Parameterizable and Jerk-Limited Trajectories with Blending for Robot Motion Planning and Spherical Cartesian Waypoints

Jianjie Lin, Markus Rickert, and Alois Knoll

***Abstract*— This paper presents two different approaches to generate a time local-optimal and jerk-limited trajectory with blends for a robot manipulator under consideration of kinematic constraints. The first approach generates a trajectory with blends based on the trapezoidal acceleration model by formulating the problem as a nonlinear constraint and a non-convex optimization problem. The resultant trajectory is locally optimal and approximates straight-line movement while satisfying the robot manipulator's constraints. We apply the bridged optimization strategy to reduce the computational complexity, which borrows an idea from model predictive control by dividing all waypoints into consecutive batches with an overlap of multiple waypoints. We successively optimize each batch. The second approach is a combination of a trapezoidal acceleration model with a 7-degree polynomial to form a path with blends. It can be efficiently computed given the specified blending parameters. The same approach is extended to Cartesian space. Furthermore, a quaternion interpolation with a high degree polynomial under consideration of angular kinematics is introduced. Multiple practical scenarios and trajectories are tested and evaluated against other state-of-the-art approaches.**

I. Introduction

Trajectory generation is a fundamental topic in the robotics community that deals with the calculation of a time-optimal, smooth, jerk-limited, and accurate motion for a well-defined task. Manually programming and optimizing paths for complex robot systems is no longer viable when it comes to flexible production with small lot sizes and multiple robot manipulators, a common use case in small and medium-sized enterprises [1]. In order to quickly adapt to new processes, new paths have to be generated automatically by modern path planning algorithms that are able to calculate complex motions for multiple manipulators in a narrow space. As cycle times should be as short as possible, globally optimal path planning algorithms [2] present a considerable advantage over classical algorithms that are followed by a local optimization step. In order to avoid stopping at every waypoint in a path, supporting various forms of blending is a desired property in a trajectory generation algorithm to further increase the performance of a robot system. Kinodynamic path planning algorithms with velocity information however are proven to be PSPACE hard [3] and therefore lead to a large increase in computation time. Industrial robot controllers and open-source implementations commonly support blending via linear parabolic motions and cubic spline interpolation. Trajectories based on linear parabolic blending, that only limit the acceleration, suffer from infinite jerk around the blend waypoints [4]. Although cubic spline interpolation can improve the smoothness of a path by limiting the jerk, it can result in a more significant deviation of the straight-line movement. This is especially important when calculating trajectories for position-based solutions in robot path planning. Trajectory generation without explicit error bounds in the path deviation can lead to undesired behavior. Deviating too far from the collision-free solution path can result in collisions. Based on these observations, we present two different approaches to generate a trajectory for following multiple waypoints. They support an explicit upper bound in deviation and are jerk-limited around the blended waypoints. In our previous work [4], we consider the situation of performing an accurate motion for a robot manipulator by forcing the trajectory to precisely pass through all waypoints, which are either manually specified or generated via a path planning algorithm. In this work, we extend this to a more general application by considering blending around the waypoints. In contrast to most state-of-the-art blending algorithms, the jerk limitation is followed throughout the trajectory. In the same way as Haschke et al. [5] and Kröger et al. [6], the trapezoidal acceleration profile is used to generate the trajectory between two consecutive waypoints. As stated in [4], the trapezoidal acceleration profile increases the optimization complexity while considering phase synchronization. In comparison to our previous work [4], we relaxed the objective function by introducing two additional weights to control the distribution of acceleration, deceleration, and cruising phases, which reduces the optimization complexity and shows a better performance from the perspective of straight-line movement. We continue to utilize the principle of model predictive control [4] for optimizing all waypoints by decomposing them into many consecutive waypoint batches and bridging each two adjacent batches with an overlapping waypoint. In addition to the optimization approach, we present another new approach that combines the trapezoidal acceleration model with a high-degree polynomial to perform a blending trajectory in joint and Cartesian space. Notably, quaternion interpolation is integrated and extended to a high degree polynomial, which considers the angular jerk and results in a smooth quaternion trajectory.

Jianjie Lin, Markus Rickert, Alois Knoll are with Robotics, Artificial Intelligence and Real-Time Systems, Department of Informatics, Technische Universität München, Munich, Germany jianjie.lin@tum.de {rickert,knoll}@in.tum.de

II. Related Work

Generating time-optimal and smooth trajectories has been studied extensively for decades in the robotics community.

The proposed trajectory planning techniques are roughly divided into two categories: online and offline planning.

Online real-time trajectory generation is mainly used to deal with unforeseen events and a dynamic and fast modification of the planned trajectory. Macfarlane et al. [7] proposed a jerk-bounded fifth-order polynomial with parabolic blends between two waypoints. Haschke et al. [5] presented an online trajectory planner by considering arbitrary initial kinematics and stopping at each waypoint. Kröger et al. [6] extended the online planner in a more general approach, which can handle arbitrary start and goal states. Lange et al. [8] proposed a path-accurate and jerk-limited online trajectory generation in configuration space. However, this cannot be easily extended to multiple waypoints and it is also not possible to blend the trajectory around the target.

Offline trajectory planning is suitable for a well-defined task, such as assembly or welding applications. The standard trajectory generator utilizes polynomials based on splines such as cubic splines or polynomials of higher degrees to provide a jerk-bound smooth trajectory. B-splines and their extension method [9] are also widely used to generate smooth trajectories. Although polynomial-based and B-spline-based algorithms can generate a smooth trajectory, they cannot fully explore the robot's capabilities and show a significant deviation from a straight line. Pham et al. [10] proposed a new approach based on reachability analysis for the time-optimal path parameterization (TOPP) problem. Similarly, Nagy and Vajk [11] applied a linear programming-based (LP) solver to tackle TOPP. Furthermore, Barnett et al. [12] introduced a bisection algorithm (BA) by extending the dynamic programming approaches to generate a trajectory. However, those algorithms are expensive to perform a trajectory with blends. Kunz et al. [13] proposed a path-following algorithm by adding circular blends that consider the acceleration bounds in joint space. Dantam et al. [14] presented spherical, parabolic blends by using the SLERP function, where no interpolation in a Cartesian pose is considered. These algorithms however do not take jerk limitation into account.

III. Problem Formulation

The goal is to find a time-optimal, jerk-limited, and smooth trajectory that blends an intermediate waypoint without violating kinematic constraints. Furthermore, it is required to minimize the deviation to a straight line in either joint or Cartesian space. The trapezoidal acceleration-based trajectory model [4], also called seven-segment model, has the capability to generate a smooth and jerk-limited trajectory. At segment $h \in [0, \cdots, 6]$, the kinematics are formulated as

$$\begin{aligned} a^k_{i,h+1}(t) &= a^k_{i,h} + j^k_{i,h}\Delta t_{i,h}\,, \\ v^k_{i,h+1}(t) &= v^k_{i,h} + a^k_{i,h}\Delta t_{i,h} + \frac{1}{2} j^k_{i,h}\Delta t^2_{i,h}\,, \\ p^k_{i,h+1}(t) &= p^k_{i,h} + v^k_{i,h}\Delta t_{i,h} + \frac{1}{2} a^k_{i,h}\Delta t^2_{i,h} + \frac{1}{6} j^k_{i,h}\Delta t^3_{i,h} \end{aligned} \tag{1}$$

at the waypoint i in the axis k. The parameter $\Delta t_{i,h}$ is the time difference, defined as $t_{i,h+1} - t_{i,h}$. For a phase synchronization [15] trajectory, position, velocity, acceleration, and jerk in each axis at the same segment should be synchronized. The time evolution of a position is interpreted by a third-order polynomial, which can increase the smoothness of trajectories by bounding the jerk. In this paper, we present two approaches: In the first approach, we extend our previous work, which enforces a precise pass through all waypoints, denoted as TrajOpt-Pass-Joint (TOPJ), to generate a blending trajectory by formulating it as a nonlinear constraint optimization problem, indicated as TrajOpt-Blend-Joint (TOBJ). In the second approach, we combine the trapezoidal acceleration model with a high-dimensional polynomial (7-degree) to generate a blending trajectory both in joint space TrajPoly-Blend-Joint (TPBJ) and Cartesian space TrajPoly-Blend-Cart (TPBC).

A. Blending by Optimization in Joint Space (TOBJ)

A blending trajectory is formulated as a nonlinear constraint optimization problem by applying a nonlinear optimization solver (SQP) [16]. The work presented in this paper introduces a newly designed objective function and additional inequality and equality constraints. We follow the same optimization strategy as introduced in [4].

1) Objective Function: The purpose of the objective function f is to find a trajectory that is optimal in time and moves as linearly as possible in joint space as

$$\begin{aligned} f = \sum_{i=1}^{i=n} \Bigg(\lambda_3 \Big(\big((\Delta t_{i,\mathrm{acc}} + \Delta t_{i,\mathrm{dec}})\lambda_1 \big)^2 + \big(\Delta t_{i,\mathrm{cruis}}\lambda_2 \big)^2 \Big) \\ + \lambda_4 \sum_{k=1}^{k=m} (v^k_{i,3})^2 \exp\Big(-\frac{\Delta t^2_{i,\mathrm{cruis}}}{2\sigma^2} \Big) \Bigg), \end{aligned} \tag{2}$$

where n is the number of waypoints, m are the degrees of freedom of a robot. The time interval of the acceleration phase is indicated as $\Delta t_{i,\mathrm{acc}}$, the cruising phase as $\Delta t_{i,\mathrm{cruis}}$, and the deceleration phase as $\Delta t_{i,\mathrm{dec}}$. The weight values λ_1 and λ_2 are used to control the distribution of the acceleration/deceleration and cruising phase. The choice of $\big((\Delta t_{i,\mathrm{acc}} + \Delta t_{i,\mathrm{dec}})\lambda_1 \big)^2 + (\Delta t_{i,\mathrm{cruis}}\lambda_2)^2$ has advantages over the formulation $\big((\Delta t_{i,\mathrm{acc}} + \Delta t_{i,\mathrm{dec}})\lambda_1 + \Delta t_{i,\mathrm{cruis}}\lambda_2 \big)^2$, which avoids the product of $(\Delta t_{i,\mathrm{acc}} + \Delta t_{i,\mathrm{dec}})\Delta t_{i,\mathrm{cruis}}$, so that acceleration/deceleration phase and cruising phase cannot affect each other. On top of this, λ_3 is used to minimize the whole trajectory time. In this formulation, λ_3 and λ_1/λ_2 conflict with each other. λ_1/λ_2 is used to achieve a straight-line motion, while minimizing the time with λ_3 requires a longer acceleration/deceleration phase that can lead to an overshooting trajectory. In [4], the straight-line deviation bound is added in the objective function. In this work, we relax this constraint by emphasizing a straight-line in joint space. Furthermore, the overshooting is observed with

$$\Delta t_{i,\mathrm{cruis}} = \big((p^k_{i,\mathrm{target}} - p^k_{i-1,0}) - \big(\Delta p^k_{i,\mathrm{acc}} + \Delta p^k_{i,\mathrm{dec}} \big) \big) / v^k_{i,3}\,, \tag{3}$$

where $\Delta p^k_{i,\mathrm{acc}} = \sum_{h=0}^{h=2} p^k_{i,h}(t_{i,h}, v^k_{i,0}, a^k_{i,0})$ and $\Delta p^k_{i,\mathrm{dec}} = \sum_{h=4}^{h=7} p^k_{i,h}(t_{i,h}, v^k_{i,4}, a^k_{i,4}, v^k_{i,7}, a^k_{i,7})$. If the time $\Delta t_{i,\mathrm{cruis}}$ is neg-

ative, the reached position overshoots the target. To eliminate this undesired behavior, the cruising phase should be omitted. Since $\Delta p^k_{i,\text{acc}} + \Delta p^k_{i,\text{dec}}$ has a longer distance than $p^k_{i,\text{target}} - p^k_{i-1,0}$, we can reduce the cruising velocity $v^k_{i,3}$ to shorten $\Delta p^k_{i,\text{acc}}$. Besides, the trapezoidal end velocity $v_{i,7}$ influences the position $\Delta p^k_{i,\text{dec}}$, which can be automatically tuned by the optimization solver. We add a regular term $(v^k_{i,3})^2 \exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ in the objective function with σ, which controls the decay rate. It can be shown that at $\Delta t_{i,\text{cruis}} \approx 0$, the regular term $(v^k_{i,3})^2 \exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ is approximated as $(v^k_{i,3})^2$, and the velocity is reduced by minimizing the objective function. In the case of $\Delta t_{i,\text{cruis}} > 0$, the Gaussian value $\exp(-\frac{\Delta t^2_{i,\text{cruis}}}{2\sigma^2})$ is exponentially decayed to zero, which has no effect on the cruising phase.

2) Kinematic Constraints: Instead of passing through the waypoints, we define a blending bound between two consecutive line segments. Furthermore, we set a non-zero velocity and acceleration for each waypoint, except for the first and last waypoint. The constraints are described as

$$\begin{aligned}
p^k(t_{i,h_1}) &= p^k_{i+1,\text{bl,start}},\ h_1 \in [4,\dots,6]\,,\\
p^k(t_{i+1,h_2}) &= p^k_{i+1,\text{bl,end}}\,, h_2 \in [0,\dots,3]\,,\\
v^k(t_{0,0}) &= v^k(t_{n,0}) = 0,\ \Delta t_{i,h} \geq 0\,,\\
a^k(t_{0,0}) &= a^k(t_{n,0}) = a^k(t_{i,3}) = 0\,,\\
|j^k(t_{i,h})| &\leq j^k_{\max}\,, |a^k(t_{i,h})| \leq a^k_{\max}\,, \forall\, h \in [0,\dots,6]\\
|v^k(t_{i,h})| &\leq v^k_{\max}\,, |p^k(t_{i,h})| \leq p^k_{\max}\,, \forall\, h \in [0,\dots,6]\\
p^k_{i,\text{bl,lower}} &\leq p^k_{i+1,\text{bl}} \leq p^k_{i+1,\text{bl,upper}}\,,
\end{aligned} \tag{4}$$

where $p^k_{i+1,\text{bl,start}}$ is the start blending segment position at waypoint $i+1$ in axis k and $p^k_{i+1,\text{bl,end}}$ is the end blending segment position at waypoint $i+1$ in axis k. The variable $p^k_{i,\text{bl,lower}}$, $p^k_{i,\text{bl,upper}}$ is a lower and upper blending bound, respectively. The corresponding optimized blending waypoint at $i+1$ in axis k is indicated as $p^k_{i+1,\text{bl}}$. The variable h_1 and h_2 are predefined values that depend on the blending percentage. One relaxation of the jerk $\boldsymbol{j}$ constraint [5] is made by allowing double acceleration or deceleration phases: $\text{sign}(\boldsymbol{j})$ is no longer strictly defined as $[\pm, 0, \mp, 0, \mp, 0, \pm]$ but changed to $\text{sign}(\boldsymbol{j}) = [\pm, 0, \pm, 0, \pm, 0, \pm]$. This relaxation allows reaching the next waypoint without slowing down.

3) Blending Bound Constraints: The blending constraint $\boldsymbol{p}_{i,\text{bl,con}}$ is computed as $\boldsymbol{p}_{i+1} + \hat{\boldsymbol{y}} r\eta$, where $\hat{\boldsymbol{y}}$ is defined as $\frac{\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1}{\|\hat{\mathbf{y}}_2 - \hat{\mathbf{y}}_1\|}$ with $\hat{\mathbf{y}}_1 = \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$ and $\hat{\mathbf{y}}_2 = \frac{\mathbf{p}_{i+2} - \mathbf{p}_{i+1}}{\|\mathbf{p}_{i+2} - \mathbf{p}_{i+1}\|}$. Additionally, $r = l_i/(\tan\alpha_{i+1}/2)$ with $\alpha_{i+1} = \arccos(\hat{\boldsymbol{y}}_1^T \hat{\boldsymbol{y}}_2)$ and $l_i = \min\{\frac{\|\boldsymbol{p}_{i+1} - \boldsymbol{p}_i\|}{2}, \frac{\|\boldsymbol{p}_{i+2} - \boldsymbol{p}_{i+1}\|}{2}, \frac{\delta \sin(\alpha_{i+1}/2)}{(1-\cos(\alpha_{i+1}/2))}\}$, where δ is the predefined blending distance from q_{i+1}. η is the percentage value for controlling the blending bound. The deviation $\epsilon_{\text{bl}} = \|\boldsymbol{p}_{i,\text{bl,con}} - \boldsymbol{p}(t_{i,7})\|$ is bound. Utilizing the blending constraints, we have $p^k_{i,\text{bl,lower}} = \min\{p^k_{i,\text{bl,con}}, p^k(t_{i,7})\}$ and $p^k_{i,\text{bl,upper}} = \max\{p^k_{i,\text{bl,con}}, p^k(t_{i,7})\}$.

B. Blending with Polynomial in Joint Space (TPBJ)

The second approach combines the trapezoidal with a high-dimensional polynomial to form a blending path. The blending segment is described as a high-degree polynomial under consideration of initial $f_0 = (p_0,\ v_0,\ a_0,\ j_0)$ and final $f_1 = (p_1,\ v_1,\ a_1,\ j_1)$ kinematic constraints. These require a total of eight coefficients, therefore we utilize a 7-degree polynomial function in one dimension: $f(t) = b_7 t^7 + b_6 t^6 + \cdots + b_2 t^2 + b_1 t + b_0$. The coefficients b_0 - b_3 can be computed using f_0 with $b_0 = p_0$, $b_1 = v_0$, $b_2 = \frac{a_0}{2}$ and $b_3 = \frac{j_0}{6}$. The coefficients b_4 to b_7 depending on f_1 and polynomial time t can be described as

$$\underbrace{\begin{bmatrix} t_1^7 & t_1^6 & t_1^5 & t_1^4 \\ 7t_1^6 & 6t_1^5 & 5t_1^4 & 4t_1^3 \\ 42t_1^5 & 30t_1^4 & 20t_1^3 & 12t_1^2 \\ 210t_1^4 & 120t_1^3 & 60t_1^2 & 24t_1 \end{bmatrix}}_{\boldsymbol{A}_1} \underbrace{\begin{bmatrix} b_7 \\ b_6 \\ b_5 \\ b_4 \end{bmatrix}}_{\boldsymbol{x}_1} = \underbrace{\begin{bmatrix} p_1 \\ v_1 \\ a_1 \\ j_1 \end{bmatrix}}_{\boldsymbol{y}_1} - \underbrace{\begin{bmatrix} 1 & t_1^1 & t_1^2 & t_1^3 \\ 0 & 1 & 2t_1 & 3t_1^2 \\ 0 & 0 & 2 & 6t_1 \\ 0 & 0 & 0 & 6 \end{bmatrix}}_{\boldsymbol{A}_2} \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}}_{\boldsymbol{y}_2}.$$

Due to this, $\boldsymbol{x}_1(t_1, f_0, f_1) = \boldsymbol{A}_1^{-1}(\boldsymbol{y}_1 - \boldsymbol{A}_2 \boldsymbol{y}_2)$ with time matrices A_1, A_2. The polynomial is simplified as

$$f(t, f_0, f_1) = [t^7, t^6, t^5, t^4]\boldsymbol{x}_1(t_1, f_0, f_1) + h_2(t, f_0)\,, \tag{5}$$

where $h_2(t, f_0)$ is described as $\frac{1}{6}j_0 t^3 + \frac{1}{2}a_0 t^2 + v_0 t + p_0$. Firstly, we assume that the initial f_0 and final f_1 kinematic constraints are available. Therefore, $f(t)$ depends only on the time t. To find a polynomial blending trajectory that satisfies all kinematic constraints, we need to verify the extreme point of the polynomial by computing the root of its derivative. For example, the extreme point of jerk can be found at the position where the first derivative of the jerk (snap) is equal to zero. In addition, a n-degree polynomial has at most n real roots. The constraints can be mathematically formulated as

$$|f^{(n-1)}(\lambda(f^{(n)}))\chi_\lambda(\lambda(f^{(n)}))| \leq k_{\max}\,, \tag{6}$$

where $f^{(n)}$ is n-th derivative of f with $n \in [1, 4]$ and the corresponding constraints $k_{\max} \in [p_{\max}, v_{\max}, a_{\max}, j_{\max}]$. The root-finding function $\lambda(f^{(n)})$ for a given polynomial is used to find the extreme value position for $f^{(n-1)}$. $\chi_A(x)$ is the indicator function of A and will be set to one if $x \in [0, t]$, otherwise to zero. Note that if the $\chi_A(x)$ function is not derivable, the gradient-based optimization solver will diverge. To find a time-optimal polynomial trajectory that satisfies initial/final conditions and lies within the kinematic constraints, we iteratively check the constraints (6) by adding a small delta to $t = t + \Delta t$, where in our case Δt is set to 0.001. Furthermore, to obtain the initial f_0 and final f_1 kinematics, we compute a Point to Point (P2P) trapezoidal acceleration profile movement between each two waypoints, which can be in joint space or Cartesian space, with zero initial and end conditions, and the computed traveling time is indicated as $t_{i->i+1}$. After that, we predefine a blending percentage η to set a start blending time $t_{\text{b}_{\text{start}}} = (1-\eta)t_{i->i+1}$ and end blending time $t_{\text{b}_{\text{end}}} = \eta t_{i+1->i+2}$ for each two consecutive trajectories. By querying the trapezoidal model at $t_{\text{b}_{\text{start}}}$ and $t_{\text{b}_{\text{end}}}$, we obtain the kinematics f_0 and f_1.

C. Blending with Polynomial in Cartesian Space (TPBC)

Utilizing the same principle, we extend the algorithm to the Cartesian space, which is widely used in industrial applications. The blending trajectory in Cartesian space requires separate blending for position and orientation. Interpolation of the Cartesian space is executed in the same way as described in Section III-B. For the orientation part, which has to consider a spherical interpolation, we apply the formulation:

$$\boldsymbol{h}(t) = \boldsymbol{h}_i \Delta\boldsymbol{h}(t) = \boldsymbol{h}_i \begin{pmatrix} \boldsymbol{u}(t)\sin(\theta(t)/2) \\ \cos(\theta(t)/2)\,, \end{pmatrix} \tag{7}$$

where $\boldsymbol{h}_i$ is the initial quaternion, and $\Delta\boldsymbol{h}(t)$ transforms the quaternion from $\boldsymbol{h}_i$ to $\boldsymbol{h}(t)$. The Eigen axis between two quaternions is defined as $\boldsymbol{u}(t) = \boldsymbol{\theta}(t)/\left\|\boldsymbol{\theta}(t)\right\| \in \mathcal{R}^3$ with a rotation angle $\theta(t) = \|\boldsymbol{\theta}(t)\|$. Therefore, the quaternion interpolation depends only on $\boldsymbol{\theta}(t) \in \mathcal{R}^3$. To consider the angular velocity $\boldsymbol{\omega}$, angular acceleration $\dot{\boldsymbol{\omega}}$, and angular jerk $\ddot{\boldsymbol{\omega}}$ at the start and end state for the quaternion blending, the time evolution function $\boldsymbol{\theta}(t)$ is described as: $\boldsymbol{\theta}(t) = \boldsymbol{a}_1(x-1)^7 + \boldsymbol{a}_2 x(x-1)^6 + \cdots + \boldsymbol{a}_8 x^7$ with $x = \frac{t}{t_f - t_0} \in [0,\,1]$. Its roots and its derivative are computed at point $x = 0$ and $x = 1$. Based on $\dot{\boldsymbol{h}} = \frac{1}{2}\boldsymbol{h}\boldsymbol{\omega}$, we can derive the relationship between $\boldsymbol{\omega} \in \mathcal{R}^3$ and $\dot{\boldsymbol{\theta}} \in \mathcal{R}^3$ as $\boldsymbol{\omega} = \boldsymbol{u}\dot{\theta} + \sin(\theta)\boldsymbol{w} \times \boldsymbol{u} - (1 - \cos(\theta))\boldsymbol{w}$, where $\boldsymbol{w} = \frac{(\boldsymbol{u} \times \dot{\boldsymbol{\theta}})}{\theta} \in \mathcal{R}^3$ and $\dot{\theta} = \boldsymbol{u}^T\dot{\boldsymbol{\theta}}$ is a scalar value. We further simplify $\boldsymbol{\omega}$ with the skew-symmetric matrix $(\cdot)^\times$ as $\boldsymbol{\omega} = (\boldsymbol{u}\boldsymbol{u}^T - \frac{\sin(\theta)}{\theta}\boldsymbol{u}^\times\boldsymbol{u}^\times - \frac{(1-\cos(\theta))}{\theta}\boldsymbol{u}^\times)\dot{\boldsymbol{\theta}} = \boldsymbol{A}_{\boldsymbol{\omega},1}\dot{\boldsymbol{\theta}}$. In the same way, we can compute the angular acceleration $\dot{\boldsymbol{\omega}}$ and jerk $\ddot{\boldsymbol{\omega}}$. We combine Cartesian position and quaternion interpolation to form a trajectory with blends.

IV. Experimental Evaluation

We compare the performance of the presented approaches in this paper with the work by Kunz et al. [13] (TO-BAV), Pham et al. [10] (TOPP-RA) and our previous work [4]. The work in [13] generates a trajectory with a designed circular blend under consideration of velocity and acceleration bounds and the work [10] used the reachability-analysis (RA) to solve the time optimal path parameterization (TOPP) problem. Our previous work [4] generates a trajectory by forcing a precise pass through all desired waypoints without stopping. For the evaluation of the motion planning scene, the collision-free paths in the examples are computed using the Robotics Library [17], which is also used for kinematic calculations and simulation. In the evaluation, we set the weights in [4] and TOBJ as $\lambda_1 = 1.2$, $\lambda_2 = 1$, $\lambda_3 = 5000$, and $\lambda_4 = 10$. All evaluations were performed on a laptop with a 2.6 GHz Intel Core i7-6700HQ and 16 GB of RAM.

A. Evaluation of Deviation from Straight-Line in Joint Space

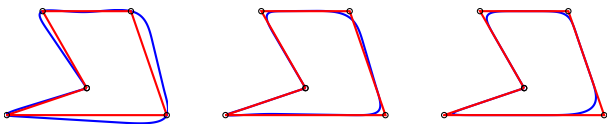
For the first evaluation, we consider a subset of the well-known ISO 9283 [18] cube industrial benchmark as shown in Fig. 1. Here, the robot's end effector has to follow a number of waypoints that are part of a two-dimensional rectangle inside a three-dimensional cube while applying blending. The corresponding results are shown in Fig. 1. The previous work TOPJ [4] in Fig. 1a completed the ISO cube task within 6.26 s by passing through all waypoints and the generated trajectory has to deviate from the straight-line movement to avoid stopping at each waypoint. For improving the quality of the trajectory, TOBJ presented in this work extends the TOPJ by blending around the target position, which results in a better straight-line movement and completed the ISO cube task with a shorter time of 6.1 s due to a shortened path length. The polynomial based algorithm (TPBJ) can further reduce the completion time to 5.94 s by setting a bigger blending circle radius.

B. Comparison between TO-BAV, TOPP-RA and TPBJ

In this section, we compare the algorithm of [10], [13] and our polynomial-based one, as using an extreme jerk value in the optimization-based one may not converge since the gradient value in jerk direction is not at the same order of magnitude with other gradient values in the gradient vector. The results are shown in Figs. 2a to 2o. The first column shows the results from TO-BAV [13], where the velocity arrives at the peak value very quickly with execution time 2.2577 s. The second column illustrates the results of TOPP-RA [10], which is forced precisely to pass through desired waypoints with the traveling time 2.88 s. The other columns show the results of the polynomial-based algorithm with increasing jerk limits, starting from 5, to 100, and finally 10 000 times the maximum velocity value. The respective trajectory travel time reduces from a value of 3.577 s to 2.28 s and our algorithm gradually approaches the profile of [13], while still limiting the jerk. We plot the first and third DOF path without loss of generality, shown in the first row. TOPP-RA produces a trajectory shown in Fig. 2b which has a noticeably bigger straight-line deviation in joint space than other two algorithms. The bigger straight-line deviation can lead to a collision, which is not desirable for the industrial application. From the perspective of velocity and acceleration performance, the velocity profile from TOPP-RA shown in Figs. 2g and 2l is less smooth than TO-BAV and TPBJ and exhibits several vibration points.

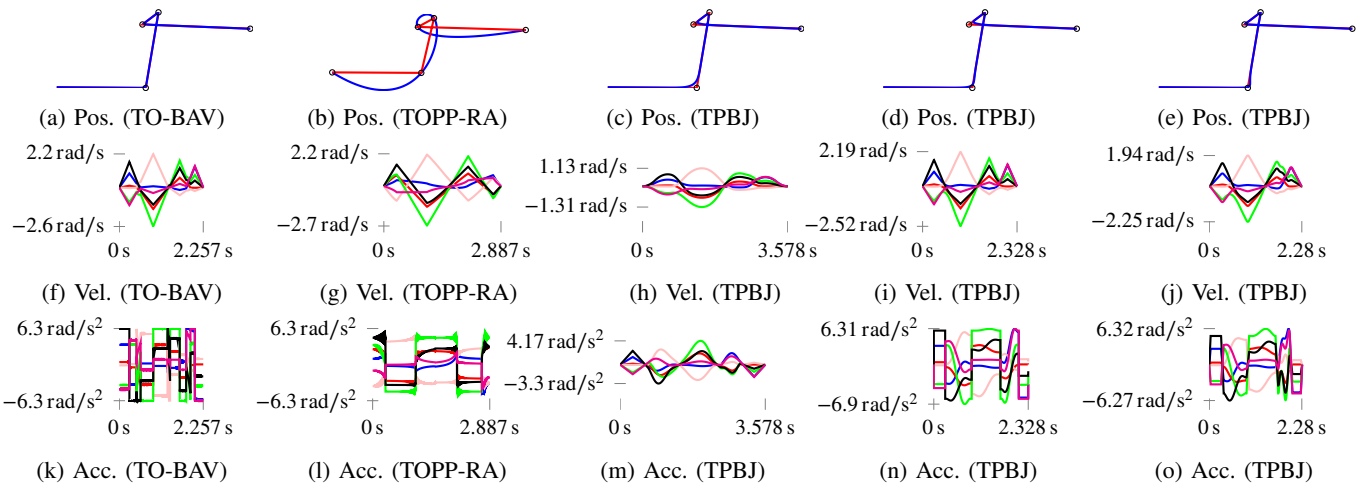
C. Evaluation of the Algorithm in a Real Robot Workcell

We evaluate our algorithms on a Universal Robots UR5 robot manipulator by looking at the actual velocity and current measured by the UR5 controller with an update rate of 125 Hz over the native Real-Time Data Exchange protocol. We compare our approaches against the algorithm developed



(a) (b) (c)

Fig. 1: Comparison of joint space trajectories for the ISO cube scenario. The plots show the first and second DOF of different algorithms. The straight line reference in joint space is shown in *red*, the calculated joint trajectory in *blue* with (a) TOPJ [4], (b) TOBJ, (c) TPBJ.

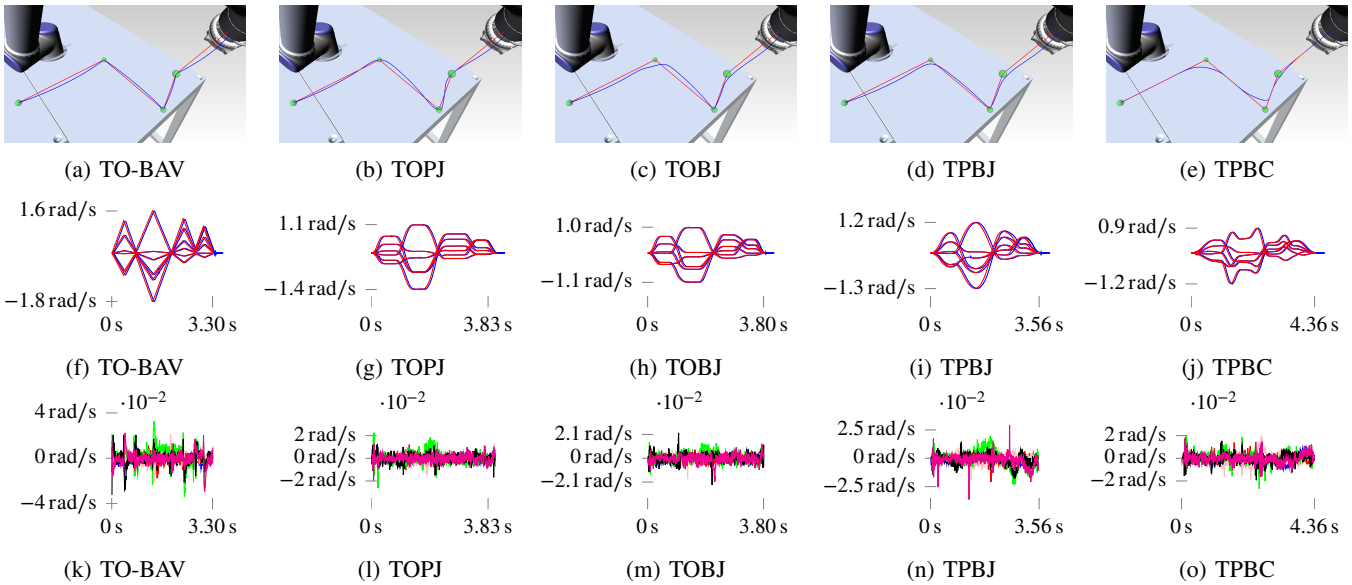


(a) Pos. (TO-BAV) (b) Pos. (TOPP-RA) (c) Pos. (TPBJ) (d) Pos. (TPBJ) (e) Pos. (TPBJ)

(f) Vel. (TO-BAV) (g) Vel. (TOPP-RA) (h) Vel. (TPBJ) (i) Vel. (TPBJ) (j) Vel. (TPBJ)

(k) Acc. (TO-BAV) (l) Acc. (TOPP-RA) (m) Acc. (TPBJ) (n) Acc. (TPBJ) (o) Acc. (TPBJ)

Fig. 2: Comparison of results from TO-BAV [13], TOPP-RA [10] and TPBJ with increasing jerk constraints. (a)–(e) show the plot of first and third DOF with generated trajectory (blue) and target straight-line path (red). (f)–(j) is the velocity profile. (k)–(o) is the acceleration profile. In our approach, TPBJ gradually increases jerk constraints from $j_{\max} = \{5, 100, 10000\} v_{\max}$.



(a) TO-BAV (b) TOPJ (c) TOBJ (d) TPBJ (e) TPBC

(f) TO-BAV (g) TOPJ (h) TOBJ (i) TPBJ (j) TPBC

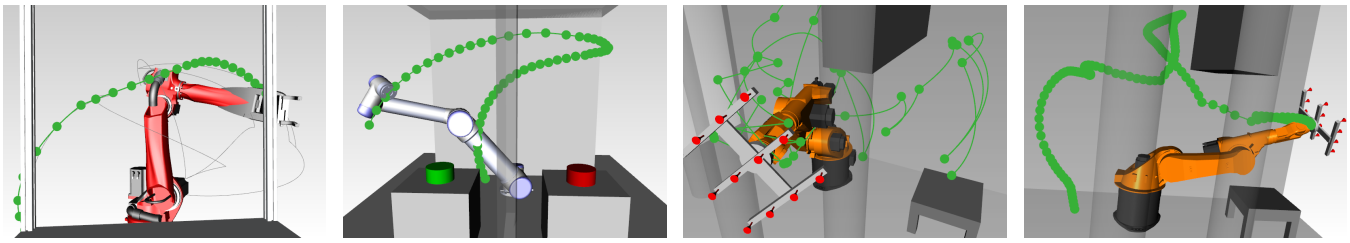
(k) TO-BAV (l) TOPJ (m) TOBJ (n) TPBJ (o) TPBC

Fig. 3: A comparison of different trajectory profiles for the UR5 example. The individual plots show (a)–(e) position, (f)–(j) velocity with the controller's target (red) and actual (blue) value, and (k)–(o) corresponding velocity differences.

by TO-BAV [13]. The evaluation results are illustrated in Fig. 3. The maximum velocity and acceleration values are identical for all trajectory generators. The only difference is that [13] does not consider any jerk limitation. The remaining algorithms limit the maximum jerk to a value of five times faster than the maximum velocity. We evaluate the computational complexity of each algorithm by running the experiment 40 times. The computed trajectories are visualized in a 3D environment [17] as shown in Figs. 3a to 3e. The blending directly implemented in Cartesian space follows a straight line. The other four algorithms have a similar trajectory performance. It needs be pointed out that the blending mode in [4] is different from the other algorithms, as it can pass precisely through all desired waypoints without stopping. From the perspective of velocity performance, the result from [13] indicates a large gap between desired and real velocity which occurs at each turning point, shown in Fig. 3k, as the robot controller is not able immediately to execute a trajectory that contains an infinite jerk. If the maximum specified acceleration is further increased, the additional burden on the motors may lead to hardware issues and a reduced lifetime. In contrast to this, the algorithm presented in this work considers the jerk limitation. The robot controller can follow the desired waypoints continuously and demonstrates reduced motor current values. Figures 3g and 3h exhibit a clear period of cruising phase in comparison to the other algorithms, as the objective function in the optimization step emphasizes a longer constant velocity

TABLE I: Benchmark results of path planning scenarios. The best results are highlighted in bold and smaller values are better. The maximum blending deviation δ is set to 0.1. The jerk limitation used by TPBJ is set to 100x and 500x of the maximal velocity constraints, denoted as TPBJ1 and TPBJ5, respectively. TOBJ is set to 100x. N_{point} describes the number of waypoint, t_{comp} is the computation time, and t_{tra} is the traveling time. L_{alg} is the traveling length, L_{stra} is the base straight-line length, and we present the percentage.

	Scenario 1 ($N_{\text{point}} = 42$)					Scenario 2 ($N_{\text{point}} = 55$)					Scenario 3 ($N_{\text{point}} = 42$)					Scenario 4 ($N_{\text{point}} = 181$)				
Alg.	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5	TO-BAV	TOPP-RA	TOBJ	TPBJ1	TPBJ5
t_{comp} [s]	0.23	0.28	54.15	0.15	**0.088**	0.28	**0.186**	65.7	0.65	0.319	0.32	0.45	149.52	0.41	**0.25**	1.68	0.54	231.68	0.70	**0.41**
t_{tra} [s]	4.08	**3.759**	7.09	7.02	5.95	3.67	**3.40**	6.35	6.58	5.88	25.20	**23.95**	26.17	26.32	25.48	11.75	**10.55**	26.48	26.44	18.91
$L_{\text{alg}}/L_{\text{stra}}$	0.999	1.0027	**0.997**	1.0001	0.999	0.999	1.003	0.998	1.0001	**0.998**	0.998	1.174	**0.997**	1.001	**0.997**	**0.996**	1.001	1.001	1.001	0.997



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 4: Benchmarks on different motion planning scenarios. (a) a Comau Racer 7-1.4 moves to a workstation with a parallel gripper. (b) a UR5 moving between two walls. (c)–(d) a Kuka KR60-3 next to a wall and 3 columns with a vacuum gripper.

phase over the acceleration and deceleration phases. This behavior is desirable in certain industrial robot task such as welding and gluing, as the cruising segments show a smoother overall behavior compared to the other segments.

D. Evaluation of Computation Complexity in Motion Planning Scenarios

We evaluate the algorithms for generating trajectories in different path planning scenarios with an increasing number of waypoints (from 42 to 181) and different point distribution characteristics, as illustrated in Fig. 4. We compare our approaches against TO-BAV, which is currently a standard trajectory generator in the MoveIt! framework, and TOPP-RA. We summarize the results in Table I. Regarding computation time, TPBJ is faster in most scenarios apart from scenario 2. In comparison to TOBJ and TPBJ under the same jerk limitation (100x of maximal velocity constraints), they show a similar performance. However, if the jerk limitation is bigger than 100x, TOBJ has a convergence problem, because jerk and time have huge differences in numerical magnitude. In comparison to TO-BAV and TPBJ with jerk limitation set to 100x and 500x of the maximum velocity constraint, we can conclude that with a higher jerk limitation, TPBJ can significantly reduce the traveling time. This conclusion can be drawn from subsection IV-B as well. The trajectory optimizer TOPP-RA without jerk limitation generates a trajectory with minimal traveling time. From the perspective of straight-line deviation, TOPP-RA generated a trajectory with significant overshooting in scenario 3 with 117.4% path length with respect to a straight-line movement, since TOPP-RA utilizes cubic spline interpolation for path parameterization. The drawbacks of using cubic spline interpolation are shown in [4].

V. Conclusion

In this work, we have extended our previous work by including the capability to blend around the target position. We have presented two different approaches to finding a time-optimal and jerk-limited trajectory. In the first approach, the algorithm follows the same principle as in our previous work by using a bridged optimization procedure, which reduces the computational complexity to a linear complexity with respect to the number of waypoints and degrees of freedom. In contrast to our previous work, we redesigned the objective function and blending constraints to achieve a better straight-line movement in joint space and allow the trajectory to blend around the target position. However, TOBJ has a convergence problem when the jerk constraint exceeds 100x of the maximum velocity constraint due to the huge differences in numerical magnitude between jerk und time. Further improvement will be left to future work by using more advanced optimization strategies. The second approach combines a trapezoidal trajectory with a seven-degree polynomial. In this approach, we compute a point-to-point motion for every two waypoints by using a standard trapezoidal acceleration model. By specifying a blend percentage, the seven-degree polynomial can be used to find a curve segment around the target position. To find a time-optimal trajectory that fully considers all kinematic constraints, we iteratively increase the trajectory time until these constraints are no longer violated for all degrees of freedom. The second approach can be directly extended to the Cartesian space by using the introduced quaternion interpolation algorithm. These two approaches do not suffer from convergence problems and show good performance in our experiments when compared against state-of-the-art approaches without jerk limitations.

REFERENCES

[1] A. Perzylo, M. Rickert, B. Kahl, N. Somani, C. Lehmann, A. Kuss, S. Profanter, A. B. Beck, M. Haage, M. R. Hansen, M. T. Nibe, M. A. Roa, O. Sornmo, S. G. Robertz, U. Thomas, G. Veiga, E. A. Topp, I. Kesslar, and M. Danzer, "SMErobotics: Smart robots for flexible manufacturing," *IEEE Robotics and Automation Magazine*, vol. 26, no. 1, pp. 78–90, Mar. 2019.

[2] E. F. Sertac Karaman, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[3] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM*, vol. 40, no. 5, pp. 1048–1066, Nov. 1993.

[4] J. Lin, N. Somani, B. Hu, M. Rickert, and A. Knoll, "An efficient and time-optimal trajectory generation approach for waypoints under kinematic constraints and error bounds," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 2018, pp. 5869–5876.

[5] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, Sept. 2008, pp. 3248–3253.

[6] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, Feb. 2009.

[7] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[8] F. Lange and A. Albu-Schäffer, "Path-accurate online trajectory generation for jerk-limited industrial robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 82–89, Jan. 2016.

[9] R. Saravanan, S. Ramabalan, and C. Balamurugan, "Evolutionary optimal trajectory planning for industrial robot with payload constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 38, pp. 1213–1226, Sept. 2008.

[10] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, June 2018.

[11] Á. Nagy and I. Vajk, "Sequential time-optimal path-tracking algorithm for robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, Oct. 2019.

[12] E. Barnett and C. Gosselin, "A bisection algorithm for time-optimal trajectory planning along fully specified paths," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 1–15, Feb. 2021.

[13] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Proceedings of Robotics: Science and Systems*, July 2012.

[14] N. Dantam and M. Stilman, "Spherical parabolic blends for robot workspace trajectories," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, Sept. 2014, pp. 3624–3629.

[15] T. Kröger, "Opening the door to new sensor-based robot applications—the Reflexxes motion libraries," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[16] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[17] M. Rickert and A. Gaschler, "Robotics Library: An object-oriented approach to robot applications," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vancouver, BC, Canada, Sept. 2017, pp. 733–740.

[18] "Manipulating industrial robots, Performance criteria and related test methods," ISO, International Standard 9283, Apr. 1998.