Time-Optimal Trajectory Generation with Multi-level Continuous Kinodynamics Constraints

Ruixuan Liu[1,2], Changliu Liu[2], and Jessica Leu[1]

***Abstract*— Time-optimal trajectory generation (TOTG) is critical in robotics applications to minimize travel time and increase robot task efficiency. To ensure the trajectory is feasible and executable by the robot, it is important to constrain the trajectory kinodynamics subject to the robot actuator limits. A typical actuator has multiple limits, 1) peak limit, and 2) multi-level continuous limits with different operation time windows. The peak limit bounds the instantaneous kinodynamics (IKD), whereas the continuous limits bound the system continuous kinodynamics (CKD). Existing works only constrain IKD, usually by the actuator peak limit, to achieve time optimality. However, a joint capable of operating at its peak limit momentarily will overheat and damage robot life if the motion continues. Alternatively, users can constrain the IKD with a reduced peak limit to avoid violating continuous limits. However, the reduced peak limit would inevitably sacrifice task efficiency. To address the challenge, this paper studies TOTG with both IKD and CKD, and proposes TOTG-C. It formulates the TOTG as a nonlinear programming (NLP). In particular, it proposes a novel formulation to encode the multi-level CKD constraints efficiently. To the best of our knowledge, TOTG-C is the first work that explicitly considers multi-level CKD constraints. We demonstrate the effectiveness and robustness of the proposed TOTG-C both in simulation and real robot experiments.**

***Index Terms*— Time-Optimal Trajectory Generation, Continuous kinodynamics Constraints, Constrained Nonlinear Programming.**

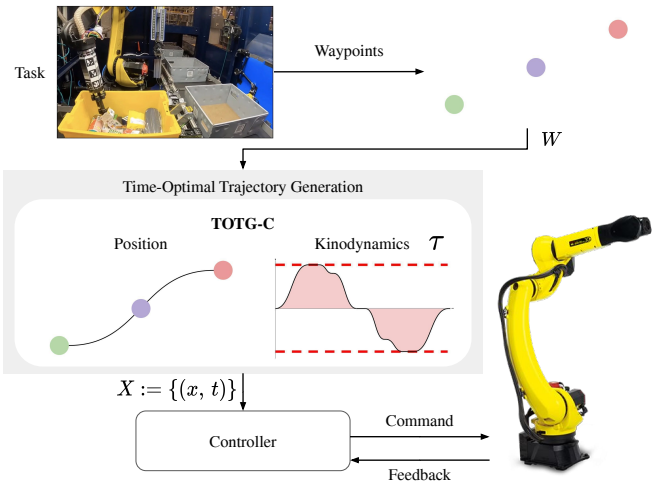

Fig. 1: Illustration of the time-optimal trajectory generation problem. Given a sequence of task-level reference waypoints W, TOTG generates a time-optimal trajectory $X(t)$ for the robot to execute the task. Conventional TOTG has the kinodynamics profile τ of the planned trajectory satisfying the robot hardware limits using instantaneous kinodynamics, *i.e.*, red dashed lines. This paper enables enforcing the continuous kinodynamics, *i.e.*, shaded red areas, to be bounded by the hardware specifications as well.

I. INTRODUCTION

Robots are widely used in applications such as assembly [1], [2], package sorting [3], object manipulation [4], [5], human-robot collaboration [6], [7], etc. To increase automation efficiency, time-optimal trajectory generation (TOTG) is critical. It minimizes travel time and maximizes robot task efficiency. To ensure the time-optimal trajectory is executable by the robot, it is important to constrain the time-optimal trajectory based on robot hardware constraints. One important type of constraint is the robot actuator kinodynamics limit. In general, a typical actuator has multiple limits, including 1) peak limit, and 2) multi-level continuous limits with different operation time windows. The peak limit bounds the instantaneous kinodynamics (IKD). It specifies, for example, the maximum velocity and torque that an actuator can reach instantaneously. On the other hand, the continuous limits bound the system continuous kinodynamics (CKD), which specify the average output of the actuator for a period of time.

TOTG has been widely studied [8], [9], [10]. However, existing works lack the ability to incorporate full robot capabilities since they only constrain IKD. To achieve time optimality, people usually constrain IKD by the actuator peak limit. However, a joint that can momentarily operate at the peak limit may not be able to operate at such a high level continuously. This is because the robot will then exceed the actuators' continuous limits, which can lead to overheating, excessive wear, and ultimately, reduce robot life. Alternatively, conventional approaches reduce the peak limit and constrain the IKD with the reduced peak limit to avoid violating continuous limits. However, the reduced peak limit results in slower motion and significantly decreases task efficiency.

To address the challenge, this paper studies TOTG with both IKD and CKD as illustrated in Fig. 1 and 2. Instead of only constraining the system IKD, *i.e.*, the red dashed bounds, we also explicitly constrain the system CKD, *i.e.*, the shaded areas, when generating the time-optimal trajectory. Given a sequence of task waypoints as shown in Fig. 1, an illustrative comparison is shown in Fig. 2. Conventional TOTG only considers the system IKD constraints and pushes the performance to its limits as depicted in Fig. 2(1), which maximizes the task efficiency but harms the robot life. In this work, we aim to integrate CKD constraints by enabling the robot to fully utilize its original peak capability but simultaneously avoid violating CKD constraints. In particular, we

[1]Amazon Robotics, MA, USA. leujessi@amazon.com
[2]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. {ruixuanl, cliu6}@andrew.cmu.edu

Manuscript 1164 submitted to 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Received March 1, 2025.

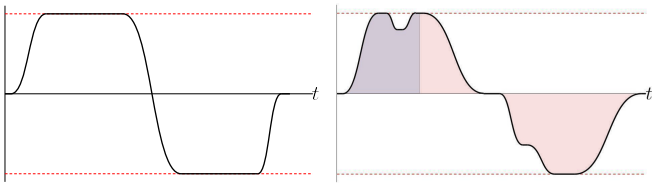

Fig. 2: A comparison between conventional TOTG and ours (TOTG-C). Given the task waypoints in Fig. 1, (1) Example kinodynamics (*e.g.*, acceleration) profile planned using conventional TOTG, and (2) Example kinodynamics profile planned using ours. Shaded blue and red regions: bounded CKD with different time windows. Dashed red lines: system IKD constraints, *e.g.*, acceleration bounds.

propose TOTG-C to generate time-optimal trajectories subject to both hardware peak limits and multi-level continuous limits with different time windows (*i.e.*, blue and red shaded regions). TOTG-C formulates the trajectory generation problem as nonlinear programming (NLP). In order to integrate the complex continuous limits, we propose a novel mixed-integer formulation to efficiently encode the multi-level CKD constraints with different time windows. To our knowledge, TOTG-C is the *first* work that considers multi-level CKD constraints in time-optimal trajectory generation. We test and evaluate the proposed TOTG-C in both simulations and real robot experiments. The experiment results demonstrate that the proposed TOTG-C can effectively and robustly generate time-optimal trajectories subject to both IKD and multi-level CKD constraints. Due to the additional CKD constraints, the kinodynamics profile would have more early rises and drops, as illustrated in Fig. 2(2), and the task execution would be longer. Nevertheless, we allow the robot to operate at its original peak level to minimize the loss of task efficiency. To further highlight the advantages of TOTG-C, we estimate the hardware wear of real robots executing trajectories planned by TOTG-C. By fully satisfying the hardware constraints (*i.e.*, both IKD and CKD constraints), TOTG-C can maximize task efficiency while significantly reducing hardware damage as the energy consumption is lower and the estimated robot actuator life is longer.

A. Related Works

Time-optimal trajectory generation has been widely studied. Early works [8], [11] use numerical integration (NI) to obtain a time-optimal trajectory via bang-bang control. However, NI-based methods are prone to failure due to the challenge of identifying switch points. [12], [13] solve TOTG using dynamic programming. Recent works [14], [15] leverage black-box optimization techniques to search for the time-optimal trajectory, which only apply to polynomial trajectories. On the other hand, [9] formulates the TOTG as a convex optimization (CO), which gives a general and robust formulation. [10] leverages reachability analysis to solve the TOTG for second-order systems by multiple linear programming (LP) sub-problems, which significantly improves the computation time. Recent efforts [16], [17], [18], [19] devote to extending the optimization-based formulation to address the TOTG for higher order constraints, *e.g.*, jerk bounds.

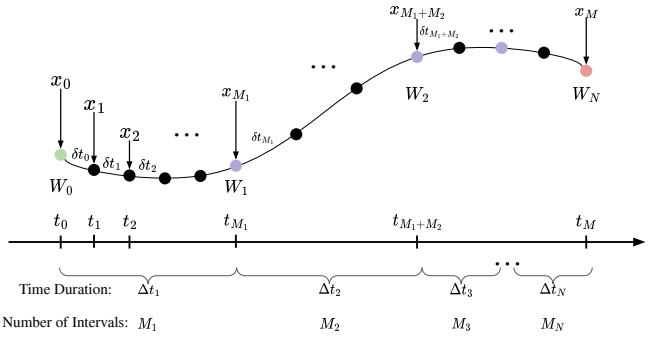

Fig. 3: Illustration of the discrete trajectory generation in the system state space. Given a sequence of $N+1$ waypoints, *i.e.*, $W = [W_0, W_1, \ldots, W_N]$, the trajectory is divided into N segments, and each has a time duration of Δt_i. The entire trajectory is interpolated to have $M+1$ reference states, where $M = M_1 + M_2 + \cdots + M_N$. The time duration between points x_i and x_{i+1} is denoted as δt_i. Black: interpolated reference states. Green: system starting state. Purple: system states that pass through the waypoints. Red: system final state.

Recent works [20], [21] formulate the TOTG problem as a nonlinear programming (NLP) problem and generate time-optimal trajectories for quadrotors. Despite TOTG has been widely studied, existing works only consider bounds on the system IKD, and thus, lack the ability to incorporate full robot capabilities.

B. Contributions

The contributions of this paper are listed as follows,

1) We propose TOTG-C, which formulates the TOTG as a general NLP to address both IKD and CKD constraints. Particularly, we propose a novel mixed-integer formulation to efficiently encode multi-level continuous kinodynamics limits.
2) We evaluate the proposed TOTG-C in simulation and demonstrate that the proposed method can effectively and robustly solve time-optimal trajectories subject to both IKD and CKD constraints.
3) We apply the TOTG-C on real robots and the experiment showcases that TOTG-C can effectively reduce the hardware wear, *i.e.*, lower energy consumption and longer estimated robot life, while achieving comparable task efficiency.

C. Organization

The rest of the paper is organized as follows. Section II introduces the key notations in this paper. Section III formulates the multi-level CKD constraints and introduces the proposed TOTG-C. Section IV demonstrates the experiment results of TOTG-C and discusses future work. Section V concludes the paper.

II. PRELIMINARIES

A. Problem Formulation

This paper formulates the TOTG as a discrete control problem, where the system state is denoted as x_i and the control is represented as u_i at time step i. Based on the robot task, a sequence of task-level waypoints W can be determined as shown in Fig. 1. Let the waypoints be

$W = [W_0, W_1, \ldots, W_N], W_i \in \mathbb{R}^n$ for an n degrees-of-freedom (DOF) robot. The waypoints divide the trajectory into N segments as shown in Fig. 3. Following the idea in [22], the i-th segment is interpolated into M_i intervals with $M_i - 1$ additional reference states (*i.e.*, black dots in Fig. 3). The goal of TOTG is to generate a time-optimal trajectory $X := \{(x, t)\}$, where $x = [x_0, x_1, \ldots, x_M]$, $t = [t_0, t_1, \ldots, t_M]$ and $M = \sum_{i=1}^{N} M_i$. We have $t_0 = 0$ and $t_M = T$, where T is the total time duration of the planned trajectory. Consequently, the delta time profile can be written as $\delta t = [\delta t_0, \delta t_1, \ldots, \delta t_{M-1}]$, where $\delta t_i = t_{i+1} - t_i$ as labeled on Fig. 3. To execute the user-specified task W, the planned trajectory X should drive the system to pass through all the waypoints W_i, *i.e.*, the purple dots in Fig. 3, while satisfying the system kinodynamics constraints. We use $\tau = [\tau_0, \tau_1, \ldots, \tau_M]$ to denote the system kinodynamics. Note that τ is a general representation, which could be the system position, velocity, acceleration, torque, etc.

B. Instantaneous kinodynamics

The instantaneous kinodynamics (IKD) depicts the system kinodynamics value at any given instantaneous time. Thus, IKD at time t_i is defined as τ_i.

C. Continuous kinodynamics

The continuous kinodynamics (CKD) evaluates the average system kinodynamics value for a period of time $\bar{t}$ (*i.e.*, a time window), which is defined as

$$\bar{\tau}_s^{\bar{t}} = \frac{\sqrt{\sum_{i=s}^{e} \tau_i^2 \delta t_i}}{\bar{t}}, \ s, e \in [0, M], \tag{1}$$

where s is the starting time index of the window and e satisfies $\sum_{i=s}^{e} \delta t_i = \bar{t}$. Depending on the hardware specifications, there could be multiple different $\bar{t}$, *e.g.*, 2s, 10s, etc.

III. TOTG-C: TOTG with Continuous Kinodynamics Constraints

We define the system state as x and control as u. The discrete system dynamics can be written as

$$x_{i+1} = \mathbf{f}(x_i, u_i, \delta t_i). \tag{2}$$

Following the idea of direct shooting [23], a general TOTG objective can be formulated as

$$\min_{x,u,\delta t} \ T = \sum_{i=0}^{M-1} \delta t_i. \tag{3}$$

To reduce the problem complexity, we adopt the method in [21] and equally divide the trajectory segments between waypoints. Thus, we have $\delta t_k = \frac{\Delta t_i}{M_i}, i \in [1, N]$ for $k \in [\sum_{j=0}^{i-1} M_j, \ \sum_{j=0}^{i} M_j - 1]$, as illustrated in Fig. 3. The TOTG can be formulated as an NLP:

$$\begin{aligned} &\min_{x,u,\Delta t} \ T = \sum_{i=1}^{N} \Delta t_i, \\ &\text{subject to (2)}, \\ &\forall i \in [0, N], \ x_k^p = W_i, \ k = \sum_{j=0}^{i} M_j, \end{aligned} \tag{4}$$

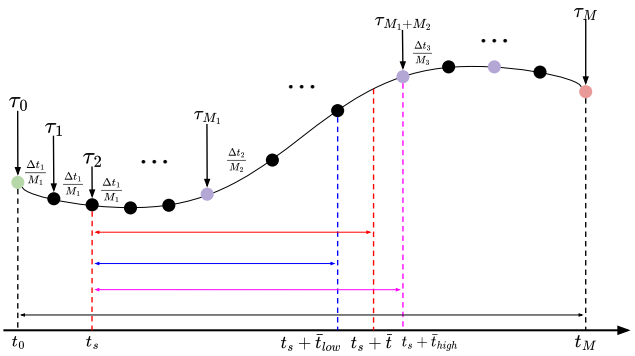

Fig. 4: Illustration of the system CKD profile. t_s, the time of the reference point at index s, is the starting time of a specified time window and $\bar{t}$ is the duration of the time window. Black interval: CKD over the whole trajectory. Red interval: CKD over a specified duration $\bar{t}$. Blue interval: CKD from t_s to the previous reference state. Purple interval: CKD from t_s to the latter reference state.

where $M_0 = 0$ and the superscript $\square^p$ denotes the position component of the system state. Note that the dimension of the decision variables in the simplified NLP (4) is significantly reduced compare to (3). More importantly, the dimension of Δt only depends on the number of waypoints N and is not influenced by the interpolation density M_i.

To ensure the planned trajectory $X := \{(x, t)\}$ is feasible and executable by the real robot in the long-term, it is important to enforce the trajectory to satisfy all the kinodynamics constraints.

A. Peak Kinodynamics Limits

From a hardware standpoint, the peak limit specifies the maximum performance the hardware can reach. It indicates that at a given time step, the system kinodynamics value cannot exceed the bound. For instance, a gantry can reach up to a certain distance; a joint can spin up to a certain velocity; an actuator can generate up to a certain amount of torque. Thus, the peak limit is directly applied on the system IKD, which is denoted as

$$\forall i \in [0, M], \ \tau_{min} \leq \tau_i \leq \tau_{max}. \tag{5}$$

From a control standpoint, the peak limit constrains the maximum output of the system at an instantaneous time. However, the robot capable of operating at the max level momentarily will overheat and be damaged if it consistently operates at the max level. Therefore, a typical actuator has not only a peak limit specified but also multi-levels of continuous limits. It is important to consider the continuous kinodynamics constraints in TOTG to ensure the feasibility and optimality of the generated trajectory.

B. Continuous Kinodynamics Limits

There are in general two types of continuous kinodynamics limits that need to be considered, *i.e.*, 1) continuous limit over the entire trajectory, and 2) continuous limit over a specified duration, *i.e.*, a time window.

a) Whole Trajectory: As defined in (1), the CKD of the whole trajectory is calculated as

$$\bar{\tau}^T = \frac{1}{T}\Big\{\sum_{i=0}^{M-1} \tau_i^2 \delta t_i\Big\}^{\frac{1}{2}}, \tag{6}$$

which corresponds to the black interval illustrated in Fig. 4. The continuous limit over the whole trajectory enforces that

$$\bar{\tau}^T \leq \bar{\tau}^T_{max}, \tag{7}$$

which imposes an additional nonlinear inequality constraint to the TOTG in (4).

b) Time Window: Given a time window duration $\bar{t}$ and a starting time t_s at the index s, the CKD is defined in (1). However, it is likely that no such e exists that satisfies $\sum_{i=s}^{e} \delta t_i = \bar{t}$ due to the discrete formulation. Thus, the CKD over a time window $\bar{t}$ is evaluated as

$$\bar{\tau}^{\bar{t}}_s = \frac{1}{\bar{t}}\Big\{\sum_{i=s}^{k-1} \tau_i^2 \delta t_i + \underbrace{\tau_k^2(\bar{t} - \sum_{i=s}^{k-1} \delta t_i)}_{\text{Residual CKD}}\Big\}^{\frac{1}{2}}, \tag{8}$$

where k is the index such that $\sum_{i=s}^{k-1} \delta t_i \leq \bar{t}$ and $\sum_{i=s}^{k} \delta t_i \geq \bar{t}$. The CKD over a time window in (8) corresponds to the red interval in Fig. 4. To enforce the continuous limit over a time window, it is necessary to satisfy

$$\forall s \in [0, M],\ \bar{\tau}^{\bar{t}}_s \leq \bar{\tau}^{\bar{t}}_{max}. \tag{9}$$

However, the CKD evaluation in (8) is not deterministic since time intervals Δt_i are decision variables. To bound the continuous limit, we have (9) formulated as

$$\begin{aligned} &\forall s \in [0, M],\ \forall k \in [s+1, M] \\ &\frac{1}{\bar{t}}\Big\{\sum_{i=s}^{k-1} \tau_i^2 \delta t_i + \tau_k^2(\bar{t} - \sum_{i=s}^{k-1} \delta t_i)\Big\}^{\frac{1}{2}} b_{s,k} \leq \bar{\tau}^{\bar{t}}_{max}, \\ &b_{s,k} = \begin{cases} 1,\ \sum_{i=s}^{k-1} \delta t_i \leq \bar{t}\ \wedge\ \sum_{i=s}^{k} \delta t_i \geq \bar{t}, \\ 0, \text{Otherwise.} \end{cases} \end{aligned} \tag{10}$$

The residual CKD in (8) makes (9) particularly difficult to encode. The resulting CKD constraints in (10) adds $\frac{M^2+M}{2}$ additional nonlinear constraints to the TOTG and makes the NLP inefficient to solve. Critically, the problem complexity grows exponentially as M increases, which would easily make the NLP intractable.

To address the challenge, instead of bounding the exact CKD, we propose to bound the CKD in (8) by its upper-bound approximation. As shown in Fig. 4, the blue and purple intervals represent the CKD at the reference states before and after $t_s + \bar{t}$, which are denoted as $\bar{\tau}^{\bar{t}_{low}}_s$ and $\bar{\tau}^{\bar{t}_{high}}_s$. We can show that $\bar{\tau}^{\bar{t}}_s \leq \max(\bar{\tau}^{\bar{t}_{low}}_s, \bar{\tau}^{\bar{t}_{high}}_s)$. Thus, the upper bound approximation of (8) can be written as

$$\lceil \bar{\tau}^{\bar{t}}_s \rceil = \max\left\{\frac{1}{\bar{t}_{low}}\Big\{\sum_{i=s}^{k-1} \tau_i^2 \delta t_i\Big\}^{\frac{1}{2}},\ \frac{1}{\bar{t}_{high}}\Big\{\sum_{i=s}^{k} \tau_i^2 \delta t_i\Big\}^{\frac{1}{2}}\right\}. \tag{11}$$

To enforce the continuous limit, it is sufficient to show

$$\forall s \in [0, M], \lceil \bar{\tau}^{\bar{t}}_s \rceil \leq \bar{\tau}^{\bar{t}}_{max}. \tag{12}$$

We propose a mixed-integer formulation of (12) to encode the CKD constraint into the NLP in (4) as

$$\begin{aligned} &\forall s \in [0, M],\ \max\{L_s, H_s\} \leq \bar{\tau}^{\bar{t}}_{max}, \\ &L_s = \frac{1}{\sum_{i=s}^{M} \delta t_i l_{s,i}}\Big\{\sum_{i=s}^{M} \tau_i^2 \delta t_i l_{s,i}\Big\}^{\frac{1}{2}}, \\ &H_s = \frac{1}{\sum_{i=s}^{M} \delta t_i h_{s,i}}\Big\{\sum_{i=s}^{M} \tau_i^2 \delta t_i h_{s,i}\Big\}^{\frac{1}{2}}, \end{aligned} \tag{13}$$

where $l_{s,i}$ and $h_{s,i}$ are integers defined as

$$\begin{aligned} l_{s,i} &= \begin{cases} 1,\ \sum_{j=s}^{i} \delta t_j \leq \bar{t}, \\ 0, \text{Otherwise.} \end{cases} \\ h_{s,i} &= \begin{cases} 1,\ \sum_{j=s}^{i-1} \delta t_j \leq \bar{t}\ \wedge\ \sum_{j=s}^{i} \delta t_j \geq \bar{t}, \\ 0, \text{Otherwise.} \end{cases} \end{aligned} \tag{14}$$

The formulation in (13) reduces the constraint complexity by removing the residual CKD. Moreover, the formulation in (13) gives M additional nonlinear constraints, which makes the NLP more tractable and can be solved efficiently.

To satisfy both peak and continuous limits, the TOTG-C solves an NLP that establishes bounds on IKD and CKD, which is formulated as

$$\begin{aligned} &\min_{x,u,\Delta t}\ T = \sum_{i=1}^{N} \Delta t_i, \\ &\text{subject to (2), (5), (7) and (13),} \\ &\forall i \in [0, N],\ x^p_k = W_i,\ k = \sum_{j=0}^{i} M_j. \end{aligned} \tag{15}$$

To further improve the computation efficiency, we warm start the TOTG-C by solving a simplified NLP.

C. Warm Start

We propose to warm start the NLP in (15) by solving its sparser relaxed version. Given the waypoints W, the trajectory is divided into N segments as shown in Fig. 3. The original NLP (15) further interpolates the i-th segment into M_i intervals. The sparse NLP interpolates the i-th segment into M'_i intervals, where $\frac{M_i}{M'_i} \in \mathbb{Z}+$. The relaxed warm-up NLP can be formulated as

$$\begin{aligned} &\min_{x',u',\Delta t}\ T = \sum_{i=1}^{N} \Delta t_i, \\ &\text{subject to (2) and (5),} \\ &\forall i \in [0, N],\ x'^p_k = W_i,\ k = \sum_{j=0}^{i} M'_j, \end{aligned} \tag{16}$$

where $x' = [x'_0, x'_1, \ldots, x'_{M'}]$, $u' = [u'_0, u'_1, \ldots, u'_{M'}]$, and $M' = \sum_{i=1}^{N} M'_i$. Essentially, (16) solves a sparser TOTG with IKD constraints only, and is easier to solve due to the reduced problem complexity. Given u', the

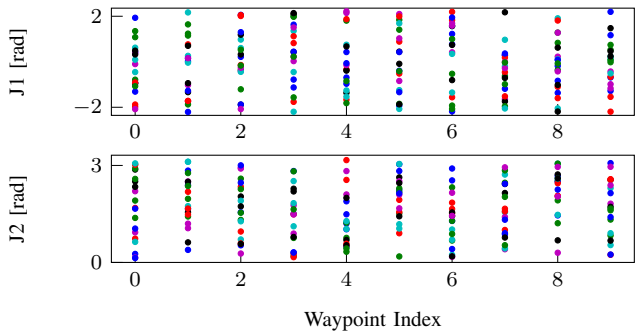

Fig. 5: Waypoints distribution (joint 1 and 2) of the testing paths with 10 waypoints. Same color indicates points from the same path.

initial guess u_{init} is obtained by upsampling as $u_{init} = [u'_0, u'_0, \ldots, u'_1, u'_1, \ldots, u'_{M'}, \ldots, u'_{M'}]$, and x_{init} is calculated by numerical integration following (2). The solution $x_{init}, u_{init}, \Delta t$ are provided to (15) to warm start the NLP.

IV. Experiment

We apply TOTG-C to a 6-DOF robot manipulator, in which the robot joint positions, velocities, and accelerations are denoted as $q, \dot{q}, \ddot{q} \in \mathbb{R}^6$. The actuator torque, $\tau \in \mathbb{R}^6$, can be calculated as

$$\tau = \mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{G}(q), \tag{17}$$

where $\mathbf{M}(q) \in \mathbb{R}^{6\times6}$ is a positive definite mass matrix, $\mathbf{C}(q, \dot{q}) \in \mathbb{R}^{6\times6}$ is the Coriolis matrix and $\mathbf{G}(q) \in \mathbb{R}^6$ accounts for gravity torques. The system dynamics is modeled using a double integrator, which can be written as

$$x_{i+1} = \begin{bmatrix} I_6 & I_6\delta t_i \\ 0 & I_6 \end{bmatrix} x_i + \begin{bmatrix} \frac{1}{2}I_6\delta t_i^2 \\ I_6\delta t_i \end{bmatrix} u_i, \tag{18}$$

where $I_6 \in \mathbb{R}^{6\times6}$ is an identity matrix. The system state $x_i = [q_i;\ \dot{q}_i]$ includes the robot joint position and velocity, and the system control is $u_i = \ddot{q}_i$. The robot has peak limits on its positions, velocities, and torques, *i.e.*, (5). The velocities and torques each have two levels of continuous limits, *i.e.*, 1) the continuous limits on velocities and torques over the whole trajectory, *i.e.*, (7), and 2) 2-second window continuous limits on velocities and torques, *i.e.*, (13). These are common hardware specifications considered in industries. The proposed TOTG-C is implemented using CasADI [24] and the NLP in (15) is solved using the Interior Point Optimizer (IPOPT) [25]. The results are generated with an Apple M3 Pro CPU.

A. Computation Efficiency

To evaluate the performance of TOTG-C, we *randomly* generate a set of paths with different numbers of waypoints, *i.e.*, 2, 4, 6, 8, and 10. We generate 20 different paths for each number of waypoints, giving a total of 100 different paths. Fig. 5 illustrates the waypoint distribution of the testing paths with 10 waypoints for joint 1 and joint 2. As shown by the distribution plot, the testing waypoints span over the robot workspace for the joints to better evaluate the performance of the proposed TOTG-C. For each path, we have an identical number of intervals between waypoints,

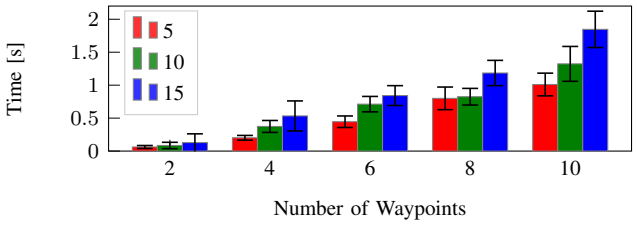

Fig. 6: TOTG-C computation time. Red: $M_i = 5$. Green: $M_i = 10$. Blue: $M_i = 15$.

	TOPP-RA	TOTG-C
Trajectory Time [s]	4.474	4.797
Peak Velocity Bound [rad/s]		
±1.89	(-1.89, 1.89)	(-1.89, 1.89)
Continuous Velocity Bound [rad/s]		
Whole Traj: 0.97	0.653	0.59
2s: 2.7	1.117	1.128
Peak Torque Bound [Nm]		
±1036	(-1018.4, 1036)	(-7.6, 1003.1)
Continuous Torque Bound [Nm]		
Whole Traj: 288.4	*382.051*	288
2s: 1036	599.396	495.67

TABLE I: Comparison of the constraint satisfaction of the trajectories generated by TOPP-RA and TOTG-C. Using joint 2 as an example, the kinodynamics bounds, *i.e.*, both peak and continuous, are listed in the left column. $(\cdot,\ \cdot)$ denotes the minimum and maximum IKD of the trajectory. *Italic numbers* indicate constraint violations.

i.e., $M_1 = M_2 = \cdots = M_N$, and test TOTG-C with different number of intervals, *i.e.*, $M_i = 5$, 10, 15. For each combination (W, M_i), the TOTG-C is executed for 10 trials, and the computation time is reported in Fig. 6. When there are only two waypoints, TOTG-C generates the trajectory within 0.1s. As the number of waypoints or the number of intervals M_i increases, the computation time increases. But most cases are solved within 1s. The variances in the computation times for different testing examples indicate that the solving time heavily depends on the difficulty of the task. Note that the computation time in Fig. 6 includes the time for solving both (16) and (15). Due to the nature of IPOPT, a warm start in general improves the solver's robustness but not always necessarily reduces the computation time. When the problem dimension is low, *e.g.*, having only 2 or 4 waypoints or $M_i = 5$, we can achieve faster solving time without the warm start, *i.e.*, approximately 0.05s to 0.1s faster.

B. Simulation Analysis

We further use the package sorting task to validate the TOTG-C performance in the real-world industrial context. Instead of randomly generating paths, we have paths with 7 to 13 waypoints sampled from package sorting tasks. In these tasks, TOTG-C on average achieves a shorter computation time than the results in section IV-A, outputting solution in 0.5s to 1s.

We compare TOTG-C with TOPP-RA [10], a state-of-the-art algorithm currently widely used in industries for time-optimal trajectory generation. Table. I and Fig. 7 show the comparison of the planned trajectories of one example task with 9 waypoints. Due to the additional continuous kinodynamics constraints, the planned trajectory by TOTG-C (4.797s) is slightly slower than the one generated by

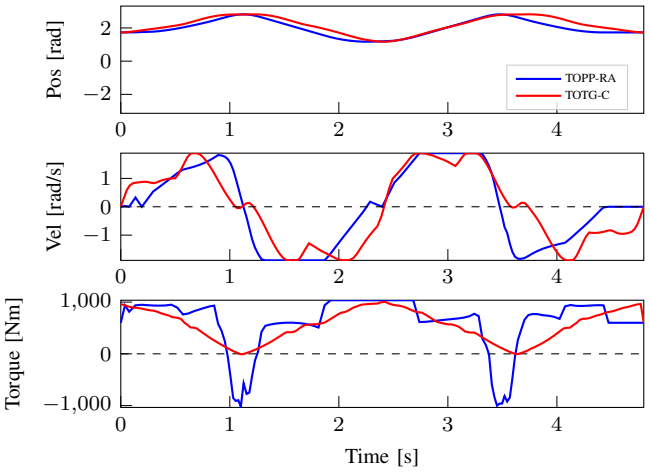

Fig. 7: Comparison of the generated trajectory profiles from TOPP-RA and the proposed TOTG-C. We plan a time-optimal trajectory for a 6-DOF robot to execute a package sorting task. We show the planned position, velocity, and torque profiles of the robot joint 2.

	Avg Power [W] ↓		Energy [Ws] ↓		Actuator Life [h] ↑
	J2	Robot	J2	Robot	J2
TOPP-RA	713.08	1029.9	3206.7	4631.6	$2.58e^4$
TOTG-C	**649.58**	**950.69**	**3149.2**	**4608.9**	$\mathbf{3.05e^4}$

TABLE II: Comparison of the robot consumption and estimated hardware life by executing the trajectories generated by TOPP-RA and TOTG-C.

TOPP-RA (4.474s). Table. I details the constraint satisfaction status of the planned trajectories. We display the results for the robot joint with the largest motion, which is joint 2 in this case. We can see that TOPP-RA maximizes the task efficiency by operating on the peak levels without violating peak bounds. However, it violates the continuous torque bound (indicated by the italic value) since it cannot address continuous kinodynamics constraints. An alternative conventional approach is to reduce the peak limits so that the generated trajectory is more conservative. In this particular example, we reduce the peak bounds to 70% of its original peak limits, and the TOPP-RA generates a 6.739s trajectory that satisfies all the constraints. This result demonstrates that reducing the peak bounds notably sacrifices the task efficiency as the trajectory time is significantly longer. More importantly, the amount of peak performance to reduce can vary from task to task, which is difficult to tune. On the other hand, by explicitly bounding the CKD, TOTG-C successfully satisfies all the hardware constraints. Moreover, TOTG-C maximizes the task efficiency by pushing the robot performance, *i.e.*, velocity and torque, to its original peak limits so that it can minimize the increase of robot motion time due to the additional continuous limits. Fig. 7 displays the planned trajectory profiles of joint 2 using TOPP-RA and TOTG-C. From the velocity and torque profiles, we can see that TOTG-C pushes the robot velocity and torque to the original peak limits, which corresponds to the results in Table. I. However, there are more early rises and falls, which demonstrates that the algorithm effectively reduces the time for the robot to continuously operate at the peak level.

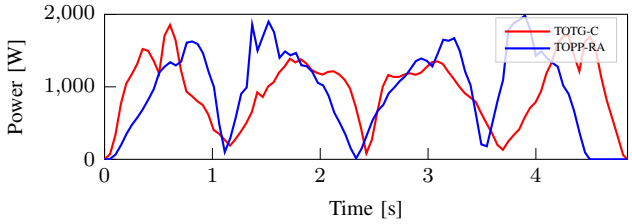

Fig. 8: Comparison of the power consumption profiles for the robot executing the trajectory generated by TOPP-RA and TOTG-C.

C. Robot Deployment

We execute the planned trajectories on a real robot to demonstrate the advantages of TOTG-C. Each trajectory is executed for 2 runs and the following results are the average of the runs. Fig. 8 illustrates the power consumption profiles of the robot executing trajectories from TOPP-RA and TOTG-C. We can see that TOTG-C significantly reduces the time for the robot to continuously operate on a high level. As shown in Table. II, TOTG-C has a lower average power consumption as well as a lower total energy consumption for both the single joint and the entire robot. Moreover, enforcing continuous kinodynamics constraints improves robot life. Following common industrial standards, we estimate the robot life using the method in [26] as $L_h = L_n \frac{V_r}{V_{avg}} (\frac{\tau_r}{\tau_{avg}})^3$, where L_n is the rated lifetime. V_r is the rated input speed in rounds per minute (RPM) and τ_r is the rated torque. V_{avg} and τ_{avg} are the average input speed in RPM and average torque, which are calculated as

$$V_{avg} = g \cdot \frac{\sum_{i=0}^{M-1} |V_i| \cdot \delta t_i}{T},$$
$$\tau_{avg} = \sqrt[3]{\frac{\sum_{i=0}^{M-1} |V_i \cdot \tau_i^3| \cdot \delta t_i}{\sum_{i=0}^{M-1} |V_i| \cdot \delta t_i}}, \tag{19}$$

where V_i is $\dot{q}_i$ in RPM and g is the actuator gear ratio. We measure the kinodynamics profiles from the real robot executing the planned trajectories and Table. II displays the estimated robot actuator life from real measurements under continuous operation for joint 2. It is clear that by enforcing both peak and continuous kinodynamics constraints, TOTG-C can significantly improve the robot hardware life while minimizing the sacrifice of task efficiency.

D. Discussion

The experiment results demonstrate that the proposed TOTG-C reduces energy consumption and extends the estimated robot life. Despite inevitable longer robot motion time due to additional continuous constraints, TOTG-C minimizes the increase of robot motion time by leveraging the robot's original peak level while enforcing continuous kinodynamics constraints. Another advantage of TOTG-C is that it formulates the problem into a general NLP form, which can be easily customized to consider different robot embodiments or additional constraints, *e.g.*, task-space motion, collision avoidance, higher-order systems, etc. Despite the promising results, the major limitation of TOTG-C lies in computation

efficiency. With additional complex CKD constraints, TOTG-C is efficient since it generally solves the NLP within a second. As a reference, the NLP for trajectory generation in [21] solves in 1-2s. However, further improvement in the computation efficiency is desired, and we aim to improve the computation time in the future.

V. CONCLUSIONS

This paper proposes TOTG-C to generate time-optimal trajectories subject to both hardware peak limits and multi-level continuous limits. TOTG-C formulates the trajectory generation problem as an NLP and proposes a novel mixed-integer formulation to encode the multi-level CKD constraints efficiently. The experiment results demonstrate that the proposed TOTG-C can efficiently and robustly generate time-optimal trajectories subject to both IKD and multi-level CKD constraints. By fully satisfying the hardware constraints, the robot can effectively reduce the wear of hardware, *i.e.*, longer estimated robot life and lower energy consumption, while maximizing the task efficiency.

VI. ACKNOWLEDGEMENT

The authors would like to thank Abhay Gupta, Beau Pollard, Nicholas Smith, and Chirag Sharma for their valuable supports with the experiment setup.

REFERENCES

[1] R. Liu, R. Chen, A. Abuduweili, and C. Liu, "Proactive human-robot co-assembly: Leveraging human intention prediction and robust safe control," in *2023 IEEE Conference on Control Technology and Applications (CCTA)*, 2023, pp. 339–345.

[2] R. Liu, Y. Sun, and C. Liu, "A lightweight and transferable design for robust lego manipulation," *arXiv preprint arXiv:2309.02354*, 2023.

[3] S. Li, A. Keipour, K. Jamieson, N. Hudson, C. Swan, and K. Bekris, "Demonstrating large-scale package manipulation via learned metrics of pick success," *arXiv preprint arXiv:2305.10272*, 2023.

[4] C. Mitash, F. Wang, S. Lu, V. Terhuja, T. Garaas, F. Polido, and M. Nambi, "Armbench: An object-centric benchmark dataset for robotic manipulation," *arXiv preprint arXiv:2303.16382*, 2023.

[5] C. Mitash, M. Hussein, J. Vanbaar, V. Terhuja, and K. Katyal, "Scaling object-centric robotic manipulation with multimodal object identification," in *ICRA 2024*, 2024.

[6] R. Liu, R. Chen, and C. Liu, "Task-agnostic adaptation for safe human-robot handover," *IFAC-PapersOnLine*, vol. 55, no. 41, pp. 175–180, 2022, 4th IFAC Workshop on Cyber-Physical and Human Systems CPHS 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896323001295

[7] "Robustifying long-term human-robot collaboration through a hierarchical and multimodal framework, author = Peiqi Yu and Abulikemu Abuduweili and Ruixuan Liu and Changliu Liu , journal=arXiv preprint arXiv:2411.15711, year=2024."

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

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

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

[11] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985. [Online]. Available: https://doi.org/10.1177/027836498500400301

[12] K. Shin and N. McKay, "A dynamic programming approach to trajectory planning of robotic manipulators," *IEEE Transactions on Automatic Control*, vol. 31, no. 6, pp. 491–500, 1986.

[13] V. Petrone, E. Ferrentino, and P. Chiacchio, "Time-optimal trajectory planning with interaction with the environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 399–10 405, 2022.

[14] Y. Dae Lee, B. Hee Lee, and H. Gyoo Kim, "An evolutionary approach for time optimal trajectory planning of a robotic manipulator," *Information Sciences*, vol. 113, no. 3, pp. 245–260, 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002002559810052X

[15] X. Zhang, F. Xiao, X. Tong, J. Yun, Y. Liu, Y. Sun, B. Tao, J. Kong, M. Xu, and B. Chen, "Time optimal trajectory planing based on improved sparrow search algorithm," *Frontiers in Bioengineering and Biotechnology*, vol. 10, 2022. [Online]. Available: https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2022.852408

[16] H. Pham and Q.-C. Pham, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2017.7989084

[17] Q. Zhang, S.-R. Li, and X.-S. Gao, "Practical smooth minimum time trajectory planning for path following robotic manipulators," in *2013 American Control Conference*, 2013, pp. 2778–2783.

[18] J. eun Lee, A. Bylard, R. Sun, and L. Sentis, "On the performance of jerk-constrained time-optimal trajectory planning for industrial manipulators," 2024. [Online]. Available: https://arxiv.org/abs/2404.07889

[19] H. Liu, X. Lai, and W. Wu, "Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 309–317, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584512000981

[20] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, Jul. 2021. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abh1221

[21] Z. Zhou, G. Wang, J. Sun, J. Wang, and J. Chen, "Efficient and robust time-optimal trajectory planning and control for agile quadrotor flight," 2023. [Online]. Available: https://arxiv.org/abs/2305.02772

[22] R. Liu, R. Chen, Y. Sun, Y. Zhao, and C. Liu, "Jerk-bounded position controller with real-time task modification for interactive industrial robots," in *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2022, pp. 1771–1778.

[23] R. Tedrake, *Underactuated Robotics*, 2023. [Online]. Available: https://underactuated.csail.mit.edu

[24] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[25] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, 2006.

[26] H. Drive. (2024) Cup type component sets housed units. [Online]. Available: https://www.harmonicdrive.net/_hd/content/catalogs/pdf/csd-shd.pdf