A Receding Horizon Online Trajectory Generation Method for Time-Efficient Path Tracking with Dynamic Factors

Shiyu Zhang, Da Sun, and Qianfang Liao

***Abstract*—Current path-tracking trajectory planning methods intrinsically suffer from a heavy computational burden, hindering them from modern autonomous robotic applications requiring real-time reactivity. To achieve flexible, accurate, and efficient motions, this article proposes a real-time trajectory planning framework for time-efficient path tracking. First, a receding horizon method is designed that generates local trajectories online while considering the global kinematic constraints. Second, an analytical closed-form method is developed for calculating the local time-minimal trajectory. Third, the models and solutions are established for handling dynamic factors. Compared to existing methods, this method exhibits lower computational complexity and can deal with path changes during motion executions while maintaining tracking accuracy and time efficiency. Experiments using Franka-Emika Panda robots are conducted in handover scenarios involving unpredictable dynamic obstacles and human interventions. The results demonstrate the low computational overhead. The robot flexibly reacts to online path changes, maintaining tracking accuracy while marginally compromising time efficiency.**

***Index Terms*—Autonomous robots, collision avoidance, motion planning, trajectory optimization.**

I. Introduction

NOWADAYS, robots are applied to increasingly diverse applications, demanding greater flexibility and autonomy. Many applications require robots to cope with unpredictable and uncertain situations, such as working in dynamic environments and interacting or collaborating with humans. To respond promptly to these uncertainties, real-time trajectory planning (within a single control cycle) becomes a critical issue.

The time-optimal path-tracking trajectory planning problem is of significant importance, which involves finding the trajectory for following a given path with minimal time duration while adhering to kinematic and dynamic constraints [1]. Accurately following specified paths is crucial for collision avoidance and various applications, such as welding [2] and advanced manufacturing [3]; while time-optimal trajectories maximize the robot's productivity and responsiveness, making them vital for efficient and effective robotic operations.

Current path-tracking trajectory planning methods usually suffer from heavy computational complexity due to extensive path-tracking constraints. This challenge is further compounded in the case of serial manipulators with multiple degrees of freedom (DOF), where the synchronization of multiple joints adds another layer of complexity. To overcome this challenge, optimal time-scaling methods have been developed for tracking given paths, leveraging a bang-bang structure to identify switch points between acceleration and deceleration through forward and backward integration [4], [5]. Following this line, methods based on numerical integration [6], [7], convex optimization [8], [9], and dynamic programming [10], [11] are proposed to improve computational efficiency, robustness, and smoothness. However, these methods require the path to be predetermined and cannot handle online changes. Although some path-tracking trajectory planning methods have emerged that can handle online changes [12], [13], they often require substantial preprocessing or involve iterative solving processes. As a result, they may not be fully competent in reliably managing frequently changing situations while ensuring feasibility and tracking accuracy.

This article aims to address the high computational complexity associated with path-tracking trajectory planning while upholding feasibility, accuracy, and time efficiency, thus enabling flexible and reliable online path-tracking. To achieve this, we propose a novel real-time trajectory planning framework for time-efficient path tracking. The method integrates a receding horizon structure and an analytical closed-form formulation to rapidly generate accurate local trajectories, while incorporating global constraints to achieve overall feasibility and time efficiency. The key contributions are listed as follows.

Received 11 January 2024; revised 7 May 2024 and 5 August 2024; accepted 3 October 2024. Recommended by Technical Editor M. Liarokapis and Senior Editor S. Katsura. This work was supported in part by the National Natural Science Foundation of China under Grant 62303067, and in part by the Fundamental Research Funds for the Central Universities under Grant 2023RC60. *(Corresponding author: Shiyu Zhang.)*

Shiyu Zhang is with the School of Intelligent Engineering and Automation, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: shiyu.zhang@bupt.edu.cn).

Da Sun and Qianfang Liao are with the Center for Applied Autonomous Sensor Systems, Örebro University, 701 82 Örebro, Sweden (e-mail: da.sun@oru.se; qianfang.liao@oru.se).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2024.3477006.

Digital Object Identifier 10.1109/TMECH.2024.3477006

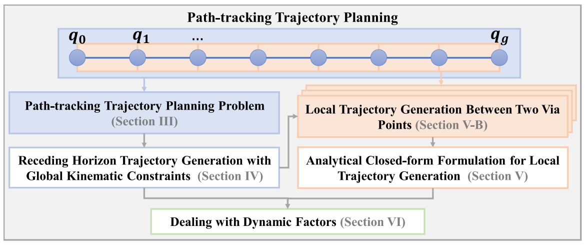

Path-tracking Trajectory Planning
q_0
q_1
...
q_g
Path-tracking Trajectory Planning Problem (Section III)
Local Trajectory Generation Between Two Via Points (Section V-B)
Receding Horizon Trajectory Generation with Global Kinematic Constraints (Section IV)
Analytical Closed-form Formulation for Local Trajectory Generation (Section V)
Dealing with Dynamic Factors (Section VI)

Fig. 1. Article organization and the relationships among the main sections. The path-tracking trajectory planning problem is first transformed into a set of local trajectory generation problems between two via points by the receding horizon method. Then, the local trajectory generation problem is solved with an analytical closed-form formulation. The receding horizon method and the analytical closed-form formulation can support dynamic factor handling.

1) A receding horizon method for path-tracking trajectory planning is designed, which supports generating local trajectories during motion execution while considering the global kinematic constraints to guarantee feasibility and minimize time duration.
2) An analytical closed-form method is developed for generating local time-minimal trajectories, which requires no iterative solving process and endows high tracking accuracy and low computational complexity.
3) Robust strategies are included in trajectory planning for effectively handling dynamic factors.

The proposed method is validated by experiments with 7-DOF robots in handover scenarios. Compared to existing methods (e.g., [14] and [15]), it achieves lower computational complexity and can operate in real time. The robot reacts efficiently to unpredictable situations, including dynamic obstacles and changing targets.

The organization of the article is illustrated in Fig. 1.

II. Related Work

A. Offline Time-optimal Path Parameterization

Time-optimal Path Parameterization (TOPP) problem consists of taking a fully specified path and finding the trajectory that minimizes the time required to follow a path while subjecting to some kinematic and dynamic constraints. Numerical integration (NI), convex optimization (CO), and dynamic programming (DP) are three typical methods for addressing the TOPP problem.

The TOPP problem was originally formulated by Bobrow et al. [4] and Shin and McKay [16], where NI methods for time-optimal time-scaling were proposed. It exploits a bang-bang structure and finds switch points between maximum acceleration and deceleration. Based on this work, Kunz and Stilman [14] introduced a path pre-processing technique to convert non-differentiable paths into differentiable ones, effectively enhancing the robustness of trajectory generation. Pham [6] further addressed dynamic singularities and provided a general, fast, and robust implementation of TOPP. A TOPP method based on reachability analysis is later proposed [15], which achieves higher computational efficiency and a 100% success rate. Shen et al. [7] summarized properties of the NI method and provided rigorous mathematical proofs.

Another approach is the CO method [17], [18], wherein Verscheure et al. [8] converted the TOPP problem into a convex optimal control problem. Debrouwere et al. [19] expanded the convex framework to accommodate nonconvex constraints in practical applications. Ma et al. [9] proposed a new convex optimization method, which deals with nonconvex jerk constraints and achieves better smoothness.

The third branch is the DP method, which relies on dividing the global problem into a series of simpler subproblems. The computational complexity of this method does not increase as the path length grows. Pfeiffer and Johanni [10] proposed an algorithm, which employs a geometric interpretation of the maximum-velocity curve (MVC) instead of relying on a numerical shooting method to find switching points. Dong and Stori [20] proposed a two-pass integration DP method, which does not rely on finding an MVC or switching points. Barnett and Gosselin [11] proposed a bisection algorithm to find the switching points, which is more numerically robust. The main drawbacks of the above methods are that they need to specify the path in advance, and the trajectory must be calculated offline, prior to motion execution.

B. Online Path-tracking Trajectory Planning

In recent times, researchers have shifted their focus towards developing online trajectory planning methods that impose a low computational burden [21].

Real-time trajectory planning is particularly crucial for applications such as human–robot interaction [22] and ball catching [23], where the robots need to constantly react to dynamic and unpredictable factors. Methods based on machine learning [24], [25] and optimization problem decoupling [26] are proposed, which significantly improved computational efficiency. However, the above methods merely focus on point-to-point trajectory planning problems without path constraints.

Kim and Croft [27] proposed an online path-tracking trajectory planning method that achieves near time-optimal performance. Each path segment is approximated with a trapezoidal velocity profile (TVP) and a smooth curve between a pair of segments is generated. Faroni et al. [28] proposed a time-scaling method based on non-lookahead model predictive control. It achieves fast computation while improving path-following accuracy and time optimality. Zanchettin and Lacevic [29] proposed a minimum-time path-following method for collaborative robots. It simultaneously handles productivity and safety criteria. He et al. [30] proposed a time-optimal switching trajectory index coordination method for continuous multi-axis trajectories. It achieves nearly the same optimality as the offline planned results and real-time performance. Solely focusing on a specified path, the above method did not discuss how to deal with the path changes during motion.

Kiemel and Kröger [12] presented a learning-based path-tracking method that enables changes to the reference path during motion. Nevertheless, this approach compromises time-optimality, and the deviation from the reference path is non-negligible. Völz and Craichen [13] introduced a predictive path-following controller that supports continuous replanning using dynamic roadmaps, enabling real-time replanning during motion. However, the iterative process carries the risk of control not being returned within a sampling cycle, potentially causing inaccuracies in path-following.

In summary, existing time-optimal path-tracking methods often face significant computational complexity. While some methods support online trajectory generation, they still demand considerable preprocessing or iterative solving processes. Addressing frequently changing situations calls for methods with simpler computations and a better guarantee of feasibility.

III. Problem Statement

A standard paradigm for controlling a robot to perform tasks involves two main steps: 1) determining a collision-free path from a starting configuration to a goal configuration; and 2) time-scaling the path to create a feasible trajectory. This trajectory is then supplied to the motion controller, which generates torques acting on the actuators to physically move the robot. In this article, we define a "path" as a purely geometric representation of the sequence of configurations achieved by the robot, and a "trajectory" as a time law that determines when the configurations on the path are reached [1].

This article focuses on the path-tracking trajectory planning problem in joint space. Given a path specified by N_p via points, each of which is represented by the configuration

$$\boldsymbol{P} = \{\boldsymbol{q_0}, \boldsymbol{q_1}, \ldots, \boldsymbol{q_{N_p-1}} := \boldsymbol{q_g}\},$$

$$\boldsymbol{q_i} = \{q_i^1, q_i^2, \ldots, q_i^{N_j}\} (i = 0, 1, \ldots, N_p - 1) \tag{1}$$

where $\boldsymbol{q_i}$ is the joint space configuration of the ith point on path $\boldsymbol{P}$. Specifically, $\boldsymbol{q_0}$ and $\boldsymbol{q_g}$ are the starting and goal configurations, respectively; the others are the via points in between. Each $\boldsymbol{q_i}$ has N_j DOFs, q_i^j represents the joint position of the jth joint of the ith point. We assume that the input path $\boldsymbol{P}$ is feasible and free from singularities, thus ensuring the absence of singularities along the trajectory.

The goal is to find a kinematically and dynamically feasible trajectory $\boldsymbol{q}(t)$. The following requirements are considered in the trajectory planning problem.

1) *Path-tracking:* The motion starts at the initial configuration, stops at the goal configuration, and passes all the via points.
2) *Time-efficiency:* The motion duration can be minimized as much as possible.
3) *Real-time:* The trajectory calculation is done within a single control cycle (usually several milliseconds).

IV. Receding Horizon Trajectory Generation With Global Kinematic Constraints

Considering a path specified by (1), calculating the time-optimized trajectory for the entire path becomes computationally expensive, particularly with an increasing number of via points.

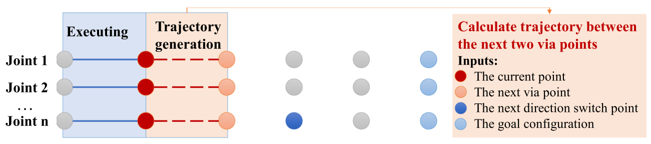


Fig. 2. Receding horizon online trajectory generation. During the motion execution, the robot takes the current point, the next via point, and the next direction switch point/ the goal configuration as inputs, and calculates the trajectory between the next two via points.

In addition, when environments or goals change, the trajectory requires replanning, leading to the discarding of the previously calculated portion. To conserve computational resources, one approach involves generating the trajectory on the fly as the robot proceeds, focusing only on a segment of the path at a time. However, it is crucial to account for global properties as well, such as ensuring the robot can stop at the goal configuration without exceeding kinematic and dynamic constraints while maximizing motion efficiency.

To prevent unnecessary computations for parts of the trajectory that will not be executed and optimize global properties, we propose a receding horizon online trajectory planning structure. This is inspired by the receding horizon control strategy, aiming to optimize the motion within each successive predefined horizon [31]. Our approach repeatedly generates local optimal trajectories while considering global kinematic feasibility. The whole process of the online path-tracking trajectory planning is shown in Algorithm 1 and Fig. 2. Upon receiving a new path, the direction-switch points for each joint are first computed on a global scale (Section IV-A). Subsequently, local trajectories between two via points are generated on the fly (Section IV-B).

A. Direction-Switch Point and Global Constraints

When we get a new path $\boldsymbol{P}$, the first step is to calculate the direction-switch points for each joint, where the motion of a joint changes direction

$$\boldsymbol{Q_{gv}^j} = \{q_{gv}^j = q_i^j \mid (q_{i+1}^j - q_i^j)(q_i^j - q_{i-1}^j) < 0$$
$$(i = 1, \ldots, N_p - 2)\} \; (j = 1, \ldots, N_j) \tag{2}$$

in which q_{gv}^j represents the position of the jth joint at a direction-switch point, $\boldsymbol{Q_{gv}^j}$ is the collection of all the direction-switch points of the jth joint.

At a direction-switch point, where the motion direction changes, the velocity is expected to be zero. Note that the direction-switch points of the different joints do not necessarily associate with the same via point. For example, for the configuration of a via point $\boldsymbol{q} = \{q^1, q^2, \ldots, q^{N_j}\}$, it is possible that $q^{j_1} = q_{gv}^{j_1}, q^{j_2} \neq q_{gv}^{j_2}, j_1 \neq j_2$. In particular, at the final goal of the entire path, it is essential for the robot to come to a stop, ensuring that the velocity of all joints is zero. Thus, a constraint is imposed to ensure that each joint can reach the direction-switch points and the final goal with zero velocity without violating any kinematic constraints. This constitutes a *global kinematic constraint* while calculating the local trajectory between two via points. Specifically, when calculating the local trajectory between two via points, we not only consider the current and the next via point, but also take into account the zero velocity constraint for the next direction-switch point of each joint or the final goal, despite that the direction-switch point does not belong to this path segment.

Incorporating this global constraint can mitigate the inherent shortsightedness of the local trajectory planning method and improve global feasibility and motion efficiency.

B. Receding Horizon Trajectory Generation

The global path-tracking trajectory planning problem consists of individual local time-optimal trajectory planning subproblems for each path segment $(\boldsymbol{q_i}, \boldsymbol{q_{i+1}})$. The local trajectory planning problem is updated and solved online during the motion execution. Employing a receding horizon structure, when the robot reaches the via point $\boldsymbol{q_i}$, the trajectory for the next path segment $(\boldsymbol{q_i}, \boldsymbol{q_{i+1}})$ is calculated, while the zero velocity constraint of the next direction-switch point for each joint $q^j_{gv} (j = 1, 2, \ldots, N_j)$ or the final goal $\boldsymbol{q_g}$ is considered. The step size of the receding horizon structure is $\boldsymbol{q_{i+1}} - \boldsymbol{q_i}$. This process is repeated till the robot reaches its final goal.

The local trajectory planning method exhibits the following properties that facilitate its online execution: 1) each subproblem is calculated within a control cycle; 2) the trajectory planning accommodates arbitrary velocities at each via point. As a result, the robot can swiftly respond to path updates in real time during motion execution. The local time-optimal trajectory generation method is detailed in Section V.

V. Analytical Closed-Form Formulation for Local Trajectory Generation

Previously, we decompose the trajectory planning problem with multiple via points into a sequence of local trajectory generation problems between each pair of via points in a receding horizon fashion. In this section, we address each local trajectory generation problem and solve the trajectory parameters for the corresponding path segment. We formulate the trajectory model and the local time-optimal trajectory planning method in an analytical form and subsequently present a closed-form solution to generate local trajectories.

A. Trajectory Model

We adopt the TVP as the trajectory model. TVP is a second-order function with low computational complexity. Further, the TVP's parameters have a straightforward kinematic interpretation, facilitating easy analytical handling. Due to these advantages, TVP is widely utilized in real-time applications [32]. TVP also achieves the fastest attainable motion under specified maximum velocity and acceleration, making it a preferred choice for time-optimal trajectory planning.

We present the TVP formulation for modeling each joint. To simplify the formulation, we avoid the superscript indicating the joint index in this section. Fig. 3 shows the standard forms of TVP, which consists of three phases: the acceleration phase

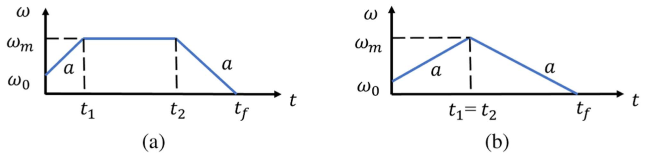


Fig. 3. Basic shapes of TVP. (a) TVP with trapezoidal shape and nonzero initial velocity. (b) TVP with triangular shape and nonzero initial velocity.

$t \in [0, t_1]$, the constant velocity phase $t \in [t_1, t_2]$, and the deceleration phase $t \in [t_2, t_f]$. Here, t_f represents the time at which the motion completes, indicating when the robot comes to a stop, and it also denotes the duration of the entire motion; t_1 and t_2 represent the time at which the acceleration phase ends and the deceleration phase starts, respectively. The acceleration and deceleration phases have the same absolute acceleration value a. The velocity during the constant velocity phase is ω_m, while the initial velocity is ω_0.

Considering the TVP for one joint, we assume that the initial velocity ω_0, initial position q_0, and the goal position q_g are known. To determine a TVP, there are five variables, $\omega_m, a, t_1, t_2, t_f$ (Fig. 3), linked by three equations

$$\frac{1}{2}\omega_m(t_f + t_2 - t_1) + \frac{1}{2}\omega_0 t_1 = q_g - q_0 \tag{3}$$

$$\omega_m = \omega_0 + a t_1 \tag{4}$$

$$\frac{\omega_m - \omega_0}{t_1} = \frac{\omega_m}{t_f - t_2}. \tag{5}$$

This implies that there are two independent variables, while the other variables can be obtained using (3)–(5).

With the above variables, the time law of the velocity is represented as

$$\omega(t) = \begin{cases} \omega_0 + at & 0 < t \leq t_1 \\ \omega_m & t_1 < t \leq t_2 \\ \omega_m - a(t - t_2) & t_2 < t \leq t_f. \end{cases} \tag{6}$$

The position profile can be accordingly obtained.

B. Local Time-optimal Trajectory Generation Between Two via Points

In this section, we proceed to formulate the subproblem of local time-optimal trajectory planning for each path segment between two via points, while accounting for the global kinematic constraints. Assume that the robot is currently at the configuration $\boldsymbol{q_i} = \{q_i^1, \ldots, q_i^{N_j}\}$ with the velocity $\boldsymbol{\omega_0} = \{{\omega_0}^1, \ldots, {\omega_0}^{N_j}\}$, and the next via point is $\boldsymbol{q_{i+1}} = \{q_{i+1}^1, \ldots, q_{i+1}^{N_j}\}$. For joint j, the next direction-switch position is q^j_{gv}. Given $(\boldsymbol{\omega_0}, \boldsymbol{q_i}, \boldsymbol{q_{i+1}}, q^1_{gv}, \ldots, q^{N_j}_{gv})$, the problem is to find the time-optimal local trajectory for the path segment from $\boldsymbol{q_i}$ to $\boldsymbol{q_{i+1}}$, while considering the kinematic constraints that ensure each joint j stops at the next direction-switch point q^j_{gv}.

In this local trajectory planning problem, motion duration synchronization is essential for serial robots with multiple DOFs. It ensures that the robot passes through a specific via point with all its joints arriving at their corresponding positions

Algorithm 1: Online Time-Minimal Path-Tracking Trajectory Planning.

```
Input: $\boldsymbol{P} = \{\boldsymbol{q_0}, \boldsymbol{q_1}, \ldots, \boldsymbol{q_{N_p-1}}\}$, $\boldsymbol{\omega_0}$
Output: $\boldsymbol{q}(t)$
1:  Calculate direction-switch points
2:    for $i = 0, \ldots, N_p - 2$ do
3:      for $j = 1, \ldots, N_j$ do
4:        $CalcuTimeMinimalTVP$ (Section V-C)
5:        Calculate $t^1_{i+1}, \ldots, t^{N_j}_{i+1}$
6:      end for
7:      $DetermineSyncTime$ (Section V-D)
8:      for $j = 1, \ldots, N_j$ do
9:        $AdjustTVP$ (Section V-E)
10:   end for
11: end for
```

Algorithm 2: Time-Minimal TVP for Each Joint.

```
Input: $\omega_0,\ q_0,\ q_g,\ a_{\max},\ \omega_{\max}$
Output: $\omega_m,\ a,\ t_1,\ t_2,\ t_f$
1:  $a = a_{\max}$
2:  $\omega_{tri} = \sqrt{{\omega_0}^2/2 + a(q_g - q_0)}$
3:  if $\omega_{tri} < \omega_{\max}$ then
4:    $\omega_m = \omega_{tri}$
5:    $t_f = \frac{2\omega_m - \omega_0}{a}$
6:  else
7:    $\omega_m = \omega_{\max}$
8:    $t_f = (q_g - q_0)/\omega_m + (\omega_m - \omega_0 + {\omega_0}^2/2\omega_m)/a$
9:  end if
10: $t_1 = (\omega_m - \omega_0)/a$
11: $t_2 = t_f - \omega_m/a$
```

simultaneously. However, this synchronization introduces coupling among the motions of different joints, leading to high-dimensional planning problems that are difficult to solve. Even with straightforward trajectory models like TVP, the optimization problems can still be high-dimensional and nonconvex [32], posing challenges for achieving efficient and accurate solutions. Consequently, this complexity hinders real-time applications in trajectory planning.

To address this issue, we propose a decoupling method to convert the multijoint coupling problem into several low-dimensional independent problems, which can be handled analytically. Specifically, it is achieved by (1) a method for determining the coupling parameter among each subproblem, i.e., the synchronous time, and (2) an algorithm for adapting TVP based on the variable synchronous time.

The overall procedure for local time-optimal trajectory generation is as follows (Lines 3–10 in Algorithm 1).

1) *$CalcuTimeMinimalTVP$ (Line 4):* For each joint j, calculate a trajectory profile $q^j(t)$ with minimal motion duration from the current configuration to the next direction-switch point or the goal configuration without concerning via points.
2) *Calculate via time (Line 5):* Based on the trajectory profile for each joint j obtained by Step (1), calculate the time $t_{via}{}^j_{i+1}$ when the expected joint position at the next via point is reached by
$$q^j(t_{via}{}^j_{i+1}) = q^j_{i+1}. \tag{7}$$
3) *$DetermineSyncTime$ (Line 7):* Select a synchronous time for the next via point (represented as $t_{via\,i+1}$), based on $t_{via}{}^1_{i+1}, \ldots, t_{via}{}^{N_j}_{i+1}$ obtained by Step 2), satisfying that all the joints can reach q^j_{i+1} at $t_{via\,i+1}$ and can stop at q^j_g without exceeding velocity and acceleration limits.
4) *$AdjustTVP$ (Line 9):* Adjust TVP parameters for each joint according to the fixed synchronous time $t_{via\,i+1}$.

The local trajectory between two via points obtained by this method is time-optimal. The motion duration is optimized by initializing the trajectory with the time-minimal TVP (Section V-C), selecting the minimal allowed synchronous time considering all the joints (Section V-D), and accordingly adjusting the trajectory segment for each joint (Section V-E). The details for solving the problem of each step are presented in the following sections.

C. Time-Minimal TVP Generation

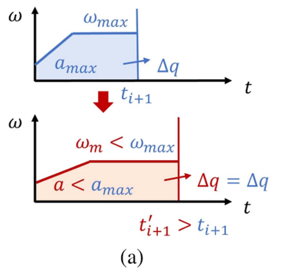
As sketched in Section V-B and Algorithm 1, in Step (1), the constraint of all the joints reaching the via point at the same time is relaxed. Instead, we calculate a time-minimal point-to-point trajectory for each joint traveling from the current position to the next position with zero velocity (either be the next direction-switch point or the final goal). The algorithm is shown in Algorithm 2. To simplify the notation, we omit the superscript joint index. $a_{\max}$ and $\omega_{\max}$ represent the maximal allowed acceleration and the maximal allowed velocity for a joint, respectively.

D. Synchronous Time Determination

Synchronous time determination plays a crucial role in solving the problem by contributing in the following two main aspects: 1) it guarantees the satisfaction of global kinematic constraints, ensuring that each joint can stop at the next direction-switch point or the final goal; 2) it enables the decoupling of the trajectory planning problem into N_j simpler problems. Each of these individual problems involves finding a TVP for each joint while incorporating the additional synchronous time constraint.

After getting the time-minimal TVP parameters for each joint, the trajectory is formulated by (6). As $q(t)$ is a monotonous function, we can get the moment when the joint will reach the position for the next via point, denoted as $t_{via}{}^j_{i+1}$. To guarantee the robot passes the next via point, all the joints need to reach the position for the next via point at the same time (dubbed as the synchronous time and denoted as $t_{via\,i+1}$), i.e., $t_{via}{}^1_{i+1} = t_{via}{}^2_{i+1} = \ldots = t_{via}{}^{N_j}_{i+1} = t_{via\,i+1}$.

The synchronous time $t_{via\,i+1}$ is determined by considering the global kinematic constraints for all the joints. If the TVP starts at the acceleration phase or constant velocity phase [shown as Fig. 4(a)], i.e., ${t_2}^j > 0$, the traveling time should be no smaller than $t_{via}{}^j_{i+1}$ to ensure the kinematic feasibility. Selecting a



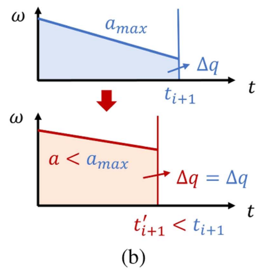


Fig. 4. Kinematically feasible adjustment of time passing the next via point. (a) When starting at the acceleration phase or constant velocity phase, the kinematically feasible next via time is no smaller than the next via time obtained by the time-minimal TVP. (b) When starting at the deceleration phase, the kinematically feasible next via time is no larger than the next via time obtained by the time-minimal TVP.

smaller a or/and ω_m leads to a larger $t_{via}{}_{i+1}^{j}$. Otherwise, if the TVP starts at the deceleration phase [shown as Fig. 4(b)], the traveling time should be no larger than $t_{via}{}_{i+1}^{j}$ to ensure the kinematic feasibility. Selecting a smaller a leads to a smaller $t_{via}{}_{i+1}^{j}$. Thus, we get the minimum allowed time for reaching the next via point of each joint

$$t_{via}{}_{i+1\min}^{j} = \begin{cases} t_{via}{}_{i+1}^{j} & t_2{}^j > 0 \\ 0 & t_2{}^j = 0. \end{cases} \tag{8}$$

To ensure the kinematic feasibility of all the joints, the synchronous time is selected as $t_{via}{}_{i+1} = \max_j t_{via}{}_{i+1\min}^{j}$.

E. TVP Adjustment With Motion Duration

With the fixed synchronous time, we can treat the joints separately. The problem now becomes finding a trajectory for each joint j (represented as $q^j(t)$), given the current position (at the ith via point) q_i^j, the current velocity ω_0^j, and the position of the next via point q_{i+1}^j, satisfying

$$q^j(t_{i+1}) = q_{i+1}^j \tag{9}$$

which means the joint will pass the next via point position q_{i+1}^j at the synchronous time $t_{via}{}_{i+1}$.

We propose a method to adapt the TVP based on the variable synchronous time. In contrast to the traditional TVP calculations, our method eliminates the need for fixed velocities of the via points or other free variables. Instead, it categorizes potential motion patterns based on the relationship between the initial velocity and traveling distance. For each recognized situation, a corresponding algorithm is developed.

Since the trajectory for each joint is solved independently, we focus on the TVP for a single joint in our explanation. For simplicity, we avoid the superscript joint index. The current joint position, the current velocity, the joint position for the next via point, and the synchronous time for reaching the next via point are represented as q_0, ω_0, q_{via}, and t_{via}, respectively.

As mentioned in Section V-A, a TVP can be specified using five variables, namely $\omega_m, a, t_1, t_2, t_f$. Equations (3)–(5) and (9) constitute four equation constraints, leaving one free variable for the TVP adjustment. The time-minimal TVP can be an initialization for this adjustment. If the TVP starts at the acceleration or constant velocity phases (i.e., $t_2 > 0$), we can increase the

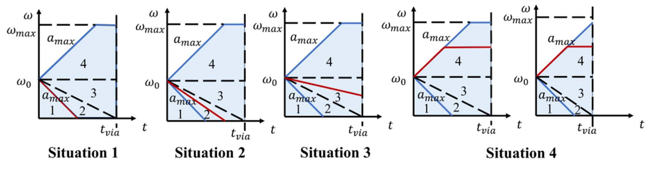


Fig. 5. TVPs for moving q_{via} within t_{via} (an example TVP for each situation is illustrated with the red line.).

motion duration by decreasing a and/or ω_m. Otherwise, if the TVP starts at the deceleration phase ($t_2 = 0$), we can reduce the motion duration by decreasing a.

Given a fixed t_{via}, we define four situations based on the relation between the initial velocity ω_0 and the moving distance $q_{via} - q_0$. Fig. 5 depicts the feasible ranges of moving distances for a certain ω_0 with shadowed regions. These situations are defined in ascending order of moving distances and are referred to as Situations 1–4. Examples of Situations 1–4 are illustrated in Fig. 5. The calculation of TVP parameters is detailed in Algorithm 3. "——" indicates that the parameter is not applicable in this situation.

Situation 1: With a fixed t_{via}, the initial velocity ω_0 and the moving distance $q_{via} - q_0$ satisfy

$$q_{via} - q_0 \in \left(0, \frac{1}{2}\ \omega_0 t_{via}\right] \tag{10a}$$

$$q_{via} - q_0 < \frac{\omega_0{}^2}{2a_{\max}}. \tag{10b}$$

Inequality (10a) indicates that $q_{via} - q_0$ is so small and the joint has to stop before t_{via}. Inequality (10b) indicates that even the joint decelerates with $a_{\max}$, it still cannot stop at q_{via}.

In this situation, the TVP only includes the deceleration phase, resulting in $t_1 = t_2 = 0$ and $\omega_m = \omega_0$. To respect the kinematic constraint, we select $a = a_{\max}$, but this compromise the tracking accuracy, leading to the joint exceeding the expected position. Note that this situation only occurs when approaching the end of the motion, where $q_{via} - q_0$ is small while ω_0 remains relatively large. Imposing global kinematic constraints (as introduced in Section IV-A) discourages infeasibly large ω_0 and prevents this situation. Accordingly we can determine the motion duration t_f.

Situation 2: With a fixed t_{via}, the initial velocity ω_0 and the moving distance $q_{via} - q_0$ satisfy

$$q_{via} - q_0 \in \left(0, \frac{1}{2}\ \omega_0 t_{via}\right] \tag{11a}$$

$$q_{via} - q_0 \geq \frac{\omega_0{}^2}{2a_{\max}}. \tag{11b}$$

Inequality (11a) is the same as Situation 1. Inequality (11b) is in contrast to Situation 1, indicating that feasible acceleration exists to allow the robot to stop at q_{via}.

Similar to Situation 1, the TVP includes only the deceleration phase, resulting in $t_1 = t_2 = 0$ and $\omega_m = \omega_0$. By considering the robot stops at q_{via}, We can calculate the motion duration t_f, and accordingly obtain the acceleration a.

Situation 3: With a fixed t_{via}, the initial velocity ω_0 and the moving distance $q_{via} - q_0$ satisfy $q_{via} - q_0 \in (\frac{1}{2}\omega_0 t_{via}, \omega_0 t_{via}]$. Compared to Situation 1 and 2, as the moving distance

Algorithm 3: TVPs for Passing q_{via} at t_{via}.

```
  Input: $\omega_0,\ q_0,\ q_{via},\ t_{via},\ a_{\max},\ \omega_{\max}$
  Output: $\omega_m,\ a,\ t_1,\ t_2,\ t_f$
 1: if $q_{via} - q_0 \in (0, \frac{1}{2}\omega_0 t_{via}]$ then
 2:    if $q_{via} - q_0 < \frac{{\omega_0}^2}{2a_{\max}}$ then (Situation 1)
 3:       $\omega_m = \omega_0$, $a = a_{\max}$, $t_1 = 0$, $t_2 = 0$, $t_f = \frac{\omega_0}{a_{\max}}$
 4:    else (Situation 2)
 5:       $\omega_m = \omega_0$, $t_1 = 0$, $t_2 = 0$, $t_f = \frac{2(q_{via}-q_0)}{\omega_0}$,
          $a = \frac{\omega_0}{t_f}$
 6:    end if
 7: else if $q_{via} - q_0 \in (\frac{1}{2}\omega_0 t_{via}, \omega_0 t_{via}]$ then (Situation 3)
 8:    $\omega_m = \omega_0$, $t_1 = 0$, $t_2 = 0$, $a = 2\frac{\omega_0 t_{via} - (q_{via}-q_0)}{{t_{via}}^2}$,
       $t_f = --$
 9: else (Situation 4)
10:    $a = a_{\max}$, $t_1 = t_{via} - \sqrt{{t_{via}}^2 - \frac{2(q_{via}-q_0-\omega_0 t_{via})}{a}}$,
       $\omega_m = \omega_0 + a t_1$, $t_2 = --$, $t_f = --$
11: end if
```

becomes larger, the joint passes t_{via} with nonzero velocity. Thus, t_f is not applicable in this situation. However, the moving distance is still relatively small and the TVP still contains the deceleration phase only.

Situation 4: With a fixed t_{via}, the initial velocity ω_0 and the moving distance $q_{via} - q_0$ satisfy $q_{via} - q_0 \geq \omega_0 t_{via}$. In this situation, the traveling distance $q_{via} - q_0$ is relatively large, the joint must accelerate to reach q_{via} at t_{via}. The TVP contains the acceleration phase and the constant velocity phase. The joint passes t_{via} with nonzero velocity. Thus, t_2 and t_f is not applicable. As previously mentioned, there is one free variable. The joint is set to work with $a_{\max}$, which has a longer cruise stage, resulting in smoother motion. Then, t_1 and ω_m can be calculated accordingly.

Algorithms 2 and 3 only require simple arithmetic operations, resulting in low computational overhead.

VI. Dealing With Dynamic Factors

One notable advantage of the presented method is its real-time trajectory calculation, which means that the trajectory can be computed within one control loop. As soon as the path is updated, the trajectories are promptly recalculated based on the new path, enabling the robot to immediately start following the updated trajectory. This capability empowers the method to effectively handle dynamic and unpredictable situations, making it highly responsive and adaptable to changes in the environment or task requirements. In this section, we show how the presented method deals with two typical dynamic situations: 1) dynamic obstacles; and 2) changing targets.

A. Dynamic Obstacles

Consider a dynamic obstacle is present in the common workspace and poses interference with the robot's motion, resulting in intersections between the moving obstacle's path and the robot's path. We make no assumption about the shape of

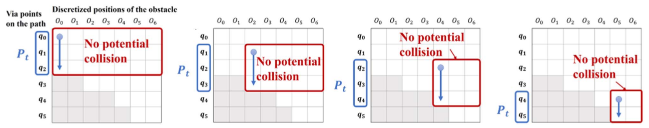


Fig. 6. Dynamic obstacle avoidance by updating temporary path and trajectory replanning. Potential collisions between pairs of sample robot configurations and obstacle positions are detected and depicted with gray color. As the obstacle and the robot proceed, the temporary path $\boldsymbol{P_t}$ is constantly updated according to the current motion state, which is fed into the trajectory planner.

the obstacle. The potential collision is detected by representing the geometry of the robot and the obstacle in the task space and checking whether they overlap with each other.

Assuming the path of the obstacle can be predicted, we first perform the collision checking between the path of the robot and the path of the obstacle, and determine where potential collisions can happen. Then, we extract part of the original path $\boldsymbol{P}$ as a temporary path to feed into the trajectory planner, represented as $\boldsymbol{P_t} = \{\boldsymbol{q_{i_0}}, \boldsymbol{q_{i_1}}, \ldots, \boldsymbol{q_{N_{p_t}-1}}\} \subseteq \boldsymbol{P}$, where $\boldsymbol{q_{i_0}}$ is the configuration of the via point the robot is currently moving towards, N_{p_t} is the number of via points of $\boldsymbol{P_t}$. Feasible solutions leading the robot to the final goal without collision can be found by properly updating $\boldsymbol{P_t}$, as long as the robot's path is not blocked by the entire path of the obstacle [33].

An example is illustrated in Fig. 6. The potential collisions are represented with a diagram. The column represents the via points on the path ($\boldsymbol{q_i}$ is the configuration of the ith via point). The row represents the discretized positions of the obstacle along its path (O_k represents the kth sample on the path). Grid (i, k) is depicted with gray if the robot at $\boldsymbol{q_i}$ and the obstacle at O_k overlap in the task space. In the beginning, the obstacle is located at O_0, and potential collisions may occur when the robot reaches $\boldsymbol{q_2}$. To address this, we create a temporary path $\boldsymbol{P_t} = \{\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}\}$ to feed into the trajectory planner. This temporary path is entirely collision-free with the obstacle's entire path. When the obstacle moves to O_2, $\boldsymbol{q_3}$ becomes collision-free with the remaining portion of the obstacle's path. Assuming the robot is currently moving towards $\boldsymbol{q_1}$, the temporary path is updated to $\boldsymbol{P_t} = \{\boldsymbol{q_1}, \boldsymbol{q_2}, \boldsymbol{q_3}\}$. This process continues, when the obstacle reaches O_5, the rest of the robot's path is freed up, and the final configuration of the temporary path is set to $\boldsymbol{q_5}$.

As both the obstacle and the robot progress, the temporary path $\boldsymbol{P_t}$ is continually updated. Whenever the trajectory planner receives a path update, Algorithm 1 is initiated again, with the updated temporary path $\boldsymbol{P_t}$ serving as the input. To ensure velocity continuity, the initial velocity $\boldsymbol{\omega_0}$ is set to the expected velocity for reaching $\boldsymbol{q_{i_0}}$ based on the current trajectory. As the trajectory planner receives the updated $\boldsymbol{P_t}$, the robot may be moving between via points, heading towards $\boldsymbol{q_{i_0}}$. The trajectory planner promptly calculates the direction-switch points (Line 1 in Algorithm 1). Upon reaching $\boldsymbol{q_{i_0}}$, the trajectory from $\boldsymbol{q_{i_0}}$ to $\boldsymbol{q_{i_1}}$ is calculated using Lines 3–10 in Algorithm 1. This calculation is completed within one control cycle, enabling real-time trajectory adjustments during motion execution. Consequently, the robot replans and executes the trajectory on the fly while maintaining continuous velocity.

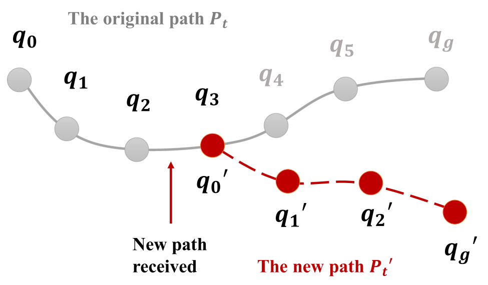


Fig. 7. Changing target handling by online trajectory re-planning according to path update. When the robot is moving towards q'_0 along path $\boldsymbol{P}$, new path $\boldsymbol{P}'$ is received and the trajectory is replanned online accordingly.

B. Changing Targets

In some situations, the targets may change during motion execution, requiring prompt reaction of the robot.

A new path $\boldsymbol{P}' = \{q'_0, q'_1, \dots, q'_g\}$ is recalculated and fed into the trajectory planner. The starting configuration of the new path q'_0 is the configuration of the via point the robot is currently moving towards, q'_1 is the first new via point on the new path, and the final configuration q'_g is the new target. An example is illustrated in Fig. 7. The original path is depicted with a grey solid line. The path is updated when the robot is moving towards q_3. The new path is $\boldsymbol{P}' = \{q'_0, q'_1, q'_2, q'_g\}$, where $q'_0 = q_3$, as depicted with the red dashed line.

Similar to Section VI-A, when the trajectory planner receives the path update, Algorithm 1 is started over. The input path is $\boldsymbol{P}'$. The initial velocity ω_0 is set to the expected velocity reaching q'_0 according to the current trajectory, which guarantees the velocity continuity. When the trajectory planner receives a new path, the robot may be moving between via points towards q'_0. The trajectory planner starts calculating the direction-switch points immediately (Line 1 in Algorithm 1). When the robot reaches q'_0, Lines 3–10 in Algorithm 1 is performed to calculate the trajectory from q'_0 to q'_1, which is performed within one control cycle. In this way, with the path updated during the motion execution, the trajectory is replanned and executed on the fly with continuous velocity.

VII. Discussions

A. Computational Efficiency

The combination of the receding horizon structure (Section IV) and the analytical closed-form solution (Section V) empowers the presented framework with promising computational efficiency and the capability to handle dynamic factors.

With the receding horizon structure, the trajectory planning problem following the path specified by N_p via points is decoupled into $N_p - 1$ subproblems, each considering only the current via point, the next via point, and the next direction-switch position for each joint. This not only largely reduces the computational complexity, but also renders the computational overhead irrelevant to the number of via points. In contrast, global time-optimal scaling methods (such as [6], [11], [14], [15]) experience an increase in computational effort as the number of the via points grows. The computational concern limits the selection of via point size and consequently leads to compromises of accuracy and time-optimality.

In addition to the receding horizon structure, each subproblem for a path segment is further decoupled into N_j problems of a single joint. By employing the TVP model (Section V-A), the calculation involves only simple algebraic operations, as shown in Algorithms 1–3. The computational complexity of calculating a path segment is $O(N_j)$. The computation time is tested in Section VIII-B.

Further, the receding horizon structure of calculating just one path segment is well suited to handle online changes, as shown in Section VI. It avoids wasting resources in calculating trajectories that will be discarded due to path changes. The joint decoupling, the TVP model and the analytical closed-form solution collectively achieve low computational complexity, enabling rapid trajectory calculations capable of responding to updated paths during motion execution. The experimental tests are shown in Sections VIII-A2 and VIII-A3.

B. Time Efficiency

Since the trajectory is generated locally without considering the full global information of the path, and online adjustments may be made to deal with unpredictable situations, the obtained trajectory is not guaranteed to be globally time-optimal. However, we aim to reduce the motion duration of the whole process by minimizing the time duration of the trajectory for each path segment.

In comparison, TOPP-RA [15] globally calculates the time-optimal path-tracking trajectory and achieves asymptotic time-optimal, meaning that the motion duration converges to the optimal value as the discretization step size approaches zero. However, with the larger path discretization size N_p, the computation time increases (the computational complexity is $O(mN_p)$, where m is the number of inequality constraints). In practice, achieving actual time optimality is unattainable as we cannot make the discretization step size infinitely small. The experimental comparison of the motion duration is presented in Section VIII-D.

C. Tracking Accuracy

The trajectory passes the via point accurately if a feasible solution exists (in Situations 2, 3, and 4). This accuracy is guaranteed by incorporating the subsequent via point as an input to the algorithm (q_{via} in Algorithm 3) during local trajectory planning for each path segment. Tracking error manifests in Situation 1, as kinematic constraints hinder reaching the expected position. However, as discussed in Section V, Situation 1 rarely happens in practical cases, and is further discouraged by considering the global kinematic constraints.

VIII. Results

To demonstrate the proposed approach in achieving path-tracking trajectory generation with high motion efficiency, high tracking accuracy and dynamic factor dealing capability, a series

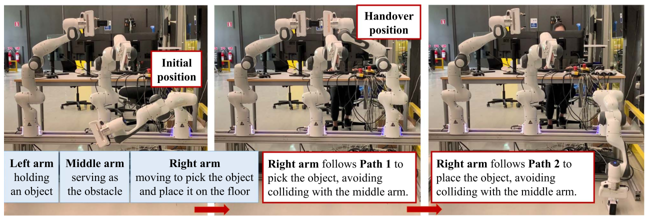

Initial position
Handover position
Left arm holding an object
Middle arm serving as the obstacle
Right arm moving to pick the object and place it on the floor
Right arm follows Path 1 to pick the object, avoiding colliding with the middle arm.
Right arm follows Path 2 to place the object, avoiding colliding with the middle arm.

Fig. 8. Static obstacle avoidance experiment. The right arm picks up the object held by the left arm (following path 1) and places it on the floor (following path 2), avoiding colliding with the middle arm.

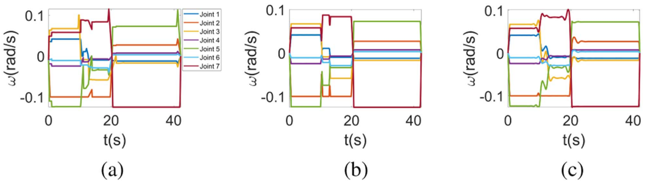


Fig. 9. Velocity profiles for static obstacle avoidance example. (a) Our method. (b) TOTG. (c) TOPP-RA.

of experiments and evaluations are conducted. We first separately evaluate the computational efficiency, tracking accuracy and motion efficiency of the present method. Furthermore, the performance is validated through experiments, which demonstrate the capability to handle both static and dynamic obstacle avoidance and effectively track changing targets. We compare our method against the classical offline NI-based method TOTG [14] and TOPP-RA [15].

The algorithm is implemented with C++ and conducted on a laptop equipped with an Intel Core i9-9980HK 5.0 GHz CPU with 32 GB RAM. The experiments are performed with Franka–Emika Panda robots and their control platform, with a control cycle of 1 ms. The maximum allowed velocity and acceleration are set to $\boldsymbol{\omega}_{\max} = \{0.1000, 0.1000, 0.1000, 0.1000, 0.1250, 0.1250, 0.1250\} rad/s$ and $\boldsymbol{a}_{\max} = \{0.3750, 0.1875, 0.2500, 0.3125, 0.3750, 0.5000, 0.5000\}$ rad/s^2, respectively.

A. Experiments

We performed experiments in a hand-over scenario. Three experiments are performed, as shown in the video attachment. The first one (Section VIII-A1) shows the robot following a given path to avoid collision with static obstacles. The second one (Section VIII-A2) and the third one (Section VIII-A3) show dynamic factors handling with the presented method. The dynamic factors consist of other moving robots in the shared workspace and human intervention.

1) Static Obstacle Avoidance: As depicted in Fig. 8, the left arm is holding an object, while the right arm is tasked with picking up the object and placing it on the floor, without colliding with the middle arm. The whole process consists of two paths for the right arm. Path 1 spans from the bottom (initial position) to the top (handover position), while Path 2 spans from the top to the floor. The paths are planned by MoveIt.[1] The two paths, respectively, consist of 42 and 36 configurations (including initial point, via points, and the target). Then trajectories are generated by the presented method, which ensures that the robot passes all the configurations of the path and reaches the final target in the shortest possible time. The motion curves are shown in Fig. 9.

2) Dynamic Obstacle Avoidance: We consider a similar task to that in Section VIII-A1; however, here, the middle arm initially obstructs the path of the right arm and moves away later. To handle this, the right arm adjusts the trajectory online according to the motion state of the middle arm.

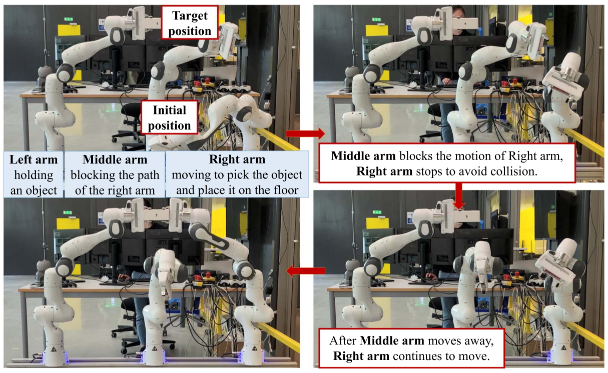

Target position
Initial position
Left arm holding an object
Middle arm blocking the path of the right arm
Right arm moving to pick the object and place it on the floor
Middle arm blocks the motion of Right arm, Right arm stops to avoid collision.
After Middle arm moves away, Right arm continues to move.

Fig. 10. Dynamic obstacle avoidance experiment. The right arm picks up the object held by the left arm following path 3, which is blocked by the middle arm. When entering the collision region, the right arm stops to avoid collisions. After the middle arm moves away, the right arm continues to move.

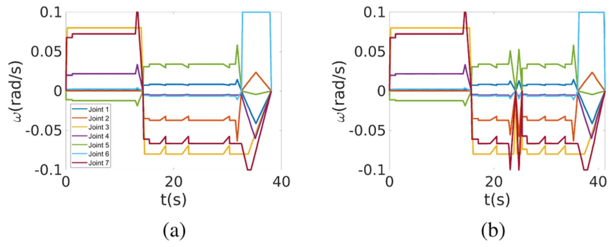


Fig. 11. Velocity profile for dynamic obstacle avoidance example. (a) Continuous. (b) Stop.

While the path of the middle arm is known, when and how fast it moves are uncertain. We conduct collision checks to identify potential collisions between the paths of the right and the middle arm in the way presented in Section VI-A. During the motion of the middle arm, the temporary path for the right arm is constantly updated based on the collision checking results. The trajectory is accordingly recalculated based on the updated path. The motion of the right arm flexibly reacts to the motion of the middle arm. In the situation the middle arm moves away earlier, and the right arm moves to the targets without stopping. In another situation shown in Fig. 10, the middle arm has not yet moved away when the right arm reaches the potential collision region, the right arm stops to avoid collision and resumes its motion once the middle arm moves away from the collision region. The motion curves of these two situations are shown in Fig. 11.

[1]Ioan A. Sucan and Sachin Chitta, “MoveIt”, [Online] Available at moveit.ros.org.

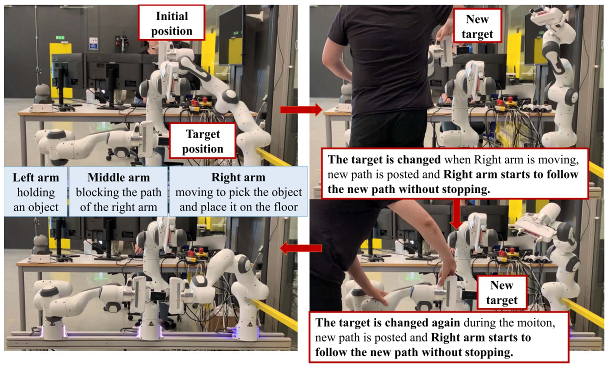

Initial position
New target
Target position
Left arm holding an object
Middle arm blocking the path of the right arm
Right arm moving to pick the object and place it on the floor
The target is changed when Right arm is moving, new path is posted and Right arm starts to follow the new path without stopping.
New target
The target is changed again during the moiton, new path is posted and Right arm starts to follow the new path without stopping.

Fig. 12. Changing targets experiment. When the right arm is moving toward the target, the target is changed manually to an arbitrary position. The right arm starts to follow the new path without stopping.

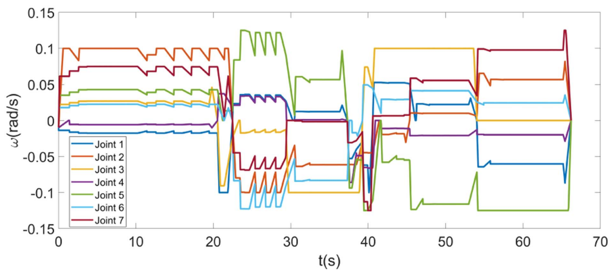


Fig. 13. Velocity profile for changing targets example.

3) Changing Targets: As shown in Fig. 12, while the right arm is moving from the initial position towards the target to pick the object, we manually change the target to an arbitrary new position, which is not predicted by the right arm. In response, the path is re-planned based on the new target, and the trajectory is then calculated according to the updated path. The robot smoothly transitions to the new target without any stopping. The process is repeated twice, and the robot consistently follows the updated path, ultimately reaching the final target for handover. Note that after the target is altered, there are some delays before the robot changes the moving direction. This is attributed to the time taken for path replanning. Nevertheless, as soon as the path is updated, the trajectory is immediately calculated according to the new path.

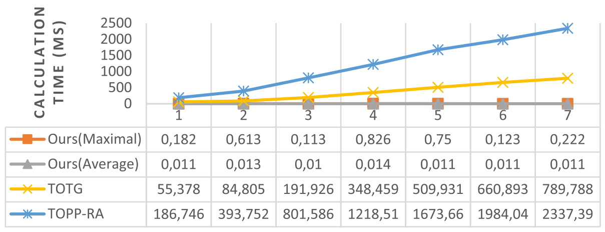
The velocity profile for the whole process is shown in Fig. 13. The smoothness of the motion is compromised for flexibility and motion efficiency. The velocity undergoes frequent changes during some periods. This behavior arises from the method considering only the next via point but not the subsequent via points, as they may change in the future. In addition, the minimization of time duration also contributes to the nonsmoothness of the motion, as the method strives to work at the maximal acceleration, leading to switches between acceleration and deceleration.

B. Computational Efficiency

The calculation time for the paths in Section VIII-A is shown in Table I. Paths 1–4 represent initial position to handover position in Section VIII-A1, handover position to placing at the floor in Section VIII-A1, initial position to handover position in Section VIII-A2, and initial position to the final handover position in Section VIII-A3, respectively. We compare our method against TOTG and TOPP-RA. In the TOTG implementation, the time step and the maximally allowed deviation from the path are set to 0.001 and 0.001, respectively. For the TOPP-RA implementation, the grid size is set to ten times the via points.

TABLE I
CALCULATION TIME (MS)

	Nvia	Ours(Avg.)	Ours(Max.)	TOTG	TOPP-RA
Path 1	42	0.011	0.405	54.527	165.682
Path 2	36	0.011	0.152	30.841	135.023
Path 3	31	0.010	0.258	39.156	122.108
Path 4	73	0.011	0.201	108.013	299.639



	1	2	3	4	5	6	7
Ours(Maximal)	0,182	0,613	0,113	0,826	0,75	0,123	0,222
Ours(Average)	0,011	0,013	0,01	0,014	0,011	0,011	0,011
TOTG	55,378	84,805	191,926	348,459	509,931	660,893	789,788
TOPP-RA	186,746	393,752	801,586	1218,51	1673,66	1984,04	2337,39

Fig. 14. Calculation time with different numbers of via points.

Our method is orders of magnitude faster than TOTG and TOPP-RA. The calculation is done in every control cycle (1 ms). Table I shows the maximum and average calculation time, which is the time taken for calculating the desired joint position and velocity of the next step within each control cycle. Note that the calculation time for the first control cycle of a path segment is comparatively longer due to the calculation of the TVP parameters (Lines 3–10 in Algorithm 1), which contributes to the reported maximum calculation time. The subsequent control cycles entail solely the calculation of positions and velocities using (6), which is straightforward. In either situation, the calculation time is much less than 1 ms, ensuring real-time execution.

The fast calculation is achieved by virtue of the combination of the receding horizon structure and the analytical closed-form solution, as discussed in Section VII-A. Furthermore, we test the calculation time with varying numbers of via points. As shown in Fig. 14, unlike TOTG and TOPP-RA, the calculation time remains constant regardless of the number of via points, owing to the receding horizon structure.

C. Tracking Accuracy

We measure the tracking accuracy by tracking error. The tracking error for the ith path segment is calculated as $\min_{t_{f_{i-1}} \leq t < t_{f_i}} \|\boldsymbol{q}(t) - \boldsymbol{q_i}\|$, indicating the discrepancy between the calculated trajectory $\boldsymbol{q}(t)(t_{f_{i-1}} \leq t < t_{f_i})$ and the predefined via point $\boldsymbol{q_i}$. The average and maximum tracking errors over the entire path are shown in Table II. As presented in Section V, the trajectories are analytically calculated using configurations of via points as inputs, leading to accurate tracking in typical scenarios. Minor errors stem from trajectory discretization. Situation 1 introduced in Section V-E presents a rare

TABLE II
TRACKING ERROR

	Path 1	Path 2	Path 3	Path 4
Average	2.00×10^{-3}	2.30×10^{-3}	3.40×10^{-5}	1.20×10^{-3}
Maximal	7.59×10^{-2}	7.21×10^{-2}	1.25×10^{-4}	5.30×10^{-2}

TABLE III
MOTION DURATION OF THE ENTIRE PATHS (S)

	Path 1	Path 2	Path 3	Path 4
Ours	42.229	26.504	40.374	66.209
TOTG	42.417	26.814	40.231	65.831
TOPP-RA	42.039	26.401	40.481	64.573

case where kinematic constraints prevent reaching the expected position, resulting in larger tracking error. Nevertheless, such situation is infrequent, and the resulting errors remain within the acceptable bounds.

D. Time Efficiency

The global time efficiency is evaluated by the motion duration of the entire path, shown in Table III. We select TOTG and TOPP-RA as baselines for global time-optimality, as they globally calculate the time-optimal path-tracking trajectory. The time duration achieved by our method is comparable to the global optimal results, and in some cases, even shorter. This is because our method always selects the largest possible acceleration, as illustrated in Fig. 9. While TOTG and TOPP-RA are not executed with the resolution required to obtain the true time-optimal solutions, as doing so would largely increase the calculation time.

IX. Conclusion

In this article, we introduce a real-time trajectory planning method designed to track a given path, specified by a list of via points, with near-minimal traveling time. The proposed approach incorporates a receding horizon method for calculating local trajectories while considering global kinematic constraints. We also develop algorithms for calculating local TVPs in closed form and introduce methods for handling dynamic factors. In our experiments, we test the method in handover scenarios involving interactions between the robot and other robots and humans. The computational overhead is on the order of 10^{-5} seconds, enabling real-time re-planning of the remaining trajectory during motion execution. The experiments demonstrate the proposed method's capability to react online to unpredictable situations, such as dynamic obstacles and changing targets, with only a marginal compromise in time efficiency. Importantly, in all experiments, the trajectory accurately tracked the given via points.

For future improvements, we plan to enhance the smoothness of the motion. In addition, we are interested in exploring the applications of these technologies in multirobot coordination and human–robot interaction scenarios.

Acknowledgement

The authors would like to thank Prof. Federico Pecora (Örebro University) for providing the experimental equipments and engaging in valuable discussions.

References

[1] K. M. Lynch and F. C. Park, *Modern Robotics*. New York, NY, USA: Cambridge Univ. Press, 2017.

[2] A. Rout, B. Deepak, and B. Biswal, "Advances in weld seam tracking techniques for robotic welding: A review," *Robot. Comput.-Integr. Manuf.*, vol. 56, pp. 12–37, 2019.

[3] S. Yuwen, J. Jinjie, X. Jinting, C. Mansen, and N. Jinbo, "Path, feedrate and trajectory planning for free-form surface machining: A state-of-the-art review," *Chin. J. Aeronaut.*, vol. 35, no. 8, pp. 12–29, 2022.

[4] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *Int. J. Robot. Res.*, vol. 4, no. 3, pp. 3–17, 1985.

[5] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[6] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1533–1540, Dec. 2014.

[7] P. Shen, X. Zhang, and Y. Fang, "Essential properties of numerical integration for time-optimal path-constrained trajectory planning," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 888–895, Apr. 2017.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. D. Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.

[9] J.-w. Ma, S. Gao, H.-t. Yan, Q. Lv, and G.-q. Hu, "A new approach to time-optimal trajectory planning with torque and jerk limits for robot," *Robot. Auton. Syst.*, vol. 140, 2021, Art. no. 103744.

[10] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE J. Robot. Autom.*, vol. 3, no. 2, pp. 115–123, Apr. 1987.

[11] E. Barnett and C. Gosselin, "A bisection algorithm for time-optimal trajectory planning along fully specified paths," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 131–145, Feb. 2021.

[12] J. C. Kiemel and T. Kröger, "Learning time-optimized path tracking with or without sensory feedback," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*. IEEE, 2022, pp. 4024–4031.

[13] A. Völz and K. Graichen, "A predictive path-following controller for continuous replanning with dynamic roadmaps," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3963–3970, Oct. 2019.

[14] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," in *Proc. Robot.: Sci. Syst. VIII*, 2012, pp. 1–8.

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

[16] K. Shin and N. McKay, "Minimum-time control of robotic manipulators with geometric path constraints," *IEEE Trans. Autom. Control*, vol. 30, no. 6, pp. 531–541, Jun. 1985.

[17] A. Nagy and I. Vajk, "Sequential time-optimal path-tracking algorithm for robots," *IEEE Trans. Robot.*, vol. 35, no. 5, pp. 1253–1259, Oct. 2019.

[18] T. Ardeshiri, M. Norrlöf, J. Löfberg, and A. Hansson, "Convex optimization approach for time-optimal path tracking of robots with speed dependent constraints," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 14648–14653, 2011.

[19] F. Debrouwere et al., "Time-optimal path following for robots with convex–concave constraints using sequential convex programming," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1485–1495, Dec. 2013.

[20] J. Dong and J. Stori, "A generalized time-optimal bidirectional scan algorithm for constrained feed-rate optimization," *J. Dynamic Syst., Meas., Control*, vol. 128, no. 2, pp. 379–390, 2005.

[21] T. Kröger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Trans. Robot.*, vol. 26, no. 1, pp. 94–111, Feb. 2010.

[22] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Auton. Robots*, vol. 42, pp. 957–975, 2018.

[23] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1049–1065, Oct. 2014.

[24] K. Hauser, "Learning the problem-optimum map: Analysis and application to global optimization in robotics," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 141–152, Feb. 2017.

[25] S. Zhang, S. Dai, and Y. Zhao, "Continuous trajectory planning based on learning optimization in high dimensional input space for serial manipulators," *Eng. Optim.*, vol. 54, no. 10, pp. 1724–1742, 2022.

[26] S. Zhang, A. M. Zanchettin, R. Villa, and S. Dai, "Real-time trajectory planning based on joint-decoupled optimization in human-robot interaction," *Mechanism Mach. Theory*, vol. 144, 2020, Art. no. 103664.

[27] J. Kim and E. A. Croft, "Online near time-optimal trajectory planning for industrial robots," *Robot. Comput.- Integr. Manuf.*, vol. 58, pp. 158–171, 2019.

[28] M. Faroni, M. Beschi, A. Visioli, and N. Pedrocchi, "A real-time trajectory planning method for enhanced path-tracking performance of serial manipulators," *Mechanism Mach. Theory*, vol. 156, 2021, Art. no. 104152.

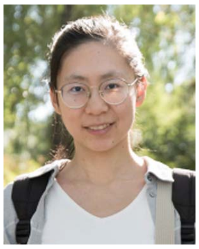
[29] A. M. Zanchettin and B. Lacevic, "Safe and minimum-time path-following problem for collaborative industrial robots," *J. Manuf. Syst.*, vol. 65, pp. 686–693, 2022.

[30] S. He, C. Hu, S. Lin, Y. Zhu, and M. Tomizuka, "Real-time time-optimal continuous multi-axis trajectory planning using the trajectory index coordination method," *ISA Trans.*, vol. 131, pp. 639–649, 2022.

[31] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Syst. Mag.*, vol. 31, no. 3, pp. 52–65, 2011.

[32] S. Zhang, S. Dai, A. M. Zanchettin, and R. Villa, "Trajectory planning based on non-convex global optimization for serial manipulators," *Appl. Math. Modelling*, vol. 84, pp. 89–105, 2020.

[33] S. Zhang and F. Pecora, "Online and scalable motion coordination for multiple robot manipulators in shared workspaces," *IEEE Trans. Automat. Sci. Eng.*, vol. 21, no. 3, pp. 2657–2676, Jul. 2024.



Shiyu Zhang received the B.Eng degree in automation from the Hefei University of Technology, Hefei, China, in 2014, and the Ph.D. degree in navigation, guidance and control from Beihang University, Beijing, China, in 2019.

From 2019 to 2022, she was a Postdoctoral Researcher with the Centre for Applied Autonomous Sensor Systems, Örebro University, Örebro, Sweden. She is currently an Associate Researcher with the School of Intelligent Engineering and Automation, Beijing University of Posts and Telecommunications, Beijing. Her research interests include multirobot coordination and human–robot interaction.

Da Sun received the B.Eng. and Ph.D. degrees in mechatronics from the University of Wollongong, Wollongong, Australia, in 2012 and 2016, respectively.

He has held positions as a Research Fellow with the National University of Singapore, Singapore, and as a permanent Senior Researcher with Örebro University, Örebro, Sweden. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. His research interests include machine learning, robotic control, and human–machine interaction.

Qianfang Liao received the B.Eng. degree in automation from Wuhan University, Wuhan, China, in 2006, the M.Eng. degree in automation from Shanghai Jiao Tong University, Shanghai, China, in 2009, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2015.

She was a tenured Researcher with Örebro University, Örebro, Sweden. She is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China,, Hefei, China. Her research interests include robot perception, learning, and control.