) in the form

$$\mathcal{Q}(t+T) = \exp\left(\frac{1}{2} [\boldsymbol{\Omega}(t, T)]_\mathcal{Q}\right) \mathcal{Q}(t). \tag{50}$$

Note that unlike in the Matrix version of Magnus' proposal from (23), $\boldsymbol{\Omega}$ does not occur with the skew symmetric matrix operator $[\cdot]_\times$. The Magnus expansion of $\boldsymbol{\Omega}$ and its approximation (32), however, remain the same.

The mapping that advance the current quaternion in time, hence is

$$\mathcal{Q}(t+T) = \boldsymbol{f}_q(\mathcal{Q}_k, \boldsymbol{\omega}_k, \boldsymbol{\alpha}_k, T) := \exp\left(\frac{1}{2}\left[\boldsymbol{M}(\boldsymbol{\alpha}_k, T)\boldsymbol{\omega}_k + \boldsymbol{\alpha}_k \frac{T^2}{2}\right]_\mathcal{Q}\right) \mathcal{Q}(t). \tag{51}$$

The definition of $\boldsymbol{M}$ in (33) as well as velocity progression (34a) are unchanged. Therefore, the only adjusted equations for the algorithm outlined in Section 3.2 are the definition of $\boldsymbol{\omega}_k^{\text{goal}}$ (38) w.r.t. quaternions

$$\boldsymbol{\omega}_k^{\text{goal}} = 2\boldsymbol{M}(\hat{\boldsymbol{\alpha}}_k, T_{\text{min,rot}})^{-1} \left(\left[\log(\mathcal{Q}^{\text{des}} \mathcal{Q}_k^{-1})\right]_{\mathcal{Q}^{-1}} - \boldsymbol{\alpha}_k \frac{T_{\text{min,rot}}^2}{2} \right) \tag{52}$$

and consequently,

$$\boldsymbol{\omega}_{k+1}^{\text{goal}} = 2\boldsymbol{M}(\hat{\boldsymbol{\alpha}}_k, T_{\text{min,rot}})^{-1} \left(\left[\log(\mathcal{Q}^{\text{des}} \mathcal{Q}_k^{-1})\right]_{\mathcal{Q}^{-1}} - \hat{\boldsymbol{\alpha}}_k \frac{T_{\text{min,rot}}^2}{2} \right) + \hat{\boldsymbol{\alpha}}_k T_\text{s}. \tag{53}$$

The algorithm for OTG on rotations using quaternions is listed in Algorithm 2.

Algorithm 2: OTG on SO(3) with magnitude constraints using quaternions

Input : current state $(\mathcal{Q}_k, \boldsymbol{\omega}_k)$, desired state $(\mathcal{Q}_k^{\text{des}}, \boldsymbol{\omega}_k^{\text{des}})$, constraints $(\omega_{\max}, \alpha_{\max})$, sampling time T_s

Output: $\mathcal{Q}_{k+1}, \boldsymbol{\omega}_{k+1}$

/* ===== find new goal velocity ===== */

$T_{\text{min,rot}} \leftarrow \max\left\{\left\lceil \frac{\|\boldsymbol{\omega}_k^{\text{des}} - \boldsymbol{\omega}_k\|_2}{\alpha_{\max}} \right\rceil, 1\right\} T_\text{s}$; // discretized minimum Time (36)

$\hat{\boldsymbol{\alpha}}_k \leftarrow \frac{\boldsymbol{\omega}_k^{\text{des}} - \boldsymbol{\omega}_k}{T_{\text{min,rot}}}$; // corresponding acceleration (37)

$\boldsymbol{M} \leftarrow \boldsymbol{I}_{3\times3} T_{\text{min,rot}} + \left[\hat{\boldsymbol{\alpha}}_k\right]_\times \frac{T_{\text{min,rot}}^3}{12} + \left[\hat{\boldsymbol{\alpha}}_k\right]_\times \left[\hat{\boldsymbol{\alpha}}_k\right]_\times \frac{T_{\text{min,rot}}^5}{240}$; // Magnus Series (33)

$\boldsymbol{\omega}_{k+1}^{\text{goal}} \leftarrow 2\boldsymbol{M}^{-1}\left(\left[\log(\mathcal{Q}^{\text{des}}\mathcal{Q}_k^{-1})\right]_{\mathcal{Q}^{-1}} - \hat{\boldsymbol{\alpha}}_k \frac{T_{\text{min,rot}}^2}{2}\right) + \hat{\boldsymbol{\alpha}}_k T_\text{s}$; // goal velocity (53)

$\mathcal{S}_\omega \leftarrow$ `collectVelocitySaturationFactors`$(\boldsymbol{\omega}_{k+1}^{goal}, \omega_{max})$;

$\boldsymbol{\omega}_{k+1}^{\text{sat}} \leftarrow \boldsymbol{\omega}_{k+1}^{\text{goal}} \min\{\mathcal{S}_\omega\}$; // saturate goal velocity

/* ===== find goal acceleration ===== */

$\boldsymbol{\alpha}_k^{\text{goal}} \leftarrow \frac{\boldsymbol{\omega}_{k+1}^{\text{sat}} - \boldsymbol{\omega}_k}{T_\text{s}}$; // goal acceleration (43)

$\mathcal{S}_\alpha \leftarrow$ `collectVelocitySaturationFactors`$(\boldsymbol{\alpha}_k^{goal}, \alpha_{max})$;

$\boldsymbol{\alpha}_k^{\text{sat}} \leftarrow \boldsymbol{\alpha}_k^{\text{goal}} \min\{\mathcal{S}_\alpha\}$; // saturate goal acceleration (45)

/* ===== advance current state (34a) and (51) ===== */

$\boldsymbol{\omega}_{k+1} \leftarrow \boldsymbol{f}_\omega(\boldsymbol{\omega}_k, \boldsymbol{\alpha}_k^{\text{sat}}, T_\text{s})$;

$\mathcal{Q}_{k+1} \leftarrow \boldsymbol{f}_\alpha(\mathcal{Q}_k, \boldsymbol{\omega}_k, \boldsymbol{\alpha}_k^{\text{sat}}, T_\text{s})$;

3.3. Synchronization of translation and rotation

While $T_{\text{min,tra}}$ and $T_{\text{min,rot}}$ from (7) and (36) assure time synchronization within the translational and rotational DOF, respectively, the two groups are not yet synchronized with each other. Complete synchronization of the whole 6D movement can be achieved by coupling the translation and rotation at two distinct places in the algorithm. First, the more restrictive minimum time

$$T_{\min} := \max\left(T_{\text{min,tra}}, T_{\text{min,rot}}\right) \tag{54}$$

is used for finding both acceleration vectors $\boldsymbol{a}_k$ and $\boldsymbol{\alpha}_k$ in (9) and (37). Second, the saturation of the goal velocities are matched by combining their saturation factor sets (12) and (42) as

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \mathcal{S}_\omega \tag{55a}$$

$$\mathcal{S}_\omega \leftarrow \mathcal{S}_v \cup \mathcal{S}_\omega \tag{55b}$$

An example of this synchronization is shown in Fig. 2.

4. Extensions for HRC Scenarios

While the presented algorithm considers a free 6D motion of, for example, the tool center point (TCP) of a robot, it does not depend on any specific robot kinematic. In the following section, we present some interesting constraint extensions particularly relevant in HRC context. The examples highlight the flexibility of our approach and demonstrate the straightforward integration for additional requirements by adding appropriate factors to the saturation sets $\mathcal{S}$. The factors are found via a three-step process:

Step 1 formulate a scalar inequality condition, for example,

$$\|\boldsymbol{v}\|_2 \leq v_{\max}$$

Step 2 derive the normalization factor μ that yields equality to 1, for example,

$$\|\boldsymbol{v}\|_2\, \mu = 1 \quad \text{where} \quad \mu = \frac{v_{\max}}{\|\boldsymbol{v}\|_2}$$

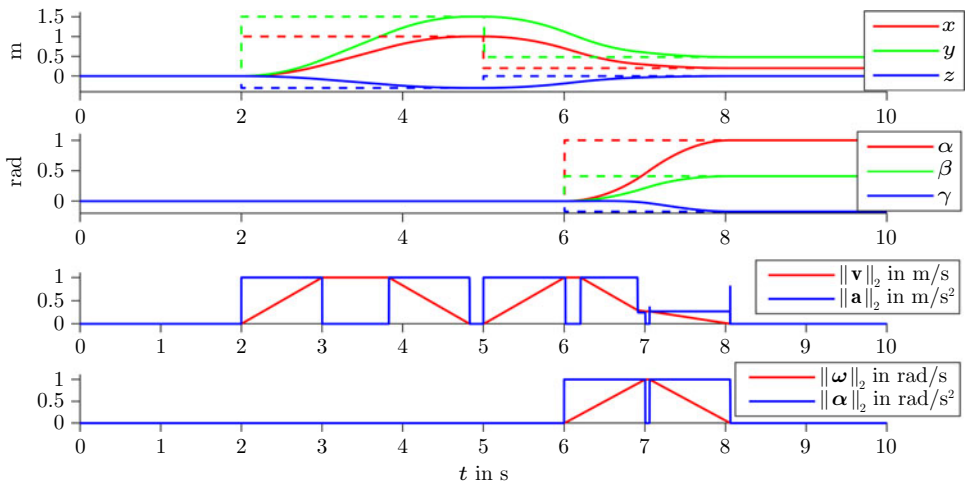

Fig. 2. Time synchronization of translation and rotation.
Dashed lines are set points and solid lines are the actual values. At $t = 6$ s, the jump in desired rotation causes an adjustment of the deceleration phase of the translation trajectories in order to synchronize the movements. Note that absolute speed and acceleration are plotted as magnitude values.

Step 3 add the factor to the corresponding saturation factor set, for example,

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu\}$$

Note that in the following, all constraints refer to the variables at the next discrete-time instance $k+1$. Therefore, we suppress the discrete-time index $k+1$ in the remainder of this section for better readability. Further, Sections 4.2–4.4 require that translation and rotation are synchronized as outlined in Section 3.3.

4.1. Direction-specific constraints

In robot applications, it is often very useful to constrain different directions of movement independently. While the OTGs mentioned in the related work of Section 1.1 offer the possibility to constrain certain axes of the underlying inertial coordinate system directly, our framework can be easily extended to the more general case of incorporating constraints in arbitrary directions. Possible scenarios in HRC that require these type of constraints, are, for example, the dynamic limitation of robot movements toward the human or obstacles. The requirement of constraining the velocity $\boldsymbol{v}_{\text{tra}}$ along a given direction vector $\boldsymbol{r}_{\text{dir}} \in \mathbb{R}^3$ is formulated as

$$\boldsymbol{v}_{\text{tra}}^T \frac{\boldsymbol{r}_{\text{dir}}}{\|\boldsymbol{r}_{\text{dir}}\|_2} \leq \boldsymbol{v}_{\max,r,} \tag{56}$$

where the $\boldsymbol{v}_{\text{tra}}$ is projected onto the direction vector $\boldsymbol{r}_{\text{dir}}$. The necessary saturation factor that complies with this constraint is given as

$$\mu_{\text{dir}} := \frac{\boldsymbol{v}_{\max,r} \, \|\boldsymbol{r}_{\text{dir}}\|_2}{\boldsymbol{v}_{\text{tra}}^T \boldsymbol{r}_{\text{dir}}} \tag{57}$$

and adding it to the velocity saturation set

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu_{\text{dir}}\} \tag{58}$$

incorporates this constraint in (13) of the algorithm. Note that this strategy does not only allow constraining orthogonal axes, but several arbitrary directions.

4.2. Combining translation and rotation constraints

Whenever humans work in close proximity with robots, robot constraints should be defined as intuitive as possible. Besides constraining linear and angular velocity separately as done in Section 3, we outline a strategy to combine the two into a single intuitive constraint.

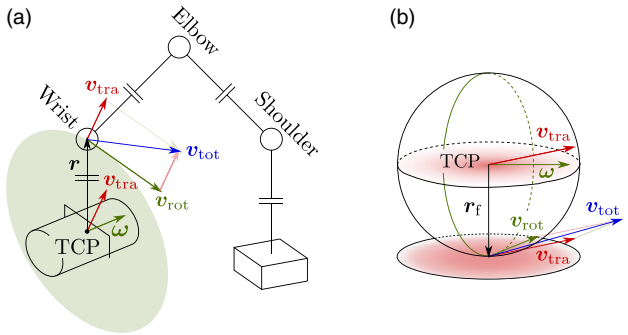

Fig. 3. Illustration of (a) total velocity of, for example, the robot wrist at a fixed vector $\boldsymbol{r}$ and (b) projection of the angular velocity $\boldsymbol{\omega}$ to identify the fastest point $\boldsymbol{r}_\mathrm{f}$ within a user-defined safety sphere of radius r.

Constraining Total Velocity of Specific Points. Instead of constraining the angular speed $\|\boldsymbol{\omega}\|_2 \leq \omega_{\max}$ of the TCP directly, it is sometimes more intuitive to constrain the total linear velocity

$$\|\boldsymbol{v}_{\mathrm{tot}}(\boldsymbol{r})\|_2 := \|\boldsymbol{v}_{\mathrm{tra}} + \boldsymbol{v}_{\mathrm{rot}}(\boldsymbol{r})\|_2 \leq v_{\max} \tag{59}$$

of a given vector $\boldsymbol{r} \in \mathbb{R}^3$, relative to the TCP. The translational and rotational velocities at the TCP is denoted as $\boldsymbol{v}_{\mathrm{tra}} := \boldsymbol{v}$ and $\boldsymbol{v}_{\mathrm{rot}} := \boldsymbol{\omega} \times \boldsymbol{r}$ resp. This can be a specific point on the geometric robot-object compound that is fixed to the TCP, for example, the robots wrist as illustrated in Fig. 3(a). Note that all variables in (59) are given in a world-fixed inertial frame I. In case a specific point is given w.r.t. the end effector coordinate system E, it needs to be transformed, that is,

$${}_I\boldsymbol{r} = \boldsymbol{R}_{IE}\,{}_E\boldsymbol{r}, \tag{60}$$

where $\boldsymbol{R}_{IE}$ rotates the vector from E to the inertial frame I. Let θ denote the angle between $\boldsymbol{\omega}$ and $\boldsymbol{r}$. The fact that the magnitude

$$\|\boldsymbol{v}_{\mathrm{rot}}(\boldsymbol{r})\|_2 = \|\boldsymbol{\omega} \times \boldsymbol{r}\|_2 = \|\boldsymbol{\omega}\|_2\,\|\boldsymbol{r}\|_2 \sin(\theta) \tag{61}$$

is linear in $\boldsymbol{\omega}$ allows us to saturate $\|\boldsymbol{v}_{\mathrm{tot}}\|_2 \leq v_{\max}$ by calculating a common factor

$$\mu_{\mathrm{tot}}(\boldsymbol{r}) := \frac{v_{\max}}{\|\boldsymbol{v}_{\mathrm{tot}}(\boldsymbol{r})\|_2} \tag{62}$$

and adding it to the factorization set

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu_{\mathrm{tot}}\} \tag{63}$$

If n multiple points are to be considered, for example, to outline a convex polytope around the TCP, multiple factors $\mu_{\mathrm{tot},i}$ for $i = [1, n]$ can be calculated and added to the factorization sets in the same fashion. Note that for the TCP itself, $\boldsymbol{r}$ is zero by definition and thus $\boldsymbol{v}_{\mathrm{tot}} = \boldsymbol{v}_{\mathrm{tra}}$, which is already considered in the basic algorithm (13).

Constraining Total Velocity within Safety Sphere. Instead of defining specific points around the TCP, it also possible to define a safety sphere $\mathbb{S} := \{\boldsymbol{x} \in \mathbb{R}^3 : \|\boldsymbol{x}\|_2 \leq r\}$ with radius r around the TCP. It is then desired that no point within the sphere is allowed to exceed the speed limit $v_{\max}$. This extends the before mentioned strategy to first identifying the fastest point $\boldsymbol{r}_\mathrm{f}$ within the sphere.

To maximize (61), $\boldsymbol{r}$ must (a) lie on the sphere surface, that is, have length r and (b) be perpendicular to the rotation vector $\boldsymbol{\omega}$. Further, to maximize $\boldsymbol{v}_{\mathrm{tot}}$, the vector $\boldsymbol{r}$ has to be chosen such that (c) the cross product $\boldsymbol{\omega} \times \boldsymbol{r}$ lies in the plane spanned by $\boldsymbol{\omega}$ and $\boldsymbol{v}_{\mathrm{tra}}$, and (d) the angle between $\boldsymbol{v}_{\mathrm{tra}}$ and $\boldsymbol{v}_{\mathrm{rot}}$ is below 90°. See Fig. 3(b) for an illustration.

Thus, the fastest point $\boldsymbol{r}_\mathrm{f} \in \mathbb{S}$ is given as

$$\boldsymbol{r}_\mathrm{f} := \begin{cases} -\dfrac{\boldsymbol{\omega} \times \boldsymbol{v}_{\mathrm{tra}}}{\|\boldsymbol{\omega} \times \boldsymbol{v}_{\mathrm{tra}}\|_2} r & \text{for } \boldsymbol{\omega} \times \boldsymbol{v}_{\mathrm{tra}} \neq \boldsymbol{0} \\ \dfrac{\boldsymbol{\omega} \times \boldsymbol{r}_\mathrm{p}}{\|\boldsymbol{\omega} \times \boldsymbol{r}_\mathrm{p}\|_2} r & \text{else} \end{cases} \tag{64}$$

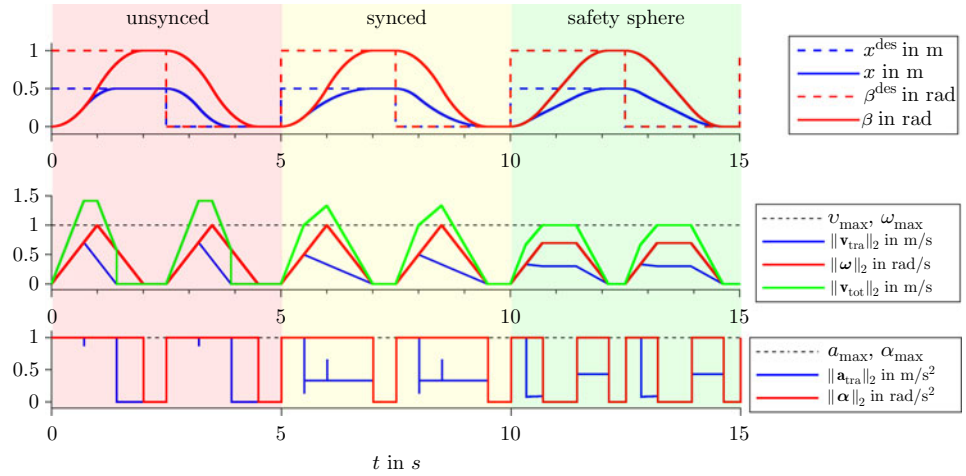

Fig. 4. Combination of translational and angular velocity constraints, by defining a safety sphere of radius $r = 1\,\text{m}$ around the TCP.
For demonstration purposes, the PTP set points are chosen in single-axes directions x and rotation around y by an angle β for translation and rotation, respectively. Three different modes are shown, applying norm constraints $v_{\max} = \omega_{\max} = 1$. The first motion 1–5 s is without synchronization. The second motion 5–10 s is time-synchronized according to Section 3.3, and the third motion 11–15 s is time-synchronized while additionally applying the safety sphere constraint from Section 4.2. Note that the change in the slope of $\|\boldsymbol{v}_{\text{tot}}\|_2$ at 10.3 s stems from the deceleration of $\boldsymbol{v}_{\text{tra}}$ to cruising speed. Both phases of the movement last 0.7 s and are dictated by $a_{\max}$ during deceleration.

where $\boldsymbol{r}_{\text{p}} \in \{\boldsymbol{x} \in \mathbb{R}^3 \backslash \{\boldsymbol{0}\} \mid \langle \boldsymbol{\omega}, \boldsymbol{x} \rangle = 0\}$ is an arbitrary vector perpendicular to $\boldsymbol{\omega}$. Its total speed is

$$\|\boldsymbol{v}_{\text{tot}}(\boldsymbol{r}_{\text{f}})\|_2 = \|\boldsymbol{v}_{\text{tra}} + \boldsymbol{\omega} \times \boldsymbol{r}_{\text{f}}\|_2 \tag{65}$$

which leads to the saturation factor

$$\mu_{\text{sphere}} := \frac{v_{\max}}{\|\boldsymbol{v}_{\text{tot}}(\boldsymbol{r}_{\text{f}})\|_2}. \tag{66}$$

Figure 4 shows the effect of this safety sphere constraint, if added to the set

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu_{\text{sphere}}\} \tag{67}$$

and applied in the algorithm.

4.3. Robot joint constraints

While the presented OTG algorithm can be directly used to generate 6D trajectories of, for example, the robot end effector, joint velocity limits were not yet considered, as they depend on the robot-specific kinematic structure as well as the inverse kinematics solver in use. Because kinematic relations are usually highly nonlinear due to rotary joints, the inverse kinematic solvers often make use of the linear relation

$$\dot{\boldsymbol{z}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{68}$$

between task space velocities, for example, $\dot{\boldsymbol{z}} = \left[\dot{\boldsymbol{v}}^T, \dot{\boldsymbol{\omega}}^T\right]^T \in \mathbb{R}^6$ and joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, via the Jacobian $\boldsymbol{J} = \frac{\partial \dot{\boldsymbol{z}}}{\partial \dot{\boldsymbol{q}}}$. If the expression $\dot{\boldsymbol{q}} = \text{IK}(\dot{\boldsymbol{z}})$ of the inverse kinematics solver in use is known, the joint velocity limits can directly be considered in the algorithm. For demonstration purposes we assume that a damped pseudo-inverse solver of the form

$$\dot{\boldsymbol{q}} = \text{IK}(\dot{\boldsymbol{z}}) := \underbrace{\boldsymbol{J}^T \left(\boldsymbol{J}\boldsymbol{J}^T + \alpha^2 \boldsymbol{I}_{n\times n}\right)^{-1}}_{=:\boldsymbol{J}^\dagger} \dot{\boldsymbol{z}} \tag{69}$$

is used. The identity matrix $\boldsymbol{I}_{n\times n}$ together with the parameter α introduces a damping effect on the inverse kinematics solution. This avoids singularity issues and is discussed in detail in ref. [30]. The individual constraints on the n joint velocities

$$-\dot{q}_{i,\mathrm{lim}} \leq \dot{q}_i \leq \dot{q}_{i,\mathrm{lim}} \quad \text{with } i = [1, n], \tag{70}$$

can concisely be written as

$$\|\boldsymbol{L}\dot{\boldsymbol{q}}\|_\infty \leq 1, \tag{71}$$

with the diagonal limit matrix $\boldsymbol{L} = \mathrm{diag}\left(\dot{q}_{1,\mathrm{lim}}, \ldots, \dot{q}_{n,\mathrm{lim}}\right)^{-1}$. It contains reciprocal values of the joint-specific velocity limits of all n joints in the serial robot kinematics. The linearity of the velocity mapping (68) again admits a scaling factor

$$\mu_{\dot{q}} := \|\boldsymbol{L}\dot{\boldsymbol{q}}\|_\infty^{-1} \tag{72a}$$

$$= \Big\|\boldsymbol{L}\underbrace{\boldsymbol{J}^\dagger \dot{\boldsymbol{z}}}_{\mathrm{IK}(\dot{\boldsymbol{z}})}\Big\|_\infty^{-1}, \tag{72b}$$

such that factorizing the joint velocities $\dot{\boldsymbol{q}}$ is equivalent to factorizing the task space velocities

$$\mu_{\dot{q}}\dot{\boldsymbol{z}} = \mu_{\dot{q}}\boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{73}$$

directly. Thus, adding $\mu_{\dot{q}}$ to the saturation factor set

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu_{\dot{q}}\} \tag{74}$$

constrains the joint velocities accordingly.

Note, that constraining the joint acceleration $\ddot{\boldsymbol{q}}$ in the same manner is not possible, because the kinematic relation for accelerations

$$\ddot{\boldsymbol{z}} = \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} + \boldsymbol{J}\ddot{\boldsymbol{q}} \tag{75a}$$

$$= \dot{\boldsymbol{J}}\boldsymbol{J}^\dagger \dot{\boldsymbol{z}} + \boldsymbol{J}\ddot{\boldsymbol{q}} \tag{75b}$$

couples task space velocity $\dot{\boldsymbol{z}}$ and acceleration $\ddot{\boldsymbol{z}}$. Thus, a linear relation such as (73), that elevates the joint constraint factor to task space, is not given.

4.4. Constraining movement of the whole robot kinematic

In a serial kinematic robot as usually used in a HRC context, the most important movement to constrain is the movement of the end effector or TCP. However, there might be other points on the robot structure that reach even higher velocities. Thus, it is necessary to not only constrain the TCP movement itself, but extend the given speed constrain to every point in the robot kinematic structure in task space. Only then can official safety standards, for example, ISO/TS 15066[1] be satisfied. Given a conventional rigid link structure consisting of joint axes only orthogonal or parallel to the links, such as illustrated in Fig. 3(a), it is sufficient to check the set of axes intersecting points $\mathcal{I}$ of the kinematic for

$$\|\boldsymbol{v}_i\|_2 \leq v_{\max} \quad \forall i \in \mathcal{I}. \tag{76}$$

The velocities $\boldsymbol{v}_i$ are calculated by

$$\|\boldsymbol{v}_i\|_2 := \left\|\boldsymbol{J}_i(\boldsymbol{q}_i)\dot{\boldsymbol{q}}_i\right\|_2 \tag{77}$$

using the corresponding Jacobian matrices $\boldsymbol{J}_i$ with the reduced joint velocity vectors $\dot{\boldsymbol{q}}$. In the illustrated 7DOF kinematic, these are $\mathcal{I} = \{\text{shoulder, elbow, wrist, TCP}\}$. Every constraint leads to a saturation factor

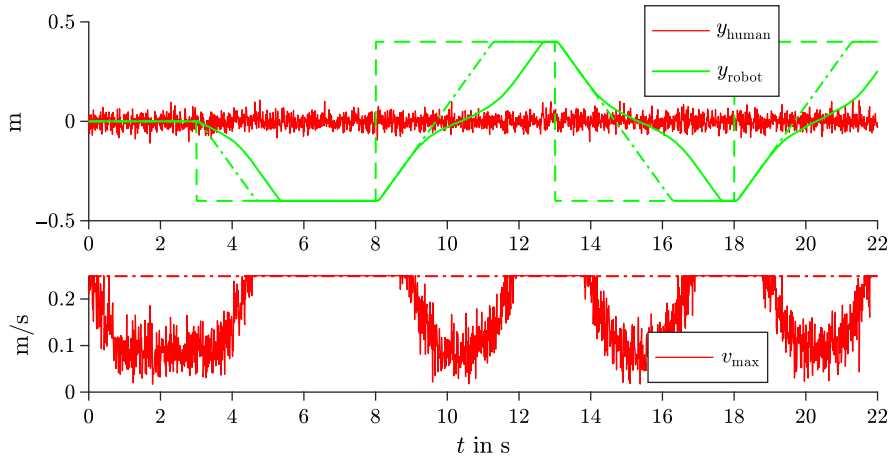

Fig. 5. Hardware Experiment: distance-sensitive $v_{\max}(d_{\text{HR}})$ with $\sigma = \frac{d_{\min}}{3}$.
Dashed lines are set points, dash-dotted lines result from a constant $v_{\max}$, and solid lines from $v_{\max}(d_{\text{HR}})$ as defined in (80). The OTG clearly slows down whenever the distance becomes smaller $d_{\min} = 20$ cm and uses the user-defined constraint $v_{\max}$ otherwise.

$$\mu_{\text{kin},i} := \frac{v_{\max}}{\|\boldsymbol{v}_i\|_2} \quad \forall i \in \mathcal{I} \tag{78}$$

that as part of the sets

$$\mathcal{S}_v \leftarrow \mathcal{S}_v \cup \{\mu_{\text{kin},i}\} \tag{79}$$

slows down the end effector velocity in the OTG whenever necessary.

Note that in the discussed conventional serial 7DOF, this method is only relevant for constraining the elbow movement. The translational shoulder velocity is always zero and can thus be omitted. Further the TCP being the general point of interest is by definition already constrained in the basic algorithm. The wrist can be considered independently of the kinematic structure in use, following the method in Section 4.2.

4.5. OTG-independent constraints

Considering time-variant velocity constraints opens up interesting application scenarios too. It allows for integrating OTG-independent metrics to be considered in the OTG algorithm. An exemplary use case in HRC is a distance-sensitive velocity limitation. In terms of safety issues in HRC, the maximal allowed robot velocity in the proposed OTG is made dependent on the Euclidean distance d_{HR} between human and robot using, for example, a Gaussian-shaped weighting term

$$v_{\max}(d_{\text{HR}}) := \begin{cases} \exp\left(-\dfrac{(d_{\text{HR}} - d_{\min})^2}{2{\sigma_A}^2}\right) v^*_{\max}, & \text{for } d_{\text{HR}} \leq d_{\min} \\ v^*_{\max}, & \text{else} \end{cases} \tag{80}$$

with the shaping factor σ_A. A shaping factor of $\frac{d_{\min}}{3}$, for example, scales the velocity to $<1\%$ for $d_{\text{HR}} \to 0$. Figure 5 shows the comparison between a fixed $v_{\max}$ and the distance-sensitive implementation (80) for the case when the robot crosses 10 cm above the human hand. In this example, it is further demonstrated how the OTG copes with measurement noise of the human position.

5. Discussion

In this Section, we will discuss the performance of the proposed algorithm in terms of transient behavior. We further give a runtime comparison to the state-of-the-art.

5.1. Transient behavior

Figure 6 shows the response of the approach to several step inputs and time-invariant constraints. Note that for such inputs, the algorithm results in a time optimal bang-bang behavior in acceleration. The user constraints (4a) and (4b) are fulfilled at all times, while (7)–(9) assures time synchronization.

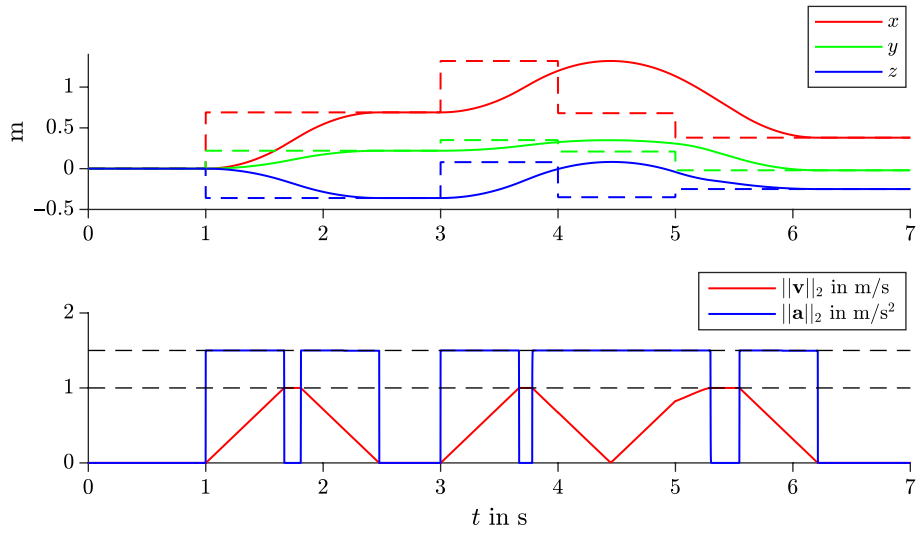

Fig. 6. Trajectory generation for translation with hanging transition.
Dashed lines are set points/constraints and solid lines are the actual time-synchronized values. The spikes in acceleration result from the transition phase due to time discretization. The change in set point at $t = 5$ s causes an adjustment in the direction of the acceleration vector, while its full magnitude $a_{\max} = 1.5$ m/s^2 is still exploited.

In case of trajectory following where the input trajectory fulfills the inequality constraints (4), the algorithm degenerates to

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k^{\text{des}} \tag{81a}$$

$$\mathcal{Q}_{k+1} = \mathcal{Q}_k^{\text{des}}. \tag{81b}$$

once the desired state is reached.

Thus, in our algorithm, no explicit switching between different strategies is necessary. This means that the algorithm uses axes-synchronized acceleration bang-bang control in cases of constraint violating goal trajectories, no matter if they result from discontinuous set point trajectories or merely from, for example, infeasible velocities. This reduces the error in position and velocity offset and eventually feeds through the target points whenever possible. An example of this behavior is shown in Fig. 7.

5.2. Analytic versus Geometric Angular velocity constraints

Current OTG algorithms mentioned in literature do either not provide synchronization treatment of the multidimensional case, or treat all DOF as independent axes for trajectory generation. Directly applying joint-related approaches to Euler angles for representing SO(3)typically are prone to so-called gimbal-lock singularities. As analyzed in Section 3.2, instead of directly regarding constraints for the time derivatives of the Euler angles, we transfer the problem from this analytical subsequent velocity vector to the geometric angular velocities $\boldsymbol{\omega}$. Note that these two velocity vectors differ clearly in their physical interpretation. The time derivative of Euler angles describes consecutive angular velocities according to the given Euler sequence, whereas $\boldsymbol{\omega}$ contains simultaneous angular velocity around the base vectors at a given time instance. Constraining these two vectors directly results obviously in different trajectories for a general case. In Fig. 8, we demonstrate the divergence of the generated trajectories for some extreme cases. The closer the configuration gets to the singularity, the larger the discrepancy between Euler angle velocities and actual geometric angular velocities. The differences can be seen in velocity as well as acceleration level. We argue that constraining the angular velocity $\boldsymbol{\omega}$ in its magnitude gives raise to more intuitive trajectories than constraining the successive Euler angle velocities.

5.3. Runtime analysis

As mentioned in the introduction, optimization-based approaches especially when considering the required nonlinear magnitude constraints as well as dynamics on SO(3) typically fail to deliver results

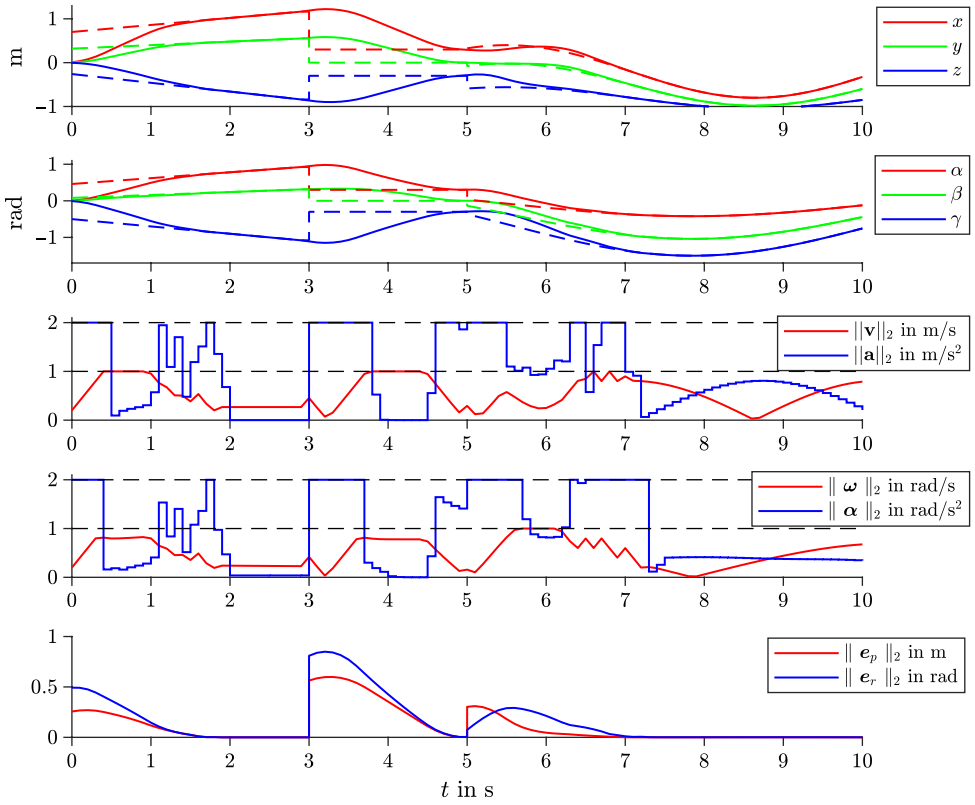

Fig. 7. Time synchronous transition between acceleration-bounded input trajectories and step inputs for a sample rate of 0.1 s.
The discontinuous desired profile consists of three distinct segments. An initial ramp with offset (0–3 s) followed by a constant (3–6 s) and a sine with distinct frequencies (6–10 s). State discrepancies at time $t = 0$ s that usually result in an error state of the robot can be avoided. After the transition phase, the error for translation and rotation (bottom plot) converge to direct feedthrough of the desired trajectory as discussed in (81).

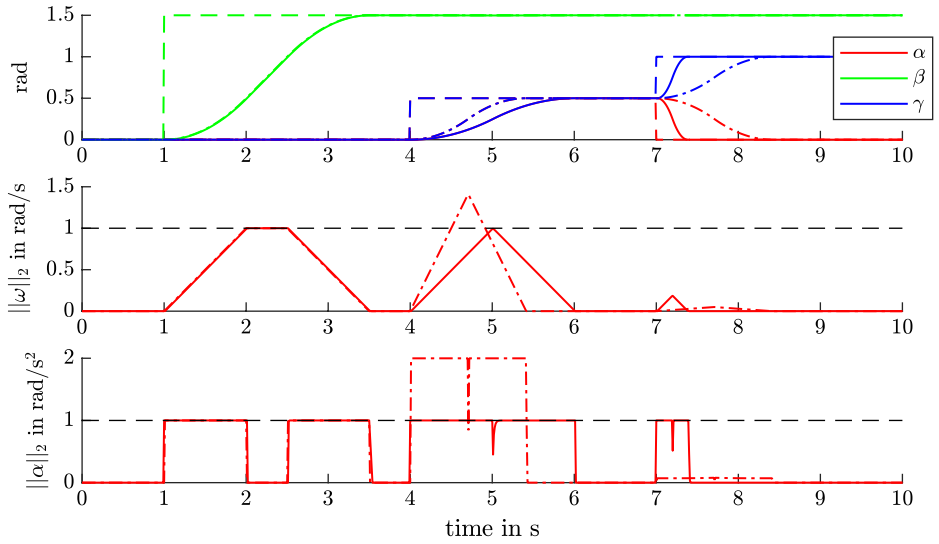

Fig. 8. Comparison of our algorithm on SO(3) to the Reflexxes Motion Library, parametrizing Euler angles, in terms of magnitude velocity and acceleration.
Note that in the given Euler angle sequence $X(\alpha) \rightarrow Y(\beta) \rightarrow Z(\gamma)$, the gimbal-lock singularity is reached at $\beta = \pi/2 \approx 1.57$ rad. Dashed lines are desired values/constraints, solid lines reflect our approach, and dash-dotted lines result from the Reflexxes Motion Library. While α and γ accelerate in the same direction at $t = 4$ s reaching almost double the constrained acceleration limit, constraining the Euler angle derivatives is overrestrictive at $t = 7$ s due to the opposite movement directions.

Table II. Runtime comparison for full 6D OTG.

Algorithm	**Reflexxes**		**Our approach**	
Representation	Euler angles[10]	Euler angles[26]	Rotation matrix	Unit quaternion
Iteration time	13.3 µs	8.3 µs	7.6 µs	**5.1 µs**

within common 1 ms iteration cycles. Therefore, they cannot be considered for the problem at hand. Representing approaches that adapt methods in $\mathbb{R}^6$ using Euler angle parametrization, we compare our algorithm with the available Reflexxes Motion Library[31] that is based on the work of Kröger.[10] For a fair comparison, the freely available Type II version (unbound jerk) is used. The algorithm was compared in terms of runtime with the OTG of the available Reflexxes Motion Library. Both algorithms were analyzed using the Simulink 8.9 Profiler evaluating 10^6 iterations on a single 3.7 GHz CPU. The resulting computational effort is listed in Table II. Although the implementation and compilation of our algorithm are not explicitly optimized for runtime, the computation time spent per iteration is below the Reflexxes Motion library for all three representations of 3D orientation in SO(3). This makes the proposed algorithm suitable as an OTG or an intermediate layer between a optional higher level trajectory planner, that may run on a slower sample rate, and the robot controller. This increases robustness without large additional computational load.

6. Conclusion

In this work, a new approach for an OTG was introduced. According to the classification by Kröger et al.[32] our method describes an online trajectory generator OTG of type II (i.e., bounded velocity and acceleration, unbounded jerk, allowing position and velocity targets) in variant B (that is time-variant constraints). It is directed—but not limited—to be used in HRC scenarios, where assuring human comfort and guaranteeing safety are prioritized over time optimality. While offering the typical advantages of OTG, such as increasing robustness against infeasible input trajectories (e.g., PTP step trajectories) and instantaneous reaction capabilities to unknown events, it offers intuitive definition of dynamic constraints. That is, constraining the magnitude of velocity and acceleration vectors of 3D translation as well as 3D rotation. This way the coordinate system-independent dynamics of the end effector, which can also be interpreted as kinetic energies on velocity level, are directly constrained. Further, rotations are constrained in their true geometric angular velocity, rather than purely analytical values such as direct derivation of Euler angles. Accurate time integration of the orientation differential equations is achieved by the means of the Magnus expansion.

This is the first real-time capable OTG algorithm to allow such constraints, while directly generating 3D translation and rotation trajectories on SE(3) in a singularity-free formulation. It was also outlined how the set of constraints can be extended to limit the velocity of the full robot structure in joint as well as 3D Euclidean space. Considering time variance of these constraints opens up many possibilities to connect other metrics such as human–robot distance. Our work introduces an algorithm that provides the advantages of an indirect approach, that is, fast computation cycles and instantaneous reaction to unforeseen inputs. At the same time, it also allows seamless transitions to directly forward trajectories that already satisfy dynamic constraints. These transitions do not require an explicit switching or blending between different strategies, but directly result from the algorithm itself measures to the OTG, as shown in experimental evaluation. The beauty of our algorithm lies clearly in its simplicity and its reduced and intuitive definition of the constraints, especially in terms of orientation. The verification of increasing acceptance by constraining magnitudes is still to be validated in user case studies. Suggestions for future development are the extension jerk limitation, which is of importance especially in industrial contexts. This requires a new step in the algorithm in which the correct acceleration profile has to be found before calculating the minimum time left.

Acknowledgment

The research leading to these results has received funding from the Horizon 2020 research and innovation program under grant agreement No. 820742 of the project “HR-Recycler—Hybrid Human-Robot RECYcling plant for electriCal and eLEctRonic equipment.”

References

1. ISO/TS 15066:2016, "Robots and robotic devices – collaborative robots," *International Organization for Standardization*, Geneva, CH (2016).
2. W. K. Chung, L.-C. Fu and T. Kröger, "Trajectory Generation and Planning," **In**: *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), 2nd ed. (Springer, Cham, 2016) ch. 8.9.
3. T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. Robot.* **26**(1), 94–111 (2010).
4. X. Broquere, D. Sidobre and I. Herrera-Aguilar, "Soft Motion Trajectory Planner for Service Manipulator Robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France (2008).
5. R. Haschke, E. Weitnauer and H. Ritter, "On-Line Planning of Time-Optimal, Jerk-Limited Trajectories," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France (2008).
6. R. Katzschmann, T. Kroger, T. Asfour and O. Khatib, "Towards Online Trajectory Generation Considering Robot Dynamics and Torque Limits," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan (2013).
7. B. Ezair, T. Tassa and Z. Shiller, "Planning high order trajectories with general initial and final conditions and asymmetric bounds," *Int. J. Robot. Res.* **33**(6), 898–916 (2014).
8. F. Lange and A. Albu-Schäffer, "Path-accurate online trajectory generation for jerk-limited industrial robots," *IEEE Robot. Auto. Lett.* **1**(1), 82–89 (2016).
9. F. Lange and A. Albu-Schäffer, "Iterative path-accurate trajectory generation for fast sensor-based motion of robot arms," *Adv. Robot.* **30**(21), 1380–1394 (2016).
10. T. Kröger, "Online trajectory generation: Straight-line trajectories," *IEEE Trans. Robot.* **27**(5), 1010–1016 (2011).
11. Z. Rymansaib, P. Iravani and M. N. Sahinkaya, "Exponential Trajectory Generation for Point to Point Motions," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Wollongong, NSW (2013).
12. J. E. Lloyd, "Trajectory Generation Implemented as a Non-linear Filter," *Technical Report* (Department of Computer Science, University of British Columbia, Vancouver, B.C., Canada, 1998).
13. L. Biagiotti and C. Melchiorri, "FIR filters for online trajectory planning with time-and frequency-domain specifications," *IFAC Control Eng. Pract.* **20**(12), 1385–1399 (2012).
14. L. Biagiotti, C. Melchiorri, and L. Moriello, "Optimal trajectories for vibration reduction based on exponential filters," *IEEE Trans. Control Syst. Tech.* **24**(2), 609–622 (2016).
15. P. Besset, R. Bearee and O. Gibaru, "FIR Filter-Based Online Jerk-Controlled Trajectory Generation," *IEEE International Conference on Industrial Technology (ICIT)*, Taipei (2016).
16. O. Gerelli and C. G. L. Bianco, "A Discrete-time Filter for the On-Line Generation of Trajectories with Bounded Velocity, Acceleration, and Jerk," *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska (2010).
17. C. G. L. Bianco and F. Ghilardelli, "A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerk," *IEEE Trans. Indust. Electron.* **61**(8), 4115–4125 (2014).
18. C. G. L. Bianco, "An efficient algorithm for the real-time generation of synchronous reference signals," *IEEE Trans. Indust. Electron.* **64**(6), 4621–4630 (2017).
19. M. M. G. Ardakani, B. Olofsson, A. Robertsson and R. Johansson, "Real-Time Trajectory Generation Using Model Predictive Control," *IEEE International Conference on Automation Science and Engineering (CASE)*, Gothenburg, Sweden (2015).
20. K. H. Dinh, P. Weiler, M. Leibold and D. Wollherr, "Fast and Close to Optimal Trajectory Generation for Articulated Robots in Reaching Motions," *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, Munich, Germany (2017).
21. O. Koç, G. Maeda and J. Peters, "Online optimal trajectory generation for robot table tennis," *Robot. Auto. Syst.* **105**, 121–137 (2018).
22. F. Gao, Y. Lin and S. Shen, "Gradient-Based Online Safe Trajectory Generation for Quadrotor Flight in Complex Environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada (2017).
23. T. Lee, M. Leok and N. H. McClamroch, "Time Optimal Attitude Control for a Rigid Body," *IEEE American Control Conference (ACC)*, Seattle, Washington (2008).
24. M. Watterson, T. Smith and V. Kumar, "Smooth Trajectory Generation on SE(3) for a Free Flying Space Robot," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (2016).
25. A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Trans. Robot.* **32**(5), 1196–1214 (2016).
26. G. Huber, V. Gabler and D. Wollherr, "An Online Trajectory Generator on SE(3) with Magnitude Constraints," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada (2017).
27. M. Boyle, "The integration of angular velocity," *Adv. Appl. Clifford Alg.* **27**(3), 2345–2374 (2017).
28. W. Magnus, "On the exponential solution of differential equations for a linear operator," *Commun. Pure Appl. Math.* **7**(4), 649–673 (1954).

29. S. Blanes, F. Casas, J. Oteo and J. Ros, "The magnus expansion and some of its applications," *Phys. Rep.* **470**(5–6), 151–238 (2009).
30. S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE J. Robot. Auto.* **17**(1–19), 16 (2004).
31. Reflexxes GmbH. The reflexxes motion libraries, type II. http://www.reflexxes.ws (accessed February 12, 2016).
32. T. Kröger, "On-Line Trajectory Generation in Robotic Systems," **In**: *Springer Tracts in Advanced Robotics* (B. Siciliano and O. Khatib, eds.) (Springer-Verlag, Berlin, Heidelberg, 2010) pp. 38–40.
33. R. M. Murray, Z. Li and S. Sastry, *A Mathematical Introduction to Robotics Manipulation* (CRC Press, Boca Raton, Florida, 1994).

A. Appendix

A.1. Implementation remarks

A.1.1. Matrix logarithm. Although the exponential map, and thus the logarithmic map, are defined with as an infinite series

$$\exp(\boldsymbol{R}) := \sum_{n=0}^{\infty} \frac{\boldsymbol{R}^n}{n!} \tag{A1}$$

in the case of $\boldsymbol{R} \in \mathfrak{so}3$ such as $[\boldsymbol{\omega}]_\times$ there is a closed-form solution

$$\exp([\boldsymbol{\omega}]_\times) := I + \sin(\|\boldsymbol{\omega}\|_2)\frac{[\boldsymbol{\omega}]_\times}{\|\boldsymbol{\omega}\|_2} + (1 - \cos(\|\boldsymbol{\omega}\|_2))\left(\frac{[\boldsymbol{\omega}]_\times}{\|\boldsymbol{\omega}\|_2}\right)^2 \tag{A2}$$

known as Rodrigues' rotation formula.[33]

A.1.2. Quaternion logarithm. Using quaternions for describing rotations in SO(3) always results in unit quaternions, that is, $\|\mathcal{Q}\|_2 = 1$. If a unit quaternion $\mathcal{Q} \in \mathbb{H}$ is considered to have a scalar and a vector part $\mathcal{Q} = (\mathcal{Q}_w, \mathcal{Q}_v)$ similar to the closed-form matrix expression above, the logarithm for a pure quaternion, that is, with zero scalar part, admits the concise closed-form solution

$$\exp(\mathcal{Q}) := \left(\cos(\|\mathcal{Q}_v\|_2\,,\, \mathcal{Q}_v \sin\left(\frac{\|\mathcal{Q}_v\|_2}{\|\mathcal{Q}_v\|_2}\right)\right) \tag{A3}$$

and the logarithm of a unit quaternion, that is, $\|\mathcal{Q}\|_2 = 1$, the logarithm simplifies to

$$\log(\mathcal{Q}) := \left(0,\, \frac{\phi}{\sin(\phi)\mathcal{Q}_v}\right) \qquad \text{with } \phi = \arctan 2(\|\mathcal{Q}_v\|_2\,,\, \mathcal{Q}_w) \tag{A4}$$

A.2. Proof that $\boldsymbol{M}(\alpha, t)$ in (33) has full rank

Proof. The symbolic calculation of the eigenvalues of $\boldsymbol{M}(\boldsymbol{\alpha}, t)$ leads to

$$\operatorname{eig}(\boldsymbol{M}(\boldsymbol{\alpha}, t)) = \begin{bmatrix} t \\ t - \|\boldsymbol{\alpha}\|_2^2 \frac{t^5}{240} + j\,\|\boldsymbol{\alpha}\|_2 \frac{t^3}{12} \\ t - \|\boldsymbol{\alpha}\|_2^2 \frac{t^5}{240} - j\,\|\boldsymbol{\alpha}\|_2 \frac{t^3}{12} \end{bmatrix} \tag{A5}$$

which is guaranteed to be of full rank, as long as $t \neq 0$. The two contradicting conditions $\boldsymbol{\alpha}$ for rank deficiency are

$$\|\boldsymbol{\alpha}\|_2^2 \frac{t^5}{240} = t \tag{A6}$$

$$\|\boldsymbol{\alpha}\|_2 \frac{t^3}{12} = 0. \tag{A7}$$