2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Jerk-limited Traversal of One-dimensional Paths and its Application to Multi-dimensional Path Tracking

Jonas C. Kiemel[1] and Torsten Kröger

***Abstract*— In this paper, we present an iterative method to quickly traverse multi-dimensional paths considering jerk constraints. As a first step, we analyze the traversal of each individual path dimension. We derive a range of feasible target accelerations for each intermediate waypoint of a one-dimensional path using a binary search algorithm. Computing a trajectory from waypoint to waypoint leads to the fastest progress on the path when selecting the highest feasible target acceleration. Similarly, it is possible to calculate a trajectory that leads to minimum progress along the path. This insight allows us to control the traversal of a one-dimensional path in such a way that a reference path length of a multi-dimensional path is approximately tracked over time. In order to improve the tracking accuracy, we propose an iterative scheme to adjust the temporal course of the selected reference path length. More precisely, the temporal region causing the largest position deviation is identified and updated at each iteration. In our evaluation, we thoroughly analyze the performance of our method using seven-dimensional reference paths with different path characteristics. We show that our method manages to quickly traverse the reference paths and compare the required traversing time and the resulting path accuracy with other state-of-the-art approaches.**

I. INTRODUCTION

One of the most common tasks in industrial robotics is to follow a given reference path. The problem of traversing a reference path in a time-optimal manner, while respecting the kinematic limits of the robot joints, is known as time-optimal path parameterization (TOPP). For velocity and acceleration constraints, the problem has been studied extensively, and open source implementations are readily available [1], [2]. However, constraints on the derivative of the acceleration, commonly known as jerk, are often ignored. By considering jerk constraints, wear and tear on the mechanical components can be reduced, which not only increases the reliability of a robotic system but also contributes to its overall cost-effectiveness. While TOPP with jerk constraints is still an ongoing research topic, progress has been made in addressing a related problem. Specifically, it is possible to compute a time-optimal trajectory considering jerk constraints when provided with an initial kinematic state and a desired target kinematic state of a robot joint [3]. These kinematic states consist of a position p, velocity v and acceleration a. With this in mind, we analyze the problem of traversing a one-dimensional path subject to jerk constraints. As shown in Fig. 1, a one-dimensional path can be described by its initial position p_0, its final position p_{III} and intermediate positions

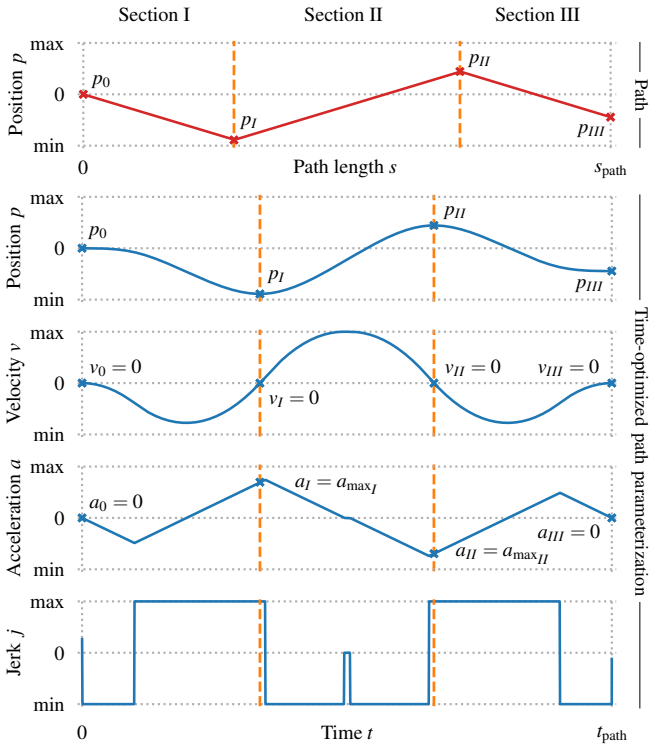


Fig. 1: A time parameterization for an exemplary one-dimensional path with an initial position p_0, final position p_{III} and two intermediate positions p_I and p_{II}.

(p_I and p_{II}) at which the direction of the path changes. The corresponding velocities v_0, v_I, v_{II}, v_{III} need to be zero. The same applies to the initial acceleration a_0 and the final acceleration a_{III}. To fully specify kinematic target states for the intermediate points p_I and p_{II}, appropriate accelerations a_I and a_{II} need to be found. The main contributions of this paper can be summarized as follows:

- We show how a binary search can be used to compute a range of feasible target accelerations for each intermediate point of a one-dimensional path. Selecting the highest feasible acceleration of each range results in a time-optimized traversal of the reference path (see $a_{\max_I}$ and $a_{\max_{II}}$ in Fig. 1).
- Based on these results, we compute trajectories leading to minimum and maximum progress along the one-dimensional path.
- Using these trajectories, we propose an iterative scheme to quickly traverse a multi-dimensional path and benchmark the resulting traversing time and path accuracy with other state-of-the-art methods.

[1]Institute for Anthropomatics and Robotics – Intelligent Process Automation and Robotics (IAR-IPR), Karlsruhe Institute of Technology (KIT), jonas.kiemel@kit.edu

II. Related work

Early research on time-optimal path parameterization (TOPP) for multi-dimensional paths dates back to the 1980s [4], [5]. Nowadays, common TOPP algorithms are designed to handle both first-order and second-order constraints, ensuring that the joint velocity and joint acceleration stay within predefined limits. An exemplary implementation of such an algorithm based on reachability analysis is provided by the open-source library TOPP-RA [2], which serves as a benchmark for our evaluation.

In order to consider jerk constraints, a TOPP algorithms supporting third-order constraints is required. While several partial solutions based on numerical integration have been proposed [6]–[8], an efficient algorithm for the general problem is still the subject of ongoing research. Singularities pose a major challenge in this context, as they make numerical integration difficult.

TOPP with first-order constraints and second-order constraints can also be effectively handled by convex optimization [9]. However, the inclusion of third-order constraints leads to a non-convex optimization problem [9], [10]. To address this issue, a convex approximation based on linear constraints is proposed in [10]. As an alternative, third-order constraints can be approximated by adding a penalty term to the objective function of the optimization problem [11].

While TOPP is typically performed offline, there are also online methods capable of considering jerk constraints. In [12] and [13], a jerk-limited path tracking technique is proposed, however, without focusing on the traversing time. As mentioned in the introduction, it is also possible to compute a time-optimal trajectory from one kinematic state to another considering jerk constraints [3], [14]–[19]. The Reflexxes motion library [17] can process a combination of a target position and a target velocity as a target state. A more recent development, the Ruckig library [3], goes a step further by also supporting target accelerations. We use the open-source community version of Ruckig as a backend for our calculations. There is also a commercial closed-source version called Ruckig pro, which additionally supports intermediate waypoints. By densely sampling waypoints from a given path, Ruckig pro can be used to generate an approximate path parameterization. In our evaluation, we consider the results of this approximation as a benchmark.

In [20], it is shown that an upper and a lower trajectory can be computed for each joint such that position, velocity, acceleration and jerk constraints are not violated. This insight is used to construct a continuous action space for reinforcement learning in which each action leads to a feasible trajectory. Using this action space, it is possible to train a neural network to follow a path without violating jerk constraints [21]. As a reference, we also provide the results of this method in our evaluation section.

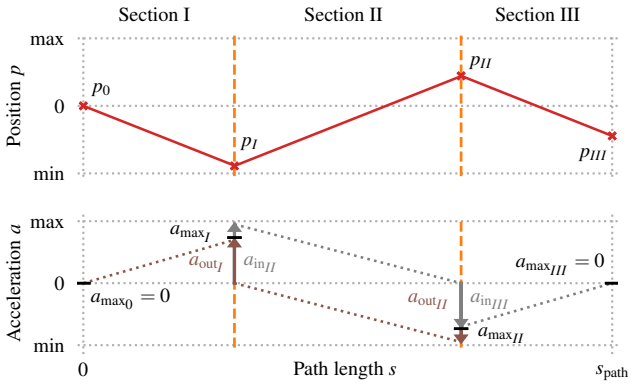


Fig. 2: The range of valid target accelerations for each waypoint $p_0, p_I, p_{II}, p_{III}$ includes zero as lower limit and $a_{\max_0}$, $a_{\max_I}$, $a_{\max_{II}}$ and $a_{\max_{III}}$ as upper limit, respectively.

III. Approach

A. Overview

In a first step, we analyze the traversal of a one-dimensional path subject to the following constraints:

$$v_{\min} \leq \dot{p} \leq v_{\max} \tag{1}$$
$$a_{\min} \leq \ddot{p} \leq a_{\max} \tag{2}$$
$$j_{\min} \leq \dddot{p} \leq j_{\max}, \tag{3}$$

where p, v, a and j stand for position, velocity, acceleration and jerk, respectively.

As shown in Fig. 2 and explained in (III-B), a range of feasible target accelerations can be computed for each intermediate waypoint of a one-dimensional path. Given a feasible target acceleration, the Ruckig library [3] can be used to compute a time-optimal trajectory from one waypoint to another without violating the constraints (1) - (3).

In section (III-C), we explain how the traversal on the path can be controlled by repeatedly computing an upper and a lower trajectory.

In section (III-D), we finally introduce an iterative scheme to closely follow a multi-dimensional path.

B. Computing feasible target accelerations

In this section, we analyze feasible kinematic states (p, v, a) for each waypoint p_0, p_I, p_{II}, p_{III} of a one-dimensional path shown in Fig. 2. While our exemplary path has two intermediate waypoints p_I and p_{II}, the same principle can be applied to arbitrary one-dimensional paths. It is evident that the velocity and acceleration of the first waypoint p_0 and the last waypoint p_{III} must be zero. The intermediate waypoints p_I and p_{II} are local extrema of the path. Consequently, their corresponding velocities also need to be zero. In order to compute feasible target accelerations for the intermediate waypoints, we first consider the movement from p_0 to p_I. Since p_I is a local minimum of the path, the target acceleration must be greater than or equal to zero. Starting from stillstand, a target acceleration of zero is always possible as it results in a normal point-to-point motion. The range of potential target accelerations a_{out_I} is

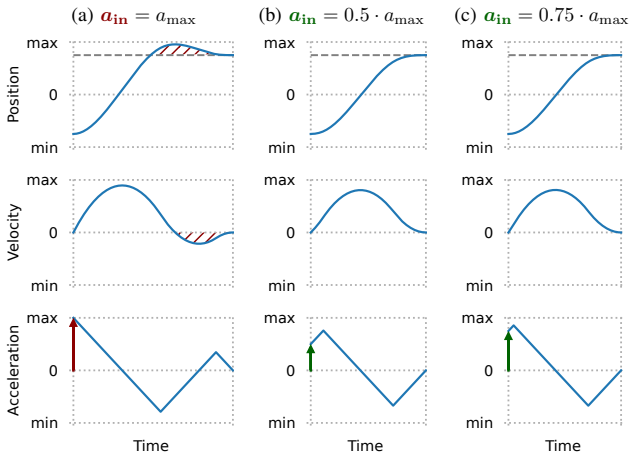


Fig. 3: Binary search to find $a_{\text{in}_{\max}}$ assuming $a_{\text{out}} = 0$. After the steps (a), (b) and (c), it can be said that $a_{\text{in}_{\max}}$ is greater than or equal to $0.75 \cdot a_{\max}$ but less than $a_{\max}$.

continuous. For that reason, all potential target accelerations are known, once the maximum target acceleration $a_{\text{out}_{\max_I}}$ is found. However, it could happen that $a_{\text{out}_{\max_I}}$ is greater than the maximum input acceleration of the next section $a_{\text{in}_{\max_{II}}}$. Consequently, the maximum target acceleration $a_{\max_I}$ is calculated as follows:

$$a_{\max_I} = \min(a_{\text{out}_{\max_I}}, a_{\text{in}_{\max_{II}}}) \tag{4}$$

Both, $a_{\text{out}_{\max_I}}$ and $a_{\text{in}_{\max_{II}}}$ are determined by a binary search. In Fig. 3, the principle is illustrated for the maximum input acceleration $a_{\text{in}_{\max}}$. In a first step (a), $a_{\text{in}_{\max}}$ is assumed to be $a_{\max}$. Using Ruckig, a trajectory from the current waypoint p_{current} to the next waypoint p_{next} is computed, selecting both the target velocity and the target acceleration to be zero. If the resulting trajectory is valid, the maximum input acceleration is $a_{\max}$. If not, another test (b) is performed, assuming $a_{\text{in}_{\max}} = 0.5 \cdot a_{\max}$. A trajectory is considered as valid if:

- The kinematic limits (1) - (3) are not violated.
- The velocity is never negative if $p_{\text{next}} > p_{\text{current}}$ or never positive if $p_{\text{next}} < p_{\text{current}}$.

In case of (a), the position overshoots the target. Consequently, the trajectory is considered as invalid as the velocity must become negative to compensate for the overshoot. In step (b) and step (c), the resulting trajectory is considered as valid. As a result, $a_{\text{in}_{\max}}$ must be greater than or equal to $0.75 \cdot a_{\max}$ but less than $a_{\max}$. The binary search is continued until $a_{\text{in}_{\max}}$ is approximated sufficiently well.

In order to compute $a_{\text{in}_{\max_{II}}}$ for the intermediate section II, we assume a target acceleration of zero. To compute $a_{\text{out}_{\max_{II}}}$, we assume an input acceleration of zero. As shown in Fig. 4, it is nevertheless possible to compute a valid trajectory choosing an input acceleration of $a_{\text{in}_{\max_{II}}}$ and a target acceleration of $a_{\text{out}_{\max_{II}}}$. In fact, any combination of an input acceleration $\alpha \cdot a_{\text{in}_{\max}}$ and a target acceleration $\beta \cdot a_{\text{out}_{\max}}$ leads to a valid trajectory, with $\alpha, \beta \in [0.0, 1.0]$. The higher the value of α or β, the faster the traversal. The smallest traversal time $t_{\min}$ results from selecting $\alpha = \beta = 1.0$. Thus, a time-optimized traversal

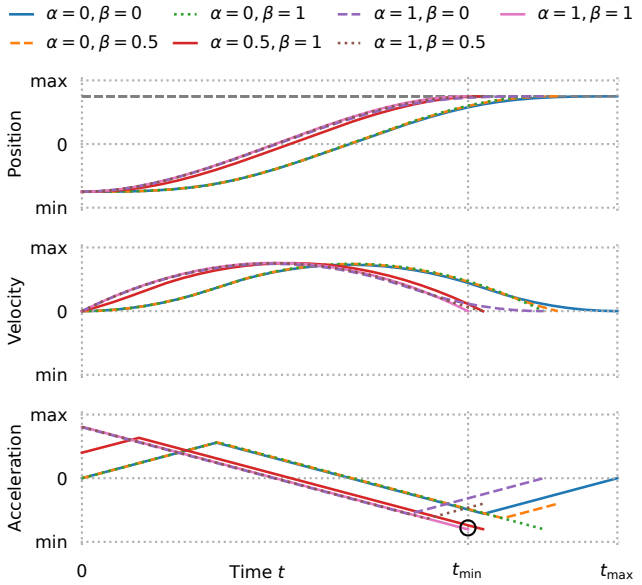


Fig. 4: Resulting trajectories when selecting an input acceleration of $\alpha \cdot a_{\text{in}_{\max}}$ and a target acceleration of $\beta \cdot a_{\text{out}_{\max}}$

of the path shown in Fig. 1 and Fig. 2 is achieved by selecting $a_I = a_{\max_I}$ and $a_{II} = a_{\max_{II}}$. As highlighted by a circle in Fig. 4, the acceleration of the fastest trajectory, shown in pink, slightly goes up before reaching $a_{\text{out}_{\max}}$. Thus, a slightly higher target acceleration could be selected, leading to a slightly smaller traversal time. However, this target acceleration could no longer be reached from an input acceleration of zero. In practice, we found the loss of time caused by including zero as a valid input acceleration and a valid target acceleration to be small.

C. Controlling the traversal of a one-dimensional path

In Fig. 5, a path with one intermediate waypoint p_I is traversed. As explained before, a time-optimized path traversal is composed of the following two parts: A trajectory from (p_0, 0, 0) to (p_I, 0, $a_{\max_I}$), followed by a trajectory from (p_I, 0, $a_{\max_I}$) to (p_{II}, 0, 0). We call the resulting trajectory upper trajectory. In contrast, the smallest feasible progress on the path is attained through a so-called lower trajectory. At the beginning of a section, the lower trajectory corresponds to a simple braking trajectory. The braking trajectory can be calculated with Ruckig by specifying a target velocity and a target acceleration of zero. However, if the resulting braking trajectory does not stay on the desired path, the calculation is adjusted. More precisely, when traversing section I, a target state (p_I, 0, $a_{\min_I}$) is selected. The acceleration $a_{\min_I}$ is less than or equal to $a_{\max_I}$ and is computed in a similiar way using a binary search. As a next step, we discretize the time t into small time steps with a time distance of Δt and compute the kinematic states ($p_{\text{lower}_{\Delta t}}$, $v_{\text{lower}_{\Delta t}}$, $a_{\text{lower}_{\Delta t}}$) and ($p_{\text{upper}_{\Delta t}}$, $v_{\text{upper}_{\Delta t}}$, $a_{\text{upper}_{\Delta t}}$) at the following time step. As shown in Fig. 5, a position and its corresponding section can be uniquely mapped to a path length s. We map $p_{\text{lower}_{\Delta t}}$ and $p_{\text{upper}_{\Delta t}}$ to $s_{\text{lower}_{\Delta t}}$ and $s_{\text{upper}_{\Delta t}}$ and define:

$$s_{\text{desired}_{\Delta t}} = s_{\text{lower}_{\Delta t}} + m \cdot (s_{\text{upper}_{\Delta t}} - s_{\text{lower}_{\Delta t}}), \tag{5}$$

with m being a mapping factor $\in [0.0, 1.0]$.

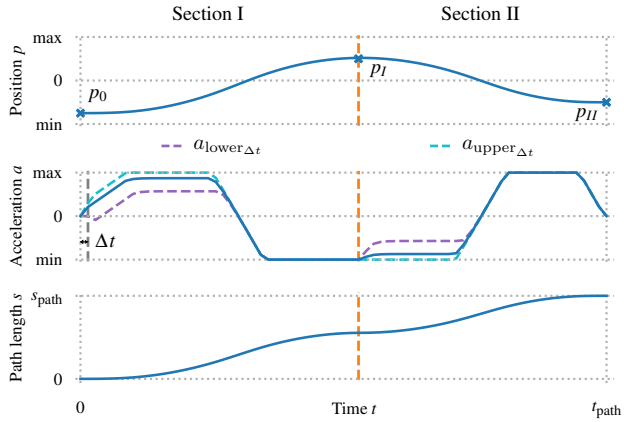


Fig. 5: Traversing a path with a fixed mapping factor of 0.7.

For small time distances Δt, an intermediate trajectory leading to a path length close to $s_{\text{desired}_{\Delta t}}$ can be found. Consequently, the mapping factor allows us to control the traversal of the path. After each time step, the lower and the upper trajectory are recomputed. As shown in Fig. 5, both trajectories are almost identical prior to a section change. Thus, the traversal can hardly be influenced during this phase, which motivates the following iterative scheme.

D. An iterative scheme to track multi-dimensional paths

In this section, we present an iterative scheme to select the mapping factors such that a multi-dimensional path is approximately tracked. Fig. 6 visualizes the first two iterations of the method for a two-dimensional path. As an initial step, we compute the time-optimized traversal of each individual path dimension and identify the slowest dimension. For the slowest dimension, we determine the path length $s(t)$ over time corresponding to the time-optimized traversal. The path length $s(t)$ of an individual dimension can be uniquely mapped to a path length $u(t)$ of the multi-dimensional path. The corresponding $u(t)$ of the slowest dimension serves as a reference u_{ref_1} for the first iteration of our method. The basic idea of the next step is to track this reference with the other path dimensions by selecting suitable mapping factors. To do so, we define a small range around u_{ref} where the other dimensions should stay in. The deviation from the reference u_{ref} is denoted as Δu_{ref}. Likewise, the corresponding position deviation is denoted as Δp_{ref}. Looking at Δu_{ref_1} in Fig. 6, it can be seen that the mapping factors m_1 are selected such that dimension 2 oscillates within the desired Δu_{ref} range. However, if it is not possible to keep the dimension within the desired range, the mapping factors are selected such that the lower bound of the range is undershot. The corresponding areas and their resulting position deviation are hatched in red. We now look for the area outside the desired Δu_{ref} range that causes the largest integrated position deviation of all dimensions. As indicated by a dashed black line in Fig. 6, we update the reference u_{ref_1} in the selected area such that dimension 2 stays within the desired Δu_{ref} range in the following iteration. As a consequence, the resulting position deviation after the second iteration Δp_{ref_2} is significantly

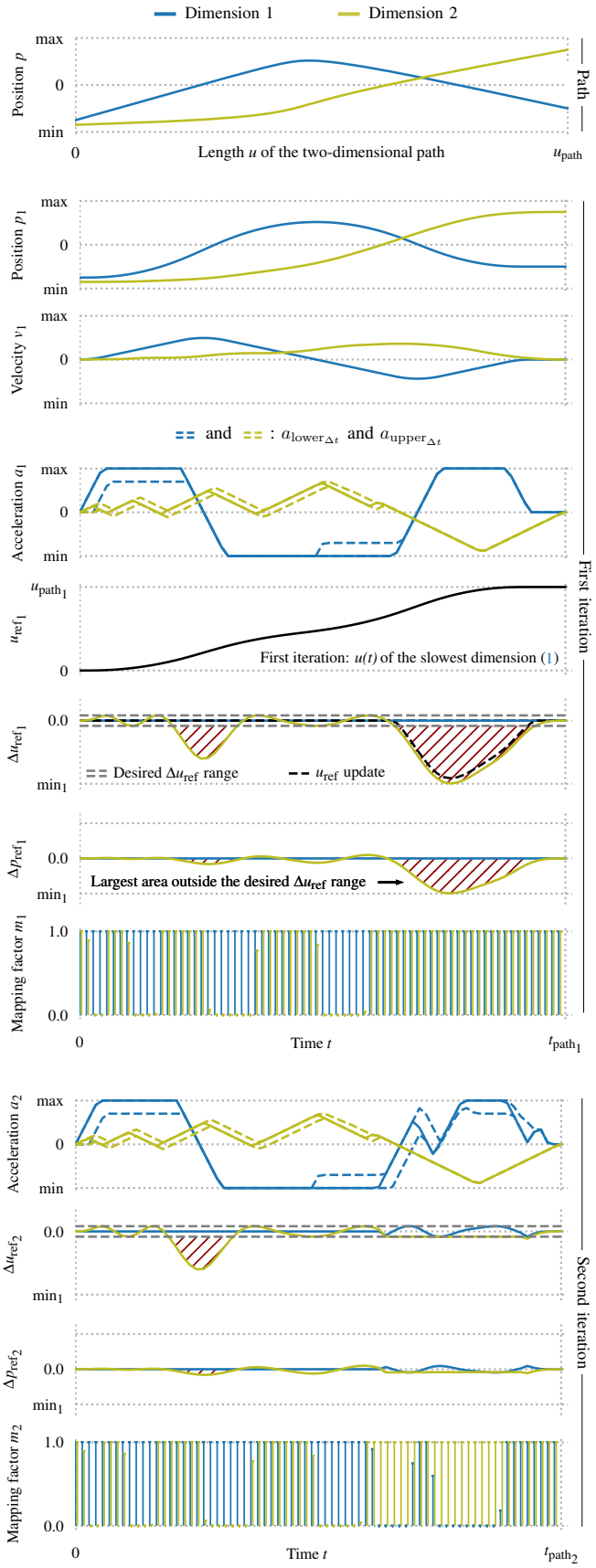


Fig. 6: The first two iterations of the proposed iterative tracking scheme shown for an exemplary two-dimensional path.

reduced compared to $\Delta p_{\mathrm{ref}_1}$. By repeating this iterative procedure, the tracking accuracy can be further increased. We note that the updated reference can decrease the tracking accuracy of other path dimensions. In practice, we repeat the procedure for a fixed number of iterations and choose the iteration with the best overall tracking performance as the final path parameterization.

IV. Evaluation

For our evaluation, we use a KUKA iiwa robot with seven degrees of freedom. The selected kinematic limits for each joint are shown in TABLE I. If not noted otherwise, the jerk limits from the table apply. However, for a more thorough evaluation, we additionally perform experiments with different jerk limits. In these cases, we report a so-called jerk limit factor that is multiplied with the jerk limits given in TABLE I.

TABLE I: Kinematic limits of the robot joints.

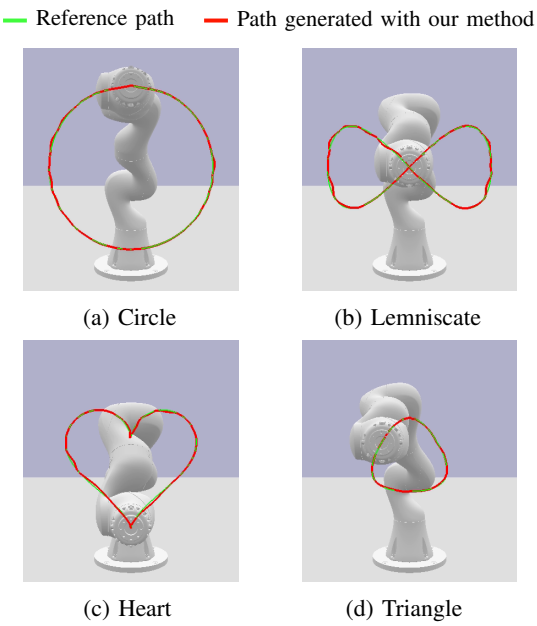
	Joint						
	1	2	3	4	5	6	7
Velocity [rad/s]	1.71	1.71	1.74	2.27	2.44	3.14	3.14
Acceleration [rad/s^2]	15.0	7.5	10.0	12.5	15.0	20.0	20.0
Jerk [rad/s^3]	300	150	200	250	300	400	400

A. Time-optimized traversal of one-dimensional paths

As a first step, we evaluate the traversal of one-dimensional paths. For each of the seven robot joints, we generate 200 one-dimensional paths with randomly selected waypoints. To minimize the trajectory duration, we select the maximum acceleration for each intermediate waypoint as derived in section (III-B). As a benchmark, we report the results of the closed-source library Ruckig pro. TABLE II shows the resulting average trajectory duration and tracking accuracy for several jerk limit factors. As expected, the trajectory duration decreases if the jerk limits are increased. However, the relative impact on the trajectory duration decreases for higher jerk limit factors. Compared to Ruckig pro, the results of our method are almost identical.

TABLE II: Time-optimized traversal of one-dimensional paths for different jerk limits (Ours / Ruckig pro).

Jerk limit factor	Trajectory duration [s]	Path deviation [rad] mean	Path deviation [rad] max
• Factor 1	5.15 / 5.16	0.0007 / 0.0026	0.0014 / 0.0052
• Factor 2	5.03 / 5.03	0.0007 / 0.0006	0.0014 / 0.0012
• Factor 3	5.00 / 5.00	0.0007 / 0.0006	0.0014 / 0.0012
• Factor 40	4.96 / 4.96	0.0007 / 0.0007	0.0014 / 0.0013



(a) Circle (b) Lemniscate

(c) Heart (d) Triangle

Fig. 7: Geometric shapes used for our evaluation.

B. Tracking of geometric shapes

As shown in Fig. 7, we apply our iterative tracking method to seven-dimensional paths that resemble geometric shapes in Cartesian space. It can be seen that the resulting Cartesian paths shown in red stay close to the reference paths shown in green. For further visualization, we refer to our accompanying video. A quantitative evaluation can be found in TABLE III. Using $\Delta t = 2.5\,\mathrm{ms}$ as time step, 20 iterations were performed as described in section III-D. Finally, the path with the lowest average deviation from the reference path was selected. The path deviation is computed by densely sampling waypoints along the reference path and the generated path. Next, waypoints sampled at the same path length u are compared by calculating the norm between them. TABLE III shows the mean and the maximum norm calculated in this way.

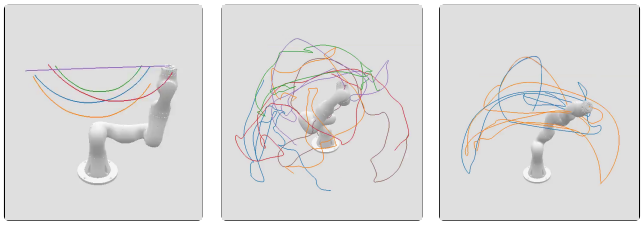
We also provide results obtained with TOPP-RA [2] and Ruckig pro. With TOPP-RA, the path tracking is very accurate but the jerk limits are ignored. Ruckig pro supports jerk limits but is designed to accurately track intermediate waypoints rather than a full path. As a reference, we provide the results for an equidistant sampling of waypoints from the reference path using a distance of 0.1 rad (A) and 0.2 rad (B). Overall, our method managed to generate fast trajectories for all of the geometric shapes shown in Fig. 7.

TABLE III: Benchmarking using geometric shapes (Ruckig pro A: 0.1 rad, Ruckig pro B: 0.2 rad).

	Trajectory duration [s]				Path deviation (mean / max) [rad]			
	Ours	TOPP-RA	Ruckig pro A	Ruckig pro B	Ours	TOPP-RA	Ruckig pro A	Ruckig pro B
• Circle	2.84	2.79	3.64	2.91	0.0041 / 0.0083	$<10^{-7}$ / $<10^{-6}$	0.0037 / 0.0096	0.0110 / 0.0229
• Lemniscate	2.83	2.75	3.68	2.95	0.0045 / 0.0108	$<10^{-7}$ / $<10^{-6}$	0.0046 / 0.0104	0.0115 / 0.0235
• Heart	2.78	2.67	3.28	2.76	0.0042 / 0.0147	0.0003 / 0.0008	0.0231 / 0.0420	0.0080 / 0.0282
• Triangle	1.64	1.24	1.52	1.29	0.0033 / 0.0058	$<10^{-7}$ / $<10^{-6}$	0.0038 / 0.0068	0.0059 / 0.0228

TABLE IV: Benchmarking with a method using neural networks based on three datasets with different path characteristics.

	Total path length	Trajectory duration [s]			Path deviation (mean / max) [rad]		Iteration
	u_{path} [rad]	Slowest dimension	Ours	Neural network	Ours	Neural network	Ours
• Dataset A	3.7	1.79	2.06	2.11	0.0027 / 0.0098	0.0394 / 0.0831	9.5
• Dataset B	22.9	5.55	6.69	8.62	0.1122 / 0.2623	0.1088 / 0.2113	12.3
• Dataset C	25.6	6.68	7.67	10.51	0.0962 / 0.2283	0.1227 / 0.2685	12.1



(a) Dataset A (b) Dataset B (c) Dataset C

Fig. 8: Exemplary paths from our datasets.

C. Tracking of paths with different characteristics

In the following, we evaluate our method using three datasets with different path characteristics. Exemplary paths for each dataset are shown in Fig. 8. Dataset A contains a wide range of semicircles and straight lines. The paths in dataset B are generated by selecting random joint accelerations. Dataset C is composed of paths obtained by moving between randomly sampled Cartesian target points. In TABLE IV, we report the results for each dataset and compare them with [21], a method that uses neural networks trained via reinforcement learning to track the reference paths. The indicated trajectory duration of the slowest individual path dimension serves as a lower limit for the resulting trajectory duration when considering the multi-dimensional path. It can be seen, that the presented method generates faster trajectories than [21] for all datasets. In addition, the tracking accuracy for dataset A and dataset C is higher. In TABLE IV, we also specify the average iteration chosen for our evaluation.

TABLE V shows additional results obtained with TOPP-RA and Ruckig pro. For dataset A, TOPP-RA generates the fastest trajectories with the highest path accuracy, however, without considering jerk limits. Using Ruckig pro with a waypoint distance of 0.1 rad leads to slower trajectories, but a higher tracking accuracy compared to our method. In contrast, selecting a waypoint distance of 0.15 rad or 0.2 rad leads to faster trajectories but a less accurate path tracking. For dataset B and dataset C, the resulting trajectories with TOPP-RA are slower than with our method but the tracking accuracy is higher.

In TABLE VI, we finally analyze the impact of the selected jerk limits on the resulting trajectory duration and path deviation. As expected, higher jerk limits lead to a faster traversal of the reference paths. Moreover, the accuracy of the tracking improves. Thus, we conclude that higher jerk limits simplify the tracking of a desired reference path length u_{ref}.

TABLE V: Additional benchmarking using datasets with different path characteristics

	Trajectory duration [s]	Path deviation [rad] mean	max
Dataset A			
• TOPP-RA	1.88	$<10^{-7}$	$<10^{-8}$
• Ruckig pro 0.1 rad	2.13	0.0019	0.0060
• Ruckig pro 0.15 rad	2.02	0.0039	0.0103
• Ruckig pro 0.2 rad	1.98	0.0064	0.0153
Dataset B			
• TOPP-RA	7.18	$<10^{-6}$	$<10^{-6}$
Dataset C			
• TOPP-RA	7.83	$<10^{-5}$	$<10^{-5}$

TABLE VI: Impact of the selected jerk limits.

	Trajectory duration [s]	Path deviation [rad] mean	max
Dataset B			
• Jerk limit factor 1.5	6.53	0.0704	0.1893
• Jerk limit factor 2.0	6.48	0.0546	0.1614
• Jerk limit factor 4.0	6.43	0.0491	0.1481
Dataset C			
• Jerk limit factor 1.5	7.50	0.0517	0.1452
• Jerk limit factor 2.0	7.47	0.0388	0.1121
• Jerk limit factor 4.0	7.38	0.0253	0.0874

V. Conclusion and Future Work

We presented a method to approximately track multi-dimensional paths considering jerk limits. As a first step, we analyzed each path dimension individually. For every intermediate waypoint of a one-dimensional path, we computed a range of feasible accelerations using a binary search algorithm. We then computed a lower and an upper trajectory to achieve minimum and maximum progress on the one-dimensional path, respectively. This allowed us to control the traversal on the path in such a way that a selected reference path length of a multi-dimensional path could be approximately tracked over time. We then applied an iterative scheme, where the reference path length was adjusted at each iteration such that the largest occurring position deviation diminished. Our evaluation on geometric shapes and datasets with different characteristics showed that our method succeeded in quickly traversing the reference paths while keeping the path deviation low.

In future work, we would like to further improve our method, e.g., by searching for continuous points in time to switch between the upper and lower trajectory.

REFERENCES

[1] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robotics: Science and Systems VIII*, pp. 1–8, 2012.

[2] 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.

[3] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," *Robotics: Science and Systems XVII*, 2021.

[4] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The international journal of robotics research*, vol. 4, no. 3, pp. 3–17, 1985.

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

[6] M. Tarkiainen and Z. Shiller, "Time optimal motions of manipulators with actuator dynamics," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*. IEEE, 1993, pp. 725–730.

[7] J. Mattmüller and D. Gisler, "Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path," *The International Journal of Advanced Manufacturing Technology*, vol. 45, pp. 1007–1016, 2009.

[8] 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, 2017, pp. 679–686.

[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] 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*. IEEE, 2013, pp. 2778–2783.

[11] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 3, pp. 415–426, 2008.

[12] F. Lange and M. Suppa, "Trajectory generation for immediate path-accurate jerk-limited stopping of industrial robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2021–2026.

[13] F. Lange and A. Albu-Schäffer, "Path-accurate online trajectory generation for jerk-limited industrial robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 82–89, 2015.

[14] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Transactions on robotics and automation*, vol. 19, no. 1, pp. 42–52, 2003.

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

[16] X. Broquere, D. Sidobre, and I. Herrera-Aguilar, "Soft motion trajectory planner for service manipulator robot," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 2808–2813.

[17] T. Kröger, "Opening the door to new sensor-based robot applications—the reflexxes motion libraries," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4.

[18] M. Hehn and R. D'Andrea, "Real-time trajectory generation for quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, 2015.

[19] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 87–96.

[20] J. C. Kiemel and T. Kröger, "Learning robot trajectories subject to kinematic joint constraints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4799–4805.

[21] J. C. Kiemel and T. Kröger, "Learning time-optimized path tracking with or without sensory feedback," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4024–4031.