

Trajectory Generation for Immediate Path-Accurate Jerk-Limited Stopping of Industrial Robots

Friedrich Lange and Michael Suppa

Abstract—Stopping the motion of industrial robots in response to warnings or unexpected sensor data is a special case of trajectory generation. In contrast to emergency stops, here the robot has to satisfy the limits of the acceleration and the jerk. In addition, during the deceleration the robot must follow the path accurately, i.e., the shape of the original path may not be left. This is usually done by scaling the desired velocity. However, for curved paths, e.g. those generated by blending of linear motion commands, by sensor corrections, or directly by splines, this method may leave the desired path. The problem is solved by interpolation using the arc length. In contrast to other methods, here the constraints are considered directly, resulting in a time-efficient computation. Finally, the proposed method prevents a rebound caused by the jerk limits when reaching zero velocity. Experiments are presented using a stiff KUKA robot whose path is exactly tracked during deceleration.

I. INTRODUCTION

This paper concerns the task of unexpected stopping a robot manipulator from continuous motion. Within this paper the robot is desired to stop as fast as possible *while keeping the desired path*. Procedures for this task are provided by robot manufacturers since the 90s. Since such algorithms are driven by industry and not by scientists, there is hardly any publication on this topic except from patents.

The common approach [1] is to scale the motion increment from the current to the next programmed pose by a factor α that is computed in such a way that the maximum acceleration is not exceeded in any axis. This factor can be computed from the acceleration limit of the most critical axis and then results in minimum stopping time. It can also be selected in order to represent a constant Cartesian braking ramp of e.g. 1 m/s². Both methods bring a path-accurate trajectory if the velocities of all axes are scaled by the same factor. This computation is the basis for the one presented in Section II.

The classical stopping procedures, however, have the following drawbacks:

- They only consider acceleration limits. In contrast, modern robot controllers additionally constrain the jerk in order to minimize mechanical abrasion in the gears.
- Path accuracy is only guaranteed for straight line motion commands (LIN) from the current to the next programmed pose. Such path segments typically end with a stop so that early stopping always ends before the final pose of the path segment. Trajectories which are curved because of blending between different straight line

segments or directly because of spline programming, are not preserved. Ref. [2] ensures a deterministic behavior for these cases; Ref. [3] accounts for singularities; both methods, however, still do not guarantee path-accurate stopping.

- Sensor data are not included even though they may modify the desired path during the deceleration phase.

In this paper a simple but efficient stopping procedure is presented that decelerates the robot motion as fast as possible while remaining on the currently desired path section and while satisfying all constraints on the acceleration and the jerk. The desired path section is updated in every sampling step (see e.g. [4]), thus allowing changes e.g. when following a contour using a sensor. In order to be independent of linear, circular, or blended path segments, the representation of the desired path is given by the desired axis positions $\mathbf{q}_d(k)$ at sampling steps k . The new computation of the braking trajectory needs far less than 1 ms computing time.

Path-accurate stopping is a special case of path-accurate trajectory generation with the distinctive feature that the target pose is not explicitly defined. Therefore we first review the generic trajectory generation topic, before the methods are transferred to the stopping case.

The authors recently presented a generic path-accurate trajectory generation method [5]. Therefore the state of the art is not repeated here in detail. As explained there, most papers treat the problem of generating a feasible trajectory that reaches the target pose in minimum time, without considering a desired path to the target, cf. [6], [7], [8], [9].

Concerning path-accurate trajectories, usually a velocity profile along the given desired path is computed. In contrast to Refs. [5], [10], in most papers the resulting trajectory is represented using a scalar parameter s , sometimes denoted as arc length. Then, the velocity profile is computed as $s(t)$ or as $\dot{s}(s)$ in the phase space. This velocity profile satisfies the constraints on the acceleration [11], [12], [13]. In later papers [14], [15], [16], [17], the constraints on the jerk are also being considered, and in [18] even the jerk derivative. The focus of these papers is on the computation of the accelerations in such a way that the velocities can be dissipated without overshooting, in order to get a time-optimal trajectory.

The methods typically feature three phases:

- 1) Express the kinematic or dynamic limits of the robot and the desired path by constraints on s , e.g. as a differential equation $f(s, \dot{s}, \ddot{s}) = 0$ or as $\dot{s} = f(s)$.
- 2) Compute the optimal solution exclusively in s , without considering the original constraints.

3) Map this scalar solution to the vectorial robot pose $\mathbf{q}(s(t))$ e.g. by computing $s(t)$.

In contrast to these phases, in Section III we present a simpler method that considers the limits directly in the time domain.

After the survey on previous work in [5] some methods have been presented that generate an end-effector trajectory in the presence of constraints. The problems of [19], [20] feature much more degrees of freedom and thus the computation is in the order of 10 to 100 ms, which is unacceptable for stopping an industrial robot. Refs. [21], [22] investigate the safety of humans. Ref. [21] computes the maximum velocity of a robot that carries humans such that braking will not exceed the acceleration limits that are tolerable for humans. Ref. [22] modifies the robot trajectory online in order to give way to a human hand.

The paper is organized as follows: In Section II the trajectory generating method of the authors in Ref. [5] is modified to path-accurate stopping. However, this method may feature undesired blending of the original path as explained in Section II-C. Therefore, in Section III an extension of that method is presented that interpolates the pose by using the arc length s . This is denoted by arc length interpolation ALI. Since using the method of Section II the vectorial pose is directly scaled, that method is denoted by direct pose interpolation (DPI) in this paper. The experiments in Section IV show the differences regarding path accuracy.

II. STOPPING BY DIRECT POSE INTERPOLATION (DPI)

The method presented in this paper is based on the procedure for trajectory generation in Ref. [5]. The task is there to limit velocity, acceleration and jerk of all robot axes in order to satisfy the given limits, while keeping the shape of the desired path. In each sampling step k the desired trajectory is recomputed, considering the current state (position, velocity and acceleration) of the robot. The goal is that the computed trajectory $\mathbf{q}_c(k + \kappa)$ will be synchronized with the given desired trajectory $\mathbf{q}_d(k + \kappa)$ as fast as possible. This is solved iteratively by forward computation (scaling) and backtracking.

For stopping, the goal is different. The goal is to stop as fast as possible while satisfying the constraints and without leaving the geometrical shape of $\mathbf{q}_d(k + \kappa)$. If $\mathbf{q}_d(k + \kappa)$ satisfies the constraints for all κ , this can be done without predicting the further desired path and thus without backtracking. Scaling is then sufficient, thus it does without iterations. In addition, the velocity constraints can be disregarded during braking.

Note that the following equations directly use sampled data, so that differentiability of the curves is not required. Instead, derivatives are computed by backward differences, omitting the sampling time T_0 (see the Appendix for the resulting representation).

A. Scaling the Velocity

Apart from the considerations in Section II-C, the original path is met if for each sampling step the velocity is scaled

by

$$\mathbf{q}_c(k) = \mathbf{q}_c(k-1) + \alpha(\mathbf{q}_d(k) - \mathbf{q}_c(k-1)) \quad (1)$$

and if $\mathbf{q}_c(k-1)$ is on the desired path.

In this equation the desired trajectory is represented by the desired position $\mathbf{q}_d(k) = (q_{d1} \cdots q_{d6})^T(k)$ in axis space, while $\mathbf{q}_c(k)$ represents the computed axis values at the current time step k , for decelerating instead of continuing the execution of the desired motion. For $\kappa < 0$, $\mathbf{q}_c(k + \kappa)$ denotes the executed trajectory, which is identical to $\mathbf{q}_d(k + \kappa)$ before the trigger of the stopping motion.

In order to limit the acceleration, α is computed by

$$\alpha_{ai} = \frac{v_{ci}(k-1) \pm \bar{a}_i}{q_{di}(k) - q_{ci}(k-1)} \quad (2)$$

if the executed velocity in the previous time step $v_{ci}(k-1) = q_{ci}(k-1) - q_{ci}(k-2)$ exceeds the acceleration limit $\pm \bar{a}_i$ and if $q_{di}(k) - q_{ci}(k-1) \neq 0$. If the latter is not true, the axis i is probably not the critical axis that finally determines α and the computation of α_{ai} can be omitted. As a precaution, α_{ai} is truncated to $0 \leq \alpha_{ai} \leq 1$. The sign of $\pm \bar{a}_i$ is such that the absolute value of the numerator is minimum. For $|v_{ci}(k-1)| < \bar{a}_i$, the stopping of axis i is possible within the current sampling step ($\alpha = 0$).

Accordingly, the limits on the jerk of \bar{j}_i give

$$\alpha_{ji} = \frac{v_{ci}(k-1) + a_{ci}(k-1) \pm \bar{j}_i}{q_{di}(k) - q_{ci}(k-1)}, \quad (3)$$

which is only evaluated if $|v_{ci}(k-1) + a_{ci}(k-1)| > \bar{j}_i$ and $q_{di}(k) - q_{ci}(k-1) \neq 0$.

Then

$$\alpha = \max_i(\alpha_{ai}, \alpha_{ji}) \quad (4)$$

will be inserted in (1).

If in addition to the stop command the desired trajectory is (sensor-based) modified in such a way that the acceleration or jerk limits are not satisfied, it is possible that $\alpha > 1$ is computed. Then prediction and backtracking according to [5] are appropriate. However, this goes beyond the scope of this paper.

B. Final Stop

Following (1) to (4), $\mathbf{v}_c = \mathbf{0}$ is reached as fast as possible. However, because of the jerk limits, it holds $\mathbf{a}_c \neq \mathbf{0}$, which generates backward motion in the next sampling steps. If this is not desired, $\bar{\mathbf{a}}$ is modified before the scaling of Section II-A. In generic trajectory generation methods, as e.g. [7], [5], this problem is solved implicitly using a target velocity or subsequent target positions.

First in each sampling step, $\bar{\mathbf{a}}'(k)$ is computed, such that $\mathbf{a}_c \rightarrow \mathbf{0}$ when $\mathbf{v}_c \rightarrow \mathbf{0}$. This is illustrated in Fig. 1. Assuming a constant jerk $j_i(k) = \cdots = j_i(k + \kappa_i) = \pm \bar{j}_i$, the number of steps κ_i until $v_{ci}(k + \kappa_i) = 0$ is rated by (26)¹

$$|v_{ci}(k-1)| = (\kappa_i + 1)|a_{ci}(k + \kappa_i)| + \kappa_i(\kappa_i + 1)/2 \bar{j}_i, \quad (5)$$

¹ k and k' in (26) correspond to $k + \kappa_i$ and $-(\kappa_i + 1)$, respectively. A final stop at $q_{ci}(k + \kappa_i - 1)$ is reached with $v_{ci}(k + \kappa_i) = 0$ and $a_{ci}(k + \kappa_i + 1) = 0$, see Fig. 1.

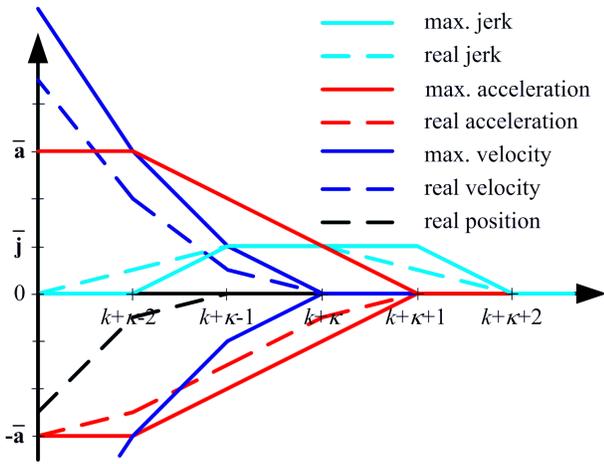


Fig. 1. Position, velocity, acceleration, and jerk of a sample trajectory (dashed), as well as the maximum values that allow a final stop at time step $k + \kappa$, for $\bar{\mathbf{a}} = 3\bar{\mathbf{j}}$ are shown. Velocities between the two solid blue curves can be decelerated until this time step, e.g. the dashed blue curve. The corresponding acceleration has to lie between the two solid red lines. They are computed from the solid cyan curve and result in the solid blue curves, using backward differences as explained in the Appendix. It is crucial that the acceleration limit $\bar{\mathbf{a}}'$ is reduced from $\pm\bar{\mathbf{a}}$ to $\pm\bar{\mathbf{j}}$ at time step $k + \kappa$, since otherwise \mathbf{a} may be nonzero afterwards.

where $v_{ci}(k-1)$ and $a_{ci}(k+\kappa_i)$ have different signs. $|a_{ci}(k+\kappa_i)| \leq \bar{j}_i$ then results in

$$|v_{ci}(k-1)| \leq (\kappa_i + 2)(\kappa_i + 1)/2 \bar{j}_i, \quad (6)$$

$$\kappa_i \geq -1.5 + \sqrt{0.25 + 2|v_{ci}(k-1)|/\bar{j}_i}, \quad (7)$$

and

$$\kappa = \max_i(\kappa_i) \quad (8)$$

with κ_i as the smallest integers that fulfill (7). Then, because of $\bar{a}'_i(k) = \bar{a}'_i(k + \kappa) + \kappa\bar{j}_i$ and $v_{ci}(k + \kappa) = 0$, (26) gives

$$\bar{a}'_i(k) = |v_{ci}(k-1)|/(\kappa + 1) + \kappa/2 \bar{j}_i. \quad (9)$$

$\mathbf{a}_c = \mathbf{v}_c = \mathbf{0}$ is not reachable if $\bar{a}'_i(k) < |a_{ci}(k-1)| - \bar{j}_i$ for any axis i . This might happen if the desired trajectory $\mathbf{q}_d(k + \kappa)$ is changed during motion. Then

$$\bar{a}'_i(k) = |a_{ci}(k-1)| - \bar{j}_i \quad (10)$$

gives the fastest possible final stop for this axis.

If $\kappa \leq 0$, $\bar{a}'_i(k) \leq 0$, or $\bar{a}'_i(k) > \bar{a}_i$ result from (8) or (9), the acceleration limit is not modified, i.e., $\bar{a}'_i(k) = \bar{a}_i$, since the ramp of the acceleration is completed or did not yet begin.

In order to compute the optimal deceleration without exceeding (10) in spite of numerical errors, \bar{a}_i and \bar{j}_i are replaced in the preceding expressions by $\bar{a}_i - \epsilon$ and $\bar{j}_i - \epsilon$, with a small $\epsilon > 0$.

In this way $\bar{\mathbf{a}}'(\cdot)$ results in a feasible trajectory for deceleration. However, it is possible that its use in Section II-A, instead of $\bar{\mathbf{a}}$, does not produce the optimal trajectory since the maximum jerk of a single axis may not be adequate for a path-accurate solution. This applies if the most restricted axis

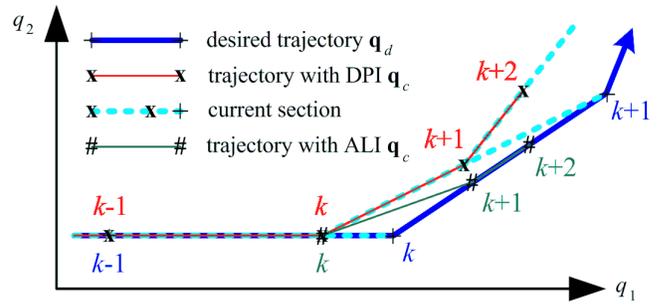


Fig. 2. Modification of the original trajectory when triggering a stopping motion at time step k , using DPI and ALI for interpolation. With DPI, $\mathbf{q}_c(k)$ is computed between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(k)$, $\mathbf{q}_c(k+1)$ between $\mathbf{q}_c(k)$ and $\mathbf{q}_d(k+1)$, before the stop is finished at $\mathbf{q}_c(k+2)$ between $\mathbf{q}_c(k+1)$ and $\mathbf{q}_d(k+2)$. Instead, with ALI all computed points lie on the desired path, i.e. $\mathbf{q}_c(k+1)$ and $\mathbf{q}_c(k+2)$ are between $\mathbf{q}_d(k)$ and $\mathbf{q}_d(k+1)$.

switches over time. Then, a final stop is reached nevertheless, but perhaps somewhat later.

The fastest possible path-accurate stop is attained by $\bar{a}'_i(k) = \bar{a}_i$. It depends on the application whether this is preferred to the main argumentation in this subsection.

C. Limitation of Direct Pose Interpolation

Since the factor α is common to all axes, the previously explained scaling of the velocities (i.e., the direct interpolation of the axis positions) results in points $\mathbf{q}_c(k)$ that lie on the straight line between $\mathbf{q}_c(k-1)$ and $\mathbf{q}_d(k)$. Since those two points may be distant from each other, for curved paths a point on the straight line may lie outside the original path. Fig. 2 illustrates this effect when performing a ‘path-accurate’ stop at time step k . Direct pose interpolation (DPI) computes the next point $\mathbf{q}_c(k+\kappa)$ in each case on the straight line between the previously executed $\mathbf{q}_c(k+\kappa-1)$ and the next desired $\mathbf{q}_d(k+\kappa)$. The resulting path errors call for a different interpolation scheme.

In addition, an almost singular denominator of (2) or (3) can cause $\alpha > 1$ which implies acceleration instead of deceleration. α can be limited to $\alpha = 1$, but this is not a consistent approach. Both problems are solved by the method explained in Section III.

III. PRESERVING THE ORIGINAL PATH

A. Stopping by Arc Length Interpolation (ALI)

In contrast to recurring to the common three phases mentioned in the Introduction, in this paper the vectorial desired path and the limits are processed directly, and the result is then mapped to s in order to get a better interpolation. This results in a much easier method and an algorithm similar to the DPI method of Section II can be applied. The difference is that $\mathbf{q}_c(k)$ is not computed by (1) with (4) but using

$$\mathbf{q}_c(k) = \mathbf{q}_r(s(k)) \quad (11)$$

with

$$s(k) = \max_i(s_{ai}(k), s_{ji}(k)), \quad (12)$$

where the $s_{*i}(k)$ are computed from the $q_{*i}(k)$ that result from the scaling of axis i with $*$ being a or j . For a path-accurate motion, $s(k)$ is the time when the original trajectory $\mathbf{q}_d(\cdot)$ is identical with the computed trajectory \mathbf{q}_c at time step k .

In order to avoid the singularities of (2) and (3) with zero denominator, these equations are concatenated with (1). Now we use

$$q_{ai}(k) = q_{ci}(k-1) + v_{ci}(k-1) \pm \bar{a}_i \quad (13)$$

and

$$q_{ji}(k) = q_{ci}(k-1) + v_{ci}(k-1) + a_{ci}(k-1) \pm \bar{j}_i \quad (14)$$

if the constraints are not satisfied. The sign of $\pm \bar{a}_i$ and $\pm \bar{j}_i$ are opposite to $v_{ci}(k-1)$ or $v_{ci}(k-1) + a_{ci}(k-1)$, respectively.

Then s_{ai} and s_{ji} are computed by

$$s_{*i}(k) = s(k-1) + \frac{(q_{*i}(k) - q_{ci}(k-1)) \cdot (\underline{s}(k-1) + 1 - s(k-1))}{q_{di}(\underline{s}(k-1) + 1) - q_{ci}(k-1)} \quad (15)$$

where $\underline{s}(k-1)$ is the largest integer not greater than $s(k-1)$.

If $s_{*i}(k)$ from (15) exceeds $\underline{s}(k-1) + 1$, $s_{*i}(k)$ is determined by testing whether $\underline{s}_{*i}(k) = \underline{s}(k-1) + 1, \dots, k-2, k-1, k$ gives a solution for

$$s_{*i}(k) = \underline{s}_{*i}(k) + \frac{q_{*i}(k) - q_{di}(\underline{s}_{*i}(k))}{q_{di}(\underline{s}_{*i}(k) + 1) - q_{di}(\underline{s}_{*i}(k))} \quad (16)$$

that satisfies

$$\underline{s}_{*i}(k) \leq s_{*i}(k) < \underline{s}_{*i}(k) + 1. \quad (17)$$

Such a solution is found for $s_{*i}(k) = \underline{s}_{*i}(k) = k$ the latest.

Then the position is interpolated by

$$\mathbf{q}_I(s(k)) = \mathbf{q}_d(\underline{s}(k)) + (s(k) - \underline{s}(k)) \cdot (\mathbf{q}_d(\underline{s}(k) + 1) - \mathbf{q}_d(\underline{s}(k))) \quad (18)$$

or, for (15), by

$$\mathbf{q}_I(s(k)) = \mathbf{q}_c(k-1) + \frac{(s(k) - s(k-1)) \cdot (\mathbf{q}_d(\underline{s}(k-1) + 1) - \mathbf{q}_c(k-1))}{\underline{s}(k-1) + 1 - s(k-1)}, \quad (19)$$

which results in $q_{Ii}(s(k)) = q_{*i}(k)$ for the most limiting constraint $*i$.

The difference with respect to the direct pose interpolation (1) is that the appropriate segment ($\mathbf{q}_d(\underline{s}(k) + 1) - \mathbf{q}_d(\underline{s}(k))$) or ($\mathbf{q}_d(\underline{s}(k-1) + 1) - \mathbf{q}_c(k-1)$) is selected instead of the whole difference between the previous and the desired pose ($\mathbf{q}_d(k) - \mathbf{q}_c(k-1)$).

If a constraint is not satisfied, the procedure is as follows:

- 1) Compute the axis positions $q_{*i}(k)$ that satisfy the limits on the accelerations and the jerk. (see (13), (14))
- 2) Determine the smallest s_{ai} and s_{ji} in the interval $s(k-1) \leq s_{*i}(k) \leq k$ for which $q_{Ii}(s_{*i}(k)) = q_{*i}(k)$. $s(k)$ is the maximum value of all s_{*i} (see (15), (16), (12)).²
- 3) Interpolate $\mathbf{q}_c(k) = \mathbf{q}_I(s(k))$ (see (18), (19)).

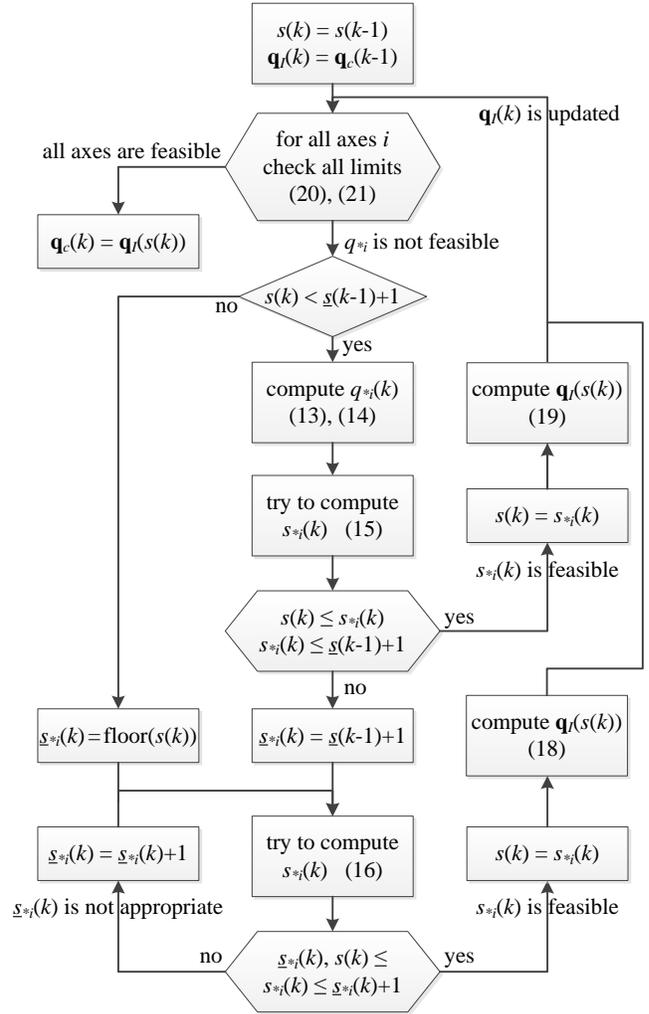


Fig. 3. Flow chart of the iterative procedure.

This procedure has to be repeated for all axes. Unfortunately, it does not always result in a feasible solution since, for non-monotone axis motion, a value $s(k) > s_{*i}(k)$ is not a guarantee for compliance with the respective limit $*i$.

An iteration is therefore required in which the three steps are repeated whenever

$$|v_{ci}(k) - v_{ci}(k-1)| > \bar{a}_i \quad (20)$$

or

$$|v_{ci}(k) - v_{ci}(k-1) - a_{ci}(k-1)| = |a_{ci}(k) - a_{ci}(k-1)| > \bar{j}_i \quad (21)$$

for any axis i with the so far computed $v_{ci}(k) = q_{Ii}(s(k)) - q_{ci}(k-1)$. With $v_{ci}(k) = 0$ this corresponds to the prior conditions of (13) and (14) that the constraints are not satisfied. $q_{*i}(k)$ is still taken from (13) and (14). The smallest $s_{*i}(k)$ according to step 2 is computed in the interval $s(k) \leq s_{*i}(k) \leq k$ with $s(k)$ from the previous iteration step. This

²Note that, in contrast, for generic trajectory generation, the minimum of the biggest s_{ai} and s_{ji} is required, which means that (16) is tested before (15).



Fig. 4. Test set-up for a high-speed path.

is also shown in Fig. 3. Note that $s_{*i}(k)$ in (15) and (16) is also checked by

$$s(k) \leq s_{*i}(k), \quad (22)$$

which accounts for a possibly non-monotone axis trajectory. The initial assumption for $s(k)$ is $s(k-1)$ which, if reachable, means that stopping is possible within the current sampling step. This applies if the prior conditions are not satisfied for any (13) or (14).

B. Discussion

The method in Section III-A solves all shortcomings reported in Section II-C. This means that, in contrast to other methods, no canonical path segment is assumed. However, it requires to check if the solution is feasible and to iterate otherwise. Nevertheless, a solution is always found ($\mathbf{q}_c(k) = \mathbf{q}_d(k)$ at least) if the original trajectory $\mathbf{q}_d(\cdot)$ is feasible.

If, instead, $\mathbf{q}_d(\cdot)$ is computed by sensors and updated during the stopping phase, it cannot be guaranteed that it will satisfy all constraints. Then a feasible trajectory has to be created first, e.g. by [5], and then the stopping procedure can be applied. In this case it is possible that no path-accurate solution will exist (see [5] for a further explanation and for a solution that is as path-accurate as possible).

The method of Section II-B is applicable for ALI as well in order to inhibit backward motion after stopping, since then \bar{a}_i has to be simply replaced by $\bar{a}'_i(k)$ in (13) and (20).

IV. EXPERIMENTS

Fig. 4 shows the test set-up for the first experiment. A KUKA KR16 robot is programmed by a vertical and a horizontal LIN command, with blending such that the sum of the vertical and the horizontal velocity remains constant at 1 m/s. The motion is recorded and reproduced using the robot

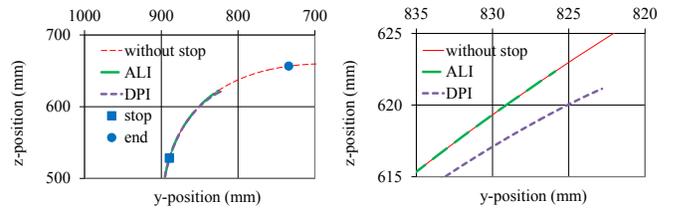


Fig. 5. Stopping during a path that is generated by a vertical and a horizontal LIN command with maximum blending. Left: Survey of the blended area. Right: Detail of stopping.

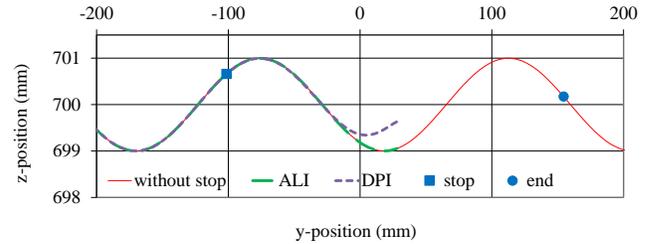


Fig. 6. Stopping during a linear motion that, for performance evaluation, is overlaid by a vertical sine function.

sensor interface (RSI) at a sampling rate of 250 Hz. Then, beginning at time step 210 (*stop* in Fig. 5), the trajectory is recomputed in each step in order to stop as fast as possible. This is completed in time step 281 (*end* in Fig. 5). Fig. 5 shows that, while ALI decelerates path-accurately, DPI has a final horizontal path error of almost 5 mm. In the video attachment it is demonstrated that this causes a collision with the blue object. The limits on the acceleration and the jerk are satisfied in both cases, since otherwise the KUKA KRC4 controller aborts execution.

A second test trajectory is generated by overlaying a constant horizontal velocity of 1 m/s with a vertical sine function (Fig. 6). In time step 855 (*stop* in Fig. 6) the deceleration is triggered. Using ALI the stopping procedure is path-accurate, while DPI loses the track since it always aims for $\mathbf{q}_d(k)$ not for $\mathbf{q}_d(s(k))$. The final $\mathbf{q}_d(k)$ is marked by *end* in Fig. 6.

This experiment is analyzed in more detail in Fig. 7. The deceleration is limited first by \bar{j}_1 , then by \bar{a}_2 , by \bar{a}_1 , and finally by $-\bar{j}_1$. The other limits of the 6-axis robot are not reached.

The effect of $\bar{a}'(k)$ is shown in Fig. 8. Without the adaptation of Section II-B, $\mathbf{v}_c = \mathbf{0}$ is reached one step earlier, but then the jerk limit prevents a permanent stop.

V. CONCLUSION

In the paper it is shown that arc length interpolation (ALI) is required in order to decelerate a robot path-accurately along a given path, while satisfying given limits on the acceleration and the jerk. Similarly, torque limits could be used instead of acceleration limits.

The method has to be applied whenever the original robot motion is not purely linear, e.g. because of blending of linear segments or because of a path definition by splines or by sensor data.

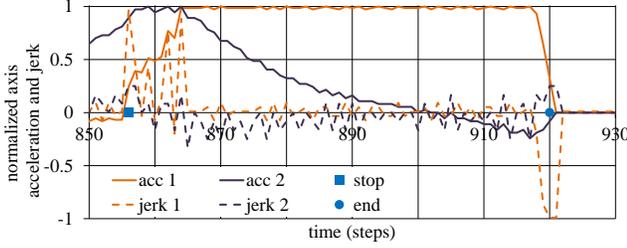


Fig. 7. Normalized acceleration and jerk of the most restricted axes 1 and 2 during the experiment of Fig. 6 using ALI.

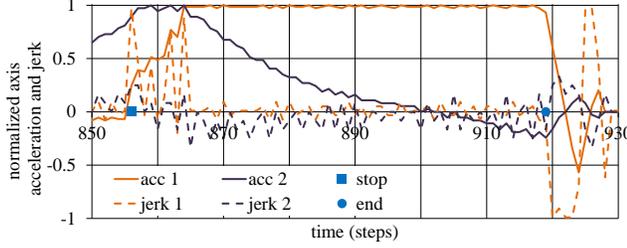


Fig. 8. Experiment of Fig. 7, but with \bar{a} instead of $\bar{a}'(k)$ from (9).

In a slightly modified form, ALI can also be applied for online trajectory generation, i.e., for the adaptation of a programmed trajectory to the sensed real environment. It then results in a more accurate path compared to the iterative method from [5], which uses direct pose interpolation (DPI).

APPENDIX

For convenience the sampling time T_0 is omitted, meaning that \bar{a} is expressed in radians per squared sampling steps instead of rad/s^2 . In addition, as in [5], the factor of 2 is omitted for a one-step prediction, which is due to a representation of the acceleration by $\mathbf{a}(k) = \mathbf{v}(k) - \mathbf{v}(k-1) = \mathbf{q}(k) - 2\mathbf{q}(k-1) + \mathbf{q}(k-2)$. With constant $\mathbf{a}(k+1) = \dots = \mathbf{a}(k+k')$ this corresponds to

$$\mathbf{q}(k+k') = \mathbf{q}(k) + k'\mathbf{v}(k) + k'(k'+1)/2 \mathbf{a}(k+k') \quad (23)$$

and

$$\mathbf{v}(k+k') = \mathbf{v}(k) + k'\mathbf{a}(k+k'). \quad (24)$$

Similar to the acceleration, the jerk is defined by $\mathbf{j}(k) = \mathbf{a}(k) - \mathbf{a}(k-1) = \mathbf{q}(k) - 3\mathbf{q}(k-1) + 3\mathbf{q}(k-2) - \mathbf{q}(k-3)$, thus omitting a further factor of 6 for $k'=1$. With constant $\mathbf{j}(k+1) = \dots = \mathbf{j}(k+k')$ this results in

$$\mathbf{q}(k+k') = \mathbf{q}(k) + k'\mathbf{v}(k) + k'(k'+1)/2 \mathbf{a}(k) + k'(k'+1)(k'+2)/6 \mathbf{j}(k+k') \quad (25)$$

and

$$\mathbf{v}(k+k') = \mathbf{v}(k) + k'\mathbf{a}(k) + k'(k'+1)/2 \mathbf{j}(k+k'). \quad (26)$$

REFERENCES

[1] J. Olonski, N. Settele, and B. Liepert. Method for a time-optimal, true-to-path braking of axle drives of numerically controlled machines. U.S. patent 5473542, 1995.

[2] M. Weiß, M. Hüttenhofer, A. Hagenauer, and G. Wiedemann. Industrial robot and method for controlling the movement of an industrial robot. European patent EP2209596B1, 2011.

[3] F. Jing, M. Tan, E. Li, Z. Liang, Z. Hou, D. Yang, K. Zhang, and Y. Qiang. System and method for robot trajectory generation with continuous accelerations. U.S. patent US8600554B2, 2013.

[4] F. Lange, W. Bertleff, and M. Suppa. Force and trajectory control of industrial robots in stiff contact. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2912–2919, Karlsruhe, Germany, May 2013.

[5] F. Lange and M. Suppa. Predictive path-accurate scaling of a sensor-based defined trajectory. In *Proc. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 754–759, Hong Kong, China, May/June 2014.

[6] T. Kröger and F. M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *IEEE Trans. on Robotics*, 26(1):94–111, Feb 2010.

[7] Reflexxes. <http://www.reflexxes.com/>, last visited 2014.

[8] R. Haschke, E. Weitnauer, and H. Ritter. On-line planning of time-optimal, jerk-limited trajectories. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3248–3253, Nice, France, Sep 2008.

[9] X. Broquère, D. Sidobre, and I. Herrera-Aguilar. Soft motion trajectory planner for service manipulator robot. In *Proc. 2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2808–2813, Nice, France, Sep 2008.

[10] C. Guarino Lo Bianco and F. Ghilardelli. A discrete-time filter for the generation of signals with asymmetric and variable bounds on velocity, acceleration, and jerks. *IEEE Trans. on Industrial Electronics*, 61(8):4115–4125, Aug 2014.

[11] Y. Bestaoui. On-line motion generation with velocity and acceleration constraints. *Robotics and Autonomous Systems*, 5:279–288, 3 1989.

[12] O. Dahl and L. Nielsen. Torque limited path following by on-line trajectory time scaling. *IEEE Trans. on Robotics and Automation*, 6(5):554–561, Oct 1990.

[13] Z. Shiller and H.-H. Lu. Computation of path constrained time optimal motions with dynamic singularities. *ASME Journal of Dynamic Systems, Measurements, and Control*, 114:34–40, 1 1992.

[14] J. Mattmüller and D. Gisler. Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path. *Int. J. Adv. Manuf. Technol.*, 45:1007–1016, 2009.

[15] M. H. Ghasemi, N. Kashiri, and M. Dardel. Near time-optimal control of redundant manipulators along a specified path with jerk constraint. *Advanced Robotics*, 25:2319–2339, 2011.

[16] C. Guarino Lo Bianco and F. Ghilardelli. Techniques to preserve the stability of a trajectory scaling algorithm. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 862–868, Karlsruhe, Germany, May 2013.

[17] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers. Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1908–1913, Karlsruhe, Germany, May 2013.

[18] A. Amthor, S. Zschäk, C. Ament, A. Lorenz, and J. Werner. Method and device for real-time-capable forth-order path planning for generating continuous target trajectories free of jerk jumps. International patent WO002010136586A1, 2010.

[19] J. Schultz and T. D. Murphey. Real-time trajectory generation for a planar crane using discrete mechanics. In *Workshop on "Real-time Motion Generation & Control - Constraint-based Robot Programming" at the 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, Sep 2014. "<http://cs.stanford.edu/people/tkr/iros2014/proceedings.html>".

[20] T. Kunz and M. Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3713–3819, Chicago, IL, USA, Sep 2014.

[21] A. Labusch, T. Bellmann, K. Sharma, and J. Bals. Worst case braking trajectories for robotic simulators. In *Proc. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3297–3302, Hong Kong, China, May/June 2014.

[22] C.-S. Tsai, J.-S. Hu, and M. Tomizuka. Ensuring safety in human-robot coexistence environment. In *Proc. 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4191–4196, Chicago, IL, USA, Sep 2014.