Real-time trajectory scaling for robot manipulators

Marco Faroni, Roberto Pagani, Giovanni Legnani

Real-time trajectory scaling for robot manipulators

Marco Faroni[1], Roberto Pagani[2], Giovanni Legnani[2]

[1]Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato
Consiglio Nazionale delle Ricerche – Milano, Italy
[2]Dipartimento di Ingegneria Meccanica e Industriale, University of Brescia – Brescia, Italy
Email: marco.faroni@stiima.cnr.it, r.pagani001@unibs.it, giovanni.legnani@unibs.it

***Abstract*— Recent developments in industrial robotics use real-time trajectory modification to improve throughput and safety in automatic processes. Online trajectory scaling is often used to this purpose. In this paper, we propose a feedback trajectory scaling approach that is able to recover from the delay introduced by the speed modulation and improves the path-following performance thanks to an additional inner control loop. Simulation and experimental results on an industrial 6-degree-of-freedom robot show the effectiveness of the proposed approach compared to standard algorithms.**

I. Introduction

Industrial robotics is facing many new challenges nowadays. Most of times, robots needs to be endowed with some degree of autonomy to adapt their behavior to the state of the environment and other agents. It is the case, for example, of Human Robot Collaboration (HRC) applications, where the robot has to take into account the operator presence to avoid/limit collisions, reduce interference, and maximize the process throughput. HRC requires a change of paradigm with respect to classic industrial robot applications: human-robot interaction has to be considered at different levels (safety, planning, control, scheduling, sociology) and an integrated approach is necessary to deploy an effective HRC framework. Among all the aforementioned aspects, it is clear that a fundamental requirements for such dynamic applications is the ability of the robot to change its motion at runtime. Such ability is of great importance as it allows the robot to react immediately to the current state of the other agents (*e.g.*, humans and other robots). This directly affects the safety of the interaction but also the maximization of the throughput, as an optimized motion re-planning might reduce or avoid robot stops and overly conservative slowdowns. Online modification of the robot motion can be performed mainly in two different ways: by modifying the whole trajectory that the robot has to follow, or by modifying only the velocity profile but keeping the original path. The first option is typically more complex as it requires to run path-planning algorithms and to check for collisions at run-time. The second option is often preferred because it only requires to slow down the execution of the trajectory along the same path. It therefore does not require to devise a new collision-free path and it is way lighter from the computational point of view.

Speed modulation is also more and more exploited to ensure safety during human-robot interaction. In particular, two of the four safety functions defined by the collaborative robot safety standard ISO/TS 15066:2016 [1] (namely, speed and separation monitoring and power force limitation) can be implemented by means of the online reduction of the robot speed according to the constraints defined by the standard. Safety speed modulation has typically been addressed in a static way. The workspace is divided in danger zones based on a risk assessment analysis that considers the reachability of the robot and the operator. Then, if an operator enters the workspace, the robot is slowed down according to the risk level of the zone accessed by the human. However, such approach usually leads to overly conservative slowdowns and stops of the robot, with consequent decay of the process throughput and of the smoothness of the collaboration. Recent approaches tend to modulate the robot speed dynamically, based on the human-robot distance to satisfy the minimum protective distance criterion or the maximum impact force in an optimized manner [2], [3]. Such approach might benefit from the use of optimized speed-modulation algorithms that take into account also the robot physical limits.

A. Related works

Recently, speed modulation algorithms for safe HRC goes toward the use of online trajectory scaling methods because the speed modulation can be optimized based on the human-robot relative distance (and velocity). This avoids overly conservative motion that would jeopardize the fruitfulness of the HRC. Online trajectory scaling methods deform the original timing law to satisfy the robot constraints. From a practical perspective, the advantage of online scaling is that the nominal motion can be devised without rigorously taking the robot limits into account, as the online algorithm will account for them at runtime. The offline planning can be therefore made simpler and less conservative, without compromising the quality of the process. Trajectory scaling algorithms are usually based upon the optimization of a performance criterion (*e.g.*, minimum time), whereas physical and safety-related limits are modeled as optimization constraints. The resulting constrained optimization problem is then solved at each cycle to find the new velocity profile to be given to the robot. As a general approach, trajectory scaling algorithms exploit the path-velocity decomposition of the task and they use the parametrization variable of the path curve as an additional degree of freedom to meet the robot constraints. The input of the algorithm is the desired motion to be followed (in

the Cartesian or the joint space) and the output is the scaled trajectory (typically the joint positions and velocities given to the robot low-level controller).

Several approaches have been proposed in the literature, including feedback techniques [4], [5], look-ahead methods [6], [7], and methods specific for redundant robots [8], [9]. Safety-oriented applications for HRC were proposed in [10], [11], [12]. All the aforementioned methods have pros and cons:

- Feedback methods (also referred to as *local*), such as [4], [5], [8], have low computational burdens, but they do not take into account future information about the tasks, the robot constraints, and possible predictions of the environment/human states. This results in poor solutions in terms of process throughput and quality (*i.e.*, the scaling are not able to avoid path deformation).
- Look-ahead/predictive techniques, such as [6], [7], [9], give much better solutions, but have higher computational burdens. In particular, they make use of nonlinear and constrained mathematical programming. To deal with sampling periods in the order of few milliseconds, they use approximated solutions such as linearization of constraints and cost functions, and down-sampling. However, their implementation might be troublesome on commercial hardware both from the software and hardware point of view due to the lack of advanced mathematical tools and limited computational resources.
- All trajectory scaling methods slow down the nominal trajectory when it is too demanding with respect to the robot physical limits or when it would violate safety or technological requirements. This results in a delay with respect to the nominal task. In some cases, such delay can be recovered by speeding up the remainder of the task. However, most of the algorithms in the literature do not show any delay-recovering feature, except for [5] and [11]. Nonetheless, also in these cases, the recovering of the delay might lead to jerky motion and overshoots due to difficulties in the tuning phase.

B. Contribution

This paper presents an online trajectory scaling method with the following novelties:

- It formulates a delay-recovering scheme as a standard control loop that can be applied to any trajectory scaling method as an outer control loop. It guarantees ease of implementation, intuitive tuning and it allows using common control strategies such as PID controllers to improve the control performance.
- It implements an additional inner loop that smooths steep changes of the scaled velocity profile. This results in a dramatic improvement of the path-following performance compared to standard feedback methods.

Moreover, the method is tested on an industrial robot and its commercial controller (namely, the 6-axis robot Efort model ER3A C-60 with controller Robox RP-1) to prove the suitability of the approach on industrial equipment. The experimental results show that the integration of scaling methods in the existing control architecture can improve the process throughput by reducing the cycle time without worsening the quality of the task. Finally, results show that the proposed scheme outperforms state-of-the-art feedback techniques without increasing the computational burden.

II. Preliminaries

A. Online trajectory scaling

Consider a curve x_d in the Cartesian space, parametrized with respect to the scalar variable γ such that:

$$x_d : \mathbb{R} \to \mathbb{R}^6, \gamma \mapsto x_d(\gamma). \tag{1}$$

The trend of γ with respect to time defines the timing law of the trajectory. Often, the timing law is given due to technological requirements (for example, constant-speed traits may be needed in the process), but it might be relaxed to ensure safety or to preserve the quality of the process.

Feedback trajectory scaling algorithms allow the relaxation of the timing law by using $\dot{\gamma}$ as a control variable. In particular, given a velocity profile $\dot{\gamma}_{\text{ref}}$ and denoting with q, $\dot{q}$, and $\ddot{q}$ the robot joint configuration, velocity, and acceleration vectors, the algorithm solves the following control problem at each time instant t_0:

$$\underset{q,v}{\text{minimize}} \ \left\| x_d\big(\gamma(t_0)\big) - f(q) \right\|_Q^2 + \left\| u(t_0) - v \right\|_R^2 \tag{2}$$

subject to $\dot{\gamma} = v$ and the robot limitations, for example:

$$q_{\min} \leq q \leq q_{\max} \tag{3}$$

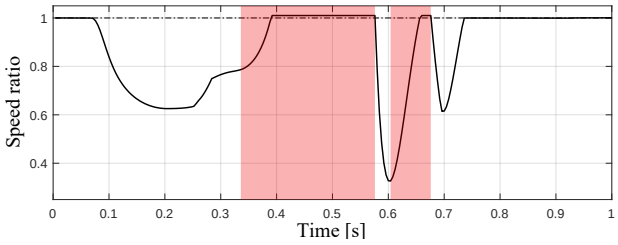
$$-\dot{q}_{\max} \leq \dot{q} \leq \dot{q}_{\max} \tag{4}$$

$$-\ddot{q}_{\max} \leq \ddot{q} \leq \ddot{q}_{\max} \tag{5}$$

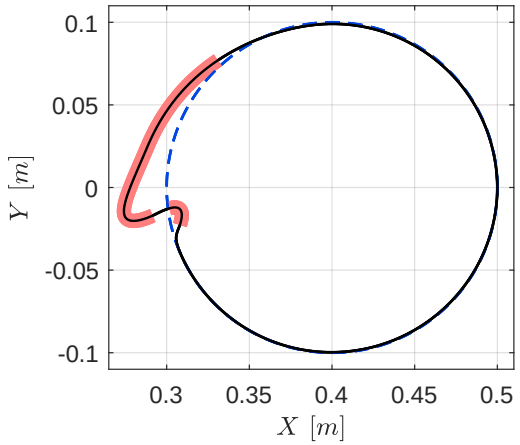
where $u = \dot{\gamma}_{\text{ref}}$, f is the forward kinematic function of the robot, Q and R are weighting matrices, $\|x\|_Q^2 = x^T Q x$, $q_{\min}$ and $q_{\max}$ are the joint mechanical limits, $\dot{q}_{\max}$ is the maximum joint velocity vector, and $\ddot{q}_{\max}$ is the maximum joint acceleration vector. Additional constraints can be added to account for safety limits [10]. Notice that (2) only concerns the trajectory and the robot state at time t_0. A more advanced approach would consider also the cost (2) and the constraints (3) in the future instants, as in [6], [7], [9].

B. Issues and illustrative examples

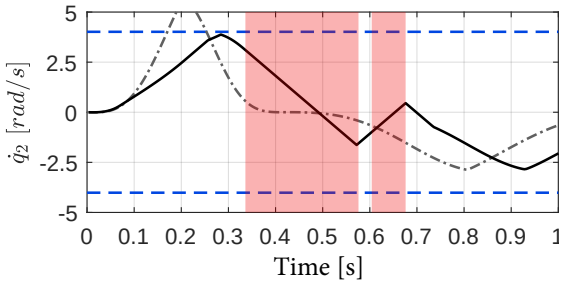
Feedback trajectory scaling algorithms suffer from large path deformation when the nominal timing law has steep acceleration and deceleration profiles. The reason is that the scaling takes action when the deceleration phase has already begun and it is therefore too late to avoid overshoots and deviations with respect to the nominal path, given the hard deceleration limits of the joints. As an illustrative example, consider a 6-degree-of-freedom Efort ER3A C-60 robot that performs a circular path in the Cartesian space. Referring to Figure 1b, the robot moves counter-clockwise from point (0.5,0) to (0.3,0), it stops, and then moves from (0.3,0) to (0.5,0), counter-clockwise. The nominal timing law would



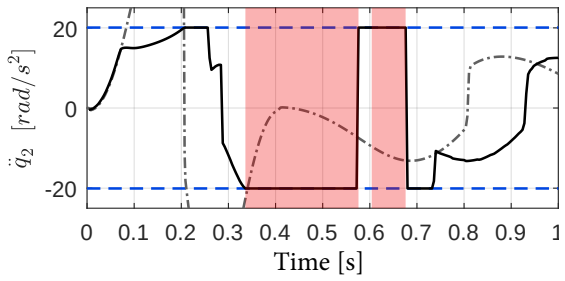
(a) Speed ratio $\dot{\gamma}/\dot{\gamma}_{\text{ref}}$



(b) Performed path



(c) Joint 2 velocity



(d) Joint 2 acceleration

Fig. 1: Issues on feedback trajectory scaling algorithms. (a) Ratio between the scaled and the nominal velocity profile. (b) Path performed by the robot. Dashed blue: nominal path; solid black: performed path. (c) Second joint's velocity. (d) Second joint's acceleration. Dashed blue: joint limits; dash-dotted gray: nominal trajectory; solid black: performed trajectory. The red areas highligths the braking phase.

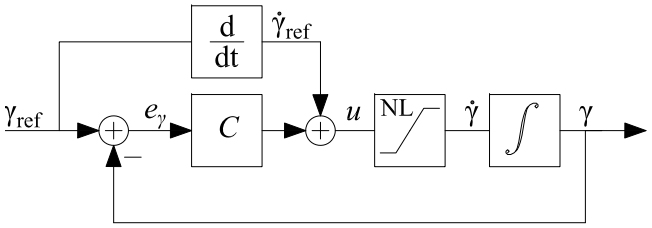


Fig. 2: Proposed delay-recovering loop scheme.

exceed the acceleration bounds of the robot (Figure 1d) in the first movement. This results in the saturation of the joint acceleration (Figure 1d) and in a significant deformation of the path when the robot is supposed to stop in (0.3,0) (see Figure 1b). Figure 1a shows the ratio between the scaled velocity and the nominal velocity computed by the scaling algorithm. It is possible to see that the algorithm is not able to prevent the steep deceleration that occurs in the highlighted area and thus lead to a significant path-following error.

Another issue of many trajectory scaling methods is that they do not recover the delay given by the slowdown. In the example, the robot might recover the delay in the second half of the circle by speeding up the nominal timing law (*i.e.*, by computing a ratio greater than one in Figure 1a), but the scaling algorithm does not implement such feature.

III. Proposed Algorithm

This section addresses the issues presented in Section II. First, a closed-loop controller is proposed to recover from the delay introduced by the scaling of the trajectory. Then, the proposed scheme is modified by adding an inner integral control loop (named *saturation fly-wheel* to improve the path-following performance of the algorithm.

A. Delay-recovering control loop

The delay-recovering feature can be implemented as an external control loop, where the controlled variable is the parameter γ, the reference signal is γ_{ref}, and the control error is therefore given by $e_\gamma = \gamma_{\text{ref}} - \gamma$. Finally, $\dot{\gamma}_{\text{ref}}$ can be seen as a feedforward action: in the case no slowdowns occur, the control input is just $\dot{\gamma}_{\text{ref}}$; when the scaling is activated, the external loop increases the control action to recover the delay. The overall control scheme is shown in Figure 2, where the online control problem (2)-(3) is represented as a nonlinear saturation block between u and $\dot{\gamma}$.

B. Saturation fly-wheel block

Look-ahead scaling techniques show significant improvements compared to feedback methods. However, as also suggested in [13], using the past history of the desired trajectory and the behavior of the algorithm might give good results in preserving the path feasibility. In [13], heuristic thresholds were set to slow down the trajectory in advance, based upon the trend of γ and its derivatives. However, the tuning of such thresholds is cumbersome and their effectiveness varies from case to case.

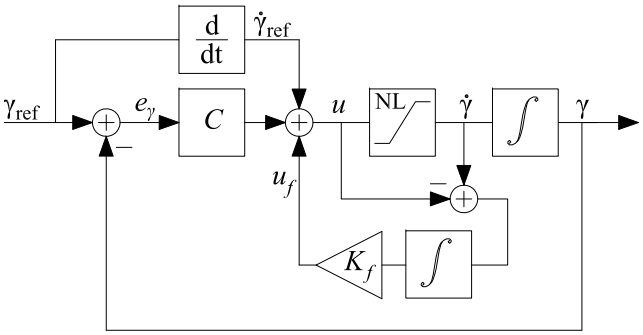


Fig. 3: Proposed scheme with delay-recovering loop and saturation fly-wheel block.

Moreover, the delay-recovering scheme in Figure 2 introduces an additional issue. When the optimal control problem returns a constrained solution, the control system behaves like a saturation block. Given that γ_{ref} is monotonically increasing with respect to time, the control error and the control variable might increase unboundedly. When the trajectory exits the slowdown phase, the large control error drives the system to saturation. To avoid this, the control error should be bounded to a value that drives the $\dot{\gamma}_{\text{ref}}$ close to the saturated value of $\dot{\gamma}$.

We propose to insert an inner control loop composed of an integral system with gain K_f. As shown in Figure 3, the control input is now computed as:

$$u = \dot{\gamma}_{\text{ref}} + C(e_\gamma) + u_f \tag{6}$$

where C is the delay-recovering control function and $u_f = K_f \int_0^{t_0} (\dot{\gamma} - u)\, dt$.

This new control loop stabilizes $\dot{\gamma}_{\text{ref}}$ to a bounded value. The higher K_f, the faster $\dot{\gamma}_{\text{ref}}$ tracks γ. A forgetting factor F_f is also used to avoid the influence of outdated data on the current iteration. By using a zero-order discrete time approximation with sampling period T, u_F becomes:

$$u_f = K_f T \sum_{i=1}^{F_f} \dot{\gamma}(t_0 - iT) - u(t_0 - iT) \tag{7}$$

The effect of this additional control loop is to keep $\dot{\gamma}_{\text{ref}}$ close to γ during saturation and to accumulate a "saturation inertia" (from which the name *saturation fly-wheel*) that reduces the growth of $\dot{\gamma}_{\text{ref}}$ that occurs when the saturation ends.

IV. Results

Simulation and experimental tests have been performed on a 6-degree-of-freedom Efort robot (model ER3A-C60) installed at the Industrial Robotics Laboratory of the University of Brescia. It has a total weight of 27 kg and a payload of 3 kg. Figure 4 shows the mechanical structure of the manipulator and its controller. The robot joints limits, reported below, have been obtained from the datasheet

Fig. 4: Robot Efort ER3A C-60 with controller Robox RP-1

provided by the manufacturer:

$$\begin{aligned} q_{\max} &= (\ 2.91,\ 1.57,\ 1.76,\ 3.14,\ 1.97,\ 6.28\ ) \text{ rad}, \\ q_{\min} &= (\ -2.91, -2.26, -1.23, -3.14, -1.97, -6.28\ ) \text{ rad}, \\ \dot{q}_{\max} &= (\ 4.0,\ 4.0,\ 4.3,\ 5.5,\ 5.5,\ 7.3\ ) \text{ rad/s}, \\ \ddot{q}_{\max} &= (\ 20,\ 20,\ 22,\ 28,\ 28,\ 36\ ) \text{ rad/s}^2. \end{aligned} \tag{8}$$

To generate the nominal trajectory, we use the motion planner embedded in the robot controller. The planner tends to compute conservative trajectories with respect to the limits of the actuators. The reason is that an optimal minimum-time planning would result in a complex mathematical programming problem and industrial planners usually use empirical methods to avoid such heavy computation.

To evaluate the algorithms performances in preserving the desired path, the nominal trajectory is made more demanding by shrinking the execution time computed by the robot planner by a factor $\lambda < 1$. In this way, the maximum velocity and acceleration values of the trajectories are increased as follows:

$$v_{m2} = v_{m1}/\lambda, \quad a_{m2} = a_{m1}/\lambda^2 \tag{9}$$

where v_{m1} and a_{m1} are the maximum velocity and acceleration values of the original trajectory and v_{m2} and a_{m2} are the maximum velocity and acceleration values of the new trajectory. In this way it is possible to generate more demanding trajectory that are expected to exceed the robot limits if the scaling algorithms are not implemented.

As regards the online trajectory scaling, two algorithms have been tested for comparison:

- The approach proposed in the previous section with the Delay-Recovering loop and the saturation Fly-wheel block. The method will be also referred to as DRF.

- The feedback method (also referred to as FB) based on the solution of (2), (3). This is equivalent to the method proposed in [8] with joint acceleration limits added as described in [14].

As regard the DRF method, a simple proportional controller $C = Ke_\gamma$ with $K = 0.1$ has been used. The integral gain and forgetting factor of the saturation fly-wheel have been set to $K_f = 100$ and $F_f = 125$.

A. Simulations

Simulations have been performed on a constrained kinematic model of the robot. The desired task consists in the interpolation of five poses in the Cartesian space through successive circular arcs. The chosen poses are:

$$\begin{aligned} X_1 &= (\,0.5\,\text{m}, \quad 0.0\,\text{m}, \quad 0.5\,\text{m}, \quad 5^\circ, -175^\circ, \quad 5^\circ\,) \\ X_2 &= (\,0.4\,\text{m}, \quad 0.2\,\text{m}, \quad 0.5\,\text{m}, \quad 5^\circ, -175^\circ, \quad 5^\circ\,) \\ X_3 &= (\,0.3\,\text{m}, \quad 0.0\,\text{m}, \quad 0.5\,\text{m}, \quad 5^\circ, -175^\circ, \quad 5^\circ\,) \\ X_4 &= (\,0.4\,\text{m}, \quad -0.2\,\text{m}, \quad 0.5\,\text{m}, \quad 5^\circ, -175^\circ, \quad 5^\circ\,) \\ X_5 &= X_1 \end{aligned} \tag{10}$$

where the first three terms of the vectors represent the (x, y, z) position of the end effector and the the last three terms represent the orientation expressed in *XYZ* Euler angles. The robot is programmed in such a way that it starts in X_1, it moves through X_2 and then stops in X_3. Then, it moves from X_3 through X_4 and finally stops in X_5.

Results compare the maximum and mean path-following errors (e_{max} and e_{mean}) and the average slowdown due to the algorithms. Position path-following error is computed as the Euclidean distance between the end-effector position and the closest point on the desired path. Orientation error is the absolute value of the angle between the end-effector reference frame and the desired reference frame. The average slowdown is the ratio between the time t_{real} taken by the robot to perform the whole task and the desired execution time t_{end}.

Results for several values of the nominal total time t_{end} are shown in Tables I, II, and III. The less demanding case ($t_{\text{end}} = 1.45$ s) does not activate the scaling mechanism and it is therefore taken as reference to evaluate the following tests. It is possible to see that as the task gets more demanding (*i.e.*, smaller values of t_{end}), the mean and maximum path-following errors tend to increase. However, the error obtained by DRF is of the same order of magnitude of the base case, while the FB method gives significantly larger error (up to two order of magnitude larger than the reference case). This is clear also from Figure 5, which shows the comparison of the performed path for three different values of t_{end}. The paths performed by FB (red lines) show a significant deviation with respect to the nominal path, especially in the breaking phase. On the contrary, DRF is able to preserve the nominal path also for small values of t_{end}. Moreover, Table III shows that the delay-recovering method enhances the performance of the algorithm from the point of view of the total execution time. As a matter of fact, the ratio between the real execution time of the task t_{real} and the nominal time t_{end} is much closer to 1 when DRF is used. Finally, Figure 6 shows the positions, velocities, and accelerations of the first, second, and third joints for the case $t_{\text{end}} = 0.85$ s. It is possible to see that the nominal trajectory is expected to exceed the velocity and acceleration limits, but the scaling algorithms adapt the motion not to overpass them. In general, DRF gives smoother trends of the joint variables and a shorter execution time.

TABLE I: Translation error for the circular task.

t_{end} [s]	e_{max} [m]		e_{mean} [m]	
	DRF	FB	DRF	FB
1.45	$2.70 \cdot 10^{-4}$	$2.91 \cdot 10^{-4}$	$9.08 \cdot 10^{-5}$	$9.37 \cdot 10^{-5}$
1.25	$3.68 \cdot 10^{-4}$	$3.77 \cdot 10^{-3}$	$1.19 \cdot 10^{-4}$	$5.59 \cdot 10^{-4}$
1.05	$4.60 \cdot 10^{-4}$	$1.43 \cdot 10^{-2}$	$1.58 \cdot 10^{-4}$	$2.89 \cdot 10^{-3}$
0.85	$5.89 \cdot 10^{-4}$	$2.44 \cdot 10^{-2}$	$2.18 \cdot 10^{-4}$	$6.06 \cdot 10^{-3}$
0.65	$7.92 \cdot 10^{-4}$	$2.54 \cdot 10^{-2}$	$3.26 \cdot 10^{-4}$	$6.53 \cdot 10^{-3}$

TABLE II: Angular error for the circular task.

t_{end} [s]	e_{max} [rad]		e_{mean} [rad]	
	DRF	FB	DRF	FB
1.45	$1.65 \cdot 10^{-5}$	$1.72 \cdot 10^{-5}$	$5.68 \cdot 10^{-6}$	$5.92 \cdot 10^{-6}$
1.25	$2.18 \cdot 10^{-5}$	$2.89 \cdot 10^{-4}$	$6.88 \cdot 10^{-6}$	$3.82 \cdot 10^{-4}$
1.05	$2.69 \cdot 10^{-5}$	$1.01 \cdot 10^{-3}$	$7.71 \cdot 10^{-6}$	$2.01 \cdot 10^{-4}$
0.85	$2.96 \cdot 10^{-5}$	$1.87 \cdot 10^{-3}$	$8.45 \cdot 10^{-6}$	$4.17 \cdot 10^{-4}$
0.65	$2.96 \cdot 10^{-4}$	$1.99 \cdot 10^{-3}$	$8.81 \cdot 10^{-6}$	$4.45 \cdot 10^{-4}$

TABLE III: Ratio between the time taken to perform the trajectory and the nominal execution time $t_{\text{real}}/t_{\text{end}}$ for the circular task.

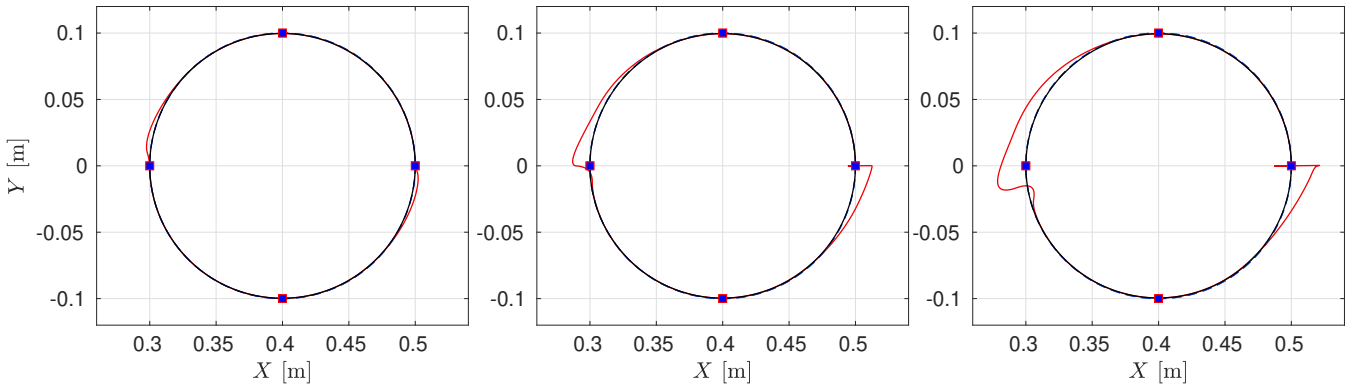
t_{end} [s]	$t_{\text{real}}/t_{\text{end}}$	
	DRF	FB
1.45	1.022	1.003
1.25	1.067	1.009
1.05	1.198	1.312
0.85	1.368	1.868
0.65	1.714	2.400

B. Experiments

The desired task consists in the pentagonal path given by the linear interpolation of the following six poses in the Cartesian space:

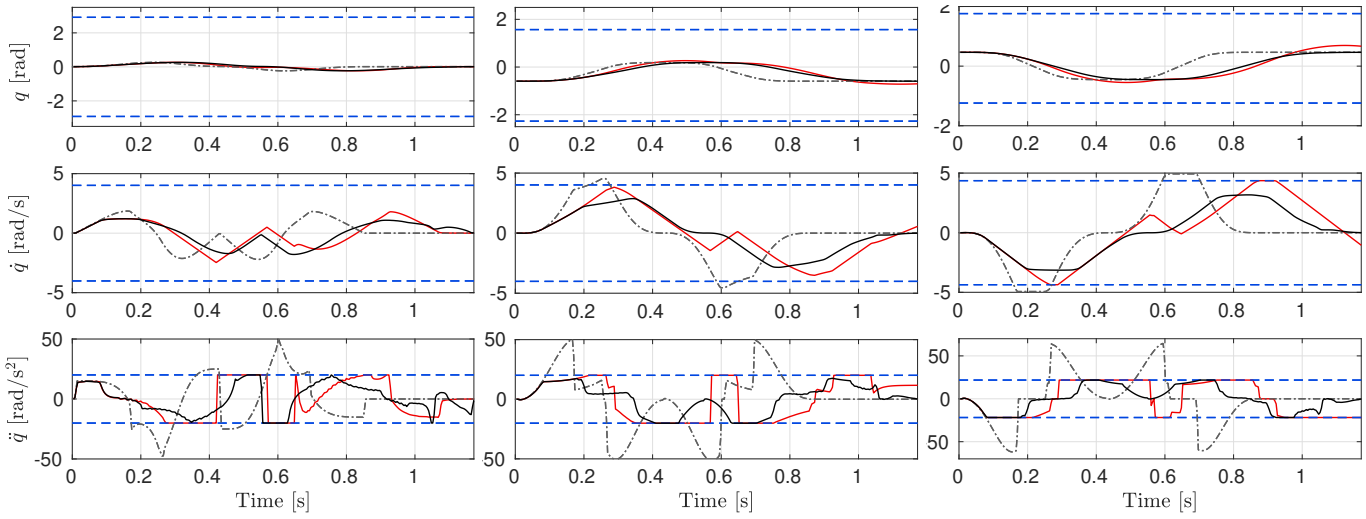
$$\begin{aligned} X_1 &= (\quad 0.5\,\text{m}, \quad 0.0\,\text{m}, \quad 0.5\,\text{m}, \; 5^\circ, -175^\circ, 5^\circ\,) \\ X_2 &= (\,0.428\,\text{m}, \; 0.121\,\text{m}, \; 0.45\,\text{m}, \; 5^\circ, -175^\circ, 5^\circ\,) \\ X_3 &= (\,0.311\,\text{m}, \; 0.075\,\text{m}, 0.368\,\text{m}, \; 5^\circ, -175^\circ, 5^\circ\,) \\ X_4 &= (\,0.311\,\text{m}, -0.075\,\text{m}, 0.368\,\text{m}, \; 5^\circ, -175^\circ, 5^\circ\,) \\ X_5 &= (\,0.428\,\text{m}, -0.121\,\text{m}, \; 0.45\,\text{m}, \; 5^\circ, -175^\circ, 5^\circ\,) \\ X_6 &= X_1 \end{aligned} \tag{11}$$

The robot starts from pose X_1 and it performs a point to point motion until X_6. Three different values of t_{end} have been



(a) $t_{\text{end}} = 1.25$ s (b) $t_{\text{end}} = 1.05$ s (c) $t_{\text{end}} = 0.85$ s

Fig. 5: Simulation results: performed paths in plain xy. Dashed blue: nominal path; solid black: DRF method; solid red: FB method (solid black and dashed blue are often indistinguishable because of overlapping).



(a) Joint 1 (b) Joint 2 (c) Joint 3

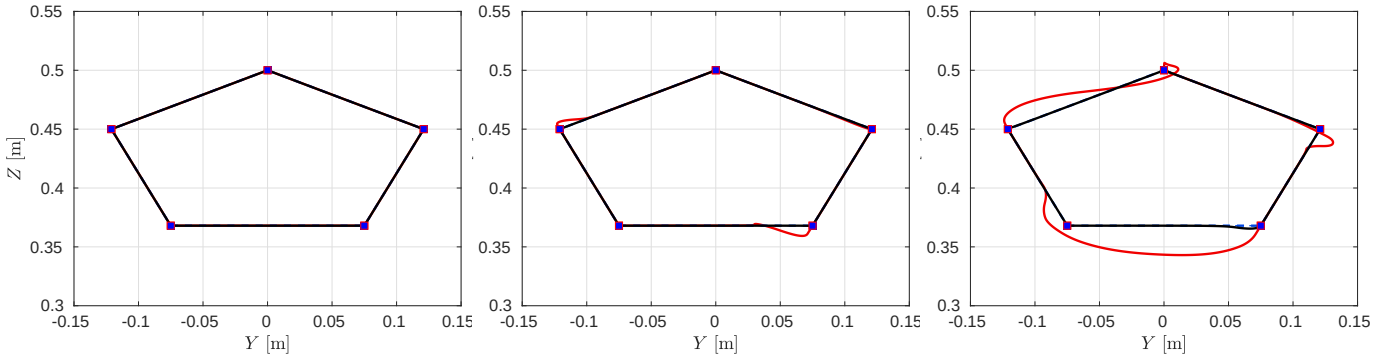
Fig. 6: Simulation results: positions (top), velocities (middle) and accelerations (bottom) of the first three joints for $t_{\text{end}} = 0.85$ s. Dash-dotted gray: nominal trajectory; solid black: DRF method; solid red: FB method; dashed blue: joint limits.

tested, namely: $t_{\text{end}} = 2.2$ s, $t_{\text{end}} = 1.6$ s, and $t_{\text{end}} = 1.0$ s. A video of the experiments can be found in the enclosed material. The first case is the less demanding and does not activate the scaling mechanism. The successive cases are more and more demanding and require the activation of the scaling algorithms.

Figure 7 shows the nominal path and the paths performed by the two methods. Moreover, Figure 8 shows a graphical reconstruction of the performed paths from the aforementioned video[1]. As expected, for the case $t_{\text{end}} = 2.2$ s does not activate the scaling, the performed paths do not differ from the nominal one (Figures 7a and 8a). As soon as the trajectories get more demanding the scaling algorithms are activated: FB method shows a visible path-following error, while DRF is able to preserve the geometry of the path (Figures 7b and 8b). Also in the extreme case $t_{\text{end}} = 1.0$ s, DRF still shows good path-following performance, while FB results in a completely deformed path (Figures 7c and 8c).

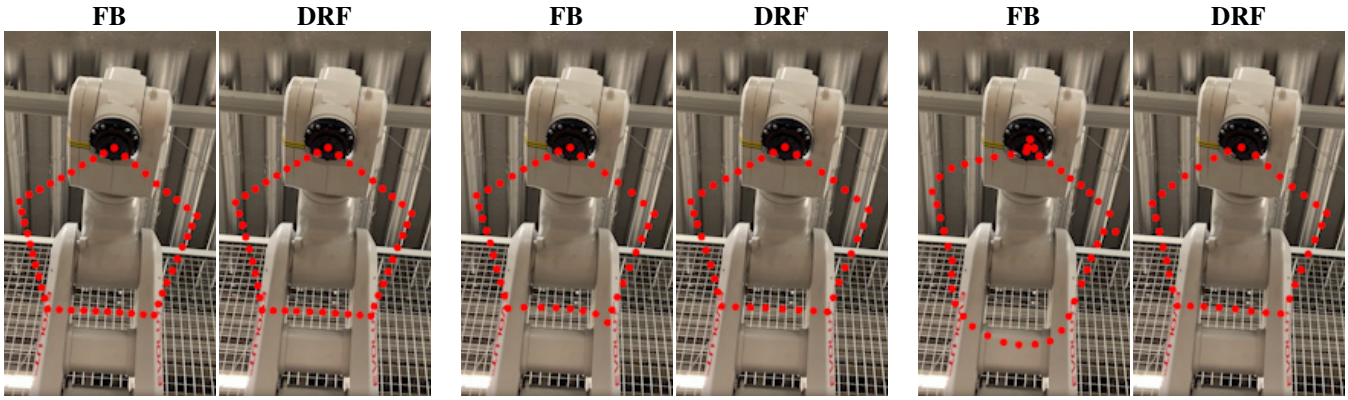
V. Conclusions

The paper presented a feedback trajectory scaling algorithm for industrial robot manipulators. The proposed method can be applied to any trajectory scaling algorithms to improve the execution time and the path-following performance of the algorithm. Simulation and experimental results showed the potentiality of the method on a 6-degree-of-freedom robot. It is worth stressing that the ease of implementation and the light computational burden allow the implementation on commercial industrial hardware, as shown in the experiments.



(a) $t_{\text{end}} = 2.2$ s (b) $t_{\text{end}} = 1.6$ s (c) $t_{\text{end}} = 1.0$ s

Fig. 7: Experimental results: performed paths in plain yz. Dashed blue: nominal path; solid black: DRF method; solid red: FB method (solid black and dashed blue are often indistinguishable because of overlapping).



(a) $t_{\text{end}} = 2.2$ s (b) $t_{\text{end}} = 1.6$ s (c) $t_{\text{end}} = 1.0$ s

Fig. 8: Experimental results: graphical reconstruction of the performed paths from the enclosed video.

ACKNOWLEDGMENT

The research leading to these results was partially funded by the European Union H2020-ICT-2017-1 – Pickplace: Flexible, safe and dependable robotic part handling in industrial environment (grant agreement: 780488).

REFERENCES

[1] "ISO/TS 15066:2016 Robots and robotic devices – Collaborative robots," International Organization for Standardization, Geneva, CH, Standard, 2016.

[2] J. A. Marvel and R. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robotics and computer-integrated manufacturing*, vol. 44, pp. 144–155, 2017.

[3] C. Byner, B. Matthias, and H. Ding, "Dynamic speed and separation monitoring for collaborative robot applications–concepts and performance," *Robotics and Computer-Integrated Manufacturing*, vol. 58, pp. 239–252, 2019.

[4] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Rob. Autom.*, vol. 6, pp. 554–561, 1990.

[5] C. Guarino Lo Bianco and O. Gerelli, "Online trajectory scaling for manipulators subject to high-order kinematic and dynamic constraints," *IEEE Trans. Rob.*, vol. 27, pp. 1144–1152, 2011.

[6] M. Faroni, M. Beschi, C. Guarino Lo Bianco, and A. Visioli, "Predictive joint trajectory scaling for manipulators with kinodynamic constraints," *Control Engineering Practice*, vol. 95, p. 104264, 2020.

[7] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 1505–1511, 2017.

[8] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. Rob.*, vol. 31, pp. 637–654, 2015.

[9] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, "Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 278–285, 2018.

[10] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 882–893, 2016.

[11] M. Lippi and A. Marino, "Safety in human-multi robot collaborative scenarios: a trajectory scaling approach," in *Proc. IFAC SYROCO*, Budapest (Hungary), 2018.

[12] M. Faroni, M. Beschi, and N. Pedrocchi, "An MPC framework for online motion planning in human-robot collaborative tasks," in *Proceedings of the IEEE International Conference on Emerging Technologies and Factory Automation*, Zaragoza (Spain), 2019.

[13] C. Guarino Lo Bianco and F. Ghilardelli, "Techniques to preserve the stability of a trajectory scaling algorithm," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe (Germany), 2013, pp. 870–876.

[14] F.-T. Cheng, T.-H. Chen, and Y.-Y. Sun, "Resolving manipulator redundancy under inequality constraints," *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 65–71, 1994.