Article

A Cartesian-Based Trajectory Optimization with Jerk Constraints for a Robot

Zhiwei Fan [1,2,3,*], Kai Jia [1,2,4], Lei Zhang [1,2,4], Fengshan Zou [1,2,4], Zhenjun Du [4], Mingmin Liu [4], Yuting Cao [1,2,3] and Qiang Zhang [5]

1. State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
2. Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China
3. University of Chinese Academy of Sciences, Beijing 100049, China
4. SIASUN Robot & Automation Co., Ltd., Shenyang 110169, China
5. School of Automation, Jiangsu University of Science and Technology, No. 666 Changhui Road, Zhenjiang 212100, China

* Correspondence: fanzhiwei@sia.cn

Abstract: To address the time-optimal trajectory planning (TOTP) problem with joint jerk constraints in a Cartesian coordinate system, we propose a time-optimal path-parameterization (TOPP) algorithm based on nonlinear optimization. The key insight of our approach is the presentation of a comprehensive and effective iterative optimization framework for solving the optimal control problem (OCP) formulation of the TOTP problem in the $(s, \dot{s})$-phase plane. In particular, we identify two major difficulties: establishing TOPP in Cartesian space satisfying third-order constraints in joint space, and finding an efficient computational solution to TOPP, which includes nonlinear constraints. Experimental results demonstrate that the proposed method is an effective solution for time-optimal trajectory planning with joint jerk limits, and can be applied to a wide range of robotic systems.

Keywords: time-optimal trajectory planning; iterative optimization; jerk limits; time-optimal path parameterization; phase plane

Citation: Fan, Z.; Jia, K.; Zhang, L.; Zou, F.; Du, Z.; Liu, M.; Cao, Y.; Zhang, Q. A Cartesian-Based Trajectory Optimization with Jerk Constraints for a Robot. *Entropy* **2023**, *25*, 610. https://doi.org/10.3390/e25040610

Academic Editors: Ahmad Taher Azar, Quanmin Zhu, Giuseppe Fusco, Amjad J. Humaidi and Ibraheem Kasim Ibraheem

Received: 28 February 2023
Revised: 31 March 2023
Accepted: 31 March 2023
Published: 3 April 2023

1. Introduction

Presently, industrial robotics has a wide range of applications, including welding, palletizing, grinding and polishing, assembly, and painting [1–3]. After decades of research, the problem of time-optimal trajectory planning (TOTP) of robots along specified paths has been extensively studied to optimize operation time and improve the efficiency of automated industrial robot operations [4]. TOTP is based on interpolation and introduces the concepts of constraint and optimization to maximize the performance of the robot and ensure the shortest time, while making the trajectory smooth and the operation run smoothly [5]. Time-optimal path parameterization (TOPP) is a fast method for determining critical conditions for navigating a pre-defined smooth path in a robot system's configuration space while respecting physical constraints [6]. Although finding the time-optimal parameterization of a path subject to second-order constraints is a well-studied problem in robotics, TOPP subject to third-order constraints (such as jerk and torque rate) has received relatively little attention and remains largely open. Moreover, joint space trajectory planning cannot visualize the end position of the robotic arm, and Cartesian space trajectory planning is often used in many specific industrial scenarios such as welding, cutting, or machining that require operation on a predetermined path. Therefore, a TOTP algorithm that satisfies the joint third-order constraints in Cartesian space is urgently needed.

1.1. Related Works

Over the years, many academics have worked on the issue of TOTP for industrial robots. This problem can be roughly divided into three main families of methods: Numerical Integration (NI), Convex Optimization (CO), and Dynamic Programming (DP).

The NI-based strategy was initiated by Bobrow et al. [7] and further developed by other researchers. Kunz et al. [8] provided a circular-blends route differentiability approach to ensure that the trajectory precisely follows the specified path of differentiable joint space. Pham [9] provided a comprehensive solution to the problem of dynamic singularities. Pham et al. [10] proposed TOPP3, a novel TOPP algorithm that addresses third-order constraints, as well as the problem of singularities that may hinder the integration of motion profiles and the smooth connection of optimal profiles.Shen et al. [11,12] proposed various new characteristics of the NI method for TOTP along the defined path, and provided explicit mathematical confirmation of these traits. Lu et al. [13] proposed a time-optimal motion planning method for sculpted surface robot machining that takes joint space and tool tip motion constraints into account. They solved the time-optimal tool motion planning in robot machining using an efficient numerical integration method based on the Pontryagin maximum principle. Methods based on NI explicitly calculate the optimal control at each position along the path, instead of performing an implicit search such as the CO-based method, which makes them very fast. However, finding the switch points between the acceleration and deceleration phases is necessary, and the main reason for their failure.

The CO-based strategy has been expanded upon by numerous researchers after being introduced by Verscheure et al. [14]. Xiao et al. [15] used the cubic polynomial fitting method to construct the maximum pseudo-speed curve that meets the torque and speed limits. Debrouwere et al. [16] proposed an effective sequential convex programming (SCP) method to solve the corresponding nonconvex optimal control problems as a difference of convex (DC) function. Pham et al. [6] presented a TOPP approach based on reachability analysis (TOPP-RA), which iteratively computes the reachable and controllable sets at discrete points along the path by solving linear programming problems (LP). Nagy et al. [17] considered kinematics and dynamics constraints and generated the time-optimal velocity distribution for the LP control problem using the sequential optimization method. Ma et al. [18] converted a nonconvex jerk limit into a linear acceleration constraint and indirectly introduced it into CO for TOTP. This method preserves CO's convexity and does not increase the number of optimization variables, resulting in a quick calculation speed. CO-based methods are easy to implement and quite robust, and can consider multiple optimization objectives beyond just time. However, the optimization problem they solve is very large. The number of variables and constraint inequalities scales with the discretization grid size, resulting in implementation that is an order of magnitude slower than the NI-based method.

The DP-based approach was first developed by [19] and has since been expanded and improved upon by numerous researchers. Kaserer et al. [20] proposed a DP-based method for solving the optimal path-tracking problem, which uses interpolation in the phase plane. This approach considers joint speed, acceleration, torque, and mechanical power, as well as joint jerk and torque rate limitations. Kaserer et al. [21] extended this method to solve the time-optimal path-tracking problem for cooperative grasping tasks involving two robots, while also accounting for robot speed, acceleration, jerk, and torque constraints. Barnett et al. [22] introduced the bisection algorithm (BA), a novel technique that extends DP approaches to tackle more complex problems with a larger number of constraints. These approaches, which break down the larger problem into smaller subproblems, become increasingly advantageous as the number of constraints grows, compared to direct transcription methods. Methods based on DP are simple to implement and do not suffer from local minima problems, and they traverse all states at each path point (rather than requiring convex space or convex function assumptions such as the CO-based method). However, the state space to be searched is huge, resulting in implementation being one (or even more) orders of magnitude slower than the CO-based method. Additionally, the DP method cannot truly achieve the global optimal point due to the issue of grid precision.

There are several alternative approaches to the TOTP problem beyond the three groups mentioned above [23–28]. Nevertheless, these approaches also neglect the third-order constraint and do not perform planning in Cartesian space. Table 1 summarizes and compares the similarities and differences of the above three methods in the following five aspects: the requirement for calculating switching points, the ability to consider multiple optimization objectives, the ability to achieve the optimal point (rather than approximately achieving the optimal point), the planning space, and the highest constraint order.

Table 1. A brief overview of related methods.

Methods		**Calculate Switch Points**	**Optimization Objectives (Simple, Multiple)**	**Achieve Optimal Point**	**Planning Space**	**Constraint Order (Second-Order, Third-Order)**
NI-based	[7,11,12]	Need	Simple	Yes	Joint/Cartesian	Second-order
	[8,9]	Need	Simple	Yes	Joint	Second-order
	[10,13]	Need	Simple	Yes	Joint	Third-order
CO-based	[6,17]	Not need	Simple	Yes	Joint	Third-order
	[14,15]	Not need	Multiple	Yes	Joint/Cartesian	Second-order
	[16]	Not need	Multiple	Yes	Joint/Cartesian	Third-order (Limit)
	[18]	Need	Simple	Yes	Joint	Third-order (Limit)
DP-based	[19,20]	Not need	Multiple	No	Joint	Third-order
	[21]	Not need	Multiple	No	Joint/Cartesian	Third-order
	[22]	Not need	Multiple	No	Joint	Second-order
	Ours	Not need	Multiple	Yes	Joint/Cartesian	Third-order

1.2. Motivations and Contributions

Motivated by previous approaches, this paper proposes aTOTP algorithm that considers joint third-order limits in a Cartesian coordinate system, maximizing the robot operation efficiency while maintaining smoothness and minimizing time. To achieve this, kinematic feasibility is ensured by introducing joint velocity, acceleration, and jerk constraints on the path parameters s, which are then relocated to the Cartesian space using a constraint transfer method based on Lie theory (We use the Lie group SE(3) to represent the motion of the robot end-effector in Cartesian space. The detailed description of using Lie theory for robot forward and inverse kinematic analysis and the Jacobian matrix derivation process is presented in the Appendix A), reducing the number of decision variables. After establishing the optimal control problem (OCP) formulation of the TOTP problem in the $(s, \dot{s})$ phase plane, the TOPP-RA algorithm is extended to the Cartesian space to obtain an initial solution, and a constraint relaxation approach is used to simplify nonconvex state-update constraints. The method is validated through simulation experiments on a ROS-based platform and real-world experiments on an actual robot, demonstrating effectiveness, generality, and robustness. This paper makes several contributions to the field of optimal trajectory generation:

- A comprehensive and effective framework for iterative optimization is presented to establish the OCP formulation of the TOTP problem, which is described by the path parameter s;
- Given an efficient computational solution for computing the nonlinear TOPP in Cartesian space while satisfying third-order constraints in joint space;

- Experiments have demonstrated that the proposed method can effectively generate smoother trajectories that satisfy jerk constraints on a wide range of robot systems.

The remainder of this paper is organized as follows. Section 2 outlines the key features of the OCP model used for the TOPP algorithm. In Section 3, we present the Cartesian-based TOPP-RA method and describe the proposed TOPP algorithm based on iterative optimization. Section 4 reports extensive experimental results. Finally, in Section 5, we provide concluding remarks.

2. Problem Statement

In this section, we establish the TOPP problem as an OCP in Cartesian space, which includes joint third-order constraints and an objective function in the $(s, \dot{s})$ phase plane. The details of these constraints will be formulated in the following subsections.

2.1. General Description

In a n-dof robot system, the state profiles in configuration space are denoted by $\mathbf{x}(t) = [\mathbf{q}(t); \dot{\mathbf{q}}(t); \ddot{\mathbf{q}}(t)]$, where $\mathbf{q} \in \mathbb{R}^n$ represents the configuration of the system. The control inputs $\mathbf{u}(t)$ represent the third derivative of the joint angles, $\dddot{\mathbf{q}}(\mathbf{t})$, in configuration space. The following is a standard OCP that can be used to describe the time-optimal speed planning problem [29]:

$$\begin{aligned} &\min J(\mathbf{x}(t), \mathbf{u}(t)) \\ &s.t.\ \dot{\mathbf{x}}(t) = f_{Status-update}(\mathbf{x}(t), \mathbf{u}(t)), \\ &\quad \mathbf{x}_{min} \le \mathbf{x}(t) \le \mathbf{x}_{max}, \mathbf{u}_{min} \le \mathbf{u}(t) \le \mathbf{u}_{max}, t \in [0, T]; \\ &\quad \mathbf{x}(0) = \mathbf{x}_{init}, \mathbf{u}(0) = \mathbf{u}_{init}, \mathbf{x}(T) = \mathbf{x}_{goal}, \mathbf{u}(T) = \mathbf{u}_{goal}. \end{aligned} \tag{1}$$

$f_{Status-update} = 0$ forms the status-update process. $[\mathbf{x}_{min}, \mathbf{x}_{max}, \mathbf{u}_{min}, \mathbf{u}_{max}]$ describes the allowable regions of state and control profiles. $[\mathbf{x}_{init}, \mathbf{u}_{init}, \mathbf{x}_{goal}, \mathbf{u}_{goal}]$ denotes the start and end conditions of the state and control profiles. T represents the total time, which is unidentified now.

To translate the above model to a TOPP problem in the $(s, \dot{s})$ phase plane, we propose a function $\mathbf{p}(s)_{s\in[0,s_{end}]}$ that represents a geometric path in the Cartesian space, and is piece-wise $\mathcal{C}^2$-continuous. We introduce a time parameterization that itself represents the parameter of the path, as a piece-wise $\mathcal{C}^2$, increasing scalar function $s : [0, T] \to [0, s_{end}]$. The trajectory is then recovered as $\mathbf{p}(s(t))_{t\in[0,T]}$ [30]. In the rest of this section, we introduce how to complete the transformation of the TOPP problem through $s : [0, T] \to [0, s_{end}]$.

2.2. Objective Function

To minimize the total time of robot movement, the objective function $J(\mathbf{x}(t), \mathbf{u}(t))$ is defined as

$$J = T = \int_{t=0}^{T} 1dt \tag{2}$$

Replace the previous equation with $ds/ds = 1$ and change the integral limits from $[0, T]$ (time) to $[0, s_{end}]$ (s) [31]. Formula (2) is updated as follows:

$$J = \int_{t=0}^{T} 1dt = \int_{t=0}^{T} \frac{ds}{ds}dt = \int_{s=0}^{s_{end}} \frac{dt}{ds}ds = \int_{s=0}^{s_{end}} \frac{1}{\dot{s}}ds \tag{3}$$

Therefore, to minimize the time, $\dot{s}^{-1}(s)$ should be as small as possible. In other words, $\dot{s}(s)$ must be as large as possible while still satisfying the various constraints mentioned later. This means that the state trajectory must follow the boundary of the phase diagram plotted by $(s, \dot{s})$, which is naturally aligned with TOPP-RA method.

2.3. Constraints

In a TOPP problem, there are generally three types of constraints: status-update constraints, constraints on the states/control profiles, and two-point boundary constraints [32].

2.3.1. Status-Update Constraints

The state-update/kinematic constraints of a robot describe the kinematic feasibility of the robot's motion. Using forward and inverse kinematics, the configuration $\mathbf{q}$ in the joint space can be converted to the corresponding Cartesian space representation $\mathbf{p}$ (see the Appendix A for transformation method). As a result, the state and control profiles can be expressed in terms of the geometric path $\mathbf{p}(s)$, which can then be further transformed to a form represented by path parameters s, as shown in Equation (4).

$$\frac{\mathrm{d}}{\mathrm{d}s}\begin{bmatrix}\dot{s}\\ \ddot{s}\end{bmatrix} = \begin{bmatrix}\frac{\ddot{s}}{\dot{s}}\\ \frac{\dddot{s}}{\dot{s}}\end{bmatrix}, s \in [0, s_{end}] \tag{4}$$

The status-update function can be rewritten by performing a second-order Taylor series expansion at s_i.

$$\begin{aligned}\dot{s} &= \dot{s}_i + \frac{\ddot{s}_i}{\dot{s}_i}\Delta(s) + \left.\frac{\mathrm{d}^2\dot{s}}{\mathrm{d}s^2}\right|_{s=\xi}(\Delta(s))^2\\ \ddot{s} &= \ddot{s}_i + \frac{\dddot{s}_i}{\dot{s}_i}\Delta(s) + \left.\frac{\mathrm{d}^2\ddot{s}}{\mathrm{d}s^2}\right|_{s=\eta}(\Delta(s))^2\end{aligned} \tag{5}$$

where $s \in [s_i, s_{i+1}], \xi, \eta \in [s_i, s]$ and $\Delta(s) = s - s_i$. Let us define the first-order status-update discretization function as follows:

$$\begin{aligned}\dot{s} &= \dot{s}_i + \frac{\ddot{s}_i}{\dot{s}_i}\Delta(s)\\ \ddot{s} &= \ddot{s}_i + \frac{\dddot{s}_i}{\dot{s}_i}\Delta(s)\end{aligned} \tag{6}$$

The error of the first-order status-update discretization function, denoted by e^{state}_{first}, is as follows:

$$e^{state}_{first} = O(\Delta^2(s)) \tag{7}$$

Similarly, by performing a third-order Tylor series expansion at s_i, the second-order status-update discretization function and its error can be, respectively, rewritten as:

$$\begin{aligned}\dot{s} &= \dot{s}_i + \frac{\ddot{s}_i}{\dot{s}_i}\Delta(s) + \left(\frac{\dddot{s}_i}{\dot{s}_i^2} - \frac{\ddot{s}_i^2}{\dot{s}_i^3}\right)\Delta^2(s)\\ \ddot{s} &= \ddot{s}_i + \frac{\dddot{s}_i}{\dot{s}_i}\Delta(s) - \frac{\ddot{s}_i\dddot{s}_i}{\dot{s}_i^3}\Delta^2(s)\end{aligned} \tag{8}$$

$$e^{state}_{second} = O(\Delta^3(s)) \tag{9}$$

2.3.2. States/Control Profiles Constraints

The state/control constraints of a robot refer to the physical constraints that the robot must adhere to during its motion process. Typically, these constraints involve the robot's state variables, such as position, velocity, acceleration, joint angles, and so on. The constraints on the robot's states and control profiles can be formulated as $\mathbf{x}_{min} \leq \mathbf{x}(t) \leq \mathbf{x}_{max}$ and $\mathbf{u}_{min} \leq \mathbf{u}(t) \leq \mathbf{u}_{max}$, respectively, where $t \in [0, T]$. These constraints essentially limit the speed, acceleration, and jerk of the robot's joints [33], as illustrated in the following equations.

$$\begin{bmatrix} \dot{\mathbf{q}}_{min} \\ \ddot{\mathbf{q}}_{min} \\ \dddot{\mathbf{q}}_{min} \end{bmatrix} \leq \begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \\ \dddot{\mathbf{q}}(t) \end{bmatrix} \leq \begin{bmatrix} \dot{\mathbf{q}}_{max} \\ \ddot{\mathbf{q}}_{max} \\ \dddot{\mathbf{q}}_{max} \end{bmatrix}, t \in [0, T] \tag{10}$$

The derivatives of the joints are projected into Cartesian space through the Jacobian matrix, as shown in Formula (11), which yields the derivatives of the path parameter s.

$$\begin{aligned} \mathbf{J}\dot{\mathbf{q}} &= \mathbf{p}'\dot{s} \\ \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}} &= \mathbf{p}''\dot{s}^2 + \mathbf{q}'\ddot{s} \\ \mathbf{J}\dddot{\mathbf{q}} + 2\dot{\mathbf{J}}\dot{\mathbf{q}} + \ddot{\mathbf{J}}\dot{\mathbf{q}} &= \mathbf{p}'''\dot{s}^3 + 3\mathbf{p}''\dot{s}\ddot{s} + \mathbf{p}'\dddot{s} \end{aligned} \tag{11}$$

where $\square'$ is defined as the differentiation of $\square$ with respect to the path parameter s. Henceforth, we shall refer to $s, \dot{s}, \ddot{s}$, and $\dddot{s}$ as the position, velocity, acceleration, and jerk, respectively. By substituting Equation (11) into Equation (10), the inequality constraints on the states/control profiles can be expressed as follows:

$$\begin{bmatrix} \dot{\mathbf{q}}_{min} \\ \ddot{\mathbf{q}}_{min} \\ \dddot{\mathbf{q}}_{min} \end{bmatrix} \leq \begin{bmatrix} \mathbf{a}(s)\dot{s} \\ \mathbf{b}(s)\dot{s}^2 + \mathbf{c}(s)\ddot{s} \\ \mathbf{d}(s)\dot{s}^3 + \mathbf{e}(s)\dot{s}\ddot{s} + \mathbf{f}(s)\dddot{s} \end{bmatrix} \leq \begin{bmatrix} \dot{\mathbf{q}}_{max} \\ \ddot{\mathbf{q}}_{max} \\ \dddot{\mathbf{q}}_{max} \end{bmatrix}, s \in [0, s_{end}], \text{where} \tag{12}$$

$$\begin{aligned} \mathbf{a}(s) &:= \mathbf{J}^{-1}(s)\mathbf{p}'(s), \\ \mathbf{b}(s) &:= \mathbf{J}^{-1}(s)(\mathbf{p}''(s) - \mathbf{J}'(s)\mathbf{J}^{-1}(s)\mathbf{p}'(s)), \\ \mathbf{c}(s) &:= \mathbf{J}^{-1}(s)\mathbf{p}'(s), \\ \mathbf{d}(s) &:= \mathbf{J}^{-1}[\mathbf{p}'''(s) - 2\mathbf{J}'(s)\mathbf{J}^{-1}(s)(\mathbf{p}''(s) - \mathbf{J}'(s)\mathbf{J}^{-1}(s)\mathbf{p}'(s)) - \mathbf{J}''(s)\mathbf{J}^{-1}(s)\mathbf{p}'(s)], \\ \mathbf{e}(s) &:= 3\mathbf{J}^{-1}(s)(\mathbf{p}''(s) - \mathbf{J}'(s)\mathbf{J}^{-1}(s)\mathbf{p}'(s)), \\ \mathbf{f}(s) &:= \mathbf{J}^{-1}(s)\mathbf{p}'(s). \end{aligned} \tag{13}$$

The formulas for calculating each order derivative of the Jacobian matrix ($\mathbf{J}', \mathbf{J}''$) will be presented in the Appendix A.

2.3.3. Boundary Constraints

Boundary constraints refer to the limitations imposed on the state and control variables of a robot during the initial and final stages of its operation. The constraints $\mathbf{x}(0) = \mathbf{x}_{init}$, $\mathbf{u}(0) = \mathbf{u}_{init}$, $\mathbf{x}(T) = \mathbf{x}_{goal}$, and $\mathbf{u}(T) = \mathbf{u}_{goal}$ define the boundary conditions. These boundary conditions ensure that the state and control profiles at the start moment $s = 0(t = 0)$ and the end moment $s = s_{end}(t = T)$ represent the necessary facts at those moments, respectively.

$$\begin{aligned} [\dot{s}(0), \ddot{s}(0), \dddot{s}(0)] &= [\dot{s}_0, \ddot{s}_0, \dddot{s}_0], \\ [\dot{s}(s_{end}), \ddot{s}(s_{end}), \dddot{s}(s_{end})] &= [\dot{s}_{s_{end}}, \ddot{s}_{s_{end}}, \dddot{s}_{s_{end}}]. \end{aligned} \tag{14}$$

In particular, more degrees of freedom are allowed in setting the control profile $\dddot{s}$ at $s = 0(t = 0)$ to ensure the normal operation of the motor.

As a summary of this section, the following OCP is established to represent the TOPP problem based on Cartesian space:

$$\begin{aligned} &\min\ (3) \\ &\ s.t.\ \ \text{Status-update constraints (4);} \\ &\qquad\ \text{States/Control profiles constraints (12) and (13);} \\ &\qquad\ \text{Two-point boundary constraints (14).} \end{aligned} \tag{15}$$

In general, when moving from the initial state to the target state along a predetermined path, speed planning aims to resolve any potential conflicts that may arise between kinematics-based constraints and environmental constraints. However, due to the nonlinear relationship between the state and control variables, an appropriate initial solution is required to solve the OCP (15). There is a problem with the state/control constraints (12) in OCP (15) because the jerk of the robot is not taken into account when solving the initial solution, which can easily lead to leaving out the free space required for kinematic feasibility. Therefore, directly solving OCP (15) may not always be effective. An alternative option we propose is to build an iterative framework in which the kinematic feasibility is adaptively adjusted when it is found to be inappropriate. The details on how to find an effective computational solution to TOPP with nonlinear constraints are described in Section 3.

3. TOPP by Iterative Optimization (TOPP-IO)

This section introduces our proposed Cartesian-based TOPP-IO method. First, we present the initial guess and control group generated by the Cartesian-based TOPP-RA method, followed by an explanation of the principle of the TOPP-IO method.

3.1. Cartesian-Based TOPP-RA Method

Combining with [6], we expanded the TOPP-RA method from joint space to Cartesian space, which we call the Cartesian-based TOPP-RA method. The geometric path in Cartesian space, denoted by $\mathbf{p}(s)$, is divided into N segments with $N+1$ grid points, where $(s_i, \dot{s}_i, \ddot{s}_i, \dddot{s}_i)$ represents the i-th stage state and control profiles, with $i \in [0, 1, \ldots, N]$. The constraints of joint acceleration can be formulated as follows, by taking into account (12) and (13):

$$\mathbf{B}\dot{s}^2 + \mathbf{C}\ddot{s} \leq \ddot{\mathbf{Q}} \tag{16}$$

where $\mathbf{B} = \begin{bmatrix} \mathbf{b}(s) \\ -\mathbf{b}(s) \end{bmatrix}, \mathbf{C} = \begin{bmatrix} \mathbf{c}(s) \\ -\mathbf{c}(s) \end{bmatrix}$ and $\ddot{\mathbf{Q}} = \begin{bmatrix} \ddot{\mathbf{q}}_{max} \\ -\ddot{\mathbf{q}}_{min} \end{bmatrix}$. The velocity constraints of the joints are expressed as a range of i-stage state variables, $\mathfrak{X}_i = [(\dot{s}_i^2)^{lower}, (\dot{s}_i^2)^{upper}]$, which reflects the allowable velocity of the joints.

$$\begin{aligned} (\dot{s}_i^2)^{lower} &= \max_j \left\{ \frac{\dot{q}_{min,j}}{a_j(s_i)} \mid a_j(s_i) > 0 \quad \text{or} \quad \frac{\dot{q}_{max,j}}{a_j(s_i)} \mid a_j(s_i) < 0 \right\}, \\ (\dot{s}_i^2)^{upper} &= \min_j \left\{ \frac{\dot{q}_{max,j}}{a_j(s_i)} \mid a_j(s_i) > 0 \quad \text{or} \quad \frac{\dot{q}_{min,j}}{a_j(s_i)} \mid a_j(s_i) < 0 \right\}. \end{aligned} \tag{17}$$

where j is the j-th element of $\mathbf{a}(s_i)$, $\dot{\mathbf{q}}_{min}$ and $\dot{\mathbf{q}}_{max}$. The state-update function for constant acceleration over $[s_i, s_{i+1}]$ is given by:

$$\dot{s}_{i+1}^2 = \dot{s}_i^2 + 2\Delta_i \ddot{s}_i \tag{18}$$

where $\Delta = s_{i+1} - s_i$.

3.1.1. Backward Pass

In considering the segment $[s_i, s_{i+1}]$ and assuming that the i+1-th feasible range, $\mathfrak{S}_{i+1}$, is known, the i-th feasible range, $\mathfrak{S}_i = [(\dot{s}_i^2)^-, (\dot{s}_i^2)^+]$, can be calculated using the following formula:

$$\begin{aligned} &(\dot{s}_i^2)^- := \min \dot{s}_i^2, \\ &(\dot{s}_i^2)^+ := \max \dot{s}_i^2, \\ &s.t. \quad \dot{s}_i^2 \in \mathfrak{X}_i, \\ &\qquad \dot{s}_i^2 + 2\Delta_i \ddot{s}_i \in \mathfrak{S}_{i+1}, \\ &\qquad \mathbf{B}\dot{s}_i^2 + \mathbf{C}\ddot{s}_i \leq \ddot{\mathbf{Q}}. \end{aligned} \tag{19}$$

Obviously, Formula (19) indicates that for any $\dot{s}_i^2 \in \mathfrak{S}_i$, there always exists a state $\dot{s}_{i+1}^2 \in \mathfrak{S}_{i+1}$ that corresponds to it. In other words, we can always move from the feasible range $\mathfrak{S}_i$ to $\mathfrak{S}_{i+1}$ using the state-update function. By applying Formula (19) recursively, we can obtain a set of transitive feasible ranges, $[\mathfrak{S}_0, \mathfrak{S}_1, \ldots, \mathfrak{S}_n]$. Any state that belongs to the transitive feasible ranges can be transferred to the ending state when the last feasible range set is determined.

3.1.2. Forward Pass

By transferring $\dot{s}_i^2 \in \mathfrak{S}_i$ from step i to $\dot{s}_{i+1}^2 \in \mathfrak{S}_{i+1}$ of step $i+1$, we can recursively reach the final state $\mathfrak{S}_n$. Furthermore, literature [6] has demonstrated that the transition process occurs on a convex polygon. Therefore, selecting control variables that can reach the upper limit of the next $\mathfrak{S}$ will result in the shortest task time. This selection exhibits locally greedy behavior while globally optimizing performance. Once the transitive feasible ranges have been derived from the backward pass, the method for transferring $(\dot{s}_i^2)^*$ to $(\dot{s}_{i+1}^2)^*$ using a greedy algorithm is as follows:

$$\begin{aligned} (\dot{s}_{i+1}^2)^* &:= \max(\dot{s}_i^2)^* + 2\Delta_i \ddot{s}_i, \\ s.t. \quad & (\dot{s}_i^2)^* + 2\Delta_i \ddot{s}_i \in \mathfrak{S}_{i+1}, \\ & \mathbf{B}(\dot{s}_i^2)^* + \mathbf{C}\ddot{s}_i \leq \ddot{\mathbf{Q}}. \end{aligned} \tag{20}$$

where $(\dot{s}_i^2)^*$ denotes the optimal solution at the i-th grid point. By setting deterministic values of $(\dot{s}_0^2)^* \in \mathfrak{S}_0$ and $\mathfrak{S}_n = \{(\dot{s}_n^2)^*\}$, the solution of Cartesian-based TOPP-RA, $[(\dot{s}_0^2)^*, (\dot{s}_1^2)^*, \ldots, (\dot{s}_n^2)^*]$, is obtained by recursively applying Formula (20).

3.2. Principle of the Proposed TOPP-IO Method

The general principle of the TOPP iterative optimization method is illustrated by the pseudo-codes in Algorithm 1. Given a path $\boldsymbol{\mathcal{P}}$ in Cartesian space, Algorithm 1 first generates an initial conjecture using the ToppraGuess() function to numerically solve (15) without joint jerk limits. This initial conjecture includes the path discretization, all the parameters required to solve (15), and the initial values of all the decision variables. Then, using the full content of this initial conjecture, Algorithm 1 establishes an iterative OCP where an intermediate optimal solution is obtained from each iteration. After the first three lines of initialization, the while loop is applied to iteratively solve the TOPP$_{OCP}$. Similar to (15), the only difference is that we add (4) as a soft constraint to the objective function. Specifically, this iterative OCP solves the following optimization problems.

$$\begin{aligned} &\min\ (3) + \omega_{soft} \cdot f_{soft}(s) \\ &s.t.\ \text{States/Control profiles constraints (12) and (13);} \\ &\qquad \text{Two-point boundary constraints (14).} \end{aligned} \tag{21}$$

where $\omega_{sift} > 0$ is a parameter used to weight the softening of the state updating, and $f_{soft}(s)$ is denoted as

$$f_{soft}(s) = \int_{s=0}^{s_{end}} \left\| \begin{bmatrix} \dot{s} \\ \ddot{s} \end{bmatrix} - f_{Status-update}(s) \right\|^2 \mathrm{d}s \tag{22}$$

In each iteration of the while loop, the function SolveIteratively(OCP$_{TOPP}$, $\boldsymbol{\mathcal{G}}$) is used to solve (21) using the initial conjecture $\boldsymbol{\mathcal{G}}$. The function StateUpdateInfeasibility($\boldsymbol{\mathcal{G}}$) evaluates the infeasibility degree of the status update determined by $f_{soft}(s)$ as given in (22). When $f_{soft}(s)$ becomes small enough, i.e., close to 0^+, the function GetTrajectoryInformation($\boldsymbol{\mathcal{G}}$) is called to extract information about the optimal trajectory from the solution $\boldsymbol{\mathcal{G}}$.

Algorithm 1: An Iterative Optimal Method for TOPP

```
   Input: Geometric path in Cartesian space $\mathcal{P}$
   Output: Optimal trajectory information $\mathbf{info}_{opti}$
1  $\mathcal{G}$ = ToppraGuess($\mathcal{P}$) ;
2  $\omega_{soft} \leftarrow \omega_{soft0}, iter \leftarrow 0, \mathbf{info}_{opti} \leftarrow \varnothing$ ;
3  while $iter < iter_{max}$ do
4      $\mathrm{OCP}_{TOPP} \leftarrow$ BuildIterativeOCP($\mathcal{G}$) ;
5      $\mathcal{G} \leftarrow$ SolveIteratively($\mathrm{OCP}_{TOPP}, \mathcal{G}$) ;
6      $f_{soft}(s) \leftarrow$ StateUpdateInfeasibility($\mathcal{G}$) ;
7      if $f_{soft}(s) < \mathfrak{e}_{soft}$ then
8          $\mathbf{info}_{opti} \leftarrow$ GetTrajectoryInformation($\mathcal{G}$) ;
9          return;
10     else
11         $\omega_{soft} \leftarrow \omega_{soft} \cdot \alpha, iter \leftarrow iter + 1$ ;
12     end
13 end
14 return;
```

3.3. Properties Discussion of Algorithm 1

This subsection describes the relevant properties of the proposed TOPP-IO method in Algorithm 1.

First, the iterative process progressively increases the feasibility and optimality of the phase state. It is assumed that the initial solution obtained by the Cartesian frame TOPP-RA does not satisfy the jerk constraint and, hence, is not status-update feasible. In such cases, restoring status-update feasibility becomes the primary goal of minimizing the objective function of OCP_{TOPP}. Therefore, the optimal solution differs from the initial guess by reducing the status-update infeasibility. Although the status-update infeasibility may not be eliminated, the resulting $(s, \dot{s})$ phase diagram is closer to being feasible, providing opportunities for further improvement in succeeding iterations.

Second, optimality is achieved when Algorithm 1 exits from line 9. As the iteration continues, the status-update infeasibility approaches 0^+ and incrementing ω_{soft} expedites the procedure. When the degree of status-update infeasibility is small, the total time (3) in the objective function of (21) dominates. Thus, the objective function of (21) is minimized, closing in on minimizing the original objective function (3) to an accuracy level of $\mathfrak{e}_{soft}$.

Third, the OCP_{TOPP} is always feasible, which is a crucial cornerstone of the entire iterative framework. With strict restrictions on CPU runtime and a willingness to accept suboptimal solutions, a feasible solution can be obtained at any point by interrupting the iterative optimization process. With very slow motion always feasible, the solution procedure for each (21) is consistently in the feasible region of the solution space when the initial solution is set to $\mathbf{0}$. Thus, as long as the obtained $(s, \dot{s})$ phase diagram's near-future period is status-update feasible, the resulting phase states can be transferred to the next iterative OCP_{TOPP} for further enhancements.

4. Simulation and Real-World Experiment Results

In this section, simulation experiments will be used to demonstrate the feasibility, performance, and generality of the proposed method, as well as an industrial robot real machine-verification experiment will be performed, which gives practical significance to the TOPP-IO algorithm. The proposed method is executed on Ubuntu using an Intel i7-7700HQ @ 2.80 GHz CPU and 16-GB RAM, and all optimization problems are solved using CasADi (CasAdi is an open-source software framework for nonlinear optimization and optimal control. It provides a flexible and efficient interface for constructing and solving various optimization problems, including trajectory optimization) [34]. We use the 6-DOF Firefox robot from SIASUN in both the simulation and the real world, in addition to the Pioneer P3-DX robot used in the simulation. The implementation of TOPP-IO was done in C++, and the required communication between systems for these experiments was established. Figure 1 illustrates the architecture of the implementation.

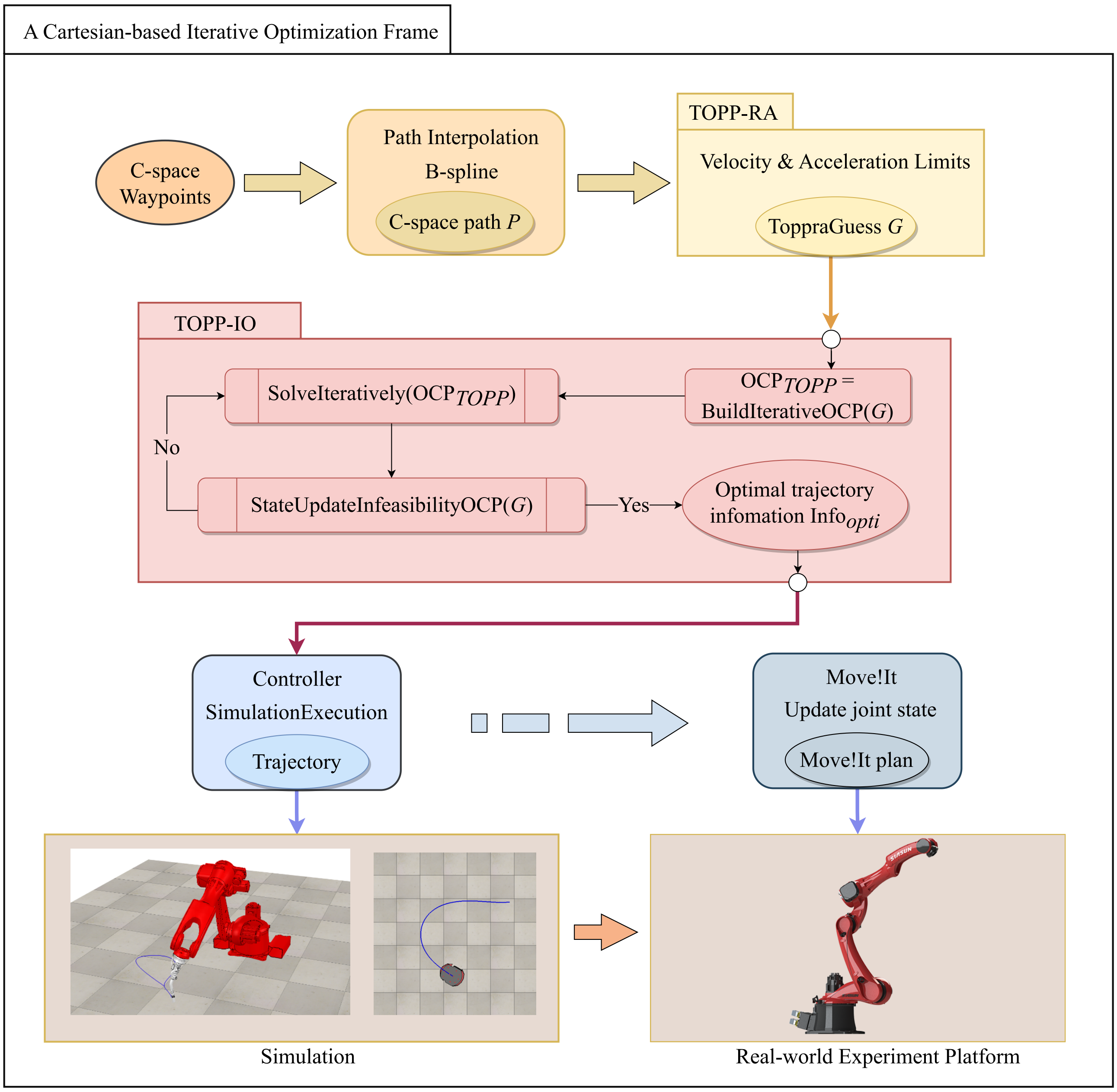

A Cartesian-based Iterative Optimization Frame
C-space Waypoints
Path Interpolation
B-spline
C-space path P
TOPP-RA
Velocity & Acceleration Limits
ToppraGuess G
TOPP-IO
SolveIteratively(OCP_TOPP)
OCP_TOPP = BuildIterativeOCP(G)
No
StateUpdateInfeasibilityOCP(G)
Yes
Optimal trajectory infomation Info_opti
Controller
SimulationExecution
Trajectory
Move!It
Update joint state
Move!It plan
Simulation
Real-world Experiment Platform

Figure 1. Architecture of implementation.

4.1. Experiment Settings

The joint and wheel velocity, acceleration, and jerk limits are presented in each experiment, respectively, which are critical factors for the safe and efficient operation of robotic systems. To assess the robustness and adaptability of our proposed algorithm, TOPP-IO, we conducted a series of experiments with varying jerk limits. Specifically, we evaluated the performance of TOPP-IO under four different jerk limits: $0.1\times$, $1\times$, $10\times$, and $100\times$ the default value. The basic parameters for the iterative optimization are carefully selected to ensure the convergence and efficiency of the optimization process, which are listed in Table 2.

Table 2. Hyperparameter setting for iterative optimization.

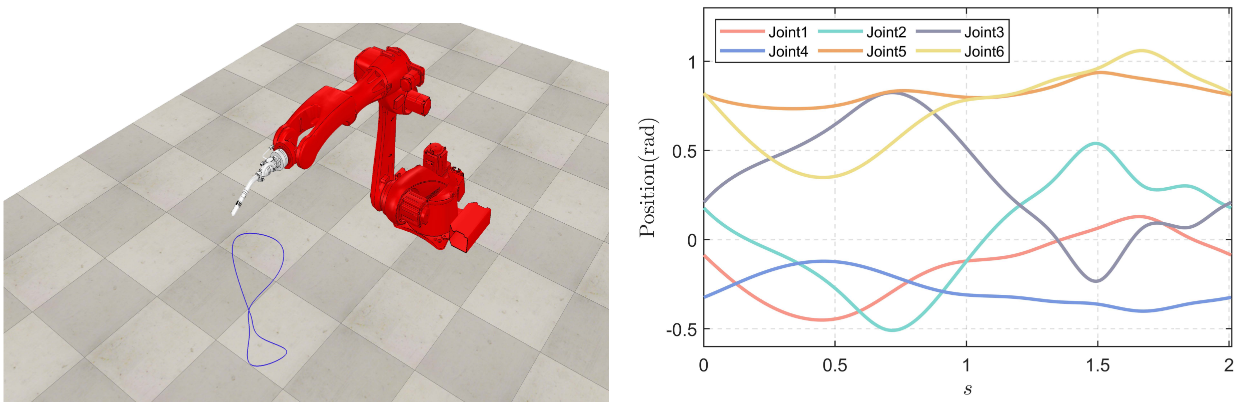
Hyperparameter	Description	Value
$iter_{max}$	Maximum iteration number	5
ω_{soft0}	ω_{soft} initial value	10^5
α	Multiplier to enlarge ω_{soft}	10
ϵ_{soft}	Softened constraints tolerance	16

4.2. Comparison with TOPP-RA Method

This method is built and tested on the random geometric route depicted in Figure 2, subject to joint velocity, acceleration, and jerk limitations which are presented in Table 3. The simulation results are compared with those obtained from the CO algorithm (TOPP-RA) presented in [6] to demonstrate the effectiveness of the proposed strategy in controlling the acceleration surge caused by ignoring the jerk constraints.

Table 3. Velocity, acceleration, and jerk limits of joints.

Limits	Joint1	Joint2	Joint3	Joint4	Joint5	Joint6
Vel. (rad/s)	2	2	2	4	4	4
Acc. (rad/s^2)	5	6	6	12	12	12
Jerk (rad/s^3)	16	16	18	20	28	28



(**a**) Cartesian space (**b**) Joint space

Figure 2. The geometric path on which this approach is implemented and tested.

The results of the two approaches, TOPP-RA and TOPP-IO, in the $(s, \dot{s})$ and $(s, \ddot{s})$ phase planes are presented in Figures 3 and 4, respectively. It can be observed from Figure 4 that TOPP-RA allows for steep slopes of acceleration due to the lack of restriction on jerk, leading to an abrupt shift in acceleration between neighboring path points. This sudden change in acceleration can be seen in the velocity curve of Figure 3, where there is no smooth transition between the acceleration and deceleration portions. Such abrupt changes in acceleration can result in jerky and unstable motion, which is not desirable in many real-world applications. To address this issue, TOPP-IO imposes explicit joint jerk limitations, leading to smoother acceleration profiles between neighboring path points. Figure 4 shows that the TOPP-IO method successfully restricts the acceleration mutation, preventing any abrupt changes in acceleration. Furthermore, the velocity curve of TOPP-IO in Figure 3 exhibits smoother transitions between the portions representing acceleration and deceleration, guaranteeing that the nearby segments will not violate the imposed restrictions.

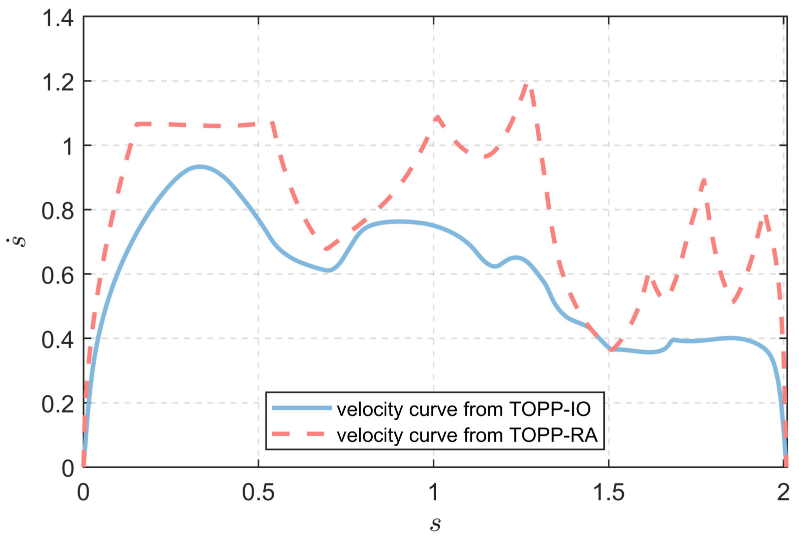


Figure 3. Comparison of the TOPP-RA resultant velocity curve without jerk limitations (red dashed line) and the one obtained from the proposed method with jerk limits (blue solid line).

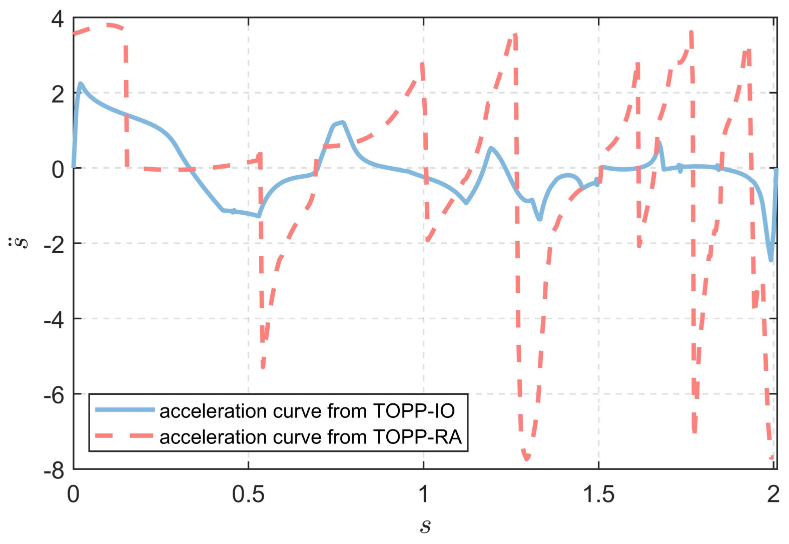


Figure 4. Comparison of the TOPP-RA resultant acceleration curve without jerk limitations (red dashed line) and the one obtained from the proposed method with jerk limits (blue solid line).

To further evaluate the performance of the two approaches, we compare their execution times in Table 4 and display the corresponding speed, acceleration, and jerk curves in Figure 5 for various jerk limits (100×, 10×, 1×, and 0.1×). In the TOPP-RA method, it is evident that the acceleration profiles are bang-bang, satisfying all joint second-order constraints.

With all third-order kinematic constraints, the jerk profiles are bang-bang in the TOPP-IO method, leading to smoother transitions between the portions representing acceleration and deceleration. Without joint jerk limits, the maximum acceleration is about 1638.4 rad/s^3. As the jerk limit is decreased from "none" to 100× and 10× jerk limits, the execution time only slightly increases from 2.81067 s to 2.89393 s and 2.90326 s, respectively, and the smoothing effect of the speed profile is not immediately noticeable. The speed profile becomes smoother as the jerk limits approach 1× jerk limits. Notably, even when the jerk limit is set to 0.1× jerk limits, TOPP-IO can still produce a valid solution, albeit with an increased execution time.

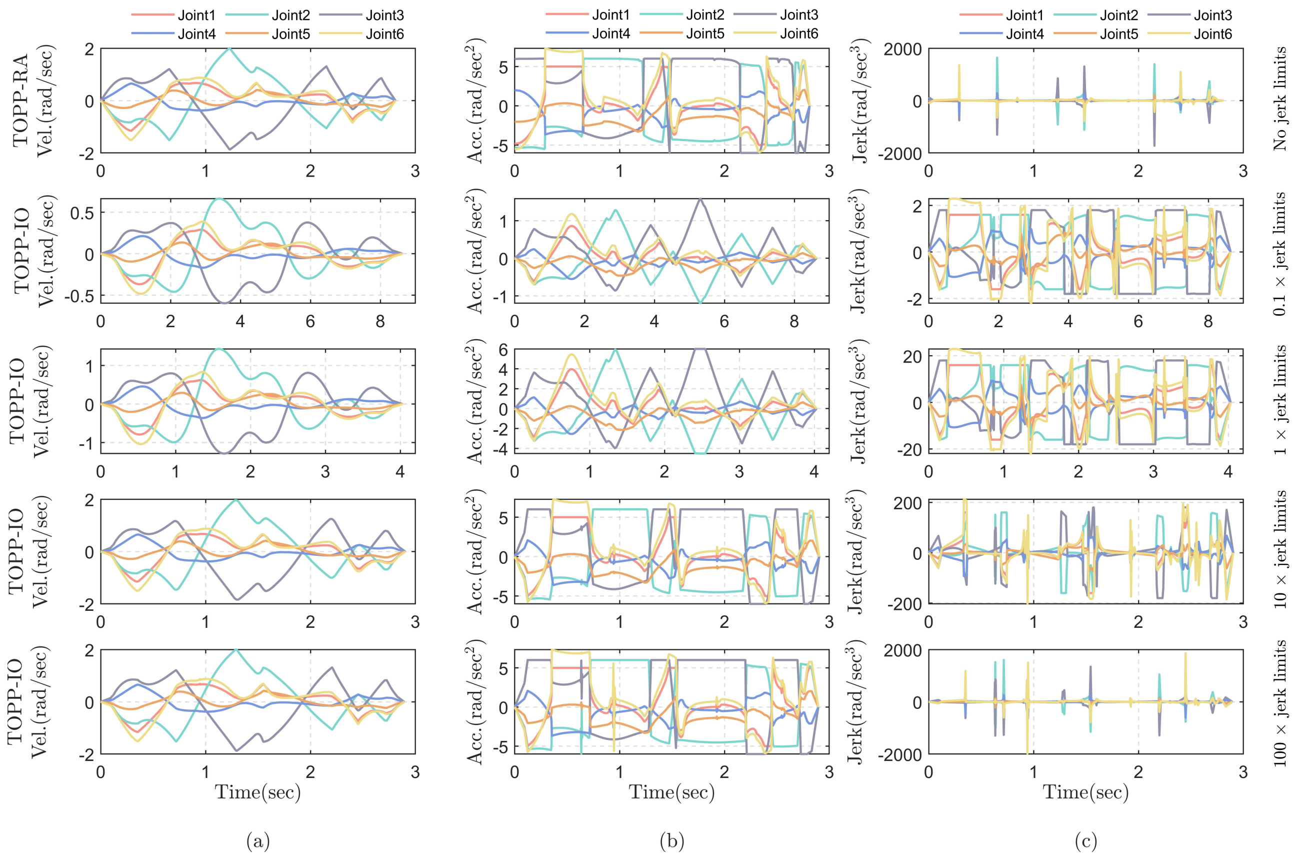


Figure 5. Velocity, acceleration, and jerk profiles for various methods and jerk restrictions. (**a**) Speed, (**b**) Acceleration, and (**c**) Jerk.

Table 4. Execution time of different trajectory planning algorithms and jerk restrictions.

Method	**TOPP-RA**	**TOPP-IO**			
Jerk Limits (rad/s^3)	-	**100×**	**10×**	**1×**	**0.1×**
t$_e$ (s)	2.81067	2.89393	2.90326	4.02941	8.63447

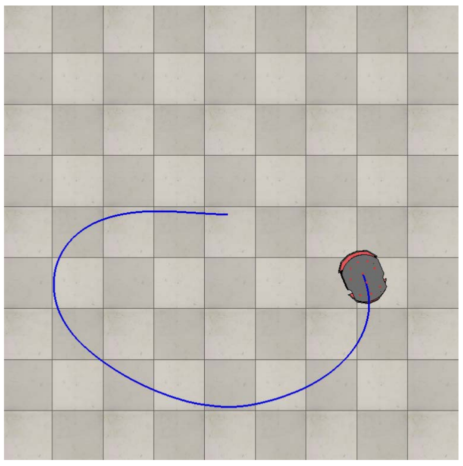
4.3. Application on Mobile Robot

Our method applies not only to manipulators but also to a wide range of robots. To demonstrate its flexibility, we computed a ground trajectory for the Pioneer P3-DX, a diff-drive mobile robot. The wheel velocity, acceleration, and jerk limitations are presented in Table 5. Screenshots of the operational phase as well as the wheel speed curve in comparison to TOPP-RA are shown in Figure 6.

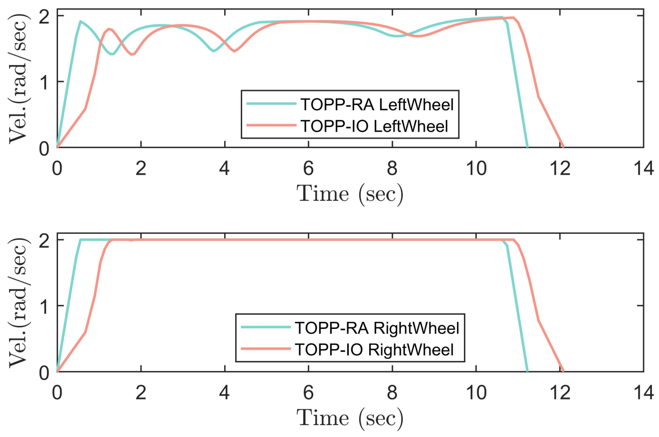
Table 5. Velocity, acceleration, and jerk limits of wheels.

Limits	**Wheel1**	**Wheel2**
Vel. (rad/s)	2	2
Acc. (rad/s^2)	4	4
Jerk (rad/s^3)	8	8

The restrictions on wheel jerk and route jerk constraints have similar effects on controlling acceleration mutation. In this study, jerk restrictions were defined as wheel jerk constraints that effectively limit acceleration mutation in the route. The robot trajectories obtained from TOPP-RA and the proposed method are presented in Figure 7. Table 6 indicates that the maximum absolute values of the robot's acceleration and jerk curves obtained from the proposed method are reduced by 60.28% and 69.82%, respectively, compared to those from TOPP-RA.



(**a**) Cartesian space

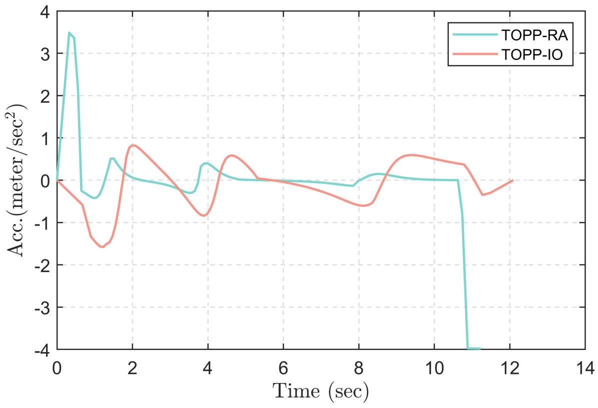


(**b**) Wheel space

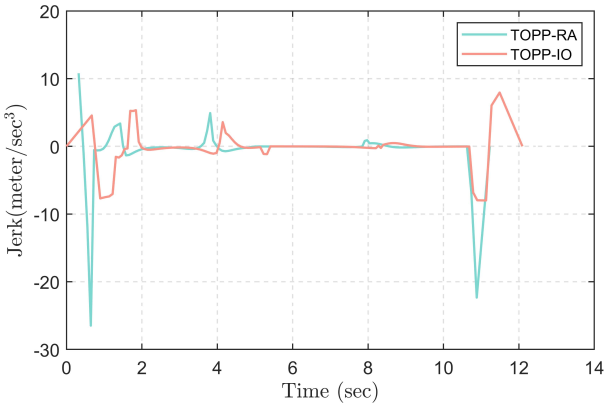
Figure 6. The ground path on which this approach is implemented and tested.

Table 6. Comparing the maximum absolute value of the robot's acceleration and jerk curves between the two approaches.

	Acceleration (m/s^2)	**Jerk (m/s^3)**
TOPP-RA	3.98086	26.4858
TOPP-IO	1.58118	7.9937
Degree of decline	60.28%	69.82%



(**a**) Mobile robot acceleration

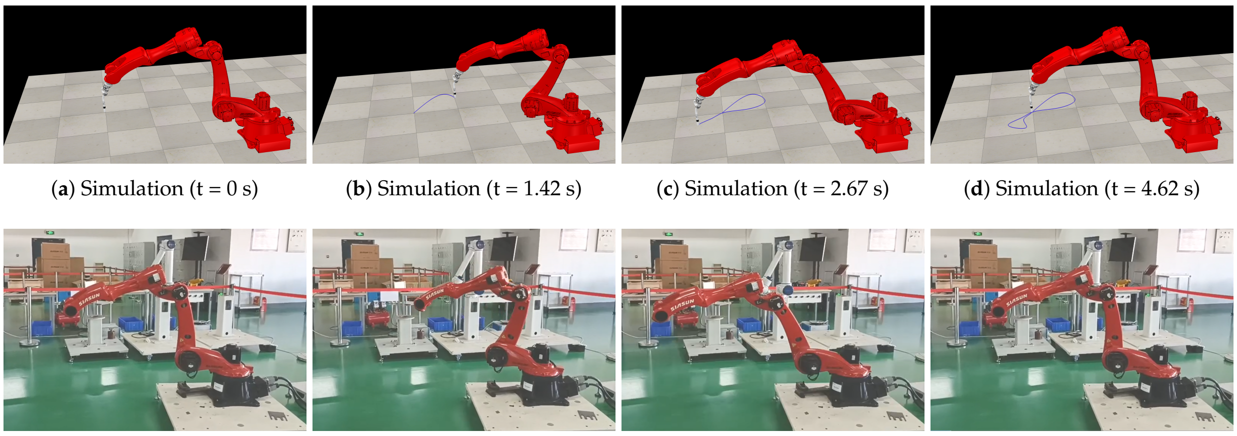


(**b**) Mobile robot jerk

Figure 7. Comparing the resulting mobile robot acceleration (**a**) and jerk (**b**) from TOPP-RA and the ones obtained from the proposed approach.

4.4. Real-World Experiments

In real-world experiments, we applied our method to the welding industry where the objective is to complete tasks as quickly, safely, and efficiently as possible. TOPP-IO succeeded in executing the assignment in a timely, safe, and stable manner. The running state of the Firefox robot in the actual world is shown in Figure 8 and is consistent with the simulation results.



(**a**) Simulation (t = 0 s) (**b**) Simulation (t = 1.42 s) (**c**) Simulation (t = 2.67 s) (**d**) Simulation (t = 4.62 s)

(**e**) Real Experiment (t = 0 s) (**f**) Real Experiment (t = 1.42 s) (**g**) Real Experiment (t = 2.67 s) (**h**) Real Experiment (t = 4.62 s)

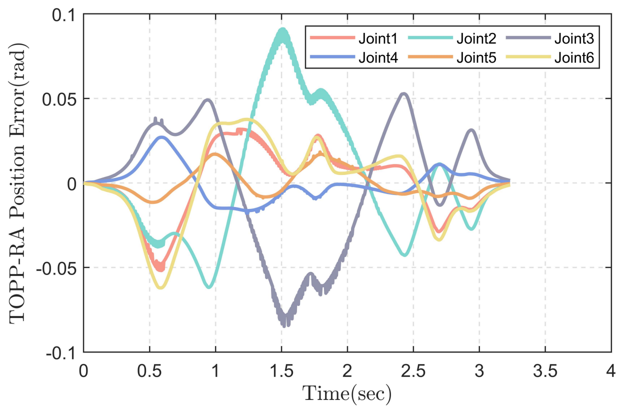
Figure 8. Real-world experiments (**e**–**h**) in accordance with the simulation (**a**–**d**).

We performed both quantitative and qualitative analyses of our method's performance during the actual operation process. Specifically, we analyzed the position error of each joint and examined the speed-tracking situation using joint1 as an example. Figure 9a shows the joint position error of the TOPP-RA method during actual operation, while Figure 9b displays the joint position error of the TOPP-IO method under the same path. In addition, Table 7 compares the performance of our TOPP-IO method with that of the TOPP-RA method. The results show that the average and maximum position errors of all joints in TOPP-IO have been reduced to different degrees during operation. The absolute values of the average position error and maximum position error have been reduced by about 29% and 27%, respectively, compared to the TOPP-RA method.

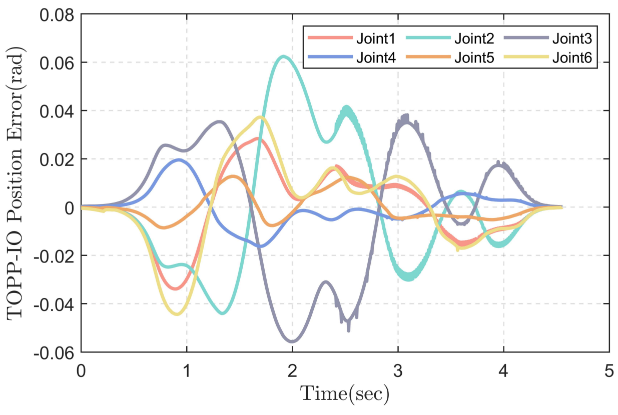
Table 7. The absolute values of the average and maximum joint position error on different trajectory planning algorithms.

		Joint1	**Joint2**	**Joint3**	**Joint4**	**Joint5**	**Joint6**
Average position error	TOPP-RA (rad)	0.0160	0.0296	0.0293	0.0075	0.0066	0.0192
	TOPP-IO (rad)	0.0112	0.0205	0.0206	0.0053	0.0047	0.0135
	Degree of decline	30.13%	30.67%	29.81%	29.60%	29.65%	29.46%
Maximum position error	TOPP-RA (rad)	0.0518	0.0911	0.0849	0.0270	0.0183	0.0621
	TOPP-IO (rad)	0.0339	0.0624	0.0557	0.0196	0.0127	0.0444
	Degree of decline	34.63%	31.54%	34.36%	27.62%	30.36%	28.47%

To examine the speed-tracking situation, we used joint1 as an example. Figure 10a shows the speed-tracking of the TOPP-RA method during actual operation, while Figure 10b presents the speed curve of our TOPP-IO method, which considers the third-order constraint. Only the second-order constraint of joint space is considered, which leads to snap-point (represented by the gray circle), or the sudden change in joint acceleration, resulting in the inability to track the given speed on the actual physical robot. As shown in the zoomed-in section of Figure 10a, the snap-point causes fluctuations in the speed curve. In contrast, our TOPP-IO method eliminates the snap-point and enables smooth tracking of joint speed. Our method ensures a smooth trajectory and efficient, steady completion of the task while maintaining high speed.

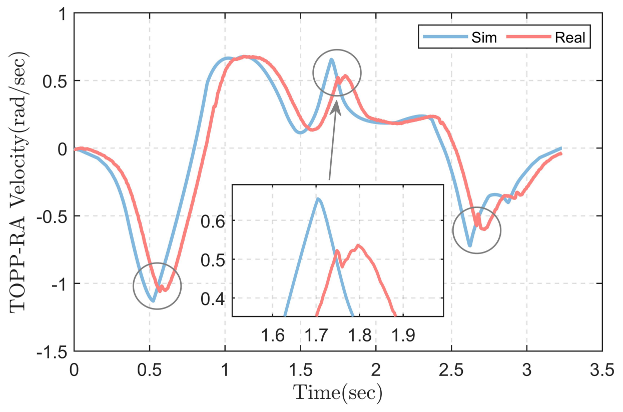


(**a**) TOPP-RA joint position error

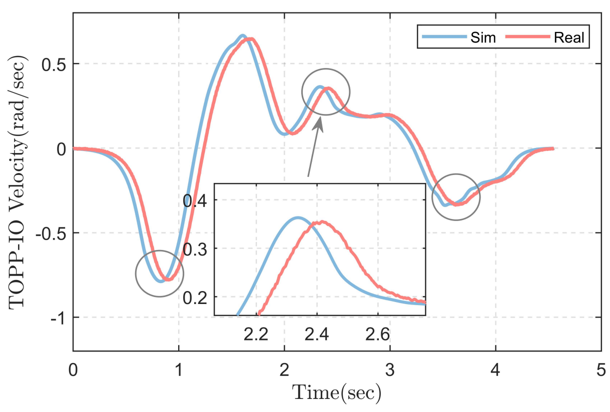


(**b**) TOPP-IO joint position error

Figure 9. Comparing the resulting joint position error from TOPP-RA (**a**) and the ones obtained from proposed approach (**b**).



(**a**) TOPP-RA velocity curve



(**b**) TOPP-IO velocity curve

Figure 10. Comparing the resulting joint1 velocity curve from TOPP-RA (**a**) and the ones obtained from proposed approach (**b**). (The gray circle represented the snap-point).

5. Conclusions

In this paper, we develop a comprehensive and efficient iterative optimization framework for solving the TOTP problem with joint third-order constraints. The main contributions and results of this paper are as follows:

- The framework is constructed from the bottom up in the Cartesian coordinate system and can be applied to both manipulator and mobile robots;
- Our study has identified two main challenges in the framework: how to consistently represent the TOTP problem in the Cartesian space using the $(s, \dot{s})$ phase plane, while imposing third-order kinematic constraints on each joint, and how to devise an efficient computational solution strategy that uses a constraint relaxation approach to simplify nonconvex constraints without violating them;
- We demonstrated the effectiveness of our proposed framework through both simulation and physical experiments. Compared to the TOPP-RA method, our approach effectively reduced the maximum absolute values of the robot's jerk and the average absolute values of the position error over 60% and 29%, respectively. These are critical factors in ensuring smooth robotic velocity tracking and reducing impact during operation.

Our framework has a few limitations. First, we assume that the path of the end-effector in Cartesian space is predetermined. We use B-spline interpolation to generate continuous, smooth end-effector poses from the given path points. Second, our approach accepts suboptimal solutions when a feasible solution can be obtained at any point by interrupting the iterative optimization process.

Our future work can be divided into two main areas:

- First, we aim to extend our framework to handle both path planning and speed planning simultaneously, which will enable our method to generate feasible solutions more efficiently;
- Second, we plan to explore the potential of the constraint relaxation approaches and achieve real-time performance. Moreover, handling dynamic environments is a challenging and interesting area for future research.

Author Contributions: Conceptualization, Z.F. and Y.C.; methodology, Z.F.; software, Z.F.; validation, Z.F., F.Z., Q.Z. and K.J.; formal analysis, L.Z. and Z.D.; investigation, Z.F.; resources, Z.D.; data curation, M.L.; writing—original draft preparation, Z.F.; writing—review and editing, Z.F. and M.L.; visualization, Z.F. and Y.C.; supervision, K.J.; project administration, L.Z.; funding acquisition, F.Z. and Q.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (No. 2020YFB1710905) and the National Natural Science Foundation of China (No. 61903162).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TOTP	time-optimal trajectory planning
TOPP	time-optimal path parameterization
NI	Numerical Integration
CO	Convex Optimization
DP	Dynamic Programming
DC	difference of convex
SCP	sequential convex programming
TOPP-RA	TOPP approach based on reachability analysis
TOPP-IO	TOPP approach based on iterative optimization
LP	linear programming
BA	bisection algorithm
NLP	nonlinear programming

Appendix A. From Configuration Space to Cartesian Space

In this appendix, we introduce how to determine the position and attitude of the end-effector and their derivatives using the robot joint variables and their derivatives of each order, and then convert them into path parameter s for representation. This process involves forward kinematics, inverse kinematics and the derivatives of Jacobian matrix, which will be described in detail in the following subsections.

Appendix A.1. Forward and Inverse Kinematics

$\boldsymbol{\psi}$ is defined as the screw coordinate of the spiral axis relative to the spatial coordinate system. Its Lie algebra representation is as follows:

$$se(3) = \left\{ \boldsymbol{\psi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix} \in \mathbb{R}^6, \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in so(3), \boldsymbol{\psi}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \right\} \tag{A1}$$

The Lie group corresponding to $\boldsymbol{\psi}$ can be expressed as

$$SE(3) = \left\{ \boldsymbol{\Psi} = \begin{bmatrix} \mathbf{R} & \boldsymbol{\varrho} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}, \mathbf{R} \in SO(3), \boldsymbol{\varrho} \in \mathbb{R}^3 \right\} \tag{A2}$$

where $\mathbf{R}$ and $\boldsymbol{\varrho}$ separately denote the directions and positions of the rigid-body. The mapping between the Lie algebra $\boldsymbol{\psi}$ and the corresponding Lie group $\boldsymbol{\Psi}$ is given by:

$$\boldsymbol{\Psi} = \exp(\boldsymbol{\psi}^\wedge) = \begin{bmatrix} \exp(\boldsymbol{\phi}^\wedge) & \frac{(\mathbf{I}-\exp(\boldsymbol{\phi}^\wedge))\boldsymbol{\phi}^\wedge\boldsymbol{\rho}+\boldsymbol{\phi}\boldsymbol{\phi}^T\boldsymbol{\rho}}{\|\boldsymbol{\phi}\|^2} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{A3}$$

Consider a robot system with n degrees of freedom, whose configuration is represented by $\mathbf{q} = [q_1, q_2, \cdots, q_n]^T \in \mathbb{R}^n$. The forward kinematics model of the robot is determined as follows:

$$\boldsymbol{\Psi}_{end}(\mathbf{q}) = \prod_{i=1}^{n} \exp(\boldsymbol{\psi}_i^\wedge q_i)\boldsymbol{\Psi}_S = \begin{bmatrix} \mathbf{R}_{end} & \boldsymbol{\varrho}_{end} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{A4}$$

where $\boldsymbol{\Psi}_S$ represents the pose matrix of the end coordinate system relative to the space coordinate system when the robot is in the initial position; $(q_i, \boldsymbol{\psi}_i)$ denote the joint position and twist, respectively, of the i-th joint; $\mathbf{R}_{end}$ and $\boldsymbol{\varrho}_{end}$ separately denote the orientations and positions of the end-effector.

For a given robot model, it is possible to convert the end pose $\mathbf{p}$ expressed in Cartesian space to the corresponding joint values $\mathbf{q}$ in joint space using inverse kinematics. Similarly, the joint values $\mathbf{q}$ can be converted to the end pose $\mathbf{p}$ through forward kinematics. Thus, the reciprocal transformation between $\mathbf{p}$ and $\mathbf{q}$ can be achieved through these two transformations.

Appendix A.2. Explicit Expressions of High-Order Jacobian Derivatives

Jacobian matrix in Formula (13) is as follows:

$$\mathbf{J} = \mathbf{J}_p \cdot \mathbf{J}_b = \begin{bmatrix} \mathbf{A}^{-1}(\mathbf{r}) & \mathbf{O} \\ \mathbf{O} & \mathbf{R}_{sb}(\mathbf{r}) \end{bmatrix} \cdot [\boldsymbol{\beta}_1, \quad \boldsymbol{\beta}_2, \quad \ldots, \quad \boldsymbol{\beta}_n] \tag{A5}$$

where $\mathbf{J}_b(\mathbf{q}) \in \mathbb{R}^{6\times n}$ represents the geometric Jacobian matrix; $\mathbf{r} \in \mathbb{R}^3$ represents the exponential coordinate of the axis angle, which is reflected in the last three lines of the $\mathbf{p}(s)$. In $\mathbf{J}_p \in \mathbb{R}^{6\times 6}$, $\mathbf{A}(\mathbf{r}) \in \mathbb{R}^{3\times 3}$ and $\mathbf{R}_{sb}(\mathbf{r}) \in \mathbb{R}^{3\times 3}$ can be expressed as:

$$\begin{aligned} \mathbf{A}(\mathbf{r}) &= \mathbf{I} - \frac{1-\cos\|\mathbf{r}\|}{\|\mathbf{r}\|^2}\mathbf{r}^\wedge + \frac{\|\mathbf{r}\| - \sin\|\mathbf{r}\|}{\|\mathbf{r}\|^3}(\mathbf{r}^\wedge)^2, \\ \mathbf{R}_{sb}(\mathbf{r}) &= \exp(\mathbf{r}^\wedge). \end{aligned} \tag{A6}$$

Since $\mathbf{r}$ can be expressed by $\mathbf{p}(s)$, the first-order and second-order path parameter derivatives of $\mathbf{J}_p$ are expressed as follows:

$$\begin{aligned} \mathbf{J}_p'(\mathbf{p}(s)) &= \begin{bmatrix} -\mathbf{A}^{-1}(\mathbf{p}(s))\dfrac{\mathrm{d}\mathbf{A}(\mathbf{p}(s))}{\mathrm{d}s}\mathbf{A}^{-1}(\mathbf{p}(s)) & \mathbf{0} \\ \mathbf{0}^T & \dfrac{\mathrm{d}\mathbf{R}_{sb}}{\mathrm{d}s} \end{bmatrix}, \\ \mathbf{J}_p''(\mathbf{p}(s)) &= \begin{bmatrix} 2\mathbf{A}^{-1}\dfrac{\mathrm{d}\mathbf{A}}{\mathrm{d}s}\mathbf{A}^{-1}\dfrac{\mathrm{d}\mathbf{A}}{\mathrm{d}s}\mathbf{A}^{-1} - \mathbf{A}^{-1}\dfrac{\mathrm{d}^2\mathbf{A}}{\mathrm{d}s^2}\mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0}^T & \dfrac{\mathrm{d}^2\mathbf{R}_{sb}}{\mathrm{d}s^2} \end{bmatrix}. \end{aligned} \tag{A7}$$

(1) First-order path parameter derivative $\mathbf{J}'$: The first-order derivative of Jacobian matrix $\mathbf{J}$ with respect to the path parameter s is as following:

$$\mathbf{J} = \mathbf{J}_p' \cdot \mathbf{J}_b + \mathbf{J}_p \cdot \mathbf{J}_b' \tag{A8}$$

The matrices $\mathbf{J}_p$ and $\mathbf{J}_p'$ depend on the path parameter s, while $\mathbf{J}_b$ is a function of the joint values $\mathbf{q}$, which can be obtained by forward and inverse kinematics. Each column of $\mathbf{J}_b(\mathbf{q})$ can be represented as an adjoint matrix, given by:

$$\begin{aligned} \boldsymbol{\beta}_i &= Ad^{-1}_{\boldsymbol{\Psi}_i}(\boldsymbol{\psi}_i) = (\boldsymbol{\Psi}_i \boldsymbol{\psi}_i^{\wedge} \boldsymbol{\Psi}_i^{-1})^{\vee} \in \mathbb{R}^6, \text{where} \\ \boldsymbol{\Psi}_i &= \prod_{j=1}^{n} \exp(\boldsymbol{\psi}_j^{\wedge} q_j) \boldsymbol{\Psi}_S. \end{aligned} \tag{A9}$$

According to the chain rule of differentiation, the first-order derivative of the geometric Jacobian matrix with respect to the path parameter s, denoted as $\mathbf{J}_b'$, and its i-th column, denoted as $\boldsymbol{\beta}_i'$, can be expressed as:

$$\begin{aligned} \mathbf{J}_b' &= [\boldsymbol{\beta}_1', \quad \boldsymbol{\beta}_2', \quad \dots, \quad \boldsymbol{\beta}_n'], \\ \boldsymbol{\beta}_i' &= \sum_{j=1}^{n} \frac{\partial \boldsymbol{\beta}_i}{\partial q_j} q_j' \\ &= \left[\frac{\partial \boldsymbol{\beta}_i}{\partial q_1}, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_2}, \quad \dots, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_n} \right] \mathbf{q}' \\ &= \left[\frac{\partial \boldsymbol{\beta}_i}{\partial q_1}, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_2}, \quad \dots, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_n} \right] \mathbf{J}^{-1} \mathbf{p}'. \end{aligned} \tag{A10}$$

In combination with the literature [35], $\dfrac{\partial \boldsymbol{\beta}_i}{\partial q_j}$ can be calculated using $(\boldsymbol{\beta}_i, \boldsymbol{\beta}_j)$ as follows:

$$\frac{\partial \boldsymbol{\beta}_i}{\partial q_j} = \begin{cases} \mathbf{0}, & j < i \\ ad_{\boldsymbol{\beta}_j}(\boldsymbol{\beta}_i) = (\boldsymbol{\beta}_j^{\wedge} \boldsymbol{\beta}_i^{\wedge} - \boldsymbol{\beta}_i^{\wedge} \boldsymbol{\beta}_j^{\wedge})^{\vee}, & j \geq i \end{cases} \tag{A11}$$

(2) Second-order path parameter derivative $\mathbf{J}''$: The second-order derivative of the Jacobian matrix $\mathbf{J}$ with respect to the path parameter s is given by:

$$\mathbf{J} = \mathbf{J}_p'' \cdot \mathbf{J}_b + \mathbf{J}_p' \cdot \mathbf{J}_b' + \mathbf{J}_p \cdot \mathbf{J}_b'' \tag{A12}$$

According to the chain rule, the second-order path parameter derivative, $\mathbf{J}_b''$, and its i-th column, $\boldsymbol{\beta}_i''$, can be denoted as:

$$\begin{aligned} \mathbf{J}_b'' =& [\boldsymbol{\beta}_1'', \quad \boldsymbol{\beta}_2'', \quad \dots, \quad \boldsymbol{\beta}_n''], \\ \boldsymbol{\beta}_i'' =& \frac{\partial}{\partial s} \left(\sum_{j=1}^{n} \frac{\partial \boldsymbol{\beta}_i}{\partial q_j} q_j' \right) \\ =& \sum_{j=1}^{n} \frac{\partial \boldsymbol{\beta}_i}{\partial q_j} q_j'' + \frac{\partial}{\partial s} \left(\sum_{j=1}^{n} \frac{\partial \boldsymbol{\beta}_i}{\partial q_j} \right) q_j' \\ =& \left[\frac{\partial \boldsymbol{\beta}_i}{\partial q_1}, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_2}, \quad \dots, \quad \frac{\partial \boldsymbol{\beta}_i}{\partial q_n} \right] \mathbf{q}'' \\ &+ \left[\frac{\partial}{\partial s} \left(\frac{\partial \boldsymbol{\beta}_i}{\partial q_1} \right), \quad \frac{\partial}{\partial s} \left(\frac{\partial \boldsymbol{\beta}_i}{\partial q_2} \right), \quad \dots, \quad \frac{\partial}{\partial s} \left(\frac{\partial \boldsymbol{\beta}_i}{\partial q_n} \right) \right] \mathbf{J}^{-1} \mathbf{p}'. \end{aligned} \tag{A13}$$

where

$$\mathbf{q}'' = \mathbf{J}^{-1} (\mathbf{p}'' - \mathbf{J}' \mathbf{J}^{-1} \mathbf{p}') \tag{A14}$$

and

$$\frac{\partial}{\partial s}(\frac{\partial \boldsymbol{\beta}_i}{\partial q_j}) = \begin{cases} \mathbf{0}, & j < i \\ ad_{\boldsymbol{\beta}_j}(\boldsymbol{\beta}_i') + ad_{\boldsymbol{\beta}_j'}(\boldsymbol{\beta}_i), & j \geq i \end{cases} \tag{A15}$$

Overall, this appendix establishes the forward kinematics of a robot using Lie theory and symbolically derives the first-order and second-order derivatives of the Jacobian matrix. Higher-order Jacobian matrices could be derived similarly. Furthermore, the inverse kinematics for a specific robot model can be easily obtained.

References

1. Mikolajczyk, T. Manufacturing Using Robot. *Adv. Mater. Res.* **2012**, *463*, 1643–1646. In *Proceedings of the Advanced Materials Research II*; Trans Tech Publications Ltd.: Baech, Switzerland, 2012. [CrossRef]
2. Oztemel, E.; Gursev, S. Literature review of Industry 4.0 and related technologies. *J. Intell. Manuf.* **2020**, *31*, 127–182. [CrossRef]
3. Chiurazzi, M.; Alcaide, J.O.; Diodato, A.; Menciassi, A.; Ciuti, G. Spherical Wrist Manipulator Local Planner for Redundant Tasks in Collaborative Environments. *Sensors* **2023**, *23*, 677. [CrossRef] [PubMed]
4. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Trajectory Planning in Robotics. *Math. Comput. Sci.* **2012**, *6*, 269–279. [CrossRef]
5. Zhang, T.; Zhang, M.; Zou, Y. Time-optimal and Smooth Trajectory Planning for Robot Manipulators. *Int. J. Control. Autom. Syst.* **2021**, *19*, 521–531. [CrossRef]
6. Pham, H.; Pham, Q.-C. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. *IEEE Trans. Robot.* **2018**, *34*, 645–659. [CrossRef]
7. Bobrow, J.E.; Dubowsky, S.; Gibson, J.S. Time-Optimal Control of Robotic Manipulators Along Specified Paths. *Int. J. Robot. Res.* **1985**, *4*, 3–17. [CrossRef]
8. Kunz, T.; Stilman, M. Time-optimal trajectory generation for path following with bounded acceleration and velocity. In *Robotics: Science and Systems VIII*; The MIT Press: Cambridge, MA, USA; London, UK, 2012; pp. 1–8.
9. Pham, Q.C. A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm. *IEEE Trans. Robot.* **2014**, *30*, 1533–1540. [CrossRef]
10. Pham, H.; Pham, Q.C. On the structure of the time-optimal path parameterization problem with third-order constraints. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 679–686. [CrossRef]
11. Shen, P.; Zhang, X.; Fang, Y. Essential Properties of Numerical Integration for Time-Optimal Path-Constrained Trajectory Planning. *IEEE Robot. Autom. Lett.* **2017**, *2*, 888–895. [CrossRef]
12. Shen, P.; Zhang, X.; Fang, Y. Complete and Time-Optimal Path-Constrained Trajectory Planning With Torque and Velocity Constraints: Theory and Applications. *IEEE/ASME Trans. Mechatronics* **2018**, *23*, 735–746. [CrossRef]
13. Lu, L.; Zhang, J.; Fuh, J.Y.H.; Han, J.; Wang, H. Time-optimal tool motion planning with tool-tip kinematic constraints for robotic machining of sculptured surfaces. *Robot.-Comput.-Integr. Manuf.* **2020**, *65*, 101969. [CrossRef]
14. Verscheure, D.; Demeulenäre, B.; Swevers, J.; De Schutter, J.; Diehl, M. Practical time-optimal trajectory planning for robots: A convex optimization approach. *IEEE Trans. Autom. Control.* **2008**, *53*, 1–10.
15. Xiao, Y.; Dong, W.; Du, Z. A time-optimal trajectory planning approach based on calculation cost consideration. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1845–1850. [CrossRef]
16. Debrouwere, F.; Van Loock, W.; Pipeleers, G.; Dinh, Q.T.; Diehl, M.; De Schutter, J.; Swevers, J. Time-Optimal Path Following for Robots With Convex-Concave Constraints Using Sequential Convex Programming. *IEEE Trans. Robot.* **2013**, *29*, 1485–1495. [CrossRef]
17. Nagy, Á.; Vajk, I. Sequential Time-Optimal Path-Tracking Algorithm for Robots. *IEEE Trans. Robot.* **2019**, *35*, 1253–1259. [CrossRef]
18. Ma, J.-w.; Gao, S.; Yan, H.-t.; Lv, Q.; Hu, G.-q. A new approach to time-optimal trajectory planning with torque and jerk limits for robot. *Robot. Auton. Syst.* **2021**, *140*, 103744. [CrossRef]
19. Shin, K.; McKay, N. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Trans. Autom. Control.* **1986**, *31*, 491–500. [CrossRef]
20. Kaserer, D.; Gattringer, H.; Müller, A. Nearly Optimal Path Following with Jerk and Torque Rate Limits Using Dynamic Programming. *IEEE Trans. Robot.* **2019**, *35*, 521–528. [CrossRef]
21. Kaserer, D.; Gattringer, H.; Müller, A. Time Optimal Motion Planning and Admittance Control for Cooperative Grasping. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2216–2223. [CrossRef]
22. Barnett, E.; Gosselin, C. A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths. *IEEE Trans. Robot.* **2021**, *37*, 131–145. [CrossRef]
23. Faulwasser, T.; Findeisen, R. Nonlinear Model Predictive Control for Constrained Output Path Following. *IEEE Trans. Autom. Control.* **2016**, *61*, 1026–1039. [CrossRef]
24. Consolini, L.; Locatelli, M.; Minari, A.; Piazzi, A. An optimal complexity algorithm for minimum-time velocity planning. *Syst. Control. Lett.* **2017**, *103*, 50–57. [CrossRef]

25. Steinhauser, A.; Swevers, J. An Efficient Iterative Learning Approach to Time-Optimal Path Tracking for Industrial Robots. *IEEE Trans. Ind. Inform.* **2018**, *14*, 5200–5207. [CrossRef]
26. Consolini, L.; Locatelli, M.; Minari, A. A Sequential Algorithm for Jerk Limited Speed Planning. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 3192–3209. [CrossRef]
27. Petrone, V.; Ferrentino, E.; Chiacchio, P. Time-Optimal Trajectory Planning With Interaction With the Environment. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10399–10405. [CrossRef]
28. Yang, Y.; Xu, H.z.; Li, S.h.; Zhang, L.l.; Yao, X.m. Time-optimal trajectory optimization of serial robotic manipulator with kinematic and dynamic limits based on improved particle swarm optimization. *Int. J. Adv. Manuf. Technol.* **2022**, *120*, 1253–1264. [CrossRef]
29. Singh, S.; Leu, M.C. Optimal Trajectory Generation for Robotic Manipulators Using Dynamic Programming. *J. Dyn. Syst. Meas. Control.* **1987**, *109*, 88–96. [CrossRef]
30. Slotine, J.J.E.; Yang, H.S. Improving the Efficiency of Time-Optimal Path-Following Algorithms. In Proceedings of the 1988 American Control Conference, Atlanta, GA, USA, 15–17 June 1988; pp. 2129–2134. [CrossRef]
31. Consolini, L.; Locatelli, M.; Minari, A.; Nagy, Á.; Vajk, I. Optimal Time-Complexity Speed Planning for Robot Manipulators. *IEEE Trans. Robot.* **2019**, *35*, 790–797. [CrossRef]
32. Li, B.; Ouyang, Y.; Li, L.; Zhang, Y. Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 15729–15741. [CrossRef]
33. Guarino Lo Bianco, C.; Faroni, M.; Beschi, M.; Visioli, A. A Predictive Technique for the Real-Time Trajectory Scaling Under High-Order Constraints. *IEEE/ASME Trans. Mechatronics* **2022**, *27*, 315–326. [CrossRef]
34. Andersson, J.A.E.; Gillis, J.; Horn, G.; Rawlings, J.B.; Diehl, M. CasADi—A software framework for nonlinear optimization and optimal control. *Math. Program. Comput.* **2019**, *11*, 1–36. [CrossRef]
35. Fu, Z.; Spyrakos-Papastavridis, E.; Lin, Y.-H.; Dai, J.S. Analytical Expressions of Serial Manipulator Jacobians and their High-Order Derivatives based on Lie Theory. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 7095–7100. [CrossRef]