2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Improving the Performance of Auxiliary Null Space Tasks via Time Scaling-Based Relaxation of the Primary Task

Nico Mansfeld, Youssef Michel, Tobias Bruckmann and Sami Haddadin

Abstract— **Kinematic redundancy enhances the dexterity and flexibility of robot manipulators. By exploiting the redundant degrees of freedom, auxiliary null space tasks can be carried out in addition to the primary task. Such auxiliary tasks are often formulated in terms of a performance or safety criterion that shall be minimized. If the optimization criterion, however, is defined in global terms, then it is directly affected by the primary task. As a consequence, the task achievement of the auxiliary task may be unnecessarily detrimented by the main task. In addition to modifying the primary task via constraint relaxation, a possible solution for improving the performance of the auxiliary task is to relax the primary task temporarily via time scaling. This gives the null space task more time for achieving its objective. In this paper, we propose several such time scaling schemes and verify their performance for a DLR/KUKA Lightweight Robot with one redundant degree of freedom. Finally, we extend the concept to multiple prioritized tasks and provide a simulation example.**

I. Introduction

Many of today's robots are equipped with more degrees of freedom (DOF) than necessary to accomplish a desired task, e.g., a six-DOF Cartesian trajectory of the end-effector. The redundancy enhances the overall flexibility and dexterity of the system and can be exploited to fulfill auxiliary null space tasks which do not interfere the primary task. The objective of a null space task is often formulated in terms of a performance or safety criterion that shall be minimized, e. g., the manipulability measure in case of singularity avoidance [1], the distance to the joint position limits [2], or the estimated contact force during a collision with the robot [3]. If the optimization criterion is defined globally, then both the primary task and the null space task influence this criterion during task execution. This means that the main task can either support or hinder the null space scheme from achieving its objective. This is also due to the fact that typical hierarchical task control schemes are defined in an instantaneous manner and do not take the entire temporal dimension of task fulfillment into account [4], [5].

One obvious solution is to use motion planning schemes for generating trajectories that respect all constraints and optimization criteria. However, motion planning is typically done offline and computationally very costly if done online. In terms of realtime control, a possible solution is to relax the main task through a suitable task scaling [6], [7] or priority switching scheme [8]–[11]. Such approaches, however, sacrifice the nominal execution of the primary task in order to satisfy the auxiliary tasks. They have been used for avoiding obstacles [8], [9], joint limits [6], [7], [10], or kinematic singularities [10], [11], for example.

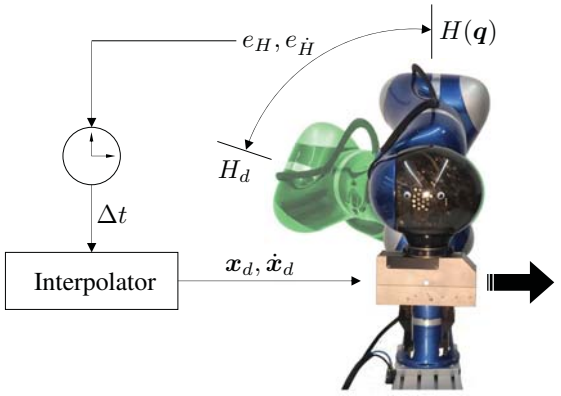

Fig. 1. Basic concept. The null space performance criterion at the current configuration is denoted $H(\boldsymbol{q})$, the desired value H_d. The difference $e_H = H(\boldsymbol{q}) - H_d$ and a suitable synchronization error $e_{\dot{H}}$ are used to modify the timing law of the interpolator, which provides the desired joint or Cartesian space trajectory (here: $\boldsymbol{x}_d, \dot{\boldsymbol{x}}_d$) to the robot. The speed reduction gives the null space scheme more time to minimize the performance criterion.

The solution developed in this paper is based on slowing down the main task temporarily while preserving its geometric description. By reducing the execution speed, the null space task is given more time to optimize for its performance criterion. The idea was – to the best of the authors' knowledge – first suggested in [12]. There, the operational speed was reduced whenever the ratio of available and desired null space velocity/torque for minimizing a certain optimization function became smaller than a threshold. In our work, we further investigate the concept of scaling the main task in time for improving the minimization performance of auxiliary tasks. The objective is to keep the same primary and null space controller but modify the timing law of the desired task trajectory online, see Fig. 1. Such relaxation leaves the principal behavior of the system unchanged. In this paper, we propose several time scaling schemes and verify them experimentally with a DLR/KUKA Lightweight Robot IV (LWR) for one redundant DOF. Furthermore, we extend the concept to multiple prioritized tasks and provide simulation results.

The paper is organized as follows. In Sec. II we describe the considered robot dynamics and the problem formulation. Then, we propose possible time scaling schemes in Sec. III. In Sec. IV, the experiments for verifying the performance of the time scaling methods are described. An extension to multiple prioritized task including a simulation of a 4R planar robot is provided in Sec. V. Finally, Sec. VI concludes the paper.

The first two authors contributed equally to this work. Nico Mansfeld is with the Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Wessling, Germany. Youssef Michel is with the Department of Human-Centered Assistive Robotics. Tobias Bruckmann is with the Chair of Mechatronics, University of Duisburg-Essen, Duisburg, Germany. Sami Haddadin is with the Chair of Robotics Science and Systems Intelligence and Munich School of Robotics and Machine Intelligence, Technical University of Munich (TUM), Munich, Germany nico.mansfeld@dlr.de

978-1-5386-6027-0/19/$31.00 ©2019 IEEE

II. Problem Formulation

The considered robot dynamics can be expressed as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{ext}}\,, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ are the generalized coordinates, $\boldsymbol{M}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the symmetric, positive definite mass matrix, $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal matrix, and $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the gravity vector. The joint torques and the external torques are denoted by $\boldsymbol{\tau} \in \mathbb{R}^n$ and $\boldsymbol{\tau}_{\text{ext}} \in \mathbb{R}^n$, respectively. The main task is given by $\boldsymbol{x} = f_k(\boldsymbol{q})$, where $\boldsymbol{x} \in \mathbb{R}^m$ and $f_k(\boldsymbol{q})$ is the nonlinear forward kinematics. The $m \times n$ task Jacobian is given by $\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial f_k(\boldsymbol{q})}{\partial \boldsymbol{q}}$ and relates joint-space velocities to task-space velocities via $\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$.

For sake of clarity, let us assume that one secondary task shall be fulfilled in addition to the main task in order to resolve the robot's redundancy (cf. Sec. V for an analysis for multiple, hierarchical tasks). The goal of the secondary task shall be the local optimization of a certain performance criterion $H(\boldsymbol{q})$ that is differentiable w.r.t $\boldsymbol{q}$[1]. The joint velocity resulting from control of the main and the secondary task is given by

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_x + \dot{\boldsymbol{q}}_{\text{ns}}\,, \tag{2}$$

where $\dot{\boldsymbol{q}}_x$ can be associated to the main task and $\dot{\boldsymbol{q}}_{\text{ns}} \in \ker(\boldsymbol{J})$ is the null space velocity, which can be obtained, e.g., by projecting the gradient $\nabla H = \frac{\partial H}{\partial \boldsymbol{q}}$ of H onto the null space of the Jacobian matrix. The time derivative of the objective is given by

$$\dot{H} = \nabla H \dot{\boldsymbol{q}} = \nabla H \dot{\boldsymbol{q}}_x + \nabla H \dot{\boldsymbol{q}}_{\text{ns}} = \dot{H}_x + \dot{H}_{\text{ns}}\,. \tag{3}$$

Since the null space controller locally minimizes H, we can assume that $\dot{H}_{\text{ns}}$ is either negative, or zero if a local minimum has been reached. The rate $\dot{H}_x$, however, can be either positive or negative, meaning the main task may support or prevent the null space scheme from minimizing H. Let us consider the following two cases:

1) $\dot{H}_x < 0$, $\dot{H}_{\text{ns}} \leq 0$: Both the main and auxiliary task minimize H.
2) $\dot{H}_x > 0$, $\dot{H}_{\text{ns}} \leq 0$: The main task deteriorates the null space performance. If $|\dot{H}_x| \leq |\dot{H}_{\text{ns}}|$, then H is being minimized (slowly), otherwise H is even increases.

While the first case is certainly desirable, the second case should be avoided, most significantly if $\dot{H}_x > 0$ and $|\dot{H}_x| > |\dot{H}_{\text{ns}}|$. Clearly, the null space dynamics play an important role here. If they are (deliberately) slower than the dynamics of the main task, then H may be increasing during task execution.

We want to ensure $\dot{H} \leq 0$ by limiting the magnitude of $\dot{H}_x$. This shall be accomplished by redistributing the actuation of the main task towards the auxiliary task via time scaling. By slowing down the main task temporarily, the null space task is given more time to minimize the performance or safety criterion. In addition to keeping $\dot{H}$ negative or equal to zero, one may want to keep the magnitude of H as small as possible, respectively the difference of H and a (preferably feasible) desired value H_d. For this, the execution speed needs to be reduced even further than required to fulfill $\dot{H} \leq 0$.

In the following, we propose several time scaling methods to solve our problem. First, we briefly summarize the concept of time scaling and related work on the topic.

[1]In the remainder, we omit the dependency of $\boldsymbol{q}$ where obvious.

III. Time Scaling-Based Relaxation of the Primary Task

The desired joint space trajectory $\boldsymbol{q}_d(t) \in \mathbb{R}^n$ or Cartesian space trajectory $\boldsymbol{x}_d(t) \in SE(3)$ are typically provided by an interpolator and are parameterized w.r.t. time. In a discrete implementation of the interpolator, the current time is given by $t_i = t_{i-1} + \Delta t$, where t_{i-1} is the last time instant and Δt is the increment, which is usually the sampling time. By multiplying the time increment by a factor α

$$t_i = t_{i-1} + \alpha \Delta t\,, \tag{4}$$

we may scale the trajectory in time, i. e., slow down ($\alpha \in [0,1)$), speed up ($\alpha > 1$), or even go backwards along the trajectory ($\alpha < 0$). In the robotics literature, time scaling has been used to solve various control problems. In [13], [14] it was employed to satisfy dynamic and kinematic constraints during task execution. The authors of [15] aimed at improving the tracking performance of pre-planned trajectories, and in [16], time scaling was used for implementing a collision reaction scheme, where α was defined as a function of the estimated external torque. Depending on the amount and the direction of the external torque, the user could push the robot back and forth along the desired path.

In this work, we apply the concept of time scaling to the synchronization of main task and (prioritized) auxiliary null space tasks. The problem is to define α as a function of a suitable null space performance error.

A. General Scheme

In the previous section, we defined two goals for the time scaling scheme, namely 1) $\dot{H}$ shall be less or equal to zero, and 2) H shall be kept as small as possible. Our general time scaling scheme for achieving these goals is defined as

$$\alpha = 1 - (K_H e_H + K_{\dot{H}} e_{\dot{H}})\,, \tag{5}$$

where e_H represents the error of the performance criterion and $e_{\dot{H}}$ is the so-called synchronization error. The corresponding scalar gains for these errors are denoted K_H and $K_{\dot{H}}$, respectively. This definition is also inspired by the literature on motion coordination of two or more dynamical systems [17], [18]. In the following, we define $e_{\dot{H}}$ and propose possible definitions of e_H. Prior to this, we comment on the considered range and dynamics of α.

B. Considered Range and Dynamics of α

The time scaling factor shall satisfy $\alpha \in [0,1]$, i. e., we want to decelerate the robot but not go backwards along the planned path ($\alpha < 0$) or reach higher speeds than the nominal velocity ($\alpha > 1$). When commanding $\alpha = 0$, the robot will stop its motion, which means that the null space scheme will reach its goal (if reachable) after some time. However, in many applications a certain cycle time shall not be exceeded. Therefore, one may demand that the allowable range is $\alpha \in [\alpha_{\min}, 1]$, where $\alpha_{\min} \in (0,1)$. This ensures that the robot always executes the task at a minimum velocity. Another possibility is to supervise the evolution of α during task execution and the time which is lost due to time scaling in order to decide when the trajectory should resume to nominal speed for meeting the cycle time requirement. In this paper, however, we only consider the constant lower bound $\alpha_{\min}$ on α. Practically, one may submit α to critically damped second-order dynamics with relatively low time

constant to ensure that the scaling factor is continuous given the possibly discontinuous errors e_H and $e_{\dot{H}}$.

C. Definition of the Synchronization Error $e_{\dot{H}}$

We define the synchronization error as

$$e_{\dot{H}} = \dot{H}_x - \dot{H}_{\mathrm{ns}}\,. \tag{6}$$

Since this error is only relevant when $\dot{H}_x$ is larger than $\dot{H}_{\mathrm{ns}}$ and both rates have opposite sign, we reformulate $e_{\dot{H}}$ as

$$e_{\dot{H}} = \begin{cases} \dot{H}_x - \dot{H}_{\mathrm{ns}}, & \text{if } \mathrm{sign}(\dot{H}_x) \neq \mathrm{sign}(\dot{H}_{\mathrm{ns}}) \\ & \wedge\, |\dot{H}_x| > |\dot{H}_{\mathrm{ns}}|\,, \\ 0, & \text{otherwise}\,. \end{cases} \tag{7}$$

For determining $\dot{H}_x$ and $\dot{H}_{\mathrm{ns}}$ based on the measured Cartesian and joint velocity, we consider the kinematically decoupled decomposition of the joint space velocities suggested in [19]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\boldsymbol{W}+}\dot{\boldsymbol{x}} + \boldsymbol{Z}^T\boldsymbol{v}_n\,, \tag{8}$$

where $\boldsymbol{J}^{\boldsymbol{W}+}$ is the weighted pseudoinverse [20] and $\boldsymbol{v}_n = (\boldsymbol{Z}\boldsymbol{W}\boldsymbol{Z}^{\mathsf{T}})^{-1}\boldsymbol{Z}\boldsymbol{W}\dot{\boldsymbol{q}}$ is a minimal parameterization of the self motion velocity [21]. The weighting matrix is denoted by $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ and is typically selected as $\boldsymbol{W} = \boldsymbol{M}$. The full row rank[2] matrix $\boldsymbol{Z} \in \mathbb{R}^{(n-m)\times n}$ spans the null space of the Jacobian matrix and satisfies $\boldsymbol{J}\boldsymbol{Z}^{\mathsf{T}} = \boldsymbol{0}$. The $\boldsymbol{Z}$-matrix can be obtained numerically by Singular Value Decomposition (SVD) of $\boldsymbol{J}$ or analytically as described in [22], for example. The rates of H due to the velocity of the main and auxiliary task can now be determined by

$$\dot{H}_x = \nabla H \boldsymbol{J}^{\boldsymbol{W}+}\dot{\boldsymbol{x}}\,, \tag{9}$$

$$\dot{H}_{\mathrm{ns}} = \nabla H \boldsymbol{Z}^T\boldsymbol{v}_n\,. \tag{10}$$

D. Definitions of the Performance Criterion Error e_H

1) Gradient: Since the gradient ∇H is usually available for null space control, we can use this information also for time scaling. When the gradient is large, then the velocity shall be reduced while the robot shall resume to nominal speed when the gradient is close to zero (extremum). A similar approach can be found in [12]. Using ∇H as a measure may not reflect the actual null space capabilities, since only local constrained minimization of H is possible. Instead, we consider the weighted gradient [23]

$$\boldsymbol{h}_n = (\boldsymbol{Z}\boldsymbol{W}\boldsymbol{Z}^{\mathsf{T}})^{-1}\boldsymbol{Z}\,(\nabla H)^{\mathsf{T}}\,, \tag{11}$$

where $\boldsymbol{h}_n \in \mathbb{R}^{(n-m)}$ can be regarded as the gradient of H projected into minimal null space coordinates. In the sequel, it will be referred to as the minimal gradient. The error in performance criterion can now be defined as the magnitude of the minimal gradient, i. e.,

$$e_H = |\boldsymbol{h}_n|\,. \tag{12}$$

[2]We assume that the Jacobian matrix is non-singular.

2) Gradient & Hessian: The problem with using $|\boldsymbol{h}_n|$ as a null space performance measure is that its magnitude is equally low near a constrained local maximum and minimum. Accordingly, the minimal gradient on its own is a rather insufficient definition of the performance error, unless augmented by information about the kind of extremum (minimum/maximum), i.e., the second-order derivative. Numerically, the derivative of the gradient can be obtained by

$$h'_n = \frac{H(\boldsymbol{q}_-) - 2H(\boldsymbol{q}) + H(\boldsymbol{q}_+)}{|\boldsymbol{q}_+ - \boldsymbol{q}_-|^2}\,. \tag{13}$$

Here, the configurations $\boldsymbol{q}_-$ and $\boldsymbol{q}_+$ in the direction of the negative and positive gradient in the vicinity of the current robot configuration $\boldsymbol{q}$ can be obtained with an Euler integration step

$$\boldsymbol{q}_- = \boldsymbol{q} - \boldsymbol{Z}^T\boldsymbol{h}_n\Delta t\,, \tag{14}$$

$$\boldsymbol{q}_+ = \boldsymbol{q} + \boldsymbol{Z}^T\boldsymbol{h}_n\Delta t\,, \tag{15}$$

where Δt is a small step size. Having determined h'_n, we redesign α as

$$\alpha = \begin{cases} \alpha_{\min} & \text{if } h'_n < 0, \\ 1 - (K_H e_H + K_{\dot{H}} e_{\dot{H}}) & \text{otherwise}\,, \end{cases} \tag{16}$$

which means that the robot speed is limited to the minimum possible velocity if the current robot configuration is near a constrained local maximum in terms of the objective.

3) Difference of Current H and Desired H_d: Since the range of the minimal gradient can be large, it may be difficult to tune the gain K_H in (5). Furthermore, the gradient provides no clear information about the error in H itself. A more intuitive formulation of the null space performance error is thus given by the difference

$$e_H = H - H_d\,, \tag{17}$$

of the current value H and a desired value H_d. When selecting H_d the kinematic self-motion capabilities of the robot should be taken into account. If H_d is selected arbitrarily, then it may occur that this goal is not reachable. If $\alpha = 0$ is allowed for, then it is possible that the robot stops its motion entirely because e_H does not converge to zero. In this situation, task scaling or task transition schemes can be employed to relax constraints on the main task and resume operation. If it is required that the objective fulfills $H \leq H_d$, then it should be considered to reformulate the task as a constraint [24].

To ensure that H_d is feasible, it should be located on a reachable part of the self-motion manifold. For local null space optimization schemes, H_d may be defined as the next local minimum in the direction of the negative gradient, which can be found by iteratively integrating the null space velocity $\dot{\boldsymbol{q}}_n = -\boldsymbol{Z}^{\mathsf{T}}\boldsymbol{h}_n$. In the optimization literature, many schemes exist to find such a minimum, e. g., via gradient descent. For the LWR this was done in [25], for example. Having determined a feasible H_d, (17) represents a meaningful error in terms of the performance criterion. Accordingly, the parameterization of the time scaling scheme should be more intuitive than for the previous schemes.

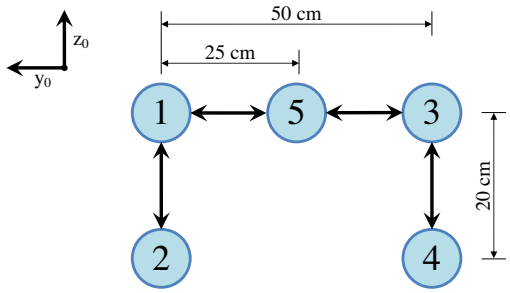

Fig. 2. Schematic of the pick and place task trajectory in the y_0/z_0 plane.

IV. Experiments

In order to verify the performance of the proposed time scaling schemes, we conducted experiments with a 7-DOF DLR/KUKA Lightweight Robot IV. The robot shall perform a pick and place task, the desired Cartesian end-effector trajectory is illustrated in Fig. 2. The motion sequence is $1 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 5$, the desired robot speed is $0.8\,\mathrm{m/s}$. We consider a Cartesian $SE(3)$ task, thus the robot has one redundant degree of freedom. The primary robot controller is a Cartesian impedance controller [26], the null space controller shall minimize the so-called reflected robot mass $H = m_u(\boldsymbol{q})$, i. e., the mass perceived at the end-effector in a certain Cartesian direction $\boldsymbol{u}$ [20]. The controller was implemented according to [25] and shall improve the collision safety of the robot. We first consider a nominal motion without null space scheme or time scaling. Then, we activate the null space controller, again without time scaling. Finally, we use the null space scheme in combination with all proposed time scaling methods. For all schemes we select $\alpha_{\min} = 0.3$ in order to allow for a significant velocity decrease but avoid for a (temporary) complete stop of the system.

The motion of the robot is shown in the attached video, the experimental results are illustrated in Fig. 3. For sake of brevity, we only depict the results for three motion segments, namely $1 \rightarrow 2$ (left column), $1 \rightarrow 3$ (middle column), and $3 \rightarrow 4$ (right column). The analysis for the other motion segments is similar. In the top row, we illustrate the reflected mass over the Cartesian end-effector position, which is represented in terms of the path parameter $s \in [0, 1]$. For motions $1 \leftrightarrow 2$ and $3 \leftrightarrow 4$, s corresponds to a distance of $25\,\mathrm{cm}$ in z_0-direction, for $1 \leftrightarrow 3$ to $50\,\mathrm{cm}$ in y_0-direction, see Fig. 2. In the middle row, we depict the time scaling factor α over time, and in the bottom row the path parameter over time. This representation allows to determine when and to which extent the robot speed is reduced and how much extra time is required to accomplish the task in comparison to the nominal motion.

The nominal trajectory without null space scheme or time scaling is represented by a black line, null space control without time scaling by a gray line, null space control with time scaling based on gradient-based scheme, gradient & Hessian, and known local minimum H_d by a blue, red, and yellow line. The next local minimum in direction of the negative gradient of H is represented by a green line.

Regarding the nominal motion (top row in Fig. 3) one can observe that the reflected mass is generally higher[3] when compared to the other schemes where a null space controller is used. However, during motion $1 \rightarrow 3$ the nominal trajectory reaches a local minimum in reflected mass (at $s \approx 0.4$) by coincidence. When the null space controller is used without any time scaling scheme, the reflected mass is increasing for the largest portion of the motion in all motion segments. This is due to the competing dynamics between the primary and the null space task. Given the relatively high velocity of the end-effector, the magnitude of $\dot{H}_x$ is typically larger than $\dot{H}_{\mathrm{ns}}$. Due to the limited null space dynamics, the minimization of the reflected mass mainly takes place at the end of a motion segment, where the velocity of the end-effector becomes zero.

The introduction of a time scaling scheme clearly improves the performance of the null space scheme, as the robot can reach a lower reflected mass over the course of the trajectory. For the considered problem and trajectory, the time scaling law based on (17) has a relatively better ability to maintain the reflected mass closer to the minimum than the other schemes. The performance of the gradient-based and gradient & Hessian schemes are identical for all motion segments except for segment $3 \rightarrow 4$[4]. The initial configuration for this segment is close to a local maximum in reflected mass and therefore, the gradient itself does not give a proper indication about the error in reflected mass.

For motion $1 \rightarrow 2$, the reflected mass is very close to the desired value (note the range of the y-axis in the top left figure) for all considered control schemes. This means that time scaling is only active for a short time period, otherwise the motion and final time remain unchanged, see the timely evolution of α and s in the first column. For the motion segments $1 \rightarrow 3$ and $3 \rightarrow 4$ the velocity is being reduced for a larger time period, which leads to a significant decrease of the reflected mass, but in turn also to an increase of the cycle time.

A. Discussion

Our experimental results confirm that time scaling can be a simple, yet effective method to improve the performance of auxiliary tasks. The influence of the time scaling scheme on the operational velocity depends on several factors. If the dynamics of the main task are much slower than those of the auxiliary task, then the time scaling scheme will be inactive most of the time. If the main task is much faster than the auxiliary task, then the error in the performance index may lead to a significant velocity reduction for a large part of the trajectory. However, being able to speed up means that the method is generally superior to only scaling the robot velocity by a constant factor for the entire trajectory.

The value of the minimum allowable scaling factor $\alpha_{\min}$ also influences the behavior of the system. Setting $\alpha_{\min}$ to very low values may result in a "stop-and-go" like motion, which can make the motion appear discontinuous. High values of $\alpha_{\min}$ yield that even when the end-effector is slowed down to the minimum admissible velocity, the dynamics of the primary task may still be faster than the

[3]For motion $1 \rightarrow 2$ the reflected mass is not illustrated because it is above the considered range.

[4]In the top right plot in Fig. 3 (motion segment $3 \rightarrow 4$), s is below zero at the beginning of the motion. Here, the robot moves < 2 mm in opposite direction due to imperfect null space projection and/or controller overshoot.

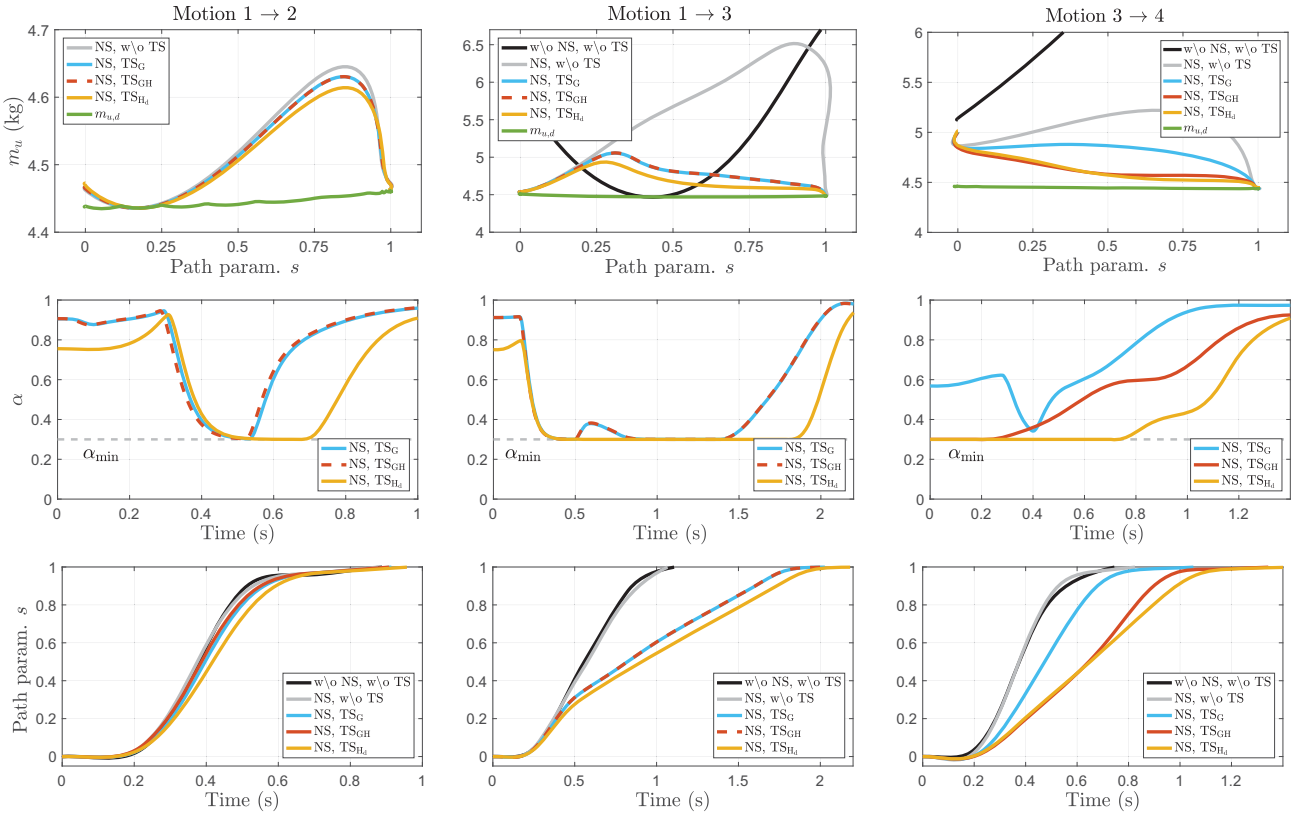

Fig. 3. Experimental results for a pick and place task using an LWR. The left column represents the motion $1 \rightarrow 2$, the middle column the motion $1 \rightarrow 3$, and the right column the motion $3 \rightarrow 4$, see Fig. 2. In the upper row, we depict reflected mass over the path parameter s, in the middle row the time scaling factor α over time, and in the lower row the path parameter over time. Legend: w\o NS, w\o TS: nominal trajectory without null space control or time scaling (black); NS, w\o TS: null space control without time scaling (gray); NS, $\{\mathrm{TS_G}, \mathrm{TS_{GH}}, \mathrm{TS_{H_d}}\}$: Null space control with gradient-based scheme, gradient & Hessian, and known local minimum H_d, respectively (blue, red, yellow); $m_{u,d}$: desired local minimum in reflected mass (green).

null space dynamics, which leads to a deterioration of the null space performance.

V. Extension to Multiple Tasks

In this section, we extend our concept to an arbitrary number of auxiliary tasks. Given a task hierarchy with r priority levels, each task $\boldsymbol{x}_i(\boldsymbol{q}) \in \mathbb{R}^{m_i \times n}$ is defined by the mapping $\boldsymbol{x}_i = \boldsymbol{f}_{k,i}(\boldsymbol{q})$ on a kinematic level and the mapping $\dot{\boldsymbol{x}}_i = \boldsymbol{J}_i(\boldsymbol{q})\dot{\boldsymbol{q}}$ on a differential level with $\boldsymbol{J}_i(\boldsymbol{q}) = \frac{\partial \boldsymbol{f}_{k,i}(\boldsymbol{q})}{\partial \boldsymbol{q}}$. Furthermore, each task consists of minimizing the performance criterion $H_i(\boldsymbol{x}_i)$ locally[5]. As before, the primary task is an end-effector positioning task, where the desired trajectory provided an interpolator is parameterized w.r.t. time. The problem is to design the time scaling factor such that the primary and the auxiliary tasks are synchronized, while taking the task priority and performance of all $r-1$ null space tasks into account.

A. Scaling Factor Design

The time scaling factor α is determined in two steps. First, we design $\alpha_i, i = 2, \ldots, r$ for each task in the hierarchy independently from the other tasks. Then we combine the scaling factors of each level to the overall time scaling factor. For each task, we consider the time scaling law

$$\alpha_i = 1 - \left(K_{H,i} e_{H,i} + K_{\dot{H},i} e_{\dot{H},i}\right), \tag{18}$$

[5]In the sequel we will again omit the dependency of $\boldsymbol{q}$ and $\boldsymbol{x}$.

where $K_{H,i}$ and $K_{\dot{H},i}$ are positive scalars. The problem is now how to define $e_{H,i}$ and $e_{\dot{H},i}$ in a hierarchy consistent way. The original task space velocities $\dot{\boldsymbol{x}}_i$ comprise couplings between the hierarchy levels and can thus not be used for our problem. Therefore, we consider the local null space velocities $\boldsymbol{v}_i \in \mathbb{R}^{m_i}$, introduced in [21], [27], which are given by

$$\underbrace{\begin{pmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_r \end{pmatrix}}_{\boldsymbol{v}} = \underbrace{\begin{pmatrix} \bar{\boldsymbol{J}}_1 \\ \vdots \\ \bar{\boldsymbol{J}}_r \end{pmatrix}}_{\bar{\boldsymbol{J}}} \dot{\boldsymbol{q}}\,. \tag{19}$$

Here, $\bar{\boldsymbol{J}}$ is the hierarchy consistent Jacobian matrix [27]. The time derivative of H_i due to the influence of the primary task and the control action on level i are given by

$$\dot{H}_{i,1} = \frac{\partial H_i}{\partial \boldsymbol{x}_i} \boldsymbol{J}_i \bar{\boldsymbol{J}}_1^{\boldsymbol{W}+} \dot{\boldsymbol{x}}_1\,, \tag{20a}$$

$$\dot{H}_{i,i} = \frac{\partial H_i}{\partial \boldsymbol{x}_i} \boldsymbol{J}_i \bar{\boldsymbol{J}}_i^{\boldsymbol{W}+} \boldsymbol{v}_i\,. \tag{20b}$$

We can now systematically define the synchronization error for each level as

$$e_{\dot{H},i} = \begin{cases} \dot{H}_{i,1} - \dot{H}_{i,i}, & \text{if } \operatorname{sign}(\dot{H}_{i,1}) \neq \operatorname{sign}(\dot{H}_{i,i}) \\ & \wedge |\dot{H}_{i,1}| > |\dot{H}_{i,i}|\,, \\ 0, & \text{otherwise}\,, \end{cases} \tag{21}$$

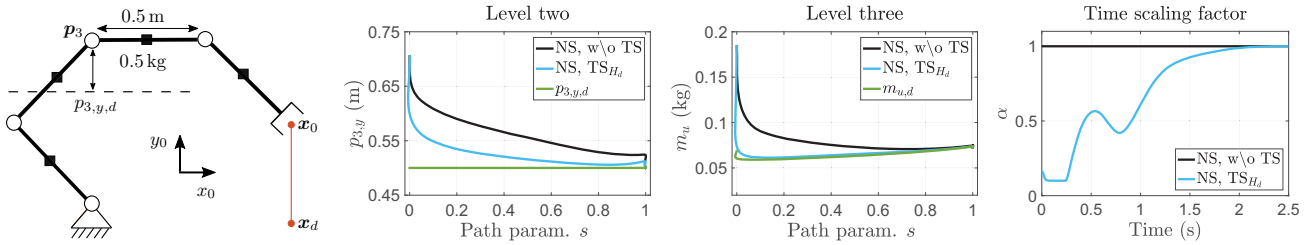

Fig. 4. Time scaling in combination with hierarchical null space control: Simulation results for a planar 4R robot. The robot is illustrated in the left figure, where the vertical position of the third joint is $\boldsymbol{p}_3 = [p_{3,x}, p_{3,y}]^\mathsf{T}$, the initial and desired Cartesian position are $\boldsymbol{x}_0$ and $\boldsymbol{x}_d$, respectively. In the second and third figure, the results for level two (vertical position of third joint) and level three (minimization of reflected mass) are depicted. The evolution of the time scaling factor is shown in the right figure. Legend: NS, w\o TS: nominal motion with null space control but without time scaling; NS, TS$_{\mathrm{H_d}}$: null space control with time scaling; $p_{3,y,d}, m_{u,d}$: Desired vertical position of second joint and reflected mass.

which means that $e_{\dot{H},i}$ is only relevant when the primary task has a negative influence on the null space task on level i. The error $e_{H,i}$ in performance criterion can be formulated in terms of any of the schemes proposed in Sec. III-D. Here, we define the error as $e_{H,i} = H_i - H_{i,d}$ with $H_{i,d}$ being a constrained local minimum for task $\boldsymbol{x}_i$. For determining $H_{i,d}$, we define the minimal gradient on each level as

$$\boldsymbol{h}_{n,i} = \bar{\boldsymbol{J}}_i \boldsymbol{W}^{-1} \boldsymbol{J}_i^\mathsf{T} \left(\frac{\partial H_i(\boldsymbol{x}_i)}{\partial \boldsymbol{x}_i} \right)^\mathsf{T}, \tag{22}$$

where the gradient of the objective (possibly defined in task space) is first mapped to configuration space using $\boldsymbol{J}_i^\mathsf{T}$, then $\boldsymbol{W}^{-1}$ compensates for the rotation of the gradient included in $\bar{\boldsymbol{J}}_i$ [28]. Finally, the gradient is mapped to local null space coordinates via $\bar{\boldsymbol{J}}_i$. The null space velocity in the direction of the gradient can be obtained by $\dot{\boldsymbol{q}}_{n,i} = \bar{\boldsymbol{J}}_i^{\boldsymbol{W}+} \boldsymbol{h}_{n,i}$. Similar to the case of only one degree of redundancy, we can now repeatedly integrate $\dot{\boldsymbol{q}}_{n,i}$ on all levels independently until the next constrained local minimum $H_{i,d}$ is found.

Having determined α_i on all levels, we now want to combine them to the overall time scaling factor. One solution is to select the most conservative α_i of all hierarchy levels, i. e.,

$$\alpha = \min(\alpha_i), \quad i = 2, \ldots, r\,. \tag{23}$$

Alternatively, each α_i can be weighted and combined as

$$\alpha = \sum_{i=2}^{r} w_i \alpha_i\,. \tag{24}$$

Here, w_i are non-negative weights which are selected such that $\sum_{i=2}^{r} w_i = 1$ and $w_i > w_j, \forall i < j$. This means that a high priority task influences the robot speed more than a low priority task.

B. Simulation Results

In order to verify the hierarchical time scaling approach, we carried out simulations with the 4R DOF planar robot shown in Fig. 4 (left). Each link has $0.5\,\mathrm{m}$ length and a $0.5\,\mathrm{kg}$ point mass located at the center. The initial configuration is $\boldsymbol{q}_0 = [135°, -90°, -45°, -45°]^\mathsf{T}$. The primary task is an X/Y linear Cartesian motion of the end-effector of $40\,\mathrm{cm}$ distance in negative y_0-direction. For the considered task, we have two redundant DOF. The secondary task is to minimize the vertical distance $p_{3,y}$, of the third joint to a horizontal plane located at $y_0 = 0.5\,\mathrm{m}$. The tertiary task is the minimization of the reflected mass perceived at the end-effector like in the previous experiment.

The simulation results are illustrated in Fig. 4. Both auxiliary tasks converge to their local minimum ($y_{3,d}$ on level two and $m_{u,d}$ on level three), which verifies that the definition of $e_{H,i}$ is consistent in terms of the task prioritization framework. This ensures that time scaling is only enabled when a potential improvement in the performance index on level i is possible. When using time scaling, all auxiliary tasks achieve better performance and converge faster to the local minimum than without using time scaling.

VI. Conclusion

In this work, we investigated how the trajectory of the robot's primary task can be relaxed by time scaling such that the performance of one or multiple auxiliary null space task(s) in terms of minimizing a performance/safety criterion can be improved. By temporarily limiting the velocity of the main task based on a suitable error in the performance index, the null space task is given more time to achieve its objective. In this work we proposed schemes, which can be implemented with minimal effort and leave the existing primary and null space controller unchanged. The schemes were verified experimentally using a DLR/KUKA Lightweight Robot. The concept was extended to the more general case of multiple prioritized task control and verified in simulation using a 4R planar robot. In summary, time scaling is a simple, yet effective method to improve the task achievement of the auxiliary tasks at the cost of slight primary task relaxation and therefore some extra cycle time. The developed method is an efficient alternative to existing task transition or priority switching schemes, as the geometric description of the main task remains unchanged. However, such schemes may also be combined with time scaling as mentioned in [12]. Future work will consider the stability analysis of the proposed concept.

Acknowledgment

We would like to thank Christian Ott and Alexander Dietrich for helpful discussions. We greatly acknowledge the funding of this work by the Alfried Krupp von Bohlen und Halbach Foundation and by the European Unionon Horizon 2020 research and innovation programme as part of the project ILIAD under grant no. 732737.

REFERENCES

[1] T. Yoshikawa, "Manipulability of robotic mechanisms," *Int. Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.

[2] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[3] R. Rossi, M. P. Polverini, A. M. Zanchettin, and P. Rocco, "A pre-collision control strategy for human-robot interaction based on dissipated energy in potential inelastic impacts," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 26–31.

[4] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *Int. Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.

[5] B. Siciliano and J.-J. E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1991, pp. 1211–1216.

[6] F. Flacco, A. D. Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[7] F. Flacco, A. D. Luca, and O. Khatib, "Prioritized multi-task motion control of redundant robots under hard joint constraints," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 3970–3977.

[8] O. Brock, O. Khatib, and S. Viji, "Task-consistent obstacle avoidance and motion behavior for mobile manipulation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2002, pp. 388–393.

[9] T. Petrič and L. Žlajpah, "Smooth continuous transition between tasks on a kinematic control level: Obstacle avoidance as a control problem," *Robotics and Autonomous Systems*, vol. 61, no. 9, pp. 948 – 959, 2013.

[10] H. Han and J. Park, "Robot control near singularity and joint limit using a continuous task transition algorithm," *Int. Journal of Advanced Robotic Systems*, vol. 10, no. 10, p. 346, 2013.

[11] A. Dietrich, A. Albu-Schäffer, and G. Hirzinger, "On continuous null space projections for torque-based, hierarchical, multi-objective manipulation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 2978–2985.

[12] V. Padois, J.-Y. Fourquet, and P. Chiron, "Kinematic and dynamic model-based control of wheeled mobile manipulators: A unified framework for reactive approaches," *Robotica*, vol. 25, no. 2, pp. 157–173, 2007.

[13] J. M. Hollerbach, "Dynamic scaling of manipulator trajectories," in *American Control Conference*, 1983, pp. 752–756.

[14] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Robotics and Automation*, vol. 6, no. 5, pp. 554–561, 1990.

[15] E. Szadeczky-Kardoss and B. Kiss, "Tracking error based on-line trajectory time scaling," in *IEEE Int. Conf. on Intelligent Engineering Systems*, 2006, pp. 80–85.

[16] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 3356–3363.

[17] C. Wah Wu and L. Chua, "A unified framework for synchronization and control of dynamical systems," *Int. Journal of Bifurcation and Chaos*, vol. 4, no. 4, pp. 979–998, 1994.

[18] I. Blekhman, A. Fradkov, H. Nijmeijer, and A. Pogromsky, "On self-synchronization and controlled synchronization," *Systems And Control Letters*, vol. 31, no. 5, pp. 299 – 305, 1997.

[19] C. Ott, A. Kugi, and Y. Nakamura, "Resolving the problem of non-integrability of nullspace velocities for compliance control of redundant manipulators by using semi-definite lyapunov functions," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 1999–2004.

[20] O. Khatib, "Inertial properties in robotic manipulation: an object-level framework," *Int. Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, 1995.

[21] J. Park, W. Chung, and Y. Youm, "On dynamical decoupling of kinematically redundant manipulators," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 1999, pp. 1495–1500.

[22] M. Z. Huang and H. Varma, "Optimal rate allocation in kinematically-redundant manipulators-the dual projection method," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1991, pp. 702–707.

[23] B. Nemec, L. Zlajpah, and D. Omrcen, "Stability of null-space control algorithms," in *12th International Workshop on Robotics in Alpe-Adria-Danube Region Cassino*, 2002.

[24] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[25] N. Mansfeld, B. Djellab, J. Raldúa Veuthey, F. Beck, C. Ott, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[26] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *Int. Journal of Robotics Research*, vol. 26, no. 1, pp. 23–39, 2007.

[27] A. Dietrich, C. Ott, and A. Albu-Schäffer, "Multi-objective compliance control of redundant manipulators: Hierarchy, control, and stability," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 3043–3050.

[28] J. Park, W. Chung, and Y. Youm, "Unified motion specification and control of kinematically redundant manipulators," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 3945–3951.