actuators

MDPI

Article

Sequential Optimal Trajectory Planning Scheme for Robotic Manipulators along Specified Path Based on Direct Collocation Method

Ziyao Xiong, Jianwan Ding *, Liping Chen, Yu Chen and Dong Yan

School of Mechanical Science & Engineering, Huazhong University of Science and Technology, Wuhan 430070, China; d201980206@hust.edu.cn (Z.X.); chenlp@hust.edu.cn (L.C.); d202180318@hust.edu.cn (Y.C.); m202270406@hust.edu.cn (D.Y.)

* Correspondence: dingjw@hust.edu.cn

Abstract: Robotic manipulators play a pivotal role in modern intelligent manufacturing and unmanned production systems, often tasked with executing specific paths accurately. However, the input of the robotic manipulators is trajectory which is a path with time information. The resulting core technology is trajectory planning methods which are broadly classified into two categories: maximum velocity curve (MVC) methods and multiphase direct collocation (MPDC) methods. This paper concentrates on addressing challenges associated with the latter methods. In MPDC methods, the solving efficiency and accuracy are greatly influenced by the number of discretization nodes. When dealing with systems with complex dynamics, such as robotic manipulators, striking a balance between solving time and path discretization errors becomes crucial. We use a mesh refinement (MR) algorithm to find a suitable number of nodes under the premise of ensuring the path discretization error. So, the actual device can effectively implement the planned solutions. Nonetheless, the conventional application of the MR algorithm requires solving the original problem in each iteration; these processes are extremely time-consuming and may fail to solve when dealing with a complex dynamic system. As a result, we propose a sequential optimal trajectory planning scheme to solve the problem efficiently by dividing the original optimal control (OC) problem into two stages: path planning (PP) and trajectory planning (TP). In the PP stage, we employ a DC method based on arc length and an MR algorithm to identify key nodes along the specified path. This aims to minimize the approximation error introduced during discretization. In the TP stage, the identified key nodes serve as boundary conditions for an MPDC method based on time. This facilitates the generation of an optimal trajectory that maximizes motion performance, considering constant velocity in Cartesian space and dynamic constraints while keeping the path discretization error. Simulation and experiment are conducted with a six-axis robotic manipulator, ROCR6, and show significant potential for a wide range of applications in robotics.

Keywords: optimal control method; robotics; trajectory planning

Citation: Xiong, Z.; Ding, J.; Chen, L.; Chen, Y.; Yan, D. Sequential Optimal Trajectory Planning Scheme for Robotic Manipulators along Specified Path Based on Direct Collocation Method. *Actuators* **2024**, *13*, 189. https://doi.org/10.3390/act13050189

Academic Editor: Zhuming Bi

Received: 31 March 2024
Revised: 7 May 2024
Accepted: 14 May 2024
Published: 15 May 2024

1. Introduction

Robotic manipulators are advanced modern equipment supporting intelligent manufacturing and unmanned production. They play an important role in smart manufacturing in Industry 5.0 [1]. The end effector (EE) is the most flexible component of robotic manipulators, enabling a diverse range of industrial application possibilities which include, but are not limited to, point-to-point (PTP) movements such as stacking [2–4], as well as movements along specific paths, such as laser cutting, spraying, etc. [5–7].

As industrial environments gradually advance towards intelligence and precision in development, the key technology of robotic manipulators can be cast as the trajectory planning (TP) problem of the EE along specific paths, where an optimal time law of the specific geometric path followed by the EE can be solved with a series of complex

constraints including robot dynamics. It is noticed that the trajectory can be viewed as a path subject to a time law [8].

Due to the high nonlinearity and coupling of robot dynamics, the TP problem is divided into two part. The first part is about the design of a high-level planner generating an optimal path or trajectory, and the other part is about the design of a lower-level controller supporting the implementation of tracking the optimal path or trajectory [9]. When it comes to TP along specific paths, the related research mainly focuses on the design of the high-level planner. In this situation, the lower-level controller is used as a discrete data receiver and enables actuators in the tracking mission, so that the specific path data from the planner can be executed.

Related research frames the above optimal trajectory planning (OTP) problem as an optimal control (OC) problem, which accounts for kinematic and dynamic constraints to achieve the most optimal performance index [10].

OC problems can be typically solved by indirect and direct methods. On the one hand, the Pontryagin Minimum Principle serves as the core technology of the indirect methods, providing optimal necessary conditions for OC problems and the derivation of an explicit solution. However, the indirect methods are often challenging to reconcile with complex engineering systems due to difficulties in formulating state–space equations. On the other hand, the direct methods, leveraging the progress in computers and numerical techniques, can discretize the OC problem and convert it into a Nonlinear Programming (NLP) problem. This transformation enables the application of a general NLP solver, effectively addressing the aforementioned challenge. Consequently, the direct collocation (DC) method which belongs to the latter direct methods finds widespread application in the industrial field [11].

Nonetheless, general DC methods are time-consuming and unstable when applied to solve the TP problem of robotic manipulators along a specified path due to the 'curse of dimensionality'. Therefore, researchers are actively investigating this issue. The related research began in 1985. J. E. Bobrow et al. solved the minimum-time manipulator control problem along a specified path by constructing the maximum velocity curve (MVC) in the phase plane [12]. This method utilizes the so-called Frenet–Serret Frames [13] where the time-independent variable arc length is used to describe the entire path. As a result, the number of states is streamlined from twice the degree of freedom (DOF) to just two variables including arc length and arc velocity, so that the problem can be simplified to find the MVC in the phase plane consisting of arc length and arc velocity. Finally, a type of numerical integration method is proposed to solve the specified path TP problem of three-axis manipulators. For simplicity, we refer to this kind of method as MVC methods.

Meanwhile, O. V. Stryk et al. studied the optimization problem of PTP motion for three-axis manipulators with several performance indices based on OC methods, and provided a numerical solution for the OC problem using a combination of a DC method and an indirect multiple shooting method in 1994 [14].

The above studies constitute prototypes of two main methods for solving OC problems. They are MVC methods and DC methods. A series of methods based on these two prototypes have evolved to the present day.

For the former, P. Shen et al. provide a summary of three time-optimal methods for TP along a specified path, which include Dynamic Programming (DP), Numerical Integration (NI), and Convex Optimization (CO) [15]. A classical DP approach presented by K. G. Shin et al. [9] was developed to solve the TP problem for three-axis robotic manipulators. This method includes discretizing the phase plane of MVC into separate grids and calculating the cost of forward movement between adjacent grids, facilitating the consideration of multiple performance indices. However, its demand for substantial data storage space is incredibly huge due to the 'curse of dimensionality'. Then, the method proposed by J. E. Bobrow et al. in 1985, as mentioned earlier, is an original NI approach. This method seeks the limit of the motion performance and possesses a bang–bang structure of torque inputs, but it can only focus on the time-optimal problem. Furthermore, Q. C. Pham [16] provides a general, fast, and robust implementation for time-optimal trajectory

planning (TOTP) using the NI method, improving algorithm robustness by addressing dynamic singularities [17]. In a related context, E. Barnett and C. Gosselin [18] propose a simpler and faster bisection algorithm to modify constraint violations during the NI process. Finally, the CO methods concentrate more on mathematical derivation. The convexity of the time-optimal path tracking problem should be explicitly studied to find the global optimal solution. D. Verscheure and F. Debrouwere [19,20] conducted related research. However, the CO methods may not be suitable for a general solver due to the necessity for unique mathematical derivations for each problem.

For the latter, since O. V. Stryk et al. proposed a numerical solution framework for OC problems, numerous methods have been derived. Z. Xiong et al. integrated the framework with a Functional Mock-up Unit (FMU), accomplishing optimal PTP TP for six-axis robotic manipulators [21]. However, the above technology is limited to solve the PTP TP problem. In response to this situation, J. T. Betts et al. published a path-constrained trajectory optimization method based on DC methods in 1993 [22], which may provide a way to solve the specified path TP problem. Furthermore, Betts summarized a general technology framework for solving OC problems in a book in 2010 [23]. Simultaneously, Victor M. Becerra implemented the program PSOPT based on the book and published it in the open-source community [24]. Afterwards, M. Kelly published an introduction in 2016 to assist others in solving TP problems using DC methods, including the utilization of multiphase direct collocation (MPDC) methods [10]. As a result, a brand-new method inspired by Kelly for solving the specified path trajectory planning problem came out; Y. Wen improved this MPDC method and successively accomplished a novel 3D path following a control framework for a robot [11] and a path-constrained and collision-free OTP scheme [25]. A simple literature review is shown in Figure 1.

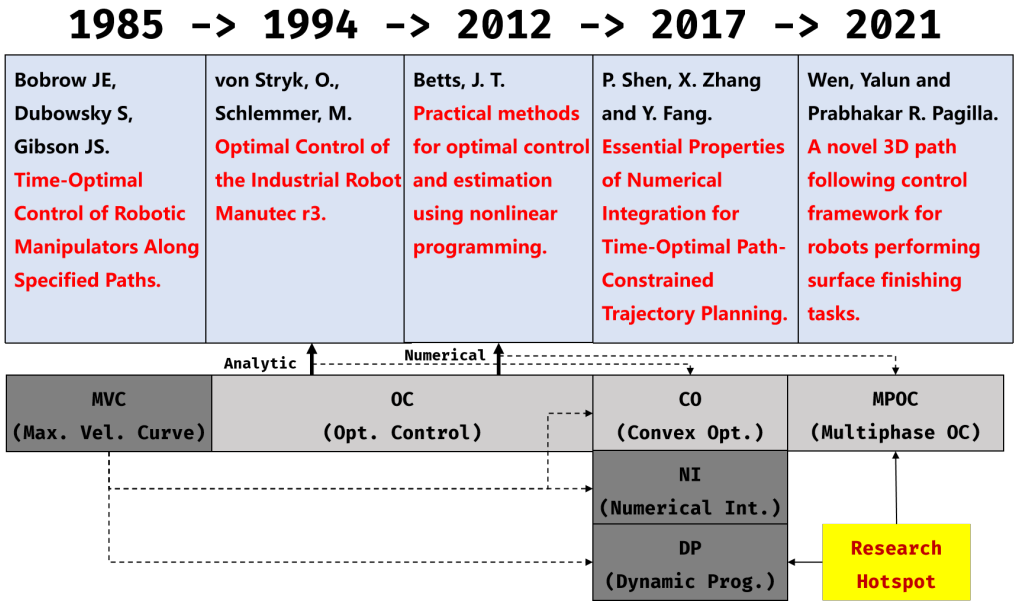

Figure 1. The literature [11,12,14,15,23] review.

At the same time, there are some other studies altogether. F. Vesentini adapts a velocity obstacle algorithm for planning collision-free trajectories for anthropomorphic arms [26]. J. Chen presents a convex TOTP method for industrial robotic manipulators with jerk constraints and achieves smooth and efficient trajectories enabling contour following and pick-and-place tasks to validate the proposed method [27]. A. Tika works on the online TOTP of cooperative robotic manipulators based on a model predictive control algorithm and realizes synchronous pick-and-place tasks [28]. JG Batista develops a TP method to avoid collisions for collaborative robotics [29]. G. Wu gives a jerk–continuous TP method for robotic manipulators by using the fourth-order S-curve to smooth the motion [30].

In summary, MVC methods and MPDC methods are two ways proven to have the ability to solve the TP problem along a specified path at present. They have different features. On the one hand, MVC methods simplify the problem but demand additional solver design, which can be categorized into NI, DP, and CO. Among them, NI is a simple and effective method but is limited to solving TOTP problems. DP can accommodate more performance indices but faces the curse of dimensionality. CO lacks universality due to the need for unique mathematical skills to transform the original problem into a convex one. On the other hand, MPDC methods support solving the target problem by a general NLP solver, and is more direct and general than MVC methods. When we use MPDC methods, the complex transformation in CO is skipped, the performance indices are not limited, the calculation storage requirement is reduced, and a local optimal solution can be provided. As we can see, the MPDC method has greater compatibility which supports its utilization in a general industrial environment.

What is more, the DC method enables the transition from OC problems to NLP problems, where each continuous variable is segmented into numerous nodes. It is evident that the quantity of these nodes significantly influences both solution accuracy and computation time. In essence, a higher number of nodes leads to increased computation time and a more accurate solution, while fewer nodes result in shorter computation time and a less accurate solution. Hence, we can find it is crucial to identify an appropriate number of nodes to strike a balance between solution accuracy and acceptable computation time. In light of the above issues, several approaches have been proposed. In 1998, John T. Betts studied a mesh refinement (MR) algorithm, which outlines a technique for changing the discretization to enhance the accuracy of the approximation [31]. This allows the nodes to start with a small quantity and gradually increase until the desired solution accuracy is achieved. However, each MR iteration happens after an entire NLP solving process. When the OC problem involves a complex system dynamics, such as a six-axis robotic manipulator, a single NLP solving process takes too much time, making the MR algorithm unsuitable for application in this scenario. In 2021, Y. Wen proposed a combined method that applies a small quantity of nodes to ensure the efficiency of the solving process and utilizes a self-designed controller to manage experimental accuracy [11].

In this paper, we present a novel sequential optimization scheme for TP problems. This scheme divides the complex specified path TP problem into two parts. In the first part, the TP problem is simplified into a path planning (PP) problem. This PP problem is designed to find a series of key nodes in the joint space to represent the specified path in Cartesian space. In this regard, a time-independent geometry optimization method is proposed based on the DC method and the MR algorithm for solving this problem. As a result, the path discretization error between the ideal path and the continuous approximate path that passes through these nodes can be minimized. What is more, the solving time is significantly reduced compared to the traditional use of the MR algorithm, and the minimum number and distribution of nodes along the specified path can be found to minimize the path discretization error. In the second part, a general MPDC method can be enabled by setting the key nodes produced in the first part as the bound conditions. In this way, we can get the time nodes corresponding to the path nodes, and the resulting continuous approximate trajectory not only minimizes the path discretization error, but also maximizes the motion performance. To sum up, a novel sequential specified path OTP scheme considering the complex system dynamics comes out.

The rest of the paper is organized as follows. Section 2 provides the problem description of the PP process, and gives the presentation of the MR algorithm to control the path discretization error. Section 3 utilizes the key nodes produced in last section to enable a MPDC method; the uniform arc-velocity condition is presented to estimate the path discretization error without another mesh optimization. Finally Sections 4 and 5 show solutions to verify the validity of the method. The technical roadmap is given in Figure 2.

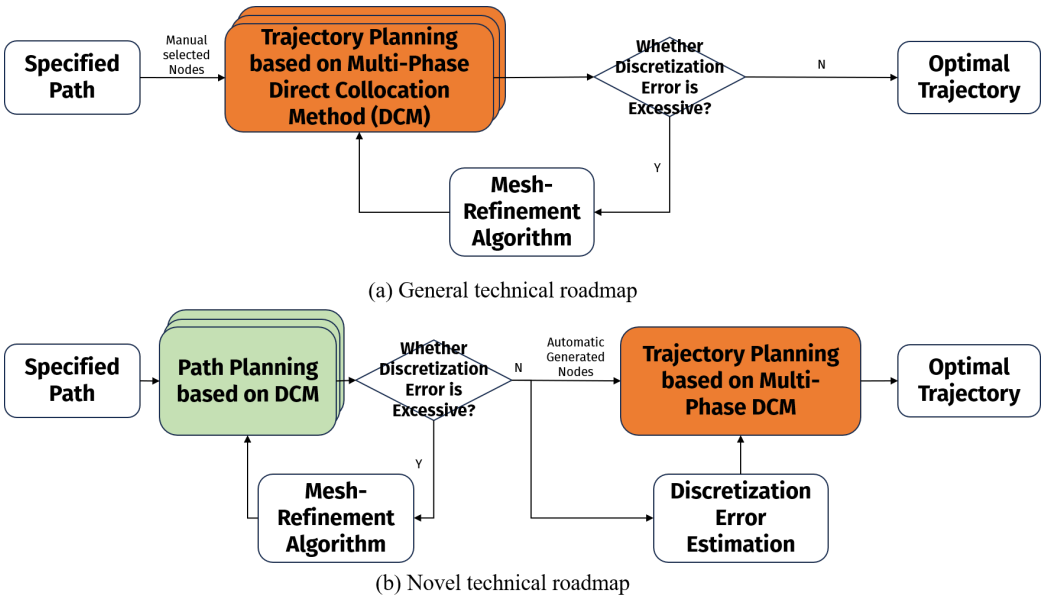

Figure 2. Technical roadmap.

2. Path Planning

In the general application of robotic manipulators, the TP problem along the specified path is typically solved by direct methods, and the resulting solution is discrete. At the same time, the general controller of robotic manipulators periodically receives the discrete desired states (referring to joint positions and joint velocities) as input and adjusts the controls (referring to joint torques and actuator currents) to achieve the trajectory tracking mission. In conclusion, we have to reduce the path discretization error in Cartesian space by applying a series of discrete nodes in joint space. It is obvious that both the discretization process and the transformation from Cartesian space to joint space introduce the path discretization error. Therefore, we can propose an optimization method that utilizes the fewest nodes possible to achieve the desired path accuracy. In this section, we regard the above optimization problem as a PP problem, and present the problem based on DC methods and the MR algorithm.

2.1. Problem Presentation

In this part, we want to present the PP problem as an OC problem. The general OC problem presentation is given as follows referring to [10,21,23].

Minimize the performance index:

$$J = \Phi(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) + \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t), t)dt, \tag{1}$$

subject to the following:
the system dynamics:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u}, t), t \in [t_0, t_f], \tag{2}$$

boundary conditions:

$$\mathbf{\Phi}(\mathbf{x}(t_0), t_0, \mathbf{x}(t_f), t_f) = 0, \tag{3}$$

constraints:

$$\mathbf{C}(\mathbf{x}(t), \mathbf{u}(t), t) \leq 0, \tag{4}$$

where J is the performance index. Φ and L are two constituents, referring to boundary cost and process cost, respectively. $\mathbf{f}$ is the system dynamics. $\mathbf{\Phi}$ is the boundary condition. $\mathbf{C}$ is the general constraint. t_0 and t_f are the start time and the end time in most situations [23], but it is noted that we replace them with the arc length s in the following path planning

problem presentation. u refers to the system control variable. x refers to the system state variable. $\dot{x}$ refers to the system differential states.

According to the above problem formulation, the path planning problem can be given as follows.

Minimize the performance index:

$$J = \int_{s_0}^{s_f} \|\mathbf{R}(s) - \mathcal{R}(s)\| ds, \tag{5}$$

subject to the following:
the system dynamics:

$$\dot{\mathbf{q}}(s) = \mathbf{u}(s), \tag{6}$$

path constraints:

$$\mathbf{R}(s) = \mathbf{fKine}[\mathbf{q}(s)] = \mathcal{R}(s) \tag{7}$$

boundary conditions:

$$\mathbf{q}(s_0) = \mathbf{q}_0, \tag{8}$$

$$\mathbf{q}(s_f) = \mathbf{q}_f, \tag{9}$$

constant constraints:

$$|\mathbf{u}(s)| \leq \mathbf{u}_{max}, \tag{10}$$

$$|\mathbf{q}(s)| \leq \mathbf{q}_{max}. \tag{11}$$

This problem description involves finding an approximate path in the joint space, aiming to execute the path in Cartesian space as closely as possible to the ideal path through forward kinematics. Thus, we consider setting joint positions as the states and the differentials of joint positions with respect to the arc length as the controls. In this way, the performance index J refers to the integration of the path discretization error along the ideal path in Cartesian space. $\mathbf{R}$ and $\mathcal{R}$ refer to positions of the executed path and the ideal path in Cartesian space, respectively. s refers to the arc length. $\mathbf{q}$ refers to the joint position. $\dot{\mathbf{q}}$ refers to the differential of joint positions with respect to the arc length, and is set as the control variable $\mathbf{u}$ at the same time. **fKine** refers to the forward kinematic function, which takes joint positions as inputs and takes positions in Cartesian space as outputs. $\mathbf{q}_0$ and $\mathbf{q}_f$ refer to the initial and the final joint position, respectively. $\mathbf{u}_{max}$ and $\mathbf{q}_{max}$ refer to the constant constraints of $\dot{\mathbf{q}}$ and $\mathbf{q}$, respectively.

As a result, DC methods can be applied to discretize the OC problem into an NLP problem. The Hermite–Simpson (H-S) collocation method is used here, and the NLP problem presentation after discretization referring to [10,21,23] is given as below.

Minimize the performance index:

$$J = \sum_{k=0}^{N-1} \frac{h_k}{6}(\omega_k + 4\omega_{k+\frac{1}{2}} + \omega_{k+1}), \tag{12}$$

$$\omega_k = \|\mathbf{R}(s_k) - \mathcal{R}(s_k)\|, \tag{13}$$

subject to the following:
the system dynamics:

$$\mathbf{q}_{k+1} - \mathbf{q}_k = \frac{h_k}{6}(\mathbf{u}_k + 4\mathbf{u}_{k+\frac{1}{2}} + \mathbf{u}_{k+1}), \tag{14}$$

$$\mathbf{q}_{k+\frac{1}{2}} = \frac{1}{2}(\mathbf{q}_k + \mathbf{q}_{k+1}) + \frac{h_k}{8}(\mathbf{u}_k - \mathbf{u}_{k+1}), \tag{15}$$

path constraints:

$$\mathcal{R}(s_m) = \mathbf{fKine}[\mathbf{q}(s_m)], m = \begin{cases} k \\ k+\frac{1}{2} \\ k+1 \end{cases}, \tag{16}$$

boundary conditions:

$$\mathbf{q}(s_0) = \mathbf{q}_0, \tag{17}$$

$$\mathbf{q}(s_N) = \mathbf{q}_f, \tag{18}$$

constant constraints:

$$\max\left(|\mathbf{u}(s_k)|, |\mathbf{u}(s_{k+\frac{1}{2}})|, |\mathbf{u}(s_{k+1})|\right) \leq \mathbf{u}_{max}, \tag{19}$$

$$\max\left(|\mathbf{q}(s_k)|, |\mathbf{q}(s_{k+\frac{1}{2}})|, |\mathbf{q}(s_{k+1})|\right) \leq \mathbf{q}_{max}. \tag{20}$$

A series of nodes are used to discretize the original problem, with a total number of nodes represented by $N+1$. Here, k denotes the serial number of the interval between nodes. s_k indicates the arc length at node k, while h_k represents the single step length between s_k and s_{k+1}. Additionally, $\mathbf{q}_k$ and $\mathbf{u}_k$ denote the states and the controls at node k, respectively. $\mathbf{q}_{k+\frac{1}{2}}$ and $\mathbf{u}_{k+\frac{1}{2}}$ denote the state and the control at the midpoint between nodes k and $k+1$. It is worth noting that the differential constraint is substituted by a number of algebraic constraints. Consequently, the original OCP transforms into an NLP, allowing for the utilization of general NLP solvers.

2.2. Mesh Refinement Algorithm

After outlining and resolving the PP problem, we consider the discrete solution nodes as a mesh. The MR algorithm can construct a new mesh by introducing a specified number of new nodes to diminish the path discretization error until it reaches an acceptable level. The execution of the MR algorithm involves adding a node to the interval with the largest path discretization error in each iteration, repeating this process until the path discretization error is deemed acceptable.

In the first step, we need to evaluate the discretization error for each interval. Thus, the ideal path and the execution path in the joint space should be given. Because the ideal path in the Cartesian space is given as a prior, a suitable inverse kinematic solution can be derived conveniently. We denote the ideal path in the joint space as $\bar{\mathbf{x}}$ and the inverse kinematic function as **iKine**, where inputs and outputs are opposite to those of the forward kinematic function **fKine**. Then, we have the following:

$$\bar{\mathbf{q}}(s) = \mathbf{iKine}[\mathcal{R}(s)]. \tag{21}$$

According to the H-S collocation method [10], a continuous execution path based on the NLP solution can be derived as follows.

$$\begin{aligned} \dot{\tilde{\mathbf{q}}}(s) = & \frac{2}{h_k^2}(\tau - \frac{h_k}{2})(\tau - h_k)\mathbf{u}_k \\ & - \frac{4}{h_k^2}\tau(\tau - h_k)\mathbf{u}_{k+\frac{1}{2}} + \frac{2}{h_k^2}\tau(\tau - \frac{h_k}{2})\mathbf{u}_{k+1}, \end{aligned} \tag{22}$$

$$\begin{aligned} \tilde{\mathbf{q}}(s) &= \int \dot{\tilde{\mathbf{q}}} ds \\ &= \mathbf{q}_k + \mathbf{u}_k(\frac{\tau}{h_k}) + \frac{1}{2}(-3\mathbf{u}_k + 4\mathbf{u}_{k+\frac{1}{2}} - \mathbf{u}_{k+1})(\frac{\tau}{h_k})^2 \\ &+ \frac{1}{3}(2\mathbf{u}_k - 4\mathbf{u}_{k+\frac{1}{2}} + 2\mathbf{u}_{k+1})(\frac{\tau}{h_k})^3, \end{aligned} \tag{23}$$

$$\tau = s - s_k. \tag{24}$$

Then, the local path discretization error in interval k, denoted as ϵ_k, is given by the following:

$$\epsilon_k = \left\| \int_{s_k}^{s_k+h_k} \|\tilde{\mathbf{q}}(s) - \bar{\mathbf{q}}(s)\| ds \right\|_\infty. \tag{25}$$

Furthermore, for ordinary differential equations (ODEs), the global error is $\mathcal{O}(h^p)$ and the local error is $\mathcal{O}(h^{p+1})$, where p refers to the order of accuracy. The H-S discretization is of order $p = 4$. As a result, the alternative representation of the local path discretization error is of the form

$$\epsilon_k \approx \|c_k\| h_k^{p+1}, \tag{26}$$

where the coefficient c_k can be solved. Then, when we insert new nodes into interval k evenly, the estimation of the local discretization path discretization error of the new mesh, denoted as η_k, is given by the following:

$$\eta_k = \|c_k\| (\frac{h_k}{1+I_k})^{p+1} = \epsilon_k (\frac{1}{1+I_k})^{p+1}, \tag{27}$$

where I_k refers to the total number of new nodes added in interval k. Consequently, an MR algorithm is designed to reduce the discretization path discretization error until it is acceptable (the tolerance error in the joint space is denoted as δ_1; the tolerance error in the Cartesian space is denoted as δ_2). Here we show the procedure of Algorithm 1.

Algorithm 1 Mesh Refinement Algorithm

1: **start**
2: Solve NLP (12)–(20);
3: Estimate path discretization error from (25);
4: Set $\eta = \max_k \epsilon_k$, $I_k = 0$;
5: **while** $(\eta > \delta_1 || J > \delta_2)$
6: Add a new node to interval k, $I_k \leftarrow I_k + 1$;
7: Refresh error from (27), $\eta \leftarrow \eta(\frac{1}{1+I_k})^{p+1}$;
8: **if** $(I_k > 0)$
9: Sort nodes and return to step 2;
10: Output the mesh;
11: **end**

In conclusion, the PP gives an improved mesh making the discretization path discretization error acceptable.

3. Trajectory Planning

In the previous section, we obtained a mesh that minimizes the path discretization error between the ideal path and the continuous approximate path passing through these mesh nodes. In this section, we utilize the MPDC method with these key mesh nodes to achieve the trajectory planning task.

3.1. Problem Presentation

In this part, our objective is to find the time law of the key mesh nodes. We assume each pair of adjacent mesh nodes as a phase and solve the time law by incorporating the mesh nodes as boundary conditions for each phase within the MPDC method framework. Here, we present the problem formulation.

Minimize the performance index:

$$J = \sum_{j=1}^{N_p} \left[\Phi^{(j)}(\mathbf{x}^{(j)}(t_0^{(j)}), t_0^{(j)}, \mathbf{x}^{(j)}(t_f^{(j)}), t_f^{(j)}) + \int_{t_0^{(j)}}^{t_f^{(j)}} L^{(j)}(\mathbf{x}^{(j)}(t), \mathbf{u}^{(j)}(t), t) dt \right], \tag{28}$$

subject to the following:
the system dynamics:

$$\dot{\mathbf{x}}^{(j)}(t) = \mathbf{f}(\mathbf{x}^{(j)}, \mathbf{u}^{(j)}, t), t \in [t_0^{(j)}, t_f^{(j)}], \tag{29}$$

boundary conditions:

$$\mathbf{Œ}^{(j)}(\mathbf{x}^{(j)}(t_0^{(j)}), t_0^{(j)}, \mathbf{x}^{(j)}(t_f^{(j)}), t_f^{(j)}) = 0, \tag{30}$$

constraints:

$$\mathbf{C}^{(j)}(\mathbf{x}^{(j)}(t), \mathbf{u}^{(j)}(t), t) \leq 0, t \in [t_0^{(j)}, t_f^{(j)}], \tag{31}$$

where the superscript (j) refers to the phase number, N_p denotes the total phase number, and other variable definitions are identical to those in (1)–(4). It is noticed that the TP problem is time-dependent, where t denotes time. $t_0^{(j)}$ and $t_f^{(j)}$ refer to the start time and the end time in phase j, respectively.

Then, the TP problem along specified path can be presented as below. To distinguish MVC methods and demonstrate their broader applicability, a comprehensive performance index including both time and energy is taken into consideration.

Minimize performance index:

$$J = \sum_{j=1}^{N_p} \left\{ t_f^{(j)} - t_0^{(j)} + \int_{t_0^{(j)}}^{t_f^{(j)}} [\mathbf{u}^{(j)}]^2 dt \right\}, \tag{32}$$

subject to the following:
the system dynamics:

$$\dot{\mathbf{x}}^{(j)} = \begin{bmatrix} \dot{\mathbf{q}}^{(j)} \\ \ddot{\mathbf{q}}^{(j)} \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}^{(j)} \\ \mathbf{f}_{FMU}(\mathbf{q}^{(j)}, \dot{\mathbf{q}}^{(j)}, \mathbf{u}^{(j)}, t^{(j)}) \end{bmatrix}, \tag{33}$$

path constraints:

$$\mathbf{R}(s^{(j)}) = \mathbf{fKine}[\mathbf{q}^{(j)}] = \mathcal{R}(s^{(j)}), \tag{34}$$

$$s^{(j)} = s_0^{(j)} + \int_{t_0^{(j)}}^{t^{(i)}} \dot{s}^{(j)} dt, \tag{35}$$

boundary conditions:

$$\mathbf{q}(t_0^{(j)}) = \mathbf{q}_0^{(j)}, \tag{36}$$

$$\mathbf{q}(t_f^{(j)}) = \mathbf{q}_f^{(j)}, \tag{37}$$

$$\dot{\mathbf{q}}(t_0^{(1)}) = \dot{\mathbf{q}}(t_f^{(N_p)}) = \mathbf{0}, \tag{38}$$

continuity conditions:

$$\dot{\mathbf{q}}(t_f^{(j)}) = \dot{\mathbf{q}}(t_0^{(j+1)}), j \neq N_p, \tag{39}$$

constant constraints:

$$| \mathbf{u}(t) | \leq \mathbf{u}_{max}, \tag{40}$$

$$| \mathbf{x}(t) | = \begin{bmatrix} |\mathbf{q}(t)| \\ |\dot{\mathbf{q}}(t)| \end{bmatrix} \leq \begin{bmatrix} \mathbf{q}_{max} \\ \dot{\mathbf{q}}_{max} \end{bmatrix}. \tag{41}$$

where the superscript (j), N_p, $t_0^{(j)}$, and $t_f^{(j)}$ have the same definitions as provided in (28). $\mathbf{u}$ denotes the system controls which refers to the torque of the joint actuator. $\mathbf{x}$ denotes the system states which refers to the joint position $\mathbf{q}$ and the joint velocity $\dot{\mathbf{q}}$ in the system of robotic manipulators. $\dot{\mathbf{x}}$ denotes the differential of the system states which refers to the joint velocity $\dot{\mathbf{q}}$ and the joint acceleration $\ddot{\mathbf{q}}$. $\mathbf{f}_{FMU}$ is a black box function constructed by MWorks and denotes the forward dynamics of the robotic system [21]. Functional Mock-up Unit (FMU) is used to name the black box function. $s^{(j)}$ is the arc length in phase j, $\dot{s}^{(j)}$ is the arc velocity in phase j. $s_0^{(j)}$ and $s_f^{(j)}$ are the initial and the end arc length in phase j. $\dot{\mathbf{R}}(s^{(j)})$ denotes the velocities of the end effector in Cartesian space and can be derived by the FMU. $t_0^{(1)}$ and $t_f^{N_p}$ refer to the initial and final time of the whole multiphase problem. As a result, (35) facilitates executing a uniform motion along the specified path. $\mathbf{q}_0^{(j)}$ and $\mathbf{q}_f^{(j)}$ are the initial joint positions and the final joint positions of phase j, respectively. Furthermore, the mesh nodes solved in the previous section are introduced here as the boundary constraints. $\mathbf{u}_{max}$, $\mathbf{q}_{max}$, and $\dot{\mathbf{q}}_{max}$ denote the max torques, positions, and velocities, respectively.

Finally, the H-S collocation method is used to represent the NLP problem shown as below.

Minimize performance index:

$$\begin{aligned} J = \sum_{j=1}^{N_p} \Bigg\{ & t_f^{(j)} - t_0^{(j)} \\ & + \sum_{k=0}^{N-1} \frac{h_k}{6} \left((\mathbf{u}_k^{(j)})^2 + 4(\mathbf{u}_{k+\frac{1}{2}}^{(j)})^2 + (\mathbf{u}_{k+1}^{(j)})^2 \right) \Bigg\}, \end{aligned} \tag{42}$$

subject to the following:
the system dynamics:

$$\begin{bmatrix} \mathbf{q}_{k+1}^{(j)} - \mathbf{q}_k^{(j)} \\ \dot{\mathbf{q}}_{k+1}^{(j)} - \dot{\mathbf{q}}_k^{(j)} \end{bmatrix} = \frac{h_k}{6} \begin{bmatrix} \dot{\mathbf{q}}_k^{(j)} + 4\dot{\mathbf{q}}_{k+\frac{1}{2}}^{(j)} + \dot{\mathbf{q}}_{k+1}^{(j)} \\ \mathbf{f}_{FMU,k}^{(j)} + 4\mathbf{f}_{FMU,k+\frac{1}{2}}^{(j)} + \mathbf{f}_{FMU,k+1}^{(j)} \end{bmatrix} \tag{43}$$

$$\begin{bmatrix} \mathbf{q}_{k+\frac{1}{2}}^{(j)} \\ \dot{\mathbf{q}}_{k+\frac{1}{2}}^{(j)} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathbf{q}_k^{(j)} + \mathbf{q}_{k+1}^{(j)} \\ \dot{\mathbf{q}}_k^{(j)} + \dot{\mathbf{q}}_{k+1}^{(j)} \end{bmatrix} + \frac{h_k}{8} \begin{bmatrix} \dot{\mathbf{q}}_k^{(j)} - \dot{\mathbf{q}}_{k+1}^{(j)} \\ \mathbf{f}_{FMU,k}^{(j)} - \mathbf{f}_{FMU,k+1}^{(j)} \end{bmatrix} \tag{44}$$

path constraints:

$$\mathcal{R}(s_m^{(j)}) = \mathbf{fKine}[\mathbf{q}_m^{(j)}], m = \begin{cases} k \\ k+\frac{1}{2} \\ k+1 \end{cases}, \tag{45}$$

$$s_{k+\frac{1}{2}}^{(j)} = \frac{s_{k+1}^{(j)} - s_k^{(j)}}{2}, \tag{46}$$

boundary conditions:

$$\mathbf{q}(t_0^{(j)}) = \mathbf{q}_0^{(j)}, \tag{47}$$

$$\mathbf{q}(t_N^{(j)}) = \mathbf{q}_f^{(j)}, \tag{48}$$

$$\dot{\mathbf{q}}(t_0^{(1)}) = \dot{\mathbf{q}}(t_N^{(N_p)}) = \mathbf{0}, \tag{49}$$

continuity conditions:

$$\dot{\mathbf{q}}(t_N^{(j)}) = \dot{\mathbf{q}}(t_0^{(j+1)}), j \neq N_p, \tag{50}$$

constant constraints:

$$\max\left(|\mathbf{u}_k^{(j)}|, |\mathbf{u}_{k+\frac{1}{2}}^{(j)}|, |\mathbf{u}_{k+1}^{(j)}|\right) \leq \mathbf{u}_{max}, \tag{51}$$

$$\max\left(\begin{bmatrix}|\mathbf{q}_k^{(j)}|\\ |\dot{\mathbf{q}}_k^{(j)}|\end{bmatrix}, \begin{bmatrix}|\mathbf{q}_{k+\frac{1}{2}}^{(j)}|\\ |\dot{\mathbf{q}}_{k+\frac{1}{2}}^{(j)}|\end{bmatrix}, \begin{bmatrix}|\mathbf{q}_{k+1}^{(j)}|\\ |\dot{\mathbf{q}}_{k+1}^{(j)}|\end{bmatrix}\right) \leq \begin{bmatrix}\mathbf{q}_{max}\\ \dot{\mathbf{q}}_{max}\end{bmatrix}. \tag{52}$$

where the superscript j denotes the phase number and N_p is the total number of phases. The subscript k denotes the nodes number, $N+1$ is the total number of nodes in the phase. And other variables follow the previous definitions. It should be noted that we set $N = 3$ when the H-S collocation method is applied. It is noticed that a key variable s is introduced to build a relationship with the path planning problem. Because the independent variable of the PP problem is arc length s, while in the TP problem it is time t, a function of $s - t$ should be constructed. For estimating the path discretization error in the next part, uniform motion is required. So we have (46) and the following equation:

$$\begin{aligned} s(t^{(j)}) &= \dot{s}^{(j)}(t^{(j)} - t_0^{(j)}) + s_0^{(j)} \\ &= \frac{s_f^{(N_p)} - s_0^{(1)}}{t_f^{(N_p)} - t_0^{(1)}}(t^{(j)} - t_0^{(j)}) + s_0^{(j)}. \end{aligned} \tag{53}$$

In this way, the original trajectory planning problem is transformed into two simplified NLP problems, and can be solved effectively. Furthermore, the PP stage can guarantee the path discretization error in the TP stage; the derivation process is given in the next section.

3.2. Path Discretization Error Estimation

This part derives the relationship of the path discretization error between the procedures of PP and TP, and proposes the key constraints in the above TP problem presentation.

The path discretization error is set as the difference between the path integrals. The ideal path integral $Q_{ideal,k+1}$ and the approximate path integral $Q_{apprx,k+1}$ in interval k are shown as below:

$$Q_{ideal,k+1} = Q_{ideal,k} + \int_{s_k}^{s_{k+1}} \bar{\mathbf{q}}(s)ds, \tag{54}$$

$$Q_{apprx,k+1} = Q_{apprx,k} + \int_{s_k}^{s_{k+1}} \tilde{\mathbf{q}}(s)ds, \tag{55}$$

then, we can derive the path discretization error in interval k:

$$\begin{aligned} \epsilon_k &= \|Q_{apprx,k+1} - Q_{ideal,k+1}\| \\ &= \|Q_{apprx,k} - Q_{ideal,k} + \int_{s_k}^{s_{k+1}} \tilde{\mathbf{q}}(s)ds - \int_{s_k}^{s_{k+1}} \bar{\mathbf{q}}(s)ds\|, \end{aligned} \tag{56}$$

we ignore the error at the beginning of the interval and set $Q_{apprx,k} = Q_{ideal,k}$, which leads to the following:

$$\begin{aligned} \epsilon_k &= \|\int_{s_k}^{s_{k+1}} \tilde{\mathbf{q}}(s)ds - \int_{s_k}^{s_{k+1}} \bar{\mathbf{q}}(s)ds\|, \\ &\leq \int_{s_k}^{s_{k+1}} \|\tilde{\mathbf{q}}(s) - \bar{\mathbf{q}}(s)\|ds. \end{aligned} \tag{57}$$

As a result, we get the general formula of the path discretization error. In addition, the solution path discretization error at interval k in the PP process is mentioned in (25). For the consistence of notions, we set the following:

$$\epsilon_{path,k} = \int_{s_k}^{s_{k+1}} \|\tilde{\mathbf{q}}(s) - \bar{\mathbf{q}}(s)\| ds, \tag{58}$$

in addition, the path discretization error at phase j in the TP problem can be written as follows:

$$\epsilon_{traj,j} = \int_{t_0^{(j)}}^{t_f^{(j)}} \|\tilde{\mathbf{q}}(t) - \bar{\mathbf{q}}(t)\| dt. \tag{59}$$

The boundary conditions at phase j are based on the states at interval k; we set the following:

$$s_k = s(t_0^{(j)}), s_{k+1} = s(t_f^{(j)}), \tag{60}$$

then, we have the following:

$$\tilde{\mathbf{q}}(s_k) = \bar{\mathbf{q}}(s_k) = \tilde{\mathbf{q}}(t_0^{(j)}) = \bar{\mathbf{q}}(t_0^{(j)}), \tag{61}$$

$$\tilde{\mathbf{q}}(s_{k+1}) = \bar{\mathbf{q}}(s_{k+1}) = \tilde{\mathbf{q}}(t_f^{(j)}) = \bar{\mathbf{q}}(t_f^{(j)}). \tag{62}$$

Moreover, (58) can be rewritten as follows:

$$\epsilon_{path,j} = \int_{t_0^{(j)}}^{t_f^{(j)}} \|\tilde{\mathbf{q}}(t) - \bar{\mathbf{q}}(t)\| \dot{s} dt, \tag{63}$$

For uniform motion, we have $\dot{s}_{avrg}^{(j)} = (s_f^{(N_p)} - s_0^{(1)})/(t_f^{(N_p)} - t_0^{(1)})$. Finally, (59) can be rewritten as follows:

$$\epsilon_{traj,j} = \frac{\epsilon_{path,j}}{\beta \dot{s}_{avrg}^{(j)}}, \tag{64}$$

where β denotes the scaling parameters, and belongs to the interval $(0, 1]$. This parameter is used to estimate the error that occurs in the arc-velocity transformation between adjacent phases. We set $\beta = 0.9$ here. In this way, a linear dependence between $\epsilon_{traj,j}$ and $\epsilon_{path,j}$ can be established by giving a suitable value of $\dot{s}_{avrg}^{(j)}$, which can be estimated through a preliminary TP process.

In conclusion, the path discretization error of the TP process can be estimated and adjusted during the process of PP. This enables the MR algorithm to be positioned at the forefront of the entire planning process, avoiding significant computation time for mesh optimization in the TP process.

4. Numerical Simulation

This section contains a complete simulation planning process for executing a specified circular path using a six-axis robotic manipulator ROCR6 manufactured by Si Vally, China. The simulation environment is a personal computer with an AMD Ryzen 7 5800H processor with 8 cores (3.20 GHz). The device information can be found in the official website or in the research [21].

The specified circular path in the PP process is shown in Figure 3. The coordinates of the circle center are $[250, 350, 300]$, the radius is 100, and the starting coordinates are

[350, 350, 300]. The length unit used in this paper is mm; the joint position unit is rad. The ideal path functions can be written as follows:

$$\mathcal{R}(s) = \begin{cases} x(s) \\ y(s) \\ z(s) \end{cases} = \begin{cases} 250 + 100cos(s/100) \\ 350 + 100sin(s/100) \\ 300 \end{cases}, \tag{65}$$

so that the path constraints (7) can be given with the forward kinematics.

The boundary and inequality conditions are set as follows:

$$s_0 = 0, s_f = 200\pi, \tag{66}$$

$$\mathbf{q}_0 = \mathbf{q}_f = [-2.61, 0.47, 0.96, 0.14, -1.57, -2.61]^T, \tag{67}$$

$$\mathbf{u}_{max} = \mathbf{inf}_{6\times 1}, \tag{68}$$

$$\mathbf{q}_{max} = [3.12, 2.55, 2.55, 3.12, 3.12, 3.12]^T. \tag{69}$$

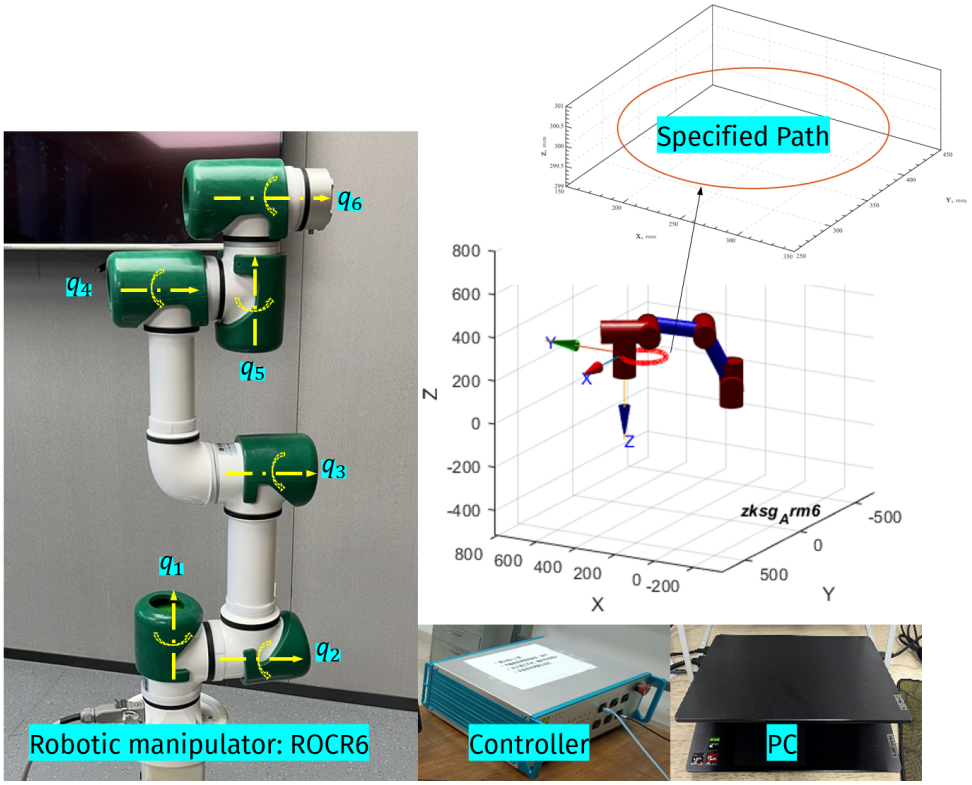

Figure 3. The device and the task path.

The initial total number of intervals is set as $N = 19$. The initial guesses of decision variables at node $k \in [0, 19]$ are given as below:

$$s_k = s_0 + \frac{k}{N}(s_f - s_0), \tag{70}$$

$$\mathbf{u}_k = \mathbf{0}, \tag{71}$$

$$\mathbf{q}_k = \mathbf{0}. \tag{72}$$

Then, a software for large-scale nonlinear optimization named Interior Point OPTimizer (IPOPT) can be applied to solve (12)–(20). In addition, δ_1 and δ_2 have to be set to

enable Algorithm 1. It is known that the location accuracy of ROCR6 is 0.02 mm. Furthermore, we can find a linear relationship between the path discretization error in the PP process and that in the TP process, as indicated by (64). A rough optimization without constant arc velocity is used to define $\dot{s}_{avrg}^{(j)} = 0.1256$ m/s. Then, we have the following:

$$\epsilon_{traj,j} = 8.85\epsilon_{path,j} \leq 10\epsilon_{path,j}. \tag{73}$$

Therefore, let δ_2 be 2×10^{-3} mm and δ_1 be 1×10^{-3} rad according to experiment experience. Finally, the solutions are presented in Figure 4, where the position function with respect to the arc length of each joint is given. In addition, the MR algorithm produces a heterogeneous mesh to minimize the path discretization error.

The PP process includes six iterations where the MR algorithm introduces 8, 11, 15, 21, and 7 nodes into the mesh, respectively. The maximum path discretization error in each iteration is presented in Table 1, where we can find that the path discretization error is gradually reduced with the introduction of nodes in each iteration. In addition, the path discretization error reduction in each iteration is shown in Figure 5. The different color blocks refer to the path discretization error at different iterations, and we observe a consistent reduction in the path discretization error across iterations. So far, the PP process is finished.

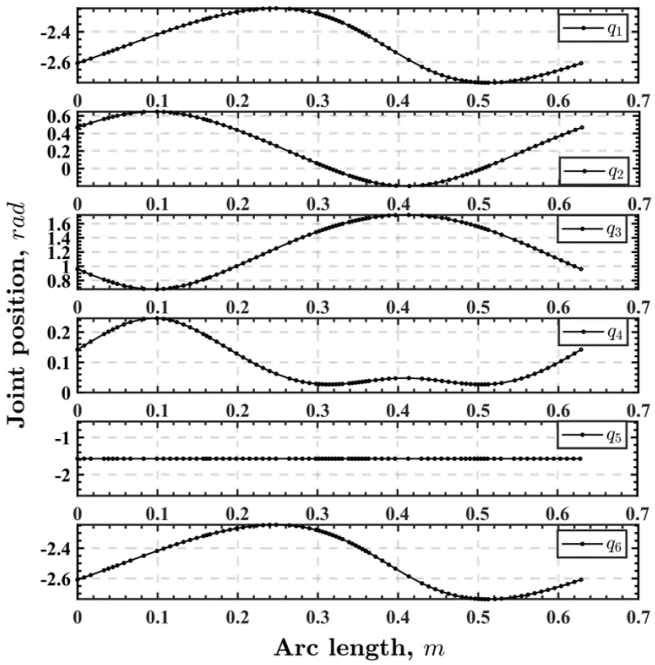

Figure 4. Path planning solutions.

Table 1. Path discretization error in each iteration.

Iteration	Nodes	δ_1 (rad)	δ_2 (m)
1	20	6.326×10^{-2}	3.533804×10^{-7}
2	28	1.423×10^{-2}	7.247751×10^{-8}
3	39	5.426×10^{-3}	3.885858×10^{-8}
4	54	2.509×10^{-3}	1.193616×10^{-8}
5	75	1.248×10^{-3}	5.273560×10^{-9}
6	82	9.342×10^{-4}	4.087400×10^{-9}

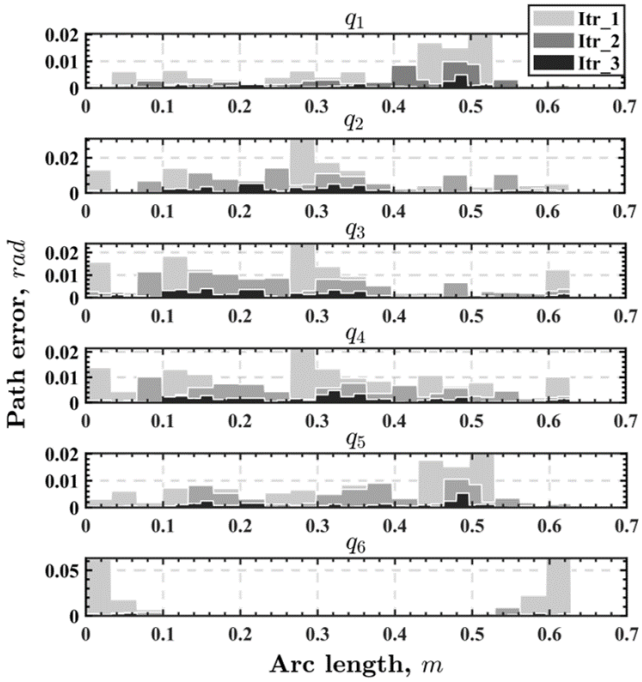

Figure 5. Path discretization error reduction in each iteration.

Next, the above solution mesh is used as boundary conditions for the TP process, where the aim is to find the time law tied to the solution mesh from the PP process. And this process solves the average arc velocity along the specified path and gives the optimal time as the result, which can be set as the input of the experimental device ROCR6. The solution is shown in Figure 6.

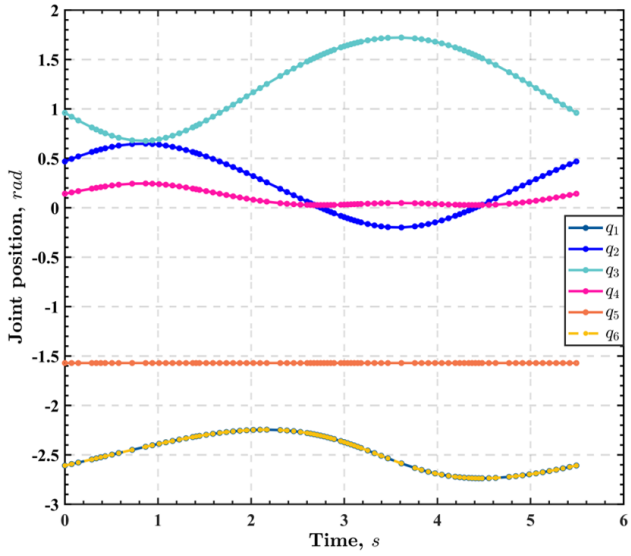

Figure 6. Trajectory planning solutions.

5. Experiment

In the above section, we proposed an efficient scheme for simplifying the original TP problem along the specified path for robotic manipulators by splitting the complex OC problem into two simple NLP problems. Thus, an optimal trajectory with achievable maximum constant arc velocity can be derived by using an open source C++ NLP solver called IPOPT. In this section, we design three groups of control experiments using different arc velocities to validate the improvement of the proposed method compared to the traditional

MPDC method. The advantages of the optimal trajectory can be analyzed from two aspects: the accuracy of path following in the Cartesian space and the ranges of joint currents.

The solutions of these three groups come from the traditional MPDC method, the proposed method, and manual setting. The traditional MPDC method solutions are unstable because of the small quantity of the modes. We set t_{2s} as the average approximate solution produced by the traditional MPDC method with less than 20 evenly spaced nodes. t_{opt} is the solution of proposed method. t_{10s} is a manual control group with longer time. The experimental solutions of controlled groups with different arc velocities are presented in Figure 7, where t_{2s}, t_{opt}, and t_{10s} refer to the solutions finished in 2 s, 5.494 s, and 10 s, respectively. It is obvious that t_{opt} and t_{10s} can keep a stable tracking performance. They have the same error in Z axis varying in the interval between 300 mm and 301.5 mm, and track the Ideal Path (IP) well in plane X-Y. However, t_{2s} is not allowed because of the dynamic constraints. We can find that the error in Z axis jumps up to 303 mm, and the IP cannot be tracked well. The currents in joint 1 to 3 are also presented in Figure 8 to validate the optimal trajectory. It is noticed that the faster the arc velocity, the more rapid the current changes. In addition, faster arc velocity also introduces wilder current boundary which might not be acceptable. In summary, we find the optimal trajectory can be executed well, and the efficiency of the scheme is validated.

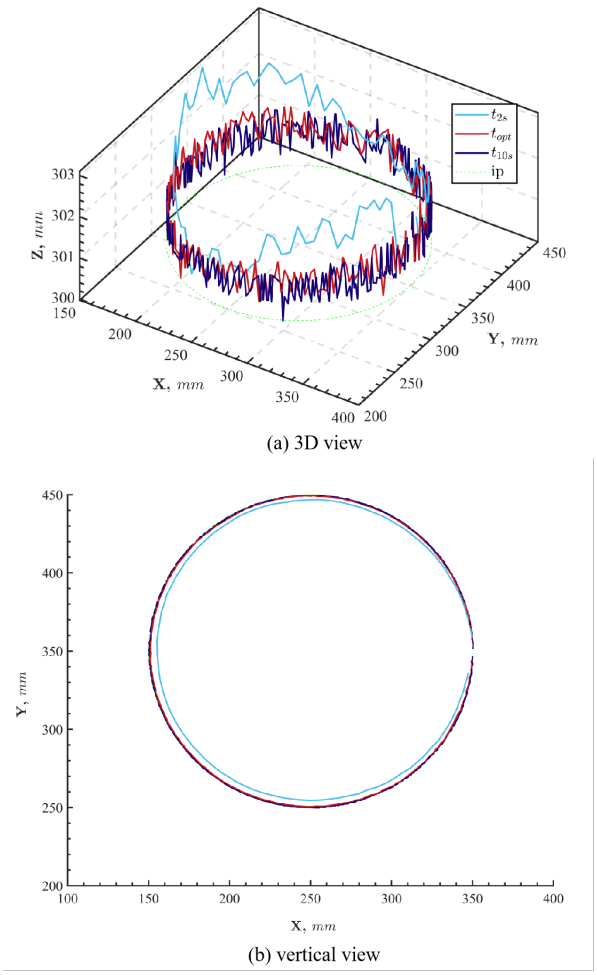

Figure 7. The experimental Cartesian paths.

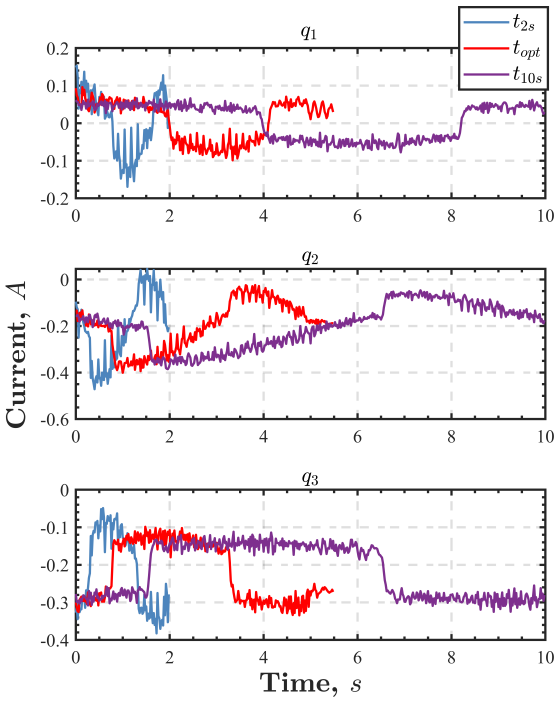

Figure 8. The experimental currents.

6. Conclusions

In this paper, we have proposed a sequential OTP scheme for robotic manipulators along a specified path, utilizing DC methods. This scheme demands a predefined Cartesian path expressed explicitly in terms of arc length and aims to optimize the motion performance of robotic manipulators to accurately track the specified path. For complex systems like six-axis robotic manipulators, maintaining desired accuracy post-discretization with DC methods can be time consuming. To address this challenge and enhance solution efficiency, we divide the original OC problem into two stages: PP and TP. In the PP stage, we employ a DC method based on arc length and an MR algorithm to identify key nodes along the specified path. This aims to minimize the approximation error introduced during discretization. In the TP stage, the identified key nodes serve as boundary conditions for an MPDC method based on time. This facilitates the generation of an optimal trajectory that maximizes motion performance, considering constant velocity in Cartesian space and dynamic constraints while keeping the path discretization error.

In summary, the proposed method has two advantages compared to the traditional MPDC method. The first one is advancing the MR iterations into the PP process. This can greatly improve the solving efficiency by avoiding the iterative calculation of the TP process. The second one is the MR algorithm design in the PP process, which helps us to identify key nodes along the specified path while minimizing the path discretization error. Therefore, the solution is guaranteed to be achieved in reality. Simulation and experimental results are presented to validate the efficiency of the optimal trajectory, demonstrating its successful execution. In the future, this planning technology can combine with the control technology to achieve model predictive control and real-time force control. This approach holds significant potential for a wide range of applications in robotics.

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

Funding: This work was supported by the Key R&D Program of Hubei Province under grant number 2021AAB001.

Data Availability Statement: Data are contained within the article. The data presented in this study can be requested from the authors.

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

References

1. Noor-A-Rahim, M.; Firyaguna, F.; John, J.; Khyam, M.O.; Pesch, D.; Armstrong, E.; Claussen, H.; Poor, H.V. Toward Industry 5.0: Intelligent Reflecting Surface in Smart Manufacturing. *IEEE Commun. Mag.* **2022**, *60*, 72–78. [CrossRef]
2. Domínguez, D.C.; Iannotta, M.; Stork, J.A.; Schaffernicht, E.; Stoyanov, T. A Stack-of-Tasks Approach Combined with Behavior Trees: A New Framework for Robot Control. *IEEE Robot. Autom. Lett.* **2022**, *7*, 12110–12117. [CrossRef]
3. Dong, X.; Jiang, Y.; Zhao, F.; Xia, J. A Practical Multi-Stage Grasp Detection Method for Kinova Robot in Stacked Environments. *Micromachines* **2023**, *14*, 117. [CrossRef]
4. Liu, C.; Jiang, D.; Lin, W.; Gomes, L. Robot Grasping Based on Stacked Object Classification Network and Grasping Order Planning. *Electronics* **2022**, *11*, 706. [CrossRef]
5. Franz, D.; Yang, Y.; Michel, L.; Esen, C.; Hellmann, R. Evaluation of an ultrashort pulsed laser robot system for flexible and large-area micromachining. *J. Laser Appl.* **2023**, *35*, 042057. [CrossRef]
6. Honigmann, P.; Hofer, M.; Hirsch, S.; Morawska, M.; Müller-Gerbl, M.; Thieringer, F.M.; Coppo, E. Cold ablation robot-guided laser osteotomy in hand, wrist and forearm surgery—A feasibility study. *Int. J. Med. Robot.* **2022**, *18*, e2438. [CrossRef]
7. Kumar, V.; Kalita, K.; Chatterjee, P.; Zavadskas, E.K.; Shankar, C. A SWARA-CoCoSo-Based Approach for Spray Painting Robot Selection. *Informatica* **2021**, *33*, 35–54. [CrossRef]
8. Olofsson, B.; Nielsen, L. Path-tracking velocity control for robot manipulators with actuator constraints. *Mechatronics* **2017**, *45*, 82–99. [CrossRef]
9. Shin, K.; McKay, N. A dynamic programming approach to trajectory planning of robotic manipulators. *IEEE Trans. Autom. Control* **1986**, *31*, 491–500. [CrossRef]
10. Kelly, M. An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation. *SIAM Rev.* **2017**, *59*, 849–904. [CrossRef]
11. Wen, Y.; Prabhakar, R.P. A novel 3D path following control framework for robots performing surface finishing tasks. *Mechatronics* **2021**, *76*, 102540. [CrossRef]
12. 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]
13. Lapierre, L.; Soetanto, D.; Pascoal, A. Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties. *Int. Robust Nonlinear Control* **2006**, *16*, 485–503. [CrossRef]
14. von Stryk, O.; Schlemmer, M. Optimal Control of the Industrial Robot Manutec r3. In *Computational Optimal Control. ISNM International Series of Numerical Mathematics*; Bulirsch, R., Kraft, D., Eds.; Birkhäuser: Basel, Switzerland; Geneva, Switzerland, 1994; Volume 115. [CrossRef]
15. 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]
16. Pham, Q.-C. A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm. *IEEE Trans. Robot.* **2014**, *30*, 1533–1540. [CrossRef]
17. Pham, Q.-C. Characterizing and addressing dynamic singularities in the time-optimal path parameterization algorithm. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 2357–2363. [CrossRef]
18. Barnett, E.; Gosselin, C. A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths. *IEEE Trans. Robot.* **2021**, *37*, 131–145. [CrossRef]
19. 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]
20. Verscheure, D.; Demeulenaere, B.; Swevers, J.; De Schutter, J.; Diehl, M. Time-Optimal Path Tracking for Robots: A Convex Optimization Approach. *IEEE Trans. Autom. Control* **2009**, *54*, 2318–2327. [CrossRef]
21. Xiong, Z.; Ding, J.; Chen, L. Time-Optimal Trajectory Planning of Six-Axis Manipulators Based on the Improved Direct Collocation Method with FMU. *Appl. Sci.* **2022**, *12*, 6741. [CrossRef]
22. Betts, J.T.; Huffman, W.P. Path-constrained trajectory optimization using sparse sequential quadratic programming. *J. Guid. Control Dyn.* **1993**, *16*, 59–68. [CrossRef]
23. Betts, J.T. Society for Industrial and Applied Mathematics. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2010.
24. Becerra, V.M. Solving complex optimal control problems at no cost with PSOPT. In Proceedings of the 2010 IEEE International Symposium on Computer-Aided Control System Design, Yokohama, Japan, 8–10 September 2010; pp. 1391–1396. [CrossRef]

25. Wen, Y.; Pagilla, P. Path-Constrained and Collision-Free Optimal Trajectory Planning for Robot Manipulators. *IEEE Trans. Autom. Sci. Eng.* **2023**, *20*, 763–774. [CrossRef]
26. Vesentini, F.; Piccinelli, N.; Muradore, R. Velocity obstacle-based trajectory planner for anthropomorphic arms. *Eur. J. Control* **2023**, *75*, 100901. [CrossRef]
27. Ji, C.; Zhang, Z.; Cheng, G.; Kong, M.; Li, R. A Convex Optimization Method to Time-Optimal Trajectory Planning with Jerk Constraint for Industrial Robotic Manipulators. *IEEE Trans. Autom. Sci. Eng.* **2023**, 1–18.
28. Tika, A.; Bajcinca, N. Predictive Control of Cooperative Robots Sharing Common Workspace. *IEEE Trans. Control Syst. Technol.* **2024**, *32*, 456–471. [CrossRef]
29. Batista, J.G.; Ramalho, G.L.B.; Torres, M.A.; Oliveira, A.L.; Ferreira, D.S. Collision Avoidance for a Selective Compliance Assembly Robot Arm Manipulator Using Topological Path Planning. *Appl. Sci.* **2023**, *13*, 11642. [CrossRef]
30. Wu, G.; Zhang, N. Kinematically Constrained Jerk–Continuous S-Curve Trajectory Planning in Joint Space for Industrial Robots. *Electronics* **2023**, *12*, 1135. [CrossRef]
31. Betts, J.T.; Huffman, W.P. Mesh refinement in direct transcription methods for optimal control. Optim. *Control Appl. Meth.* **1998**, *19*, 1–21. [CrossRef]