Proc. Int. Conf. on Robotics and Automation (ICRA), Hong Kong, China, May/June 2014

Predictive Path-Accurate Scaling of a Sensor-Based Defined Trajectory

Friedrich Lange and Michael Suppa

***Abstract*— The paper considers an a priori given robot trajectory which has to be recomputed when online sensed information on the environment is available. Then the original trajectory is adapted in order to continue the so far commanded motion by the sensed geometric shape. The adapted trajectory has to comply with restrictions on velocity, acceleration and jerk. Furthermore it is desired to converge to the original trajectory. At least if the robot is in contact with the environment it is further essential that the geometrical path is not left when modifying the trajectory. This means that preferably only the temporal profile is changed by scaling or rescaling the velocity. In order to inhibit overshooting, future restrictions are predicted and backtracked in the case of a violation. All this computation is done within a single sampling step, i.e. within 4 ms for a standard KUKA industrial robot. This precludes accurate optimization algorithms. When applied without an a priori given trajectory the method results in an near time-optimal solution.**

I. Introduction

Trajectory generation of fixed robot paths can be done offline, considering sophisticated optimization criteria. In contrast, trajectories from paths which are computed or adapted online from sensor data have to be computed in a time efficient manner. Thus optimization methods as e.g. [1], [2], [3] are too slow. Instead, the computation time should be below 1 ms, even if this results in a sub-optimal trajectory.

This is important for fast motion with high accuracy requirements, e.g. when approaching an object, where the sensed desired path abruptly changes when a contact force is measured. Then, in order to avoid excessive forces as well as a loss of contact after the first overshooting, the robot should decelerate as fast as possible and then perform the desired contact motion, e.g. polishing with the desired pressure. Another critical task for trajectory generation is stopping during constrained motion, without exerting undesired forces, i.e. without leaving the previously defined path.

Sudden changes of the desired trajectory can also be present before sensing a contact, e.g. in pick-and-place operations or during assembly, when the exact gripping pose is not known until it is sensed (e.g. by an eye-in-hand camera of limited field of view). Then, a significant change of orientation may be needed in a very short time. And this path correction should be executed without colliding with the object to be gripped.

A general formulation of the problem is, first, the definition of desired trajectories $\mathbf{q}_d(k+\kappa)$ from each sampling step k, representing the sampled positions of the next 100 ms or so, incorporating the current information on the environment.

The authors are with the German Aerospace Center (DLR), Robotics and Mechatronics Center (RMC), 82234 Wessling, Germany friedrich.lange@dlr.de

For this paper, the desired trajectories are assumed to be given. Second, there are restrictions, usually given by the robot manufacturer, limiting the velocity, the acceleration and the jerk of the individual axes. In addition, there may be restrictions on Cartesian level. If the desired trajectory does not comply with the restrictions, the task of trajectory generation is to modify the trajectory such that:

- The restrictions are met.
- The desired trajectory is reached as soon as possible.
- The modified path, i.e. the geometrical shape of the trajectory, complies with the desired path.

The latter requirement is called *path-accuracy*[1]. The resulting executable trajectory is called *limited desired trajectory*, with the first value denoted as the *commanded position* $\mathbf{q}_c(k)$ which provides the desired values for the servo loops.

Typical interpolation methods as e.g. [4], [5] generate a trajectory from the current position to a goal position, assuming that the robot stands still at both places. Other algorithms generate a trajectory through several points in the axis space, as e.g. [3], [6], [7], [8]. All this is not applicable, when the robot is in motion and the desired path suddenly changes because of unexpected sensor signals.

The Reflexxes Motion Library [9] provides a useful trajectory generator to limit the velocity, the acceleration, and the jerk, considering the full current state. However, the goal is to reach the target pose with given velocity and acceleration in minimum time. So there are two drawbacks for the task on hand: First, when using the algorithm, $\mathbf{q}_d(k)$, $\mathbf{q}_d(k+1)$, $\mathbf{q}_d(k+2)\cdots$ and the appropriate derivatives have to be tested as goal, until the algorithm confirms that a solution is possible within the given number of sampling steps. In this way, the experiments are done in Section IV. The second drawback is even worse. Since the only criterium is time-optimality, there is no desired trajectory processed. Therefore the limited desired trajectory may be quite far from the desired trajectory, not considering a possible collision.

Scanning other methods it turns out that time optimality is predominantly used as cost function. Also with methods that explicitly consider the tracking of a desired path (as e.g. [10], [11], [12], [13], [14], [15]), minimum execution time is aimed instead of synchronization with a given trajectory. The latter however, is desired with given robot programs. This is the motivation for the method of this paper.

[1] In this paper the term *path-accurate* is understood that the current *limited desired* position is on the straight line between the previous *commanded* position and the current *desired* position. So, if the desired path is not continuous or a commanded position is not on the desired path, *path-accuracy* does not guarantee that subsequent sampling steps will fit the path.

If a velocity profile for a given path is computed, the path is represented with the arc length s as parameter. Then the task is to compute the velocity $\dot{s}(s)$ in the phase space, that complies with the restrictions. Early papers as [10], [11], [12] did not consider constraints on the jerk. In contrast, [13] uses forward and backward computation and smoothing at the meeting point to account for these limits as well. [14] only computes from the beginning of the trajectory, but switches before entering a *trapped area*, since otherwise the path will be lost later.

While in most papers (as [4], [9], [12], [13], [14], [16]) the motion is divided into a limited number of phases with given characteristics, as a constant jerk or a constant acceleration, the consideration of the path-accuracy affords the generation of a continuous trajectory. Therefore the number of phases can be up to the number of sampling steps until the desired trajectory is reached. As mentioned before, the optimization of so many degrees of freedom (dof) is not possible within the given computing time.

It should be noted that in contrast to the offline generation of a whole trajectory it is not always possible to generate a path-accurate trajectory from a given state, that meets all restrictions. For example, in the above mentioned task of approaching an object it may be impossible to inhibit overshooting. In contrast to acceleration phases, the path-accuracy is always at risk if the desired deceleration[2] exceeds the limits, even more if some axes are decelerated while others are accelerated. This is discussed in more detail in [9], [14].

This problem might be relieved if not only the current time step is considered but also future restrictions, similar to a whole trajectory. [15] locally restricts the given velocity limits in order to comply with the limits on the acceleration and the jerk. Instead, in this paper it is proposed to predict future restrictions and then to modify the trajectory at previous time steps. This is outlined in Section III.

The paper is organized as follows: The next section tries to modify a desired position in order to comply with the robot restrictions, given the so far commanded trajectory. Since this may result in overshooting, Section III considers several future sampling steps. Both approaches are shown in experiments in Section IV.

II. Path-Accurate Scaling of the Current Time-Step

In this section the desired axis positions $\mathbf{q}_d(k)$ are modified to commands $\mathbf{q}_c(k)$, in order to comply with the restrictions. This is explained in three steps, considering the limitations on velocity, acceleration, and jerk.

For convenience, in the following equations we omit the sampling time $T_0 = 1$, which means that the time is expressed in steps instead of seconds. Furthermore, we denote $\mathbf{q}_c(k)$ as the output of each step, which then becomes the input of the next step, i.e. $\mathbf{q}_d(k)$. The same applies if there is an iteration within a step. Past values $\mathbf{q}_c(k-\kappa) = \mathbf{q}_d(k-\kappa)$ are identical anyway.

[2]Acceleration against the current direction of motion is similar to deceleration.

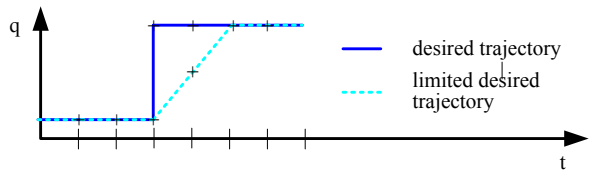

Fig. 1. Modification of a 1 dof path by scaling the velocity at the discontinuity of the desired path in order to comply with a velocity limit.

A. Scaling of the Velocity

For each axis the restriction

$$|\mathbf{v}_c(k)| = |\mathbf{q}_c(k) - \mathbf{q}_c(k-1)| \leq \bar{\mathbf{v}} \tag{1}$$

applies. If the desired position violates the restriction in at least one axis i, i.e. $|v_{di}(k)| = |q_{di}(k) - q_{di}(k-1)| > \bar{v}_i$, the velocity has to be scaled. Scaling of the velocity between time step $(k-1)$ and k always results in a path-accurate trajectory since not the path is modified but the time profile of its execution.

The modification is executed in two steps. First, the scaling factor α is computed as

$$\alpha = \max_i(|v_{di}(k)|/\bar{v}_i). \tag{2}$$

This computation is always possible and results in $\alpha > 1$. Then the velocity of all axes is scaled

$$\mathbf{q}_c(k) = \mathbf{q}_d(k-1) + \mathbf{v}_d(k)/\alpha. \tag{3}$$

For one degree of freedom (dof) this is displayed in Fig. 1.

In addition, if α exceeds a value $\bar{\alpha}_v$, e.g. 2, the motion is stopped by $\mathbf{q}_c(k) = \mathbf{q}_d(k-1)$, since then probably a sensor failure or another fault has occurred.

B. Path-Accurate Scaling of the Acceleration

For each axis the restriction

$$|\mathbf{a}_c(k)| = |\mathbf{q}_c(k) - 2\mathbf{q}_c(k-1) + \mathbf{q}_c(k-2)| \leq \bar{\mathbf{a}} \tag{4}$$

applies. If the desired position violates the restriction in at least one axis i, i.e. $|a_{di}(k)| = |q_{di}(k) - 2q_{di}(k-1) + q_{di}(k-2)| > \bar{a}_i$, the velocity is scaled (see Fig. 2). (3) gives

$$|\mathbf{q}_d(k-1) + \mathbf{v}_d(k)/\alpha - 2\mathbf{q}_d(k-1) + \mathbf{q}_d(k-2)| \leq \bar{\mathbf{a}} \tag{5}$$

and hence the scaling factor

$$\frac{1}{\alpha} = \min_i\left(\frac{\pm\bar{a}_i + v_{di}(k-1)}{v_{di}(k)}\right). \tag{6}$$

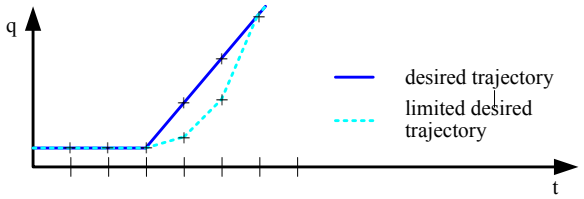

Fig. 2. Modification of a 1 dof path by scaling the velocity at the vertex of the desired path in order to comply with an acceleration limit.

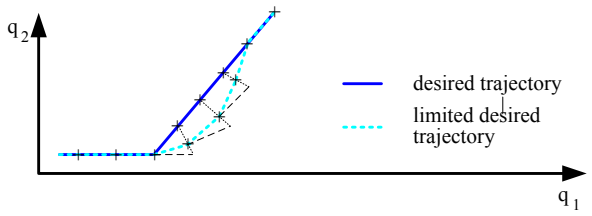

Fig. 3. Modification of a 2 dof path by scaling the acceleration at the vertex of the desired path in order to comply with acceleration limits (not path-accurate).

The sign of $\bar{a}_i$ is positive if $a_{di}(k) > \bar{a}_i$, it is negative if $a_{di}(k) < -\bar{a}_i$. Otherwise axis i does not contribute to the computation of α since the component i of (4) is met.

If the result is $\alpha > 1$, (3) meets (4) in all axes. Then the modification is path-accurate. But the condition on α is not always met.

Unfortunately, a scaling in axis i_1 may cause a limitation in another axis i_2, since the value of $\mathbf{q}_c(k)$ which is modified according to (3) has to be used in (4). Simply speaking, if the velocity has to be reduced because of a big desired acceleration in axis i_1, the reduction of the velocity of axis i_2 may exceed the maximum deceleration. Therefore all scaling is repeated iteratively, until a solution is found that meets all restrictions. The number of steps of this iteration is not greater than the number of axes. If such a solution cannot be found, the modification of the desired trajectory will not be path-accurate.

As well for the acceleration a fault detection is implemented which stops the motion if $\alpha > \bar{\alpha}_a$, where $\bar{\alpha}_a$ for the scaling of the acceleration can be bigger than $\bar{\alpha}_v$, e.g. $\bar{\alpha}_a = 30$, depending on the sampling time.

C. Other Scaling of the Acceleration

If no path-accurate scaling is possible, the acceleration is scaled directly, since the robot controller inhibits the motion if the restrictions are not met.

The scaling factor is computed by

$$\alpha = \max_i(|a_{di}(k)|/\bar{a}_i). \tag{7}$$

Then the scaling is done by

$$\mathbf{q}_c(k) = \mathbf{q}_d(k-1) + \mathbf{v}_d(k-1) + \mathbf{a}_d(k)/\alpha. \tag{8}$$

This modification of the desired trajectory is always possible, since $\alpha > 1$ if the acceleration limits are violated without scaling. Fig. 3 shows the geometrical interpretation. The scaling is done between the position which would be reached without acceleration and the desired position which is not reachable with the given restrictions on the acceleration.

D. Path-Accurate Scaling of the Jerk

The restriction of the jerk is

$$|\mathbf{j}_c(k)| = |\mathbf{q}_c(k) - 3\mathbf{q}_c(k-1) + 3\mathbf{q}_c(k-2) - \mathbf{q}_c(k-3)| \le \bar{\mathbf{j}}. \tag{9}$$

Scaling is required if the limit is exceeded by at least a single axis. As with the acceleration it is first tried to modify the desired trajectory in a path-accurate way, i.e. by scaling the velocity. (3) gives

$$|\mathbf{q}_d(k-1) + \mathbf{v}_d(k)/\alpha \\ -3\mathbf{q}_d(k-1) + 3\mathbf{q}_d(k-2) - \mathbf{q}_d(k-3)| \le \bar{\mathbf{j}}. \tag{10}$$

Thus the scaling factor

$$\frac{1}{\alpha} = \min_i\left(\frac{\pm\bar{j}_i + v_{di}(k-1) + a_{di}(k-1)}{v_{di}(k)}\right) \tag{11}$$

can be computed and inserted in (3). As with the acceleration this has to be repeated until a solution is found that meets all restrictions. Still it is possible that no solution exists.

E. Other Scaling of the Jerk

If the scaling of the velocity is not possible, as well here it is required to modify the desired trajectory in order to comply with the restrictions. This can be done by scaling the accelerations or directly the jerk. Since the scaling of the accelerations between the current and the previous time step is not always successful in order to meet the jerk limitations, it is not explained here. Instead, the jerk is scaled by

$$\alpha = \max_i(|j_{di}(k)|/\bar{j}_i). \tag{12}$$

This results in $\alpha > 1$ if the jerk limits are violated without scaling. Thus this scaling always gives a valid modification of the desired trajectory

$$\mathbf{q}_c(k) = \mathbf{q}_d(k-1) + \mathbf{v}_d(k-1) + \mathbf{a}_d(k-1) + \mathbf{j}_d(k)/\alpha. \tag{13}$$

F. Limitations of Scaling

In this way a valid scaling is possible for the restrictions in each level, but this does not always result in a path-accurate modification of the desired trajectory.

Even worse, there are desired trajectories for which no valid modification is possible at all. This may happen if the robot has accelerated until the maximum velocity is reached. But then the robot continues to accelerate because the restrictions on the jerk do not allow zero acceleration in the next step. This is called a *forbidden point* in [13].

Besides the kinematic restrictions, as in this paper, it is possible to provide for dynamic restrictions as a limited motor torque. In this case $\bar{\mathbf{a}}$ is a time variant function of the allowed motor torques, considering couplings from other axes as e.g. velocity dependent terms. (see e.g. [11], [14] for other methods accounting for torques)

III. Predictive Scaling by Backtracking

In the same way as $\mathbf{q}_d(k)$ is modified, future time steps $\mathbf{q}_d(k+\kappa)$ can be modified. If the desired trajectory is not only given for the current time step k, the knowledge of its future limitations gives the chance to avoid deviations from the desired path, i.e. modifications of the desired trajectory which cannot be done path-accurately. For example, a strong deceleration at time step $k+\kappa$ will produce overshooting if this is not inhibited by starting the deceleration before. Then the speed reduction can be executed path-accurately.

Another application for predictive scaling is the execution of a desired step, e.g. during sensor controlled motion when

the desired position differs from the actual one because of a changed model of the environment. If the desired trajectory (the step) is scaled according to Section II, the speed of the limited desired trajectory will increase until the step is executed. But then it is too late for deceleration. This is shown in Section IV (see Fig. 5).

But the algorithm which is explained in the sequel is not restricted to these examples. It is applicable as well, e.g. to a desired trajectory which offers a high curvature in time step $k+20$.

The procedure is as follows.

1) Beginning from the current time step k the desired positions are checked for limitations and modified according to Sections II-A, II-B, and II-D.
2) If a path-accurate modification of the desired trajectory is not possible, the previous time steps are scaled according to Section III-A or III-B. This is called backtracking.
3) Then the iteration continues with step $k-1$ or $k-2$, respectively, checking the time steps which are just modified.

The procedure ends when

- no more predictions on $\mathbf{q}_d(k+\kappa)$ are available (then no unsolved contradictions to a path-accurate execution have been found),
- a maximum number of iteration steps is reached, or
- the procedure tries to modify past values, as $\mathbf{q}_c(k-1)$.

A. Restriction on the Acceleration

By a path-accurate modification of $\mathbf{q}_d(k+\kappa)$ and $\mathbf{q}_d(k+\kappa-1)$ it is always possible to meet the limits of the acceleration $\mathbf{a}_c(k+\kappa)=\mathbf{q}_c(k+\kappa)-2\mathbf{q}_c(k+\kappa-1)+\mathbf{q}_c(k+\kappa-2)$.

For brevity, in the sequel we denote $k+\kappa$ as k'. Then the modification is done by two scalings of the velocity

$$\begin{aligned}\mathbf{q}_c(k') &= \mathbf{q}_d(k'-1)+\mathbf{v}_d(k')/\alpha_{k'} && (14)\\ \mathbf{q}_c(k'-1) &= \mathbf{q}_d(k'-2)+\mathbf{v}_d(k'-1)/\alpha_{k'-1}. && (15)\end{aligned}$$

In order to limit the computing time, we concatenate these scalings by selecting $\alpha_{k'}=\alpha$ and $\frac{1}{\alpha_{k'-1}}=(1+\frac{1}{\alpha})/2$. Then

$$\begin{aligned}\mathbf{q}_c(k') &= \mathbf{q}_d(k'-1)+\mathbf{v}_d(k')/\alpha && (16)\\ \mathbf{q}_c(k'-1) &= \mathbf{q}_d(k'-2)+\mathbf{v}_d(k'-1)\cdot(1+1/\alpha)/2 && (17)\end{aligned}$$

which results in α from (7), with k' instead of k.

For $\alpha\to\infty$ this converges to

$$\begin{aligned}\mathbf{q}_c(k') &= \mathbf{q}_d(k'-1) && (18)\\ \mathbf{q}_c(k'-1) &= (\mathbf{q}_d(k'-2)+\mathbf{q}_d(k'-1))/2 && (19)\end{aligned}$$

and thus results in an acceleration $\mathbf{q}_c(k')-2\mathbf{q}_c(k'-1)+\mathbf{q}_c(k'-2)$ of zero.

For the critical axis from which α has been computed, we get

$$q_{ci}(k')=q_{di}(k'-1)+v_{di}(k')\cdot\bar{a}_i/|a_{di}(k')| \quad (20)$$

$$q_{ci}(k'-1)=q_{di}(k'-2)+\frac{v_{di}(k'-1)}{2}\left(1+\frac{\bar{a}_i}{|a_{di}(k')|}\right) \quad (21)$$

and thus

$$q_{ci}(k')-2q_{ci}(k'-1)+q_{ci}(k'-2)=\pm\bar{a}_i. \quad (22)$$

B. Restriction on the Jerk

Similarly the path-accurate modification of $\mathbf{q}_d(k+\kappa)$, $\mathbf{q}_d(k+\kappa-1)$, and $\mathbf{q}_d(k+\kappa-2)$ is always possible to meet the jerk limits $\mathbf{j}_c(k+\kappa)=\mathbf{q}_c(k+\kappa)-3\mathbf{q}_c(k+\kappa-1)+3\mathbf{q}_c(k+\kappa-2)-\mathbf{q}_c(k+\kappa-3)$. Here, as well, $k+\kappa$ is denoted as k'.

Analogously to (16) and (17) the modification is done by

$$\begin{aligned}\mathbf{q}_c(k') &= \mathbf{q}_d(k'-1)+\mathbf{v}_d(k')/\alpha && (23)\\ \mathbf{q}_c(k'-1) &= \mathbf{q}_d(k'-2)+\mathbf{v}_d(k'-1)\cdot(1+2/\alpha)/3 && (24)\\ \mathbf{q}_c(k'-2) &= \mathbf{q}_d(k'-3)+\mathbf{v}_d(k'-2)\cdot(2+1/\alpha)/3 && (25)\end{aligned}$$

which results in α from (12), with k' instead of k.

For $\alpha\to\infty$ this converges to

$$\begin{aligned}\mathbf{q}_c(k') &= \mathbf{q}_d(k'-1) && (26)\\ \mathbf{q}_c(k'-1) &= (2\mathbf{q}_d(k'-2)+\mathbf{q}_d(k'-1))/3 && (27)\\ \mathbf{q}_c(k'-2) &= (\mathbf{q}_d(k'-3)+2\mathbf{q}_d(k'-2))/3 && (28)\end{aligned}$$

which corresponds to a jerk $\mathbf{q}_c(k')-3\mathbf{q}_c(k'-1)+3\mathbf{q}_c(k'-2)-\mathbf{q}_c(k'-3)$ of zero.

For the critical axis from which α has been computed, we get

$$q_{ci}(k')=q_{di}(k'-1)+v_{di}(k')\cdot\bar{j}_i/|j_{di}(k')| \quad (29)$$

$$q_{ci}(k'-1)=q_{di}(k'-2)+\frac{v_{di}(k'-1)}{3}\left(1+2\frac{\bar{j}_i}{|j_{di}(k')|}\right) \quad (30)$$

$$q_{ci}(k'-2)=q_{ci}(k'-3)+\frac{(v_{di}(k'-2)}{3}\left(2+\frac{\bar{j}_i}{|j_{di}(k')|}\right) \quad (31)$$

and thus

$$q_{ci}(k')-3q_{ci}(k'-1)+3q_{ci}(k'-2)-q_{ci}(k'-3)=\pm\bar{j}_i. \quad (32)$$

C. Limitations of Backtracking

In this way backtracking will find a path-accurate solution in most cases. If the iteration is aborted, e.g. because already time step $k+\kappa$ with $\kappa=0$ cannot be scaled path-accurately, the restrictions can be met by a scaling according to Section II-C or II-E. Thus, besides the exception[3] mentioned in Section II-F, the modified trajectory will comply with the restrictions.

By scaling and by backtracking the trajectory is always slowed down, because the positions are shifted towards positions at previous time steps. Thus the restrictions are probably met. But it will not be path-accurate if the backtracking exceeds the current time step k. Therefore with this conservative procedure it is not ensured that a possible path-accurate solution is found. On the other side it has been stated in the Introduction that a full optimization is not feasible.

The required number of future time steps that can be used for backtracking depends on the velocity and on how close the desired trajectory is from the limits. In the experiments we use $0\le\kappa<\bar{\kappa}=20$ which is usually sufficient.

[3]If sufficient backtracking is possible, the exception will be solved as well.

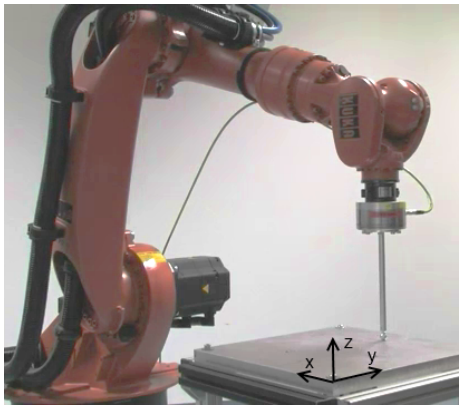

Fig. 4. KUKA robot with force sensor and pin which is in contact.

IV. Experiments

In this section we first consider the case of a vertical motion of the tcp (see setup in Fig. 4), until a reaction force is measured. This happens in time step 2801 in Figs. 5 to 7, much earlier than expected. Then the future desired trajectory is recomputed as that position, at which the desired force will be executed. This approach is explained in detail in [17]. It results in a step since the actual position at which the force is sensed is delayed with respect to the so far desired trajectory. The desired trajectory is recalculated as well in further time steps. Then the desired position slightly changes because the assumptions on the robot compliance are disturbed. In theory there is no feedback from the robot motion to the desired motion. Therefore the latter is assumed to be given in this paper irrespective of the type of sensor.

Figures 5 to 7 show the experimental results in Cartesian space, using a KUKA KR16 robot which is controlled at 250 Hz via RSI Ethernet from an external computer. In Fig. 6 the path error is displayed, i.e. the motion perpendicular to the desired path. Fig. 7 displays the computed accelerations and jerks. Their used limits in the joint space are displayed in Table I, meaning, e.g., velocity limits $\bar{v}_i/T_0$ in rad/4ms.

When the desired position is changed (because of a sensed contact), in the first step all methods are identical. An overshooting cannot be avoided. Thus this step is not path-accurate.

Then the method of Section II executes maximum acceleration until the desired value is reached. Then, as far as the jerk limit allows, the deceleration is maximal. Like this, a poorly damped oscillation around the desired position is executed. The robot motion may even become unstable if

TABLE I

Limits of the individual axes (before filtering).

i	$\bar{v}_i$	$\bar{a}_i$	$\bar{j}_i$
1	0.014 rad	0.074 mrad	0.061 mrad
2	0.014 rad	0.037 mrad	0.030 mrad
3	0.014 rad	0.085 mrad	0.069 mrad
4	0.029 rad	0.250 mrad	0.204 mrad
5	0.030 rad	0.252 mrad	0.206 mrad
6	0.055 rad	0.450 mrad	0.368 mrad

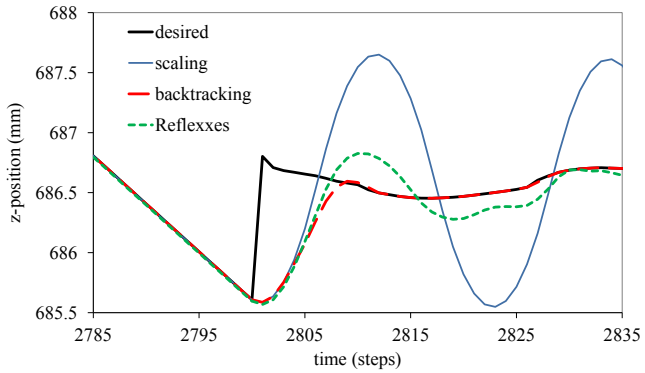

Fig. 5. Desired and limited desired trajectories in z using different methods.

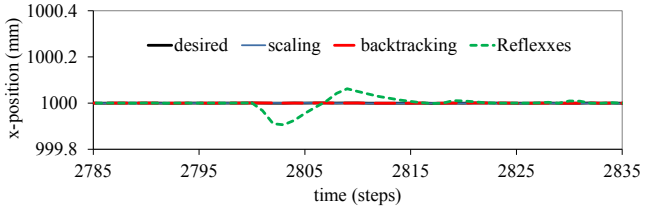

Fig. 6. Deviations in x from the desired path using different methods.

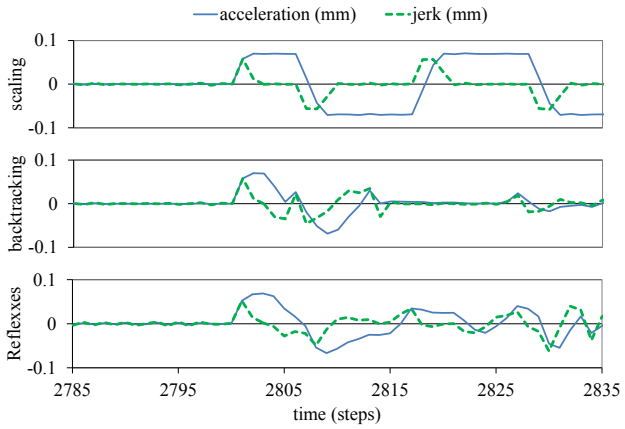

Fig. 7. Limited acceleration and jerk.

the limits of the jerk are very restrictive.

In contrast, with backtracking there is no overshooting since the acceleration is reduced in time. Apart from sampling step 2801, the trajectory is path-accurate. The plots of the acceleration and the jerk show however, that the trajectory is sub-optimal since they do not feature the typical shape as with the scaling. Tracing the experiment discloses that in step 2802 backtracking is executed for time steps 2807 (jerk), 2808 (acc.), 2807 (jerk), 2805 (jerk), 2809(acc.), 2808 (acc.), 2807 (jerk), 2805 (jerk), 2808 (jerk), 2810 (acc.). Similarly, backtracking is performed in each of the next sampling steps.

The result of the Reflexxes Motion Library [18] is similar to this, at least according to Fig. 5. It is obtained by computing trajectories from $\mathbf{q}_c(k-1)$, $\mathbf{v}_c(k-1)$, and $\mathbf{a}_c(k-1)$ to $\mathbf{q}_d(k+\kappa)$ and $\mathbf{v}_d(k+\kappa)$. This is done by

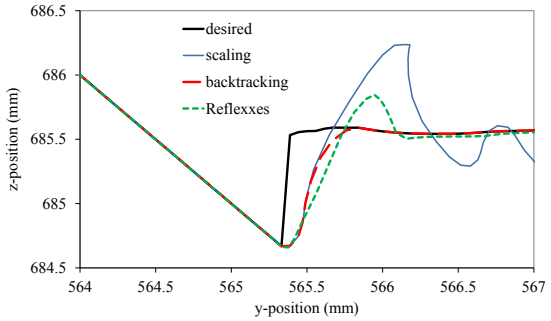

Fig. 8. Desired and limited desired trajectories in the y-z-plane using a force sensor with different methods.

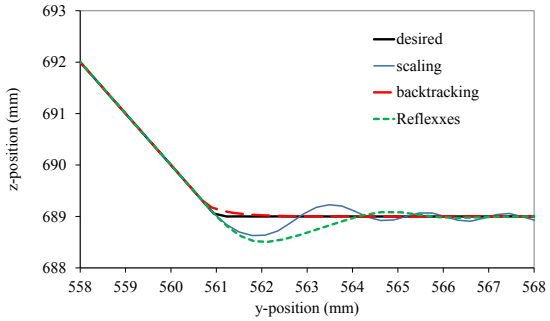

Fig. 9. Desired and limited desired trajectories in the y-z-plane using a predictive sensor with different methods (5 times as fast as Fig. 8).

tentatively calling the function *RMLPosition* for $\kappa = 0, \cdots, \bar{\kappa}$ until the target is reached within at most κ time steps.[4] The intermediate time steps $k, \cdots, k + \kappa - 1$ are not considered in each case. Therefore the motion is not path-accurate, i.e. there are small deviations in x-direction. In this example, the deviations are not essential since there is no contact in this direction. However, deviations in the sub-millimeter range may be significant if the robot is in stiff contact of e.g. 1000 N/mm and thus generates undesired forces.

In a second experiment the task is overlaid by a constant horizontal velocity in y direction. The resulting path is displayed in Fig. 8.

As a third experiment the previous one is repeated 5 times faster, where instead of the force-torque sensor a distance sensor is simulated. This sensor can predict the contact point and thus the changed desired trajectory. So the predictive approach decelerates early enough, such that there is no overshooting (Fig. 9). This cannot be exploited by the other two methods.

The video attachment gives an impression of the three experiments. However, the different methods cannot be distinguished there.

V. Conclusion

The paper presents a trajectory generator that in each sampling step continues the so far commanded robot motion towards the currently sensed path, complying with restrictions on the velocity, the acceleration, and the jerk, and synchronizing with the offline computed desired trajectory. The latter will be reached exactly or displaced after a short time, depending on the sensed environment. With expected sensor values, motion continues as in a typical industrial robot program. Thus the method is applicable for industrial robots with standard robot programs which in this way are locally modified according to the sensor data.

References

[1] F. Debrouwere, W. Van Loock, G. Pipeleers, Q. Tran Dinh, M. Diehl, J. De Schutter, and J. Swevers. Time-optmal 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.

[2] M. Lawitzky, M. Kimmel, P. Ritzer, and S. Hirche. Trajectory generation under the least action principle for physical human-robot cooperation. In *Proc. 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4270–4275, Karlsruhe, Germany, May 2013.

[3] K. Erkorkmaz and Y. Altintas. High speed CNC system design. Part I: jerk limited trajectory generation and quintic spline interpolation. *Int. J. of Machine Tools and Manufacture*, 41(9):1323–1345, July 2001.

[4] L. van Aken and H. van Brussel. On-line robot trajectory control in joint coordinates by means of imposed acceleration profiles. *Robotica*, 6(3):185–195, July 1988.

[5] B. Cao, G. I. Dodds, and G. W. Irwin. Time-optimal and smooth constrained path planning for robot manipulators. In *Proc. 1994 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1853–1858, San Diego, California, USA, May 1994.

[6] S. A. Bazaz and B. Tondu. Minimum-time on-line joint trajectory generator based on low order spline method for industrial manipulators. *Robotics and Autonomous Systems*, 29(4):257–268, 1999.

[7] S. Macfarlane and E. A. Croft. Jerk-bounded manipulators trajectory planning: Design for real-time applications. *IEEE Trans. on Robotics*, 19(1):42–52, Feb 2003.

[8] L. Biagiotti and C. Melchiorri. B-spline based filters for multi-point trajectories planning. In *Proc. 2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3065–3070, Anchorage, Alaska, USA, May 2010.

[9] 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.

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

[11] O. Dahl and L. Nielsen. Torque limited path following by on-line trajectory time scaling. In *Proc. 1989 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1122–1128, Scottsdale, Arizona, USA, May 1989.

[12] 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.

[13] 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.

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

[15] 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.

[16] B. Cao, G. I. Dodds, and G. W. Irwin. A practical approach to near time-optimal inspection-task-sequence planning for two cooperative industrial robot arms. *The Int. Journal of Robotics Research*, 17:858–867, 1998.

[17] 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.

[18] Reflexxes. `http://www.reflexxes.com/`, last visited 2013.

[4] Note that this is different to the intended usage. In addition, the used backward computation of the derivatives is not adequate for the approach of the library. Therefore overshooting and offset when using the Reflexxes Motion Library are no deficiency of that library.