2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Spherical Cubic Blends: $\mathcal{C}^2$-Continuous, Zero-Clamped, and Time-Optimized Interpolation of Quaternions

Jonas Wittmann[1,2], Lukas Cha[1], Marco Kappertz[1], Philipp Seiwald[1] and Daniel J. Rixen[1]

***Abstract*— Modern collaborative robotic applications require robot motions that are predictable for human coworkers. Therefore, trajectories often need to be planned in task space rather than configuration space ($\mathcal{C}$-space). While the interpolation of translations in Euclidean space is straightforward, the interpolation of rotations in $SO(3)$ is more complex. Most approaches originating from computer graphics do not exhibit the often desired $\mathcal{C}^2$-continuity in robotics. Our main contribution is a $\mathcal{C}^2$-continuous, zero-clamped interpolation scheme for quaternions that computes a fast synchronized motion given a set of waypoints. As a second contribution, we present modifications to two state-of-the-art quaternion interpolation schemes, *Spherical Quadrangle Interpolation (SQUAD)* and *Spherical Parabolic Blends (SPB)*, to enable them to compute $\mathcal{C}^2$-continuous, zero-clamped trajectories. In experiments, we demonstrate that for the time optimization of trajectories, our approach is computationally efficient and at the same time computes smooth trajectory profiles.**

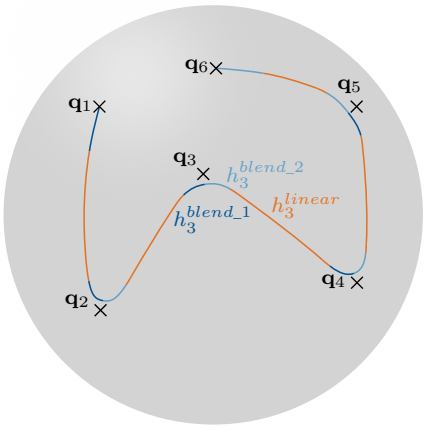

Fig. 1. Schematic visualization of the *SCB* curve on the surface of the four-dimensional hypersphere according to [5]. Orange lines show the *linear* motion phase of *SCB*. Dark blue lines show the first half of the *blend* phase and light blue lines show the second half. $h_3^\star$ indicate the time duration of the motion phases around the third waypoint. Black crosses indicate the orientation waypoints.

I. Introduction

Sampling-based planning algorithms, e.g. RRT, operate in the robot's configuration space ($\mathcal{C}$-space) and are state-of-the-art methods to compute robot motions in dynamic environments. However, when robots share the work space with human coworkers, the motions have to be predictable. Further, many robotic automation solutions, e.g. welding, define waypoints in Cartesian space. In these scenarios, trajectory generation methods that interpolate task space waypoints are applied and subsequent differential inverse kinematics (IK) transform the trajectory to $\mathcal{C}$-space. To limit stress to the actuators, the Cartesian space trajectories have to be $\mathcal{C}^2$-*continuous* and *zero-clamped*, i.e., with zero velocities and accelerations at the first and last waypoints. Besides this first objective of *smoothness*, *time optimization* is a second objective to ensure short process times.

The translational part of Cartesian space trajectories and $\mathcal{C}$-space trajectories are both part of the linear Euclidean space and various interpolation methods have been proposed to compute $\mathcal{C}^2$-continuous, zero-clamped trajectories, e.g. time-optimal cubic splines in [1]. The interpolation of the rotational part, however, is more complex. Rotations in Cartesian space are part of the nonlinear $SO(3)$ group, whose members are defined by a minimal representation with three parameters. The four state-of-the-art representations are: *Euler angles*, *axis-angle*, *rotation matrices* and *unit quaternions* [2]. *Euler angles* and *axis-angle* suffer from singularities [3]. Compared to *rotation matrices* with nine parameters, *unit quaternions* are defined by four parameters. Owing to their computational efficiency, quaternions are often used, e.g. for the pose message type in ROS [4].

State-of-the-art interpolation methods for quaternions were developed for computer graphics [6]. When applied in robotics, these approaches lack smoothness, i.e. $\mathcal{C}^2$-continuity and zero-clamped boundaries, which is required to limit stress to the actuators. We propose a smooth interpolation scheme for quaternions, *Spherical Cubic Blends (SCB)*, that computes a fast synchronized motion given a set of quaternion waypoints. Further, we contribute modifications to the state-of-the-art methods *SQUAD* [7] and *SPB* [8] that enable them to plan $\mathcal{C}^2$-continuous zero-clamped trajectories.

Robot manufacturers derive task space limits from a conservative estimation based on the joint actuators' limits, and they specify the maximum absolute angular velocity $\omega^{\max}$, acceleration $\dot{\omega}^{\max}$ and jerk $\ddot{\omega}^{\max}$, e.g. in [9]. Note that we focus on rotational trajectories only in the following. In the experiments in Section IV, we use quintic spline interpolation for the translational part of the trajectory. Further, we use the *Resolved Motion Rate Control (RMRC)* [10] to compute the control signals for the actuators in $\mathcal{C}$-space. As RMRC is able to propagate the degree of smoothness through the IK computation, the resulting $\mathcal{C}$-space trajectories maintain the $\mathcal{C}^2$-continuity of the planned task space trajectories.

[1]Technical University of Munich, TUM School of Engineering and Design, Department of Mechanical Engineering, Munich Institute of Robotics and Machine Intelligence (MIRMI), Chair of Applied Mechanics, Boltzmannstraße 15, 85748 Garching, Germany
[2]Corresponding author; e-mail: jonas.wittmann@tum.de

II. Related Work

Concerning the generation of smooth trajectories in $SO(3)$ as the first objective, [7] presents *SLERP* and *SQUAD* in the context of computer graphics. *SLERP* keeps a constant rotation axis in between the waypoints and computes discontinuous velocity profiles when applied to multiple waypoints. Therefore, in robotics, *SLERP* is generally combined with a time scaling approach. *SQUAD* smoothly changes the rotation axis and is preferred in robotics. However, the profiles are not zero-clamped and *SQUAD* ensures $\mathcal{C}^1$-continuity only in the case of equal time duration in between the waypoints. *SPB* [8] enhances *SLERP* by a parabolic blending phase at the waypoints such that the change of the rotation axis is continuous. It is $\mathcal{C}^1$-continuous and ensures constant rotation axes during most of the time. We propose *SCB*, that also blends in between waypoints. However, in contrast to *SPB*, *SCB* formulates blending on velocity and jerk level and thus obtains smoother $\mathcal{C}^2$-continuous profiles. The idea to connect constant velocity segments with blend phases stems from the $\mathcal{C}$-space method *Linear Interpolation with Parabolic Blends* [11]. [12] proposes a method of smooth orientation interpolation via element-wise quaternion interpolation and re-normalization. [13] fits a quaternion B-Spline curve to the waypoints by formulating a quadratic optimization problem. While ensuring $\mathcal{C}^2$-continuity, both approaches do not consider actuator limits which is required for time-optimized motions, our second objective besides smoothness.

[14] proposes a time optimization approach in $SO(3)$ with efficient batch-wise optimization that derives ideas from model predictive control. However, this method is unable to maintain a constant axis of rotation in between waypoints. In $\mathcal{C}$-space, time optimization can be efficiently solved with analytical gradient information and gradient-based optimization. [15], [16], [17] derive analytical gradients for cubic splines and B-splines. However, in $SO(3)$ the derivation of analytical gradient information is often not possible due to the nonlinearity and one has to resort to numerical gradient computation [18] or apply iterative methods that do not require gradient computation. Our approach is similar to [19], a trajectory generator for quaternions that is also applicable for translations. They exploit actuator limits for time-optimized motions and at the same time ensure smooth trajectories. Similar to them, we also divide the angular velocity profile into three phases (see Fig. 1) and perform interpolation on velocity level. Their approach is more generic, allows an order of smoothness that is higher and selectable and they further provide closed-form solutions to exactly hit actuator limits while we iteratively converge towards the limits in case they would be violated. However, in applications where $\mathcal{C}^2$-continuity is sufficient, our formulation is more compact. The approach in [19] requires point-to-point (PTP) trajectories as inputs and interpolates between the deceleration and acceleration phases of two subsequent trajectories. Our approach directly interpolates between two segments using constant jerk. Further, we provide a benchmark with state-of-the-art approaches and an evaluation on hardware.

III. Proposed Method

In this chapter, we propose *SCB*, a trajectory generation approach that ensures $\mathcal{C}^2$-continuous, zero-clamped and time-optimized quaternion interpolation. Further, we propose modifications of *SQUAD* and *SPB* to increase their smoothness. Note that *time-optimized* means that with the given interpolation scheme and boundary conditions a fast trajectory is computed that exploits given actuator limits. However, this is not a *time-optimal* trajectory that is commonly computed with a bang-bang profile.

A. Spherical Cubic Blends

SCB translates the *Parabolic Blends* approach in $\mathcal{C}$-space [20] to $SO(3)$. However, it blends waypoints on jerk level rather than acceleration level to increase the trajectory's smoothness. In Euclidean space, the constant jerk values during blending would result in cubic profiles on position level. Therefore, we refer to it as *Spherical Cubic Blending* although the time integration in $SO(3)$ leads to a nonlinear quaternion profile.

SCB defines the interpolation on angular velocity $\boldsymbol{\omega}$, acceleration $\dot{\boldsymbol{\omega}}$ and jerk $\ddot{\boldsymbol{\omega}}$ level. In contrast to that, the state-of-the-art approaches in Section III-B, apart from [19], interpolate trajectories on orientation level. Interpolating on position level is not a problem for translations. Positions are defined in Euclidean space and the mapping between position and its time derivatives is linear. Thus, interpolation characteristics on position level, e.g. smoothness, also propagate to velocity, acceleration and jerk level. Orientations, however, are defined in the nonlinear $SO(3)$, while $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$, $\ddot{\boldsymbol{\omega}}$ are defined in Euclidean space. The mapping between quaternions $\mathbf{q}$ and $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$, $\ddot{\boldsymbol{\omega}}$ is nonlinear. Thus, interpolation characteristics inherent to interpolation schemes on orientation level do not propagate to velocity, acceleration and jerk level. Therefore, the $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$, $\ddot{\boldsymbol{\omega}}$ profiles of *SQUAD* and *SPB* in Section III-B are not defined by smooth polynomials, which is a drawback for control approaches that use IK on velocity-level. The reference signals are of lower quality and induce vibrations due to the acceleration discontinuities.

SCB interpolates in the Euclidean space of $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$, $\ddot{\boldsymbol{\omega}}$ and computes smooth and analytically described control signals. The reference quaternions $\mathbf{q}_k$, required for drift and error compensation on orientation level, are obtained by integration. *SCB* uses two different motion phases: a *linear* phase and a *blend* phase (see Fig. 1). In the *linear* phase, the rotation axis and $\boldsymbol{\omega}$ are constant, similar to *SPB*. The *blend* phase interpolates between the angular velocities of two subsequent linear phases using linear $\dot{\boldsymbol{\omega}}$ profiles. Therefore, the blend phase is divided into two halves, *blend_1* and *blend_2*, with constant and opposite jerk at its limits. Using this approach, one can show that the blend phases lie symmetrically around the waypoints w.r.t. time. That is, after half of the blend phases, the quaternion curve is closest to the waypoints, but does not reach them.

Alg. 1 computes the containers that completely define a *SCB* trajectory: $\mathcal{Q}$, $\mathcal{T}_{\text{bStart}}$, $\mathcal{H}_b$, Ω_{lin} and $\ddot{\Omega}_b$. Note that the quaternion waypoints in $\mathcal{Q}$ are already preprocessed such

Algorithm 1 SCB Trajectory Generation

Input: $\mathcal{Q}$, $\omega^{\max}$, $\dot{\omega}^{\max}$ and $\ddot{\omega}^{\max}$
Output: $\mathcal{T}_{\text{bStart}}$, $\mathcal{H}_b$, Ω_{lin}, $\ddot{\Omega}_b$

1: **for** $i \in [1, n-1]$ **do**
2: $\quad {}_i\mathbf{q}_{i+1} = {}_0\mathbf{q}_i^{-1} \otimes {}_0\mathbf{q}_{i+1}$
3: $\quad \Theta_i = \text{abs}(2 \cdot \text{acos}({}_ia_{i+1}))$
4: $\quad {}_i\mathbf{e}_i = \text{sign}(2 \cdot \text{acos}({}_ia_{i+1})) \frac{{}_i\boldsymbol{v}_{i+1}}{\|{}_i\boldsymbol{v}_{i+1}\|}$
5: $\quad {}_0\bar{\mathbf{e}}_i = {}_0\mathbf{q}_i \otimes {}_i\bar{\mathbf{e}}_i \otimes {}_0\mathbf{q}_i^{-1}$
6: $\quad h_{\text{lin},i} = \frac{\Theta_i}{\omega^{\max}}$
7: $\quad {}_0\boldsymbol{\omega}_{\text{lin},i} = {}_0\mathbf{e}_i \cdot \omega^{\max}$
8: **end for**
9: **for** $i \in [1, n]$ **do**
10: $\quad h_{b,i} = 2 \cdot \sqrt{\frac{\|{}_0\boldsymbol{\omega}_{\text{lin},i+1} - {}_0\boldsymbol{\omega}_{\text{lin},i}\|}{\ddot{\omega}^{\max}}}$
11: **end for**
12: **while** BLENDPHASESOVERLAP() **or** MAXACCLIMIT() **do**
13: $\quad$ **for** $i \in [1, n-1]$ **do**
14: $\quad\quad$ **if** BLENDPHASESOVERLAP() **then**
15: $\quad\quad\quad h_{\text{lin},i} \leftarrow 1.05 \cdot h_{\text{lin},i}$
16: $\quad\quad\quad {}_0\boldsymbol{\omega}_{\text{lin},i} = {}_0\mathbf{e}_i \cdot \frac{\Theta_i}{h_{\text{lin},i}}$
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad$ **for** $i \in [1, n]$ **do**
20: $\quad\quad h_{b,i} = 2 \cdot \sqrt{\frac{\|{}_0\boldsymbol{\omega}_{\text{lin},i+1} - {}_0\boldsymbol{\omega}_{\text{lin},i}\|}{\ddot{\omega}^{\max}}}$
21: $\quad\quad {}_0\ddot{\boldsymbol{\omega}}_{b,i} = 4 \cdot \frac{{}_0\boldsymbol{\omega}_{\text{lin},i+1} - {}_0\boldsymbol{\omega}_{\text{lin},i}}{h_{b,i}^2}$
22: $\quad$ **end for**
23: $\quad$ **if** MAXACCLIMIT() **then**
24: $\quad\quad \ddot{\omega}^{\max} \leftarrow 0.95 \cdot \ddot{\omega}^{\max}$
25: $\quad$ **end if**
26: **end while**
27: $\mathcal{T}_{\text{bStart}} \leftarrow$ COMPUTEBLENDSTARTTIMES($\mathcal{H}_{\text{lin}}$, $\mathcal{H}_b$)

that the antipodal problem is resolved and ${}_i\mathbf{q}_{i+1}$ defines the shortest of the two possible rotation directions. $\mathcal{T}_{\text{bStart}} = \{t_{\text{bStart},i} \mid i \in [1, n]\}$ and $\mathcal{H}_b = \{h_{b,i} = h_i^{blend\_1} + h_i^{blend\_2} \mid i \in [1, n]\}$ contain the start times of the blend phases and their duration, respectively. $\Omega_{\text{lin}} = \{\boldsymbol{\omega}_{\text{lin},i} \mid i \in [1, n-1]\}$ and $\ddot{\Omega}_b = \{\ddot{\boldsymbol{\omega}}_{b,i} \mid i \in [1, n]\}$ are the constant angular velocities and jerks in the linear phases and blend phases, respectively. $\mathcal{H}_{\text{lin}}$ are the linear phase duration. $\otimes$ denotes the quaternion product and we divide a quaternion into its real and imaginary part: ${}_0\mathbf{q}_i = [a_i, x_i, y_i, z_i] = [a_i, \mathbf{v}_i^T]$. A quaternion $\mathbf{q}$ formulates a rotation w.r.t. a specific frame B_i and we denote this frame with a left subscript, e.g. ${}_i\mathbf{q}$. Note that we plan all motions w.r.t. the robot's base frame B_0.

Lines 3-5 in Alg. 1 compute the rotation axes ${}_0\mathbf{e}_i$ and angles Θ_i for the linear motion segments in between the waypoints. We define Θ_i to be positive and select ${}_0\mathbf{e}_i$ accordingly. In the coordinate transformation in line 5, ${}_i\bar{\mathbf{e}}_i$ is a quaternion with ${}_i\mathbf{e}_i$ as imaginary part and 0 as real part. For the computation of the blend duration in line 10, we add the zero-clamped boundary condition, i.e. $\boldsymbol{\omega}_{\text{lin},0} = \boldsymbol{\omega}_{\text{lin},n} = \mathbf{0}$. Line 12-26 is our iterative optimization procedure: the robot moves with $\omega^{\max}$ and $\ddot{\omega}^{\max}$. In case two blend phases overlap, the linear phase in between is extended (line 15), which is equivalent to a decrease of $\omega^{\max}$. Thereby, BLENDPHASESOVERLAP checks $0.5 \cdot (h_{b,i} + h_{b,i+1}) < h_{lin,i}$ for each segment. In case $\dot{\omega}^{\max}$ is violated during blending, $\ddot{\omega}^{\max}$ is reduced in line 24. MAXACCLIMIT computes the maximum absolute angular acceleration $\|0.5 \cdot {}_0\ddot{\boldsymbol{\omega}}_{b,i} \cdot h_{b,i}\|$ in each blend phase. Based on $\mathcal{H}_{\text{lin}}$ and $\mathcal{H}_b$, COMPUTEBLENDSTARTTIMES computes the starting times of the blend duration $\mathcal{T}_{\text{bStart}}$. To not only blend the first and last waypoint in $\mathcal{Q}$, but to hit them, COMPUTEBLENDSTARTTIMES adds half of the first and last blend phases to the corresponding linear phases: $h_{\text{lin},1} \leftarrow h_{\text{lin},1} + 0.5 \cdot h_{b,1}$; $h_{\text{lin},n-1} \leftarrow h_{\text{lin},n-1} + 0.5 \cdot h_{b,n}$.

The new ${}_0\boldsymbol{\omega}_{\text{lin},i}$ and ${}_0\ddot{\boldsymbol{\omega}}_{b,i}$ are computed in line 16 and 21 such that linear acceleration profiles result. The computed trajectory respects $\omega^{\max}$, $\dot{\omega}^{\max}$ and $\ddot{\omega}^{\max}$ at any time and Fig. 2 shows the resulting trajectory profiles for *SCB* for a motion with six waypoints.

While Alg. 1 shows the offline computation of the trajectory, Alg. 2 defines the online control function. Given an arbitrary time t_k, Alg. 2 computes the reference quaternion $\boldsymbol{q}_k$, angular velocity $\boldsymbol{\omega}_k$ and angular acceleration $\dot{\boldsymbol{\omega}}_k$. To obtain the quaternion profile, we integrate the angular velocity:

$$\boldsymbol{q}(t) = \exp\left(\frac{1}{2}\int_0^t \boldsymbol{\omega}(\tau)\ \mathrm{d}\tau\right) \otimes \boldsymbol{q}_0 \tag{1}$$

Note that $\exp(\dots)$ in (1) forms a quaternion similar to ${}_i\bar{\mathbf{e}}_i$ in Alg. 1. The integration in (1) only holds for interpolations with a constant rotation axis [21], which is not the case for the blend phases. However, as the approximation error is negligible for a limited number of waypoints, we favor the analytic integration in (1) over a numeric integration. For the experimental application scenario with six waypoints in Section IV (see Fig. 3), we did a comparison of our approximated analytic integration, line 6 and 11 in Alg. 2, with a numeric integration of steps size 1.0 ms, which corresponds to the real-time control cycle of our robot: During the trajectory execution time of 2.53 s, the rotational error increased over the waypoints and was biggest at the last waypoint with 0.02 rad. For our application, in which we only try to blend the waypoints, we consider this approximation error acceptable. For applications which demand higher accuracy at the waypoints, blending might not be an option anyhow. For applications with a lot more motion segments, the integration error might accumulate too much and we suggest numeric integration in these cases. Fig. 2 shows that *SCB* with (1) blends the defined waypoints in $\mathcal{Q}$ with the same accuracy as *SPB*.

Line 1 in Alg. 2 determines the motion segment that t_k is part of. We derived the boundary conditions for the three segment types - *blend_1*, *blend_2* and *linear* phase - and provide the interpolation formula in Alg. 2.

Fig. 2 shows the constant $\ddot{\boldsymbol{\omega}}$ profile during the blend phases that leads to linear profiles of $\dot{\boldsymbol{\omega}}$ and quadratic profiles of $\boldsymbol{\omega}$. Note that in contrast to *SPB*, the $\boldsymbol{\omega}$ profile of *SCB* does not show undesired bumps during blending (see Section III-B.2).

Algorithm 2 SCB Online Control

Input: t_k, $\mathcal{Q}$, $\mathcal{T}_{\text{bStart}}$, $\mathcal{H}_b$, Ω_{lin}, $\ddot{\Omega}_b$
Output: $\boldsymbol{q}_k$, $\boldsymbol{\omega}_k$, $\dot{\boldsymbol{\omega}}_k$

segment = determineSegmentType(t_k)
if segment is *blend_1* **then**
  $\Delta t = t_k - t_{\text{bStart},i}$
  ${}_0\dot{\boldsymbol{\omega}}_k = {}_0\ddot{\boldsymbol{\omega}}_{b,i}\Delta t$
  ${}_0\boldsymbol{\omega}_k = {}_0\boldsymbol{\omega}_{\text{lin},i} + {}_0\ddot{\boldsymbol{\omega}}_{b,i}\frac{\Delta t^2}{2}$
  ${}_0\boldsymbol{q}_k = \exp\left({}_0\boldsymbol{\omega}_{\text{lin},i}\left(-\frac{h_{b,i}}{4} + \frac{\Delta t}{2}\right) + {}_0\ddot{\boldsymbol{\omega}}_{b,i}\frac{\Delta t^3}{12}\right) \otimes {}_0\mathbf{q}_i$
else if segment is *blend_2* **then**
  $\Delta t = t_k - t_{\text{bStart},i} - \frac{1}{2}h_{b,i}$
  ${}_0\dot{\boldsymbol{\omega}}_k = {}_0\ddot{\boldsymbol{\omega}}_{b,i}\left(\frac{h_{b,i}}{2} - \Delta t\right)$
  ${}_0\boldsymbol{\omega}_k = {}_0\boldsymbol{\omega}_{\text{lin},i} + {}_0\ddot{\boldsymbol{\omega}}_{b,i}\left(\frac{h_{b,i}^2}{8} + \frac{h_{b,i}\Delta t}{2} - \frac{\Delta t^2}{2}\right)$
  ${}_0\boldsymbol{q}_k = \exp\Big({}_0\boldsymbol{\omega}_{\text{lin},i}\frac{\Delta t}{2} + \ \ldots$
    $\ldots\ {}_0\ddot{\boldsymbol{\omega}}_{b,i}\left(\frac{h_{b,i}^3}{96} + \frac{h_{b,i}^2\Delta t}{16} + \frac{h_{b,i}\Delta t^2}{8} - \frac{\Delta t^3}{12}\right)\Big) \otimes {}_0\mathbf{q}_i$
else
  $\Delta t = t_k - t_{\text{bStart},i} - \frac{1}{2}h_{b,i}$
  ${}_0\dot{\boldsymbol{\omega}}_k = \mathbf{0}$
  ${}_0\boldsymbol{\omega}_k = {}_0\boldsymbol{\omega}_{\text{lin},i}$
  ${}_0\boldsymbol{q}_k = \exp\left({}_0\boldsymbol{\omega}_{\text{lin},i}\frac{\Delta t}{2}\right) \otimes {}_0\mathbf{q}_i$
end if

Depending on the time parameterization, a great proportion of the the motion takes place around a constant rotation axis, similar to *SPB*.

B. C^2-Continuous Zero-Clamped Trajectories with State-of-the-Art Interpolation Schemes

In this section, we contribute modified versions of *SQUAD* and *SPB* that we use in Section IV to benchmark *SCB*. We refer to [5] and [8] for detailed derivations. However, their versions lack smoothness, i.e. $\mathcal{C}^2$-continuity, and the *SQUAD* derivation in [5] further lacks zero-clamped boundaries. We propose the required modifications to resolve these drawbacks as they have not been presented yet.

1) SQUAD: It is based on *SLERP* and, given n waypoints $\mathbf{q}_i$ and $n-1$ segment duration h_i, *SQUAD* uses (2) for interpolation. In related work, $u(t)$ is a linearly increasing parameter with $u(t_0) = 0$ and $u(T) = 1$. In contrast to *SLERP*, the rotation axis is not constant but smoothly changes in between the waypoints (see Fig. 2).

$$SQUAD(\boldsymbol{q}_i, \boldsymbol{s}_i, \boldsymbol{s}_{i+1}, \boldsymbol{q}_{i+1}, u)) = \ldots$$
$$SLERP(\hat{\boldsymbol{q}}_i, \hat{\boldsymbol{s}}_i, 2u(1-u)) = \hat{\boldsymbol{q}}_i \otimes (\hat{\boldsymbol{q}}_i^{-1} \otimes \hat{\boldsymbol{s}}_i)^{2u(1-u)} \tag{2}$$

$$\hat{\boldsymbol{q}}_i = SLERP(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}, u) \tag{3}$$

$$\hat{\boldsymbol{s}}_i = SLERP(\boldsymbol{s}_i, \boldsymbol{s}_{i+1}, u) \tag{4}$$

$$\boldsymbol{s}_i = \boldsymbol{q}_i \otimes \exp\left(\frac{\log(\boldsymbol{q}_i^{-1} \otimes \boldsymbol{q}_{i+1})}{-2(1 + \frac{h_i}{h_{i-1}})} + \frac{\log(\boldsymbol{q}_i^{-1} \otimes \boldsymbol{q}_{i-1})}{-2(1 + \frac{h_{i-1}}{h_i})}\right) \tag{5}$$

Our contributed modifications to *SQUAD* are the following and we refer to this $\mathcal{C}^2$-continuous zero-clamped *SQUAD* implementation as *SQUAD_C2*:

- To ensure a continuous change of the rotation axis, *SQUAD* adds two inner control quaternions in each motion segment, $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+1}$, that are calculated such that $\mathcal{C}^1$-continuity is achieved. However, the formula for $\boldsymbol{s}_i$ in [5] ignores the time parameterization of the interpolation parameter and ensures $\mathcal{C}^1$-continuity only in case all motion segment duration h_i are equal. With (5), we provide the correct formula that respects different h_i. This allows to optimize the trajectory duration by using different h_i to exploit $\omega^{\max}$, $\dot{\omega}^{\max}$ and $\ddot{\omega}^{\max}$ in each motion segment.
- The approach in [5] does not ensure zero-clamped boundaries. We introduce two additional virtual waypoints $\mathbf{q}_1^{virt}$ and $\mathbf{q}_{n-1}^{virt}$ after the first and second-last waypoint: $\mathcal{Q} = [\mathbf{q}_1, \mathbf{q}_1^{virt}, \mathbf{q}_2, \ldots, \mathbf{q}_{n-1}^{virt}, \mathbf{q}_n]$. Setting the two virtual waypoints $\mathbf{q}_1^{virt} = \mathbf{q}_1$ and $\mathbf{q}_{n-1}^{virt} = \mathbf{q}_n$ ensures zero-clamped boundary constraints on velocity level.
- The two modifications above enable *SQUAD* to be $\mathcal{C}^1$-continuous with zero-clamped boundaries on velocity level. To eliminate the discontinuities in $\dot{\boldsymbol{\omega}}(t)$, we use a piecewise quintic polynomial interpolation for $u(t)$ instead of the linear interpolation. This allows $\ddot{u}(t) = 0$ to be set at the waypoints, which removes the discontinuities in the angular acceleration profile and defines two boundary conditions for the quintic polynomials that are defined by six unknowns. Reaching the waypoints and setting $\dot{u}(t) = 0$ at the waypoints defines the other four unknowns. Note that using cubic instead of quinitic polynomials with boundary conditions only for $\dot{u}(t)$ is not sufficient for $\mathcal{C}^2$-continuous rotational trajectory profiles due to the chain rule when deriving $\dot{\boldsymbol{\omega}}(t)$ and $\ddot{\boldsymbol{\omega}}(t)$. A fifth order polynomial interpolation is thus the minimal interpolation order for $u(t)$ that results in $\mathcal{C}^2$-continuity.

Fig. 2 shows the rotation axis and the trajectory profiles for *SQUAD* and *SQUAD_C2*. Note that *SQUAD* already includes our first two modifications to achieve $\mathcal{C}^1$-continuity and zero-clamped boundaries on velocity level. Adding the two virtual waypoints leads to accelerating and decelerating motion phases around a constant rotation axes at the first and last waypoint with zero-clamped angular velocities. Note that the discontinuous change of the rotation axes for *SQUAD* and *SQUAD_C2* in these two motion phases is due to a change of the rotation direction at zero angular velocity and thus does not result in a discontinuity. Apart from these two motion phases, *SQUAD* and *SQUAD_C2* smoothly change the rotation axis over the entire motion segments and thus do not provide segments with constant rotation axis, which is in contrast to *SCB* and *SPB*. However, as opposed to *SCB* and *SPB*, *SQUAD* and *SQUAD_C2* pass through the defined waypoints. In contrast to *SQUAD*, *SQUAD_C2* computes $\mathcal{C}^2$-continuous trajectories with zero acceleration at start and goal. The increased smoothness comes with higher oscillating $\boldsymbol{\omega}$ profiles and further, stops at the waypoints, i.e.

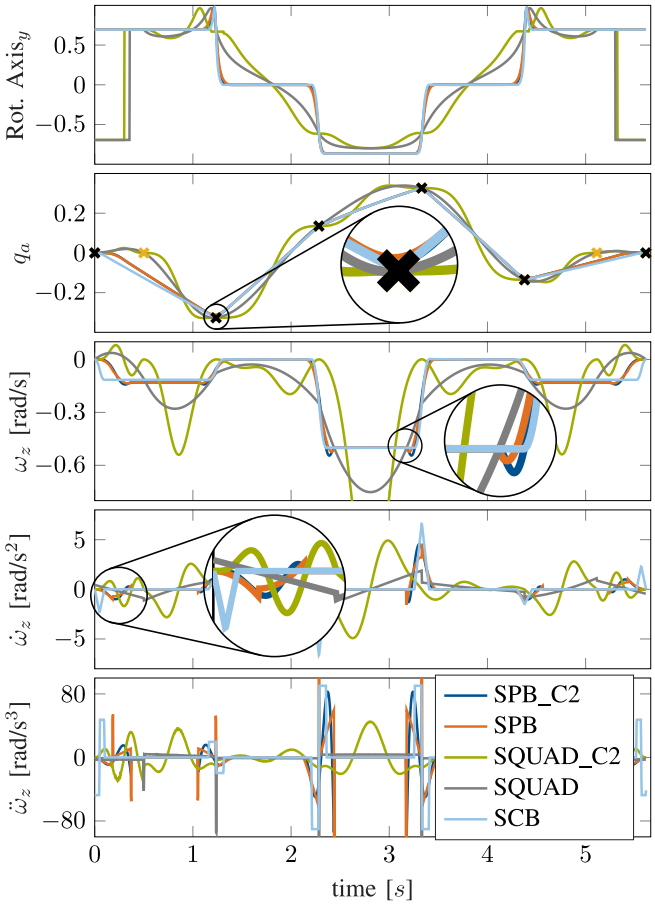

Fig. 2. Simulated trajectory profiles for the rotation axis, $\mathbf{q}$, $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$ and $\ddot{\boldsymbol{\omega}}$ for a path with six waypoints. For better readability, we only show the quaternions' real part and only the y- and z-component for the rotation axis and time derivatives, respectively. Black crosses in the quaternion profile mark the defined waypoints and yellow crosses mark the virtual waypoints of *SQUAD* and *SQUAD_C2*. The same segment duration h_i in between the waypoints are defined for all methods. Note that the blending phases for *SCB*, *SPB* and *SPB_C2* in the $\mathbf{q}(t)$ profile are short and thus their profiles are quite close to the intermediate waypoints, i.e. the black crosses, however, on a closer look, they do not reach them.

$\boldsymbol{\omega} = \mathbf{0}$ and $\dot{\boldsymbol{\omega}} = \mathbf{0}$.

2) SPB: It focuses on applications that demand constant rotation axes and is proposed in [8]. *SPB* modifies *SQUAD* and uses quadratic blending for $u(t)$. This ensures a quick and smooth change of the rotation axis and zero-clamped boundaries. However, the trajectory profiles are only $\mathcal{C}^1$-continuous. Further, $\boldsymbol{\omega}$ shows undesired bumps during blending (see Fig. 2). To obtain $\mathcal{C}^2$-continuous trajectory profiles, we use quintic blending of $u(t)$ instead of quadratic blending. Similar to *SQUAD_C2*, a quintic interpolation of $u(t)$ allows $\ddot{u}(t) = 0$ to be set at the bounds. Fig. 2 compares *SPB* and our modified version that we refer to as *SPB_C2*. Both smoothly change the rotation axis and have zero-clamped boundaries up to acceleration level. *SPB* and *SPB_C2* show a similar change of rotation axis and similar quaternion profiles. However, in contrast to *SPB*, *SPB_C2* is $\mathcal{C}^2$-continuous. Both methods suffer from the bumps within the $\boldsymbol{\omega}$ profile during blending.

Unlike *SQUAD* and *SPB*, where discontinuities in acceleration cause infinite jerk values as seen in Fig. 2, *SQUAD_C2* and *SPB_C2* avoid this problem due to their increased degree of continuity.

IV. Results

Section III compared the qualitative trajectory profiles of *SCB*, *SPB_C2* and *SQUAD_C2*. This section applies the three methods in experiments on a *FRANKA EMIKA* robot [22] in an application that targets time-optimized motions. Therefore, we integrated the proposed methods in our C++ planning framework. *SCB* inherently computes fast, $\mathcal{C}^2$-continuous and zero-clamped motions by exploiting $\omega^{\max}$, $\dot{\omega}^{\max}$ and $\ddot{\omega}^{\max}$. To obtain time-optimized motions with *SQUAD_C2* or *SPB_C2*, the nonlinear optimization given in (6) needs to be solved. h_i are the segment duration in between the waypoints.

$$\min_{\mathbf{h}} \quad T(\mathbf{h}) = \sum_{i=1}^{N-1} h_i \tag{6}$$

$$\|\boldsymbol{\omega}(t_i)\| \leq \omega^{\max} \; ; \; \|\dot{\boldsymbol{\omega}}(t_i)\| \leq \dot{\omega}^{\max} \; ; \; \|\ddot{\boldsymbol{\omega}}(t_i)\| \leq \ddot{\omega}^{\max} \tag{7}$$

As stated in Section III, *SQUAD_C2* and *SPB_C2* interpolate on orientation level. Due to the nonlinearity in $SO(3)$, there are no analytical expressions for the extreme values in $\boldsymbol{\omega}(t)$, $\dot{\boldsymbol{\omega}}(t)$ and $\ddot{\boldsymbol{\omega}}(t)$. Thus, to respect the actuator limits in (7), the trajectory is discretized into time steps t_i that are checked for constraint violations. This is a drawback of *SQUAD_C2* and *SPB_C2* compared to *SCB*, as the solution of the discretized nonlinear optimization without analytical gradients is computationally more expensive than our iterative approach in Alg. 1.

The test scenario defines the same waypoints as in Fig. 2. To solve (6) without gradient information, we use *COBYLA*, a local derivative-free optimization algorithm implemented in the open-source library *NLopt* [23]. To provide a fair benchmark, we did a parameter study for *SQUAD_C2* and *SPB_C2* and set the maximum number of iterations to 150, which showed a good trade-off between optimality and computation time. The discretization for solving (7) was set to 0.025 s. We define the symmetrical actuator limits $\omega^{\max} = 2.0$ rad/s, $\dot{\omega}^{\max} = 20.0$ rad/s^2 and $\ddot{\omega}^{\max} = 5000.0$ rad/s^3. For the initial guess, we define constant velocity motions between the waypoints with 50 % of $\omega^{\max}$. For *SPB_C2*, the initial blend phases are set to 30 % of the linear phases. For *SQUAD_C2*, the two additional waypoints are set 0.5 s after and before the first and last waypoint respectively. We use the open-source library *broccoli* [24] to implement spline interpolations and quaternion algebra. The computation runs on a 64-bit Ubuntu 18.04 operated computer with 32 GB RAM and with a 12-core *Intel i7-7800K* CPU running at 3.7 GHz.

Fig. 3 shows the measured trajectory profiles and Table I compares the interpolation methods w.r.t. computation time T_{comp}, trajectory duration T, qualitative smoothness based

TABLE I

BENCHMARK OF QUATERNION INTERPOLATION METHODS

	T_{comp}^{opt} [ms]	T_{comp}^{traj} [µs]	T [s]	Smooth	Hit
SPB_C2	80.2 ± 2.4	8.1 ± 6.1	2.94	•	no
SQUAD_C2	83.5 ± 1.1	30.0 ± 12.4	6.47	• • •	yes
SCB	7.9 ± 2.2	94.5 ± 33.2	2.59	• •	no

on the profiles and if the defined waypoints are hit. Although the problem formulation and the solver are deterministic, the computation time on the computer hardware is not due to other programs executing, operating system settings etc. Therefore, the computation times are averaged over ten runs and the mean value and the standard deviation are given. To consider the decoupled optimization and trajectory computation of *SQUAD_C2* and *SPB_C2*, we distinguish between T_{comp}^{opt} and T_{comp}^{traj}. The video of the experiment is given as supplementary material and available at: https://youtu.be/x2iKmf4R28M. It shows the robot motions that correspond to the profiles in Fig. 3. We use quintic spline interpolation for the translational part of the trajectory, which is a planar square with a side length of 0.02 m, and we use *RMRC* for the differential IK.

All three methods compute $\mathcal{C}^2$-continuous profiles that can be tracked accurately by the robot, which is shown by the good fit of continuous lines and dashed lines in Fig. 3. Minor drawbacks w.r.t. the tracking accuracy of $\boldsymbol{\omega}$ are visible for *SPB_C2* due to the bumps in the profile. In contrast to *SCB* and *SPB_C2*, *SQUAD_C2* hits the defined waypoints and provides the highest smoothness. However, the lack of constant velocity phases leads to the longest overall trajectory duration T. This is due to $\omega^{\max}$ being only exploited for certain segments at a single time point. *SCB* and *SPB_C2* both exploit $\omega^{\max}$ and $\dot{\omega}^{\max}$ in each of the motion segments. However, the bumps within the $\boldsymbol{\omega}$ profile of *SPB_C2* lead to a less smooth profile and further prevent the exploitation of $\omega^{\max}$ throughout the whole linear phase. Therefore, T is longer for *SPB_C2* compared to *SCB*. *SCB* shows a good trade-off between time optimization and a smooth trajectory profile. Note that for the defined scenario, there were no overlaps during the blend phases and thus $\omega^{\max} = 2.0$ rad/s was not modified during the optimization iterations (line 15 in Alg. 1). However, the acceleration limits were violated during the iterations, and thus $\ddot{\omega}^{\max}$ was scaled from originally 5000.0 rad/s^3 to 101.38 rad/s^3 according to line 24 in Alg. 1. The time optimization for *SPB_C2* and *SQUAD_C2* dominates the total computation time and is magnitudes higher than for the iterative approach of *SCB*.

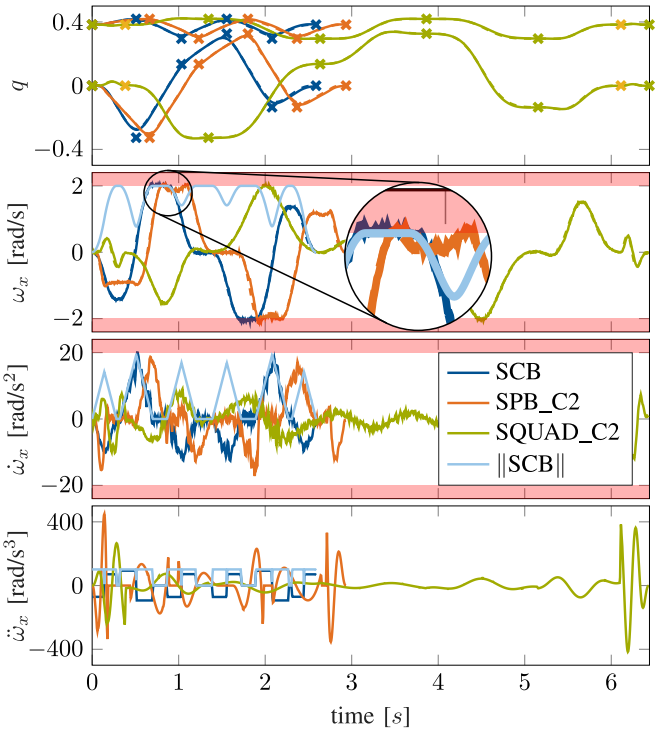

Fig. 3. Measured trajectory profiles for $\mathbf{q}$, $\boldsymbol{\omega}$, $\dot{\boldsymbol{\omega}}$ and $\ddot{\boldsymbol{\omega}}$ for a path with six waypoints. For better readability, we only show the quaternions' a- and y-components and only the x-component for the time derivatives. Black crosses in the quaternion profile mark the defined waypoints and yellow crosses mark the virtual waypoints of *SQUAD_C2*. The red patches visualize the symmetrically defined actuator limits. The light blue line indicates the computed absolute values $\|\boldsymbol{\omega}\|$, $\|\dot{\boldsymbol{\omega}}\|$, $\|\ddot{\boldsymbol{\omega}}\|$ for *SCB*. Within the $\mathbf{q}$ and $\boldsymbol{\omega}$ profiles, the dashed lines indicate the desired trajectory profile. The continuous lines show the profile executed by the robot. The acceleration profile is computed with the filtered velocity profile. The jerk profile is plotted with the computed reference values to avoid a second derivation of the noisy velocity measurements.

V. CONCLUSIONS

We propose *SCB*, a computationally efficient trajectory generation algorithm for the time-optimized interpolation of quaternions. Additionally, we contribute modifications to the state-of-the-art methods *SQUAD* and *SPB* such that their trajectory profiles are $\mathcal{C}^2$-continuous and zero-clamped, yielding *SQUAD_C2* and *SPB_C2*. We benchmark *SCB* with *SQUAD_C2* and *SPB_C2*: *SCB* achieves time optimization with lower computational cost and further computes smoother trajectory profiles than *SPB_C2*. *SQUAD* and *SPB* interpolate on orientation level and thus, due to the nonlinearity of $SO(3)$, compute less smooth profiles. For both, an additional quintic time interpolation for the underlying control parameter needs to be implemented to achieve $\mathcal{C}^2$-continuity and zero-clamped boundaries. Further, analytical gradient information cannot be derived, which prevents efficient analytical gradient-based time optimization. In contrast to other approaches, *SCB* can also be applied to translations and the modifications are straightforward. *SCB* cannot be applied to applications in which the end effector needs to pass through given waypoints as they are only blended, not hit. Further, *SCB*, as presented here, is not applicable to applications with translational motion only, i.e. when two adjacent quaternions are the same as Alg. 1 would fail. However, catching this case and adding a motion segment with no rotation would be straightforward.

References

[1] C. Lin, P. Chang, and J. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1074, 1983.

[2] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.

[3] F. S. Grassia, "Practical parameterization of rotations using the exponential map," *Journal of graphics tools*, vol. 3, no. 3, 1998.

[4] (2019) geometry_msgs pose message. Accessed: 2022-02-21. [Online]. Available: http://docs.ros.org/en/lunar/api/geometry_msgs/html/msg/Pose.html

[5] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, interpolation and animation*. Citeseer, 1998, vol. 2.

[6] M.-J. Kim, M.-S. Kim, and S. Y. Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 369–376.

[7] K. Shoemake, "Animating rotation with quaternion curves," in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985, pp. 245–254.

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

[9] FRANKA EMIKA GmbH. (2017) Robot and interface specifications. Accessed: 2022-08-21. [Online]. Available: https://frankaemika.github.io/docs/control_parameters.html#constants

[10] D. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man Machine Systems*, vol. 10, no. 2, pp. 47–53, 1969.

[11] W. Khalil and E. Dombre, *Modeling identification and control of robots*. CRC Press, 2002.

[12] M. Shahbazi, N. Kashiri, D. Caldwell, and N. Tsagarakis, "On the orientation planning with constrained angular velocity and acceleration at endpoints," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7033–7038.

[13] M. Neubauer *et al.*, "Smooth orientation path planning with quaternions using b-splines," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 2087–2092.

[14] J. Lin, M. Rickert, and A. Knoll, "Parameterizable and jerk-limited trajectories with blending for robot motion planning and spherical cartesian waypoints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 982–13 988.

[15] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 3, pp. 415–426, 2008.

[16] ——, "Optimal trajectory planning for industrial robots," *Advances in Engineering Software*, vol. 41, no. 4, pp. 548–556, 2010.

[17] J. Wittmann and D. J. Rixen, "Time-optimization of trajectories using zero-clamped cubic splines and their analytical gradients," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4528–4534, 2022.

[18] F. O. Kuehnel, "On the minimization over so (3) manifolds," Citeseer, Tech. Rep., 2003.

[19] R. M. Grassmann *et al.*, "Quaternion-based smooth trajectory generator for via poses in se(3) considering kinematic limits in cartesian space," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4192–4199, 2019.

[20] T. Kunz and M. Stilman, "Turning paths into trajectories using parabolic blends," Georgia Institute of Technology, Tech. Rep., 2011.

[21] M. Boyle, "The integration of angular velocity," *Advances in Applied Clifford Algebras*, vol. 27, no. 3, p. 2345–2374, May 2017. [Online]. Available: http://dx.doi.org/10.1007/s00006-017-0793-z

[22] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.

[23] S. G. Johnson, "The nlopt nonlinear-optimization package," https://github.com/stevengj/nlopt, 2021.

[24] P. Seiwald and F. Sygulla. (2022) broccoli: Beautiful Robot C++ Code Library. [Online]. Available: https://gitlab.lrz.de/AM/broccoli