Efficient and robust trajectory generation for robotic manipulators

Oriol Ruiz Leopold Palomo-Avellaneda Raúl Suárez Jan Rosell

Institute of Industrial and Control Engineering
Universitat Politècnica de Catalunya
Barcelona, Catalonia, Spain
oriol.ruiz.celada@estudiantat.upc.edu, leopold.palomo@upc.edu, raul.suarez@upc.edu, jan.rosell@upc.edu

Abstract—**This paper describes a procedure to generate valid robot trajectories for a given sequence of points defining a geometric path, which is a quite frequent output of many robot motion planners. The proposed approach takes into account physical constraints of the robot, as the maximum velocity and the maximum acceleration that each joint can reach, and has been implemented in C++ as a general tool that can be used with different robots.**

Index Terms—**robotics, robot trajectories, motion planning.**

I. Introduction

Current robotic applications frequently use motion planners to generate the robot paths as a sequence of joint configurations (waypoints) that solves the task and avoid potential collisions [1] [2]. This is a practical solution that still requires, in a second step, the generation of the robot trajectories, i.e., adding the velocities and accelerations to the geometric path defined by the waypoints. Usually, this is done by simple "moving" the robot from one waypoint to the next one in the sequence, but, depending on the robot controller, this generates non-smooth movements because the robot unnecessarily breaks and accelerates between the waypoints in a non-optimal way, and even more, the closer the points are (in order to better define the geometric path) the worse the quality of the robot movements could be.

To deal with this problem, this work introduces a procedure to obtain valid robot trajectories for a given sequence of waypoints defining a geometric path. The proposed approach takes into account physical constraints of the robot, as the maximum velocity and the maximum acceleration that each joint can reach.

After this introduction the paper is organized as follows. Section II briefly describes and formalize the addressed problem. Section III introduces and discusses relevant related previous works. Section IV describes the proposed approach, and Section V provides an illustrative application example. Finally, Section VI summarized the work contributions and the future work.

This work was partially supported by the Spanish Government through the project PID2020-114819GB-I00.

II. Problem definition

For a K-degrees-of-freedom robot manipulator, consider a piecewise rectilinear collision-free path defined by a sequence of N joint configurations that satisfy the joint limits. The path can be represented by a matrix WP:

$$\mathrm{WP} = \begin{bmatrix} q_1^1 & \cdots & q_1^K \\ \vdots & & \vdots \\ q_N^1 & \cdots & q_N^K \end{bmatrix},$$

where $\mathrm{WP}(i,k) = q_i^k$ represents the k-th joint value of the i-th waypoint of the path. We will refer to the interval between two consecutive waypoints as a *segment*.

The goal is to find a trajectory $\mathbf{q}(t) = (q^1(t), \ldots, q^K(t))$ through the N waypoints, i.e., to specify the time t_i $i = 1, \ldots, N$ to pass through each waypoint, as well as the velocity and acceleration profiles, while satisfying the following constraints:

- Total trajectory time $t_N = T$ as short as possible,
- $\mathbf{q}(t_i) = (q_i^1, \ldots, q_i^K),\ i = 1, \ldots, N,$
- Joint velocities limited to $\dot{q}_{max}^k,\ k = 1, \ldots, K,$
- Joint accelerations limited to $\ddot{q}_{max}^k,\ k = 1, \ldots, K,$
- Null initial and final velocities and accelerations,
- No sign change in the velocity between waypoints.

III. Previous approaches

Among the classic trajectory generation methods, the simplest one consists in assigning a total time for the whole trajectory and divide it among the waypoints assuming linear interpolation, which is used, for instance, to animate a path in a simulation. This procedure is usually adjusted to satisfy velocity constraints by setting the time between any two consecutive waypoints large enough in order not to require (for any of the K joints) a velocity greater than the maximum to cover the corresponding joint interval. However, this leads to unwanted discontinuities in the velocity, which can be avoided with a polynomial interpolation that requires the definition of velocities (or velocities and accelerations) at the waypoints. The method guarantees that the constraints at the waypoints are met, although they may not in between waypoints. Another usual solution uses linear

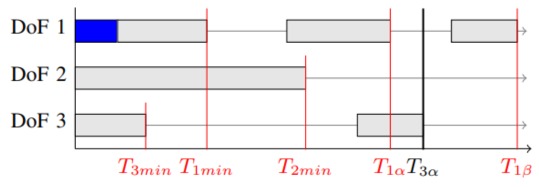

Fig. 1: Time synchronization step. All the intervals in which each DoF does not have a solution are shown in grey, and the black vertical line indicates the minimal possible time in which all DoFs can be unblocked [10].

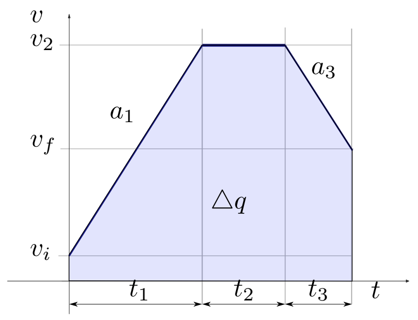

Fig. 2: Generic velocity profile for each joint in each segment.

polynomials with parabolic blends, which connects the waypoints lineally but introduces a parabolic blends to smooth the trajectories [3], but this solution does not ensure that the path passes through the waypoints. This method is used by default in MoveIt [4].

Other approaches deal with the problem of finding time-optimal trajectories under joint position, velocity, acceleration, and jerk limits by using nonlinear constraints optimization methods [5], or linear programming [6], or for the case of trajectories with blends, by combining a trapezoidal acceleration model with a 7-degree polynomial to form a path with blends [7].

The problem has also been approached using control techniques [8] [9] but in these cases the dynamic model of the robot is required.

Recently, another approach, called *Ruckig*, that allows the generation of jerk-limited real-time trajectories has been introduced [10]. The resulting trajectories are continuous to the second derivative, meaning that position, velocity and acceleration are continuous, while the jerk is not. The algorithm considers for each segment and degree of freedom (DoF) different profile types, and selects the one that guarantees the limits, obtaining an optimal time and blocked intervals for that profile (blocked intervals are time intervals in which it is impossible to reach the next waypoint using a selected profile). Then, it performs a synchronization step for each segment, in which all the DoF are stretched to the minimum time that does not fall under any blocked interval for any DoF (Figure 1). This approach is really interesting, but the use of current available implementation of the Ruckig library for the interpolation of trajectories with multiple waypoints requires the connection to a server, which in many cases greatly reduces its practical use. Moreover, the method is geared towards dynamic tasks, which is not always necessary, and requires the definition of the position, the velocity and the acceleration at each waypoint, which may not be always available.

The proposal presented here aims at solving the drawbacks of the polynomial interpolation.

IV. Proposed approach

This section describes the proposed approach to obtain feasible trajectories from a set of waypoints. The first subsection describes the generic velocity profile used for each joint in each segment, and the second subsection describes the developed procedure to compute valid velocity profiles for each joint using and adapting the generic velocity profile.

A. Generic velocity profile for each segment

As, in general, infinite trajectories are possible for each joint, we fix a generic velocity profile composed of three intervals in each segment (see Fig.2):

- Interval 1: Acceleration (or deceleration), where acceleration a_1 is applied during an interval t_1.
- Interval 2: Constant velocity, where a velocity v_2 is applied during an interval t_2.
- Interval 3: Deceleration (or acceleration), where acceleration a_3 is applied during an interval t_3.

This 3-interval profile is determined for each of the robots joint by,

$$\begin{cases} q_f = q_i + v_i t_1 + \frac{1}{2} a_1 (t_1)^2 + v_2 t_2 + v_2 t_3 - \frac{1}{2} a_3 (t_3)^2 \\ v_2 = v_i + a_1 t_1 \\ v_f = v_2 - a_3 t_3 \\ t_{segment} = t_1 + t_2 + t_3 \end{cases} \tag{1}$$

where, the known data are: the initial and final positions of the joint, q_i and q_f respectively, the initial velocity of the joint v_i, the (maximum) acceleration and deceleration, a_1 and a_3 respectively, and the unknowns are: the final velocity v_f and the duration of each interval, t_1, t_2 and t_3 and their summation $t_{segment}$. Note that t_1, t_2 or t_3 could be equal 0, making the velocity profile to actually have only one or two real intervals.

The set of positions of each joint in the waypoints can be splitted into monotonic increasing or decreasing subsets, with zero velocity at the initial and ending points of the sequence; for simplicity and to fit into the available space for the paper, we will consider here the incremental case, i.e., in (1) we assume $q_f > q_i$. The

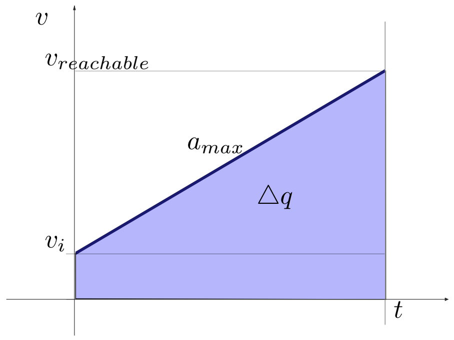

Fig. 3: Example of the velocity $v_{reachable}$ in one segment for one joint.

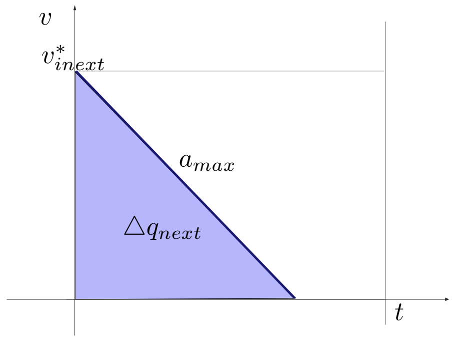

Fig. 4: Example of the velocity v^*_{inext} in one segment for one joint.

decreasing case is simply tackled with a sign change in the velocities and accelerations and if a joint has the same position in consecutive points the velocity along the segment is null.

B. *Finding the velocity profiles for each joint in each segment*

The first step of the approach is to determine the total time $t_{segment}$ required for each segment. This can be done using (1) if the final velocity v_f is known, so we look first for a proper value of v_f that allows an initial estimation of $t_{segment}$ for each joint. There are four constraints that bound the maximum value of v_f:

- 0: when the next value of the joint is a (may be local) extreme of the sequence of joint positions;
- v_{max}: the maximum velocity achievable by the joint;
- $v_{reachable}$: the maximum velocity the joint can reach starting with velocity v_i and applying maximum acceleration a_{max} until covering the required displacement Δq in the segment(see Fig. 3), i.e.,

$$v_{reachable} = \sqrt{(v_i)^2 + 2a_{max}\Delta q} \qquad (2)$$

- v^*_{inext}: the maximum starting velocity of the next segment that guarantees that the joint can decelerate with $-a_{max}$ until zero velocity without advancing more than the required displacement Δq_{next} in the segment (see Fig. 4), i.e.,

$$v^*_{inext} = \sqrt{2a_{max}\Delta q_{next}} \qquad (3)$$

Since all these are maximum constrains, selecting the minimum of them assures the validity of the selection, thus, calling $v_{adequate}$ to the valid value:

- $v_{adequate} = 0$ if the next joint position is a (local) extreme, and,
- $v_{adequate} = min(v_{max}, v_{reachable}, v^*_{inext})$ otherwise.

Once $v_f = v_{adequate}$ is known, the required $t_{segment}$ is obtained by solving (1). In this way, it is possible to obtain $t_{segment}$ for each joint and for each segment, but, in order for all the joints to be synchronized, the values of $t_{segment}$ must be equal in the same segment for all the joints. Thus, in order to assure the existence of a global solution, it is necessary to choose, for a given segment, the maximum value of $t_{segment}$ among those of all the joints.

Algorithm 1 calculateTrajectory

```
Input: WP, $v_{max}$, $a_{max}$
for $s = 1$ to $N-1$ do
    for $k = 1$ to $K$ do
        Compute $v^{s,k}_{adequate}$
        Compute $t^{s,k}_{segment}$
    end for
    $t_S = max\{t^{s,k}_{segment}\}$
    for all $k$ such that $t^{s,k}_{segment} \neq t_S$ do
        Re-compute $t^{s,k}_{segment}$
    end for
end for
Result: Joint velocity profiles
```

After this selection, the velocity profiles in the same segment of the joints with smaller $t_{segment}$ must be re-computed, and this done using again (1) to compute v_f, considering now as a known input the value of $t_{segment}$.

Considering a robot with $k = 1, ..., K$ joints and a set of N waypoints that therefore determine $s = 1, ..., N-1$ segments, this procedure is applied sequentially to all the segments, as described in Algorithm 1.

V. Simulation example

In order to illustrate the performance of the proposed approach, we present here an illustrative example simple enough to allow a clear visualization of the results. The considered case is a robot with 4 joints and a path defined by 6 waypoints given by the matrix

$$WP = \begin{pmatrix} 0.5 & -2.0 & 1.5 & 2.0 \\ 0.3 & -1.5 & 1.1 & 2.0 \\ -0.5 & -1.5 & 0.0 & 1.0 \\ -0.2 & 2.0 & -2.0 & 1.0 \\ 0.2 & -1.0 & 1.0 & 0.9 \\ 0.1 & -0.5 & 1.5 & 0.0 \end{pmatrix}$$

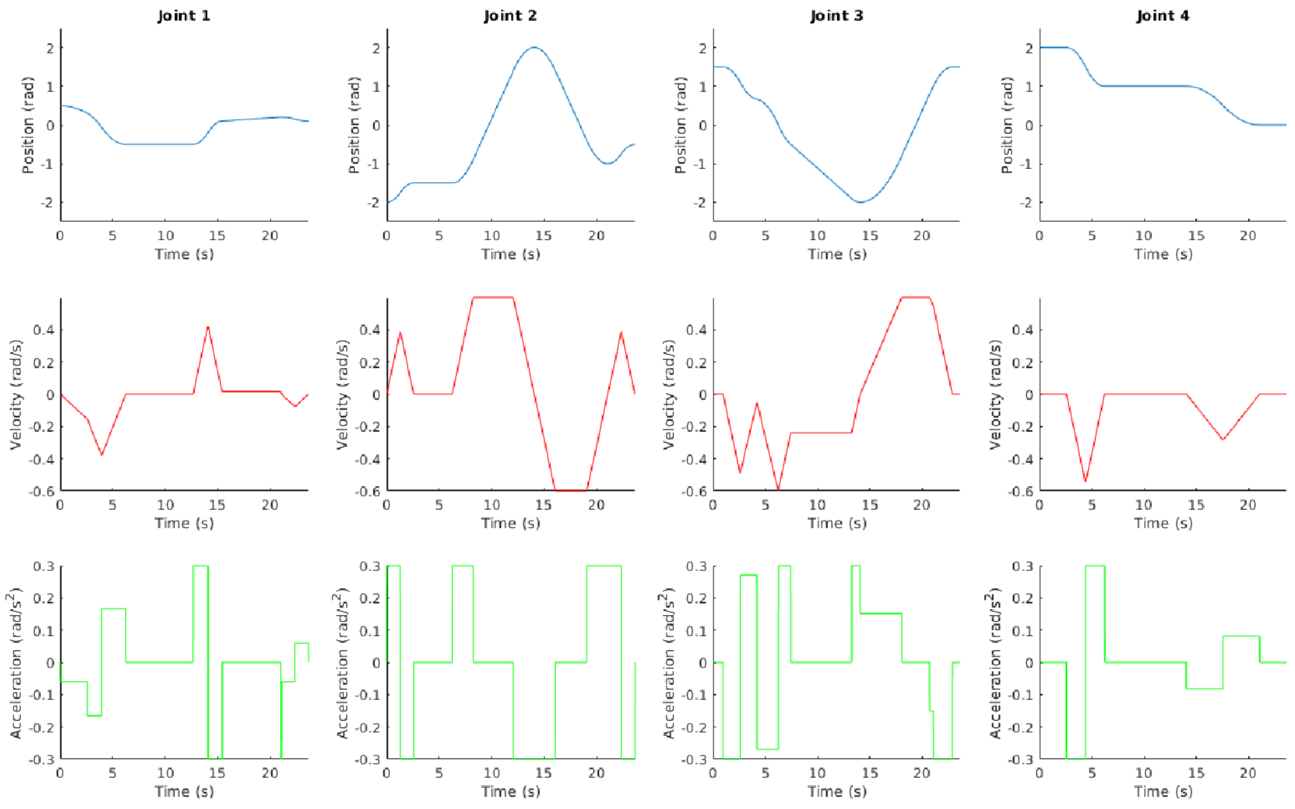

Fig. 5: Resulting position, velocity and acceleration profiles for the illustrative example.

The maximum velocity of all the joints was set to 0.6 rad/s and the maximum acceleration to 0.3 rad/s^2.

Figure 5 shows the obtained results for each joint, including the resulting position, velocity and acceleration profiles.

VI. Conclusions and future work

This work has proposed a simple yet efficient procedure to obtain a robot trajectory for a given sequence of waypoints defining a piece-wise rectilinear collision-free geometric path. The trajectories obtained are the fastest ones passing through all the waypoints and satisfying the maximum velocity and acceleration that each joint can reach. Currently the proposal is being thoroughly evaluated in a set of simulated and real scenarios in the laboratory. Future work will extend the proposal to jerk-limited trajectories.

References

[1] S. M. Lavalle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] H. Touzani, N. Seguy, H. Hadj-Abdelkader, R. Suárez, J. Rosell, L. Palomo-Avellaneda, and S. Bouchafa, "Efficient industrial solution for robotic task sequencing problem with mutual collision avoidance & cycle time optimization," *IEEE Robotics and Automation Letters*, pp. 2597–2604, 2021.

[3] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London: Springer, 2009.

[4] "Time Parameterization (MoveIt)," https://ros-planning.github.io/moveit_tutorials/doc/time_parameterization/time_parameterization_tutorial.html, accessed: 2022-07-22.

[5] J. Lin, N. Somani, B. Hu, M. Rickert, and A. Knoll, "An efficient and time-optimal trajectory generation approach for waypoints under kinematic constraints and error bounds," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5869–5876.

[6] A. Nagy and I. Vajk, "Sequential time-optimal path-tracking algorithm for robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1253–1259, 2019.

[7] J. Lin, M. Rickert, and A. Knoll, "Parameterizable and jerk-limited trajectories with blending for robot motion planning and spherical cartesian waypoints," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 982–13 988.

[8] 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.

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