Online task-space trajectory planning using real-time estimations of robot motion capabilities

Antun Skuric, Nicolas Torres Alberto, Lucas Joseph, Vincent Padois, David Daney

Online task-space trajectory planning using real-time estimations of robot motion capabilities

Antun Skuric[1], Nicolas Torres Alberto[1,2], Lucas Joseph[1], Vincent Padois[1] and David Daney[1]

***Abstract*—Planning a robot's task-space movement according to its actual capacity is a difficult problem because this capacity depends on its state and thus can evolve significantly during the execution of the movement.**

This paper proposes a method for real-time trajectory planning based on time-optimal Trapezoidal acceleration profile (TAP) trajectories, that adapts to the real-time evolution of the robot's capacity. The method is based on an efficient approach for projecting the robot's kinematic limits in the trajectory direction, based on the convex polytope algebra.

The method is experimentally validated on a Franka Emika Panda collaborative robot and compared with the classical approach considering fixed robot's Cartesian space motion capacity. The results show that the proposed method is able to better exploit true robot's motion capacity by generating faster trajectories for the same level of tracking accuracy.

I. Introduction

When planing a robot's task space point-to-point motion, a common approach in the industry is to use the teach pendant to hand tune the desired motion. This is often performed by specifying the starting and ending pose as well as a certain percentage of the robot's joint or Cartesian limits, preset by the manufacturers [1] [2]. This manual approach has to be repeated for each trajectory of interest in order to find a satisfactory trade-off between robot's speed and the tracking error, which can be time consuming and, in many cases, results in suboptimal trajectories planed without any real information about the robot's true motion capacity.

Automating the optimal trajectory generation for a robot manipulator while taking in consideration its kinematic or kinetodynamic limits is a well studied problem in literature. The approaches considering the robot's kinetodynamic constraints, limiting the actuation torque $\boldsymbol{\tau}$ as well as joint positions $\boldsymbol{q}$ and velocities $\dot{\boldsymbol{q}}$, require precise knowledge of the robot's dynamic model [3] [4]. However, this is not always possible as there are many dynamic effects that are difficult to properly identify and model, such as joint friction. Additionally, as robot dynamics are highly nonlinear, the resolution of these methods commonly relies on nonlinear optimization strategies [5] [6], increasing the solving complexity significantly. In order to avoid using nonlinear dynamical models, many approaches are developed assuming a higher level velocity control loop and thus requiring only the knowledge of the robot's kinematics [7]. These approaches commonly consider the robot's kinematic limits on the joint positions $\boldsymbol{q}$, velocities $\dot{\boldsymbol{q}}$, accelerations $\ddot{\boldsymbol{q}}$ and jerks $\dddot{\boldsymbol{q}}$ to be constant

$$\begin{gathered}\dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min}, \dddot{\boldsymbol{q}}_{max}], \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}], \\ \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}], \boldsymbol{q} \in [\boldsymbol{q}_{min}, \boldsymbol{q}_{max}]\end{gathered} \tag{1}$$

and assume that all the dynamical effects of the robot motion will be perfectly compensated for by the low-level robot control. Such an approach leads to less complex computational models as much simpler strategies can be employed to find optimal trajectories.

When searching for purely kinematic time-optimal or minimum time [8] trajectories, the classical approach are the Trapezoidal velocity profiles (TVP) [9]. TVP finds the time optimal trajectory which respects the constant velocity and acceleration limits. However, for many modern high-performance systems, such as robotic manipulators, additional requirements on the acceleration continuity are necessary, by limiting the acceleration time derivative - *jerk* [10], [11]. The extension of TVP to account for jerk limits is called trapezoidal acceleration profile (TAP) or *S-curve* velocity profile, representing a time optimal trajectory considering constant velocity, acceleration and jerk limits [12] [13]. A TAP profile example is shown on figure 3.

TAP and TVP approaches are traditionally used for planning robot trajectories in the joint space (JS), planning the robot's point-to-point movement from the start to the end robot configuration $\boldsymbol{q}$. JS is convenient for trajectory planning as the robot's kinematic limits (1) are specified in JS, the path between two joint configurations is straightforward to find and the optimal trajectory can be directly used for the robot control, since robots are commonly controlled in JS as well [9]. However, robot's tasks are usually related to the Cartesian space (CS), where

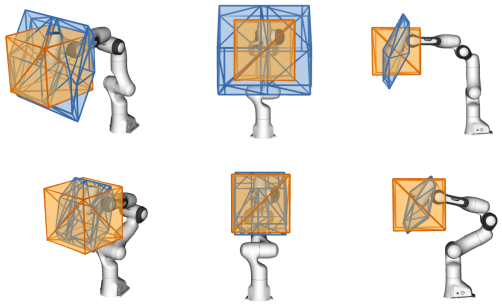

Fig. 1. End-effector linear velocity limits as computed using the fixed Cartesian limits provided by the manufacturer (orange) and using an estimation based on the joint space velocity limits and the current configuration of the robot (blue) (scale: 1 m.s^{-1}/ 10 cm). The true robot's velocity capacity are in some directions higher and in the others significantly lower than the manufacturer's limits, depending on the configuration.

[1]AUCTUS Team, Inria, Talence, France
firstname.lastename@inria.fr
[2]Stellantis, Centre Technique Vélizy, France
nicolas.torres@stellantis.fr

This work is partly funded by the ANRT and Stellantis Group and performed within the framework of the OpenLab AI.

This work is partly supported by BPI France through the LICHIE project in collaboration with Airbus.

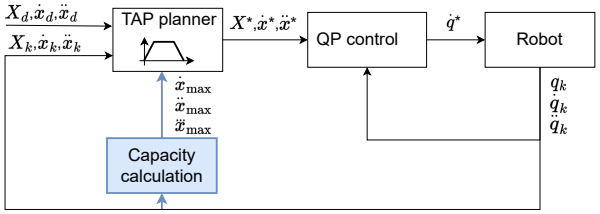

Fig. 2. Proposed method schematic overview, elements in blue present the extension of a standard CS based TAP planning.

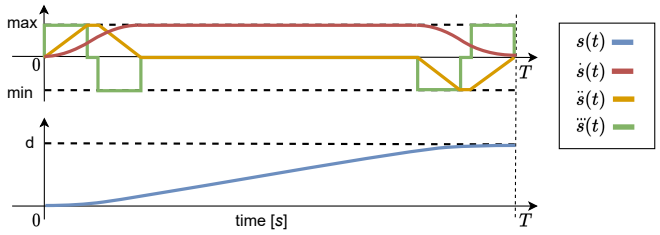

Fig. 3. Trapezoidal acceleration profile example, all the initial and final conditions are set to 0.

the robot is required to go from one pose X_a to another X_b (for example expressed as an homogeneous transform matrix in $SE(3)$) in order to execute a given task. Planning point-to-point robot motion in CS results in robot's movement in straight lines which is more intuitive for human operators [14].

When planning in CS, robot's kinematic limits (1) are no longer constant, they become robot configuration dependant and can potentially change significantly along the trajectory [15] [16], as shown on Figure 1. As most of the trajectory planning approaches consider the limits to be constant, a common approach to planning a CS trajectory is to make sure to plan with the worst-case robot kinematic capacity along the trajectory, usually determined by trial and error by a human operator. However, this method can result in trajectories underestimating the true robot capacities and prevent reaching fastest possible robot motion. Instead of planning the optimal trajectory only once, before the movement is executed, this paper proposes an approach that evaluates the robot's CS limits in real-time and uses them to replan the optimal trajectory, based on the TAP approach, during the trajectory execution. By capturing the real-time evolution of the robot's capacity the method is able to ensure that it is never exceeded and, at the same time, produces trajectories that better exploit the robot's true capacity.

The schematic diagram of the proposed approach within the robot control paradigm is shown on Figure 2. Given the robot's current configuration $\boldsymbol{q}_k$, $\dot{\boldsymbol{q}}_k$, $\ddot{\boldsymbol{q}}_k$ in step k, the approach first determines the robot's current CS velocity $\dot{\boldsymbol{x}}_{max}$, acceleration $\ddot{\boldsymbol{x}}_{max}$ and jerk $\dddot{\boldsymbol{x}}_{max}$ limits. Then it calculates the new optimal trajectory, based on the TAP planning, given the robot's current Cartesian state X_k,$\dot{\boldsymbol{x}}_k$,$\ddot{\boldsymbol{x}}_k$, the desired state X_d,$\dot{\boldsymbol{x}}_d$,$\ddot{\boldsymbol{x}}_d$ and the robot's kinematic limits $\dot{\boldsymbol{x}}_{max}$,$\ddot{\boldsymbol{x}}_{max}$,$\dddot{\boldsymbol{x}}_{max}$. Once the optimal TAP trajectory has been found, target states X_k^*,$\dot{\boldsymbol{x}}_k^*$,$\ddot{\boldsymbol{x}}_k^*$ are sent to the inverse velocity kinematics layer of the control architecture. In the case of this paper, the robot follows the trajectory using a control law formulated as a Quadratic program (QP) which in term finds the optimal desired joint level velocity $\dot{\boldsymbol{q}}^*$ that ensures the robot's trajectory following while respecting all the robot's constraints.

The paper gives an overview of the Cartesian space TAP planning and the description of the real-time replanificaiton strategy in section II. A real-time method for robot's Cartesian kinematic capacity evaluation is described in section III. The experimental setup for assessing the performance of the proposed method on a Franka Emika Panda collaborative robot is given in section IV, while the resutls of the performance analysis are described in section V. Discussion about the limitation of the method can be found in section VI.

II. Trapezoidal acceleration profile basics

The robot's point to point path can be expresses as $P(s)$, where $s \in [0, d]$ is a position on the path with a length d [17], [18]. A TAP trajectory results in a time optimal evolution of position $s(t)$, respecting the starting and end conditions

$$\dot{s}(0) = \dot{s}_{ini} \quad \dot{s}(T) = \dot{s}_{fin} \quad \ddot{s}(0) = \ddot{s}_{ini} \quad \ddot{s}(T) = \ddot{s}_{fin} \quad (2)$$

as well as the limits on all its derivatives:

$$\dot{s} \in [\dot{s}_{min}, \dot{s}_{max}], \ \ddot{s} \in [\ddot{s}_{min}, \ddot{s}_{max}], \ \dddot{s} \in [\dddot{s}_{min}, \dddot{s}_{max}] \quad (3)$$

where T is the trajectory duration, $\dot{s}_{ini}$ and $\ddot{s}_{ini}$ represent the velocity and acceleration along the path $P(s)$ at the beginning of the trajectory, while the $\dot{s}_{fin}$ and $\ddot{s}_{fin}$ represent their final values at the end of the trajectory. One example of the TAP profile is shown on Figure 3.

A. Cartesian space planning

TAP trajectory planned in CS results in time optimal evolution $X(t) = P(s(t))$ of the Cartesian pose $X(t) \in SE(3)$ of the desired robot's frame (ex. end-effector frame) with respect to the frame of interest (ex. inertial or robot base frame). When dealing with the geometric paths in CS, planning the robot's movement form a Cartesian pose X_a to X_b, it is common to separate the translation $P_t(s_t)$ and orientation $P_o(s_o)$ component of the path $P(s)$ [4]. The translation component of the path can then be expressed as:

$$P_t(s_t) = s_t \boldsymbol{u}_t, \quad s_t \in \left[0, ||X_b^t - X_a^t||_2\right] \quad (4)$$

where X_a^t and X_b^t are the translation parts of the poses X_a and X_b and $\boldsymbol{u}_t$ is the unit vector pointing from X_a^t to X_b^t. For the orientation, the common approach is to use the axis-angle representation of the rotation, and specify the difference in orientation between X_a and X_b as an angle θ around the axis $\boldsymbol{u}_o$. Then the geometric path corresponding to the orientation $P_o(s_o)$ can be written as

$$P_o(s_o) = s_o \boldsymbol{u}_o, \quad s_o \in [0, \theta] \quad (5)$$

$P_o(s_o)$ and $P_t(s_t)$ represent a relative change in the orientation and translation over the course of the trajectory from the initial pose X_a. Cartesian poses are commonly defined as homogeneous matrices in SE(3), therefore, the full desired Cartesian pose

$X^*(t)$ can be calculated using an homogeneous transformation matrix Ω, constructed from P_o and P_t

$$X^*(t) = X_a \Omega(P_o(s_o), P_t(s_t)) \tag{6}$$

and optimal Cartesian velocity and acceleration can be found

$$\dot{\boldsymbol{x}}^* = \begin{bmatrix} \dot{s}_t(t)\boldsymbol{u}_t \\ \dot{s}_o(t)\boldsymbol{u}_o \end{bmatrix}, \quad \ddot{\boldsymbol{x}}^* = \begin{bmatrix} \ddot{s}_t(t)\boldsymbol{u}_t \\ \ddot{s}_o(t)\boldsymbol{u}_o \end{bmatrix} \tag{7}$$

Cartesian velocity, acceleration and jerk limits

$$\dddot{\boldsymbol{x}} \in [\,\dddot{\boldsymbol{x}}_{min},\, \dddot{\boldsymbol{x}}_{max}], \ddot{\boldsymbol{x}} \in [\ddot{\boldsymbol{x}}_{min}, \ddot{\boldsymbol{x}}_{max}], \dot{\boldsymbol{x}} \in [\dot{\boldsymbol{x}}_{min}, \dot{\boldsymbol{x}}_{max}] \tag{8}$$

can then be projected onto the path transforming them to limits on the trajectory variables s_o and s_t, using vectors $\boldsymbol{c}_t = [\boldsymbol{u}_t^T, \boldsymbol{0}_{3\times1}]^T$ and $\boldsymbol{c}_o = [\boldsymbol{0}_{3\times1}, \boldsymbol{u}_o^T]^T$

$$\begin{gathered} \dddot{\boldsymbol{s}}_i \in [\boldsymbol{c}_i^T \dddot{\boldsymbol{x}}_{min},\, \boldsymbol{c}_i^T \dddot{\boldsymbol{x}}_{max}], \ddot{\boldsymbol{s}}_i \in [\boldsymbol{c}_i^T \ddot{\boldsymbol{x}}_{min},\, \boldsymbol{c}_i^T \ddot{\boldsymbol{x}}_{max}] \\ \dot{\boldsymbol{s}}_i \in [\boldsymbol{c}_i^T \dot{\boldsymbol{x}}_{min},\, \boldsymbol{c}_i^T \dot{\boldsymbol{x}}_{max}] \end{gathered} \tag{9}$$

where i is either the translation t or the orientation o.

Finally, once both translation and orientation TAP trajectories are found resulting in optimal $s_o(t)$ and $s_t(t)$, to synchronise the two movements, their time duration is matched, making both trajectories last the same time T, the time taken by the longer of the two trajectories $T = \max\{T_o, T_t\}$.

The inherent challenge of planning the robot motion in the Cartesian space is that its Cartesian kinematic limits (8) are configuration dependant, and over the course of the trajectory the robot's Cartesian capacity can change significantly, as shown on the Figure 1. Therefore, no set of fixed Cartesian limits as defined in (8) will be able to exploit the robot's full movement capabilities. Even though some manufacturers do specify the maximal Cartesian kinematic limits in their robot datasheets [2], these values are not representative of the true robot's capabilities. Figure 1 shows the visual comparison of the manufacturer's datasheet limits and the real robot's CS velocity $\dot{\boldsymbol{x}}$ capacity for one configuration of a Franka Emika Panda robot.

B. Real-time replannifying TAP trajectory

In order to account for constantly changing robot's kinematic limits during the trajectory execution, a simple real-time replanifying strategy is proposed. It recurrently calculates a new optimal TAP trajectory based on the robot's current position on the trajectory and its current kinematic limits. In this way, the robot follows the same path from the start to the end pose, but does it slower or faster based on its capacity.

For a given moment in time t_k and robot's position on the trajectory $s_t(t_k), s_o(t_k)$, the remaining length of the geometric path to the target position can be calculated as $l_k = s_t(T) - s_t(t_k)$ and the remaining angle of rotation $\theta_k = s_o(T) - s_o(t_k)$. Then the initial conditions for the new planning execution can be updated

$$\begin{aligned} s_t \in [0, l_k], \quad \dot{s}_{t,ini} = \dot{s}_t(t_k), \quad \ddot{s}_{t,ini} = \ddot{s}_t(t_k), \\ s_o \in [0, \theta_k], \quad \dot{s}_{o,ini} = \dot{s}_o(t_k), \quad \ddot{s}_{o,ini} = \ddot{s}_o(t_k) \end{aligned} \tag{10}$$

Finally, an updated TAP trajectory can then be calculated for the translation $s_t(t)$ and the orientation $s_o(t)$, resulting in an optimal robot trajectory given the updated robot's limits (3) and the initial conditions (10).

III. Real-time robot's capacity evaluation

The mapping between the robot's joint space and Cartesian space velocity, acceleration and jerk for a certain fixed frame of interest (ex. end-effector frame) is nonlinear and dependant on $\boldsymbol{q}$, $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$ and $\{\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}\}$ respectively

$$\begin{aligned} \dot{\boldsymbol{x}} &= J(\boldsymbol{q})\dot{\boldsymbol{q}} \\ \ddot{\boldsymbol{x}} &= J(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \\ \dddot{\boldsymbol{x}} &= J(\boldsymbol{q})\,\dddot{\boldsymbol{q}} + 2\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})\ddot{\boldsymbol{q}} + \ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})\dot{\boldsymbol{q}} \end{aligned} \tag{11}$$

For a certain joint configuration $\boldsymbol{q}$, the joint velocity $\dot{\boldsymbol{q}}$, acceleration $\ddot{\boldsymbol{q}}$ and jerk $\dddot{\boldsymbol{q}}$ are mapped to Cartesian space using the Jacobian matrix $J(\boldsymbol{q})$ and its time derivatives $\dot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\ddot{J}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}})$.

Joint space kinematic limits (1) are expressed in a form of an interval for each of the robot's n joints (degrees of freedom), forming n-dimensional hypercubes. For any given robot state $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k$, CS kinematic limits can be calculated by projecting these JS n-dimensional hypercubes (1) into the m-dimensional CS using the expressions (11). The resulting CS limits will have a form of convex polytopes.

For a certain robot pose $\boldsymbol{q}_k$ the convex polytope $\mathcal{P}_v$ of achievable CS velocities $\dot{\boldsymbol{x}}$ can be expressed as

$$\mathcal{P}_v = \{\dot{\boldsymbol{x}} \in \mathbf{R}^m | \dot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\dot{\boldsymbol{q}} + \boldsymbol{b}_v,\ \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}]\} \tag{12}$$

where $\boldsymbol{b}_v$ is a bias vector $\boldsymbol{b}_v = \boldsymbol{0}_{m\times1}$. The convex polytopes $\mathcal{P}_a$ and $\mathcal{P}_j$ of achievable CS acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ have a form

$$\begin{aligned} \mathcal{P}_a &= \{\ddot{\boldsymbol{x}} \in \mathbf{R}^m | \ddot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\ddot{\boldsymbol{q}} + \boldsymbol{b}_a,\ \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{min}, \ddot{\boldsymbol{q}}_{max}]\} \\ \mathcal{P}_j &= \{\,\dddot{\boldsymbol{x}} \in \mathbf{R}^m | \,\dddot{\boldsymbol{x}} = J(\boldsymbol{q}_k)\,\dddot{\boldsymbol{q}} + \boldsymbol{b}_j,\ \dddot{\boldsymbol{q}} \in [\dddot{\boldsymbol{q}}_{min},\, \dddot{\boldsymbol{q}}_{max}]\} \end{aligned} \tag{13}$$

where $\boldsymbol{b}_a$ and $\boldsymbol{b}_j$ are the bias acceleration and jerk produced by the effect of current joint velocity $\dot{\boldsymbol{q}}_k$ and acceleration $\ddot{\boldsymbol{q}}_k$

$$\begin{aligned} \boldsymbol{b}_a &= \dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \\ \boldsymbol{b}_j &= 2\dot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\ddot{\boldsymbol{q}}_k + \ddot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \end{aligned} \tag{14}$$

Additionally, as the $\boldsymbol{b}_j$ term produced by the second derivative of the Jacobian matrix $\ddot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k$ will produce relatively small effects on final value of $\dddot{\boldsymbol{x}}_b$, in the case of this paper it is neglected $\ddot{J}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k)\dot{\boldsymbol{q}}_k \approx 0$. Finally, the achievable CS velocity, acceleration and jerk, given current robot state $\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k, \ddot{\boldsymbol{q}}_k$ can be expressed as

$$\dot{\boldsymbol{x}} \in \mathcal{P}_v, \quad \ddot{\boldsymbol{x}} \in \mathcal{P}_a, \quad \dddot{\boldsymbol{x}} \in \mathcal{P}_j \tag{15}$$

Polytope definition (12) and (13) is an overestimation of the robot's real capacity as the joint kinematic limits are considered independent, which is not usually the case (ex. reaching $\dot{\boldsymbol{q}}_{max}$ in some configuration might violate joint acceleration $\ddot{\boldsymbol{q}}_{max}$ or jerk $\dddot{\boldsymbol{q}}_{max}$ limits). However, these limits can directly be used for TAP planning, as it inherently accounts for this issue and finds the trajectory respecting all the set constraints at the same time.

A. Finding Cartesian limits in trajectory direction

Given a m-dimensional Cartesian space vector $\boldsymbol{c}$, pointing in the direction of the trajectory, the trajectory $m-1$ dimensional normal space can be found using the singular value decomposition (SVD)

$$\boldsymbol{c} = U\Sigma V^T \quad V = [V_1,\, V_2], \quad V_2 \in \mathbf{R}^{m\times(m-1)} \tag{16}$$

where matrix V_2 represents the projector to the *null-space* of $\boldsymbol{c}$ [19]. Columns $\boldsymbol{n}_i$ of $V_2 = [\boldsymbol{n}_1, \ \dots \ , \boldsymbol{n}_{m-1}]$ are an orthonormal base of the $m-1$ dimensional space normal to the vector $\boldsymbol{c}$, or in other words $V_2^T \boldsymbol{c} = \boldsymbol{0}$

With the known trajectory direction $\boldsymbol{c}$ and the trajectory normal space V_2, the maximal value of the Cartesian variable $\boldsymbol{y} \in \mathbf{R}^m$, in the trajectory direction $\boldsymbol{c}$, within its range expressed as a convex polytope $\mathcal{P}_y$, can be found by solving the Linear programming (LP) problem [20]

$$\begin{aligned} &\max_{\boldsymbol{y}} \ c^T \boldsymbol{y} \\ &s.t. \ V_2^T \boldsymbol{y} = \boldsymbol{0}, \\ &\qquad \boldsymbol{y} \in \mathcal{P}_y \end{aligned} \tag{17}$$

whereas the minimum can be found by minimising it [21].

By substituting the generic Cartesian variable $\boldsymbol{y}$ with Cartesian velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ and its polytope $\mathcal{P}_y$ with their respective polytopic limits (15), LP formulation (17) can be directly used to calculate the limits of the velocity $\dot{s}$, acceleration $\ddot{s}$ and jerk $\dddot{s}$ in the trajectory direction $\boldsymbol{c}$.

When searching for maximal Cartesian velocity $\dot{s}_{max}$, assuming known robot configuration $\boldsymbol{q}_k$ and trajectory direction vector $\boldsymbol{c}$, the Cartesian velocity along the trajectory $\dot{s}$ can be calculated as a projection of the Cartesian velocity $\dot{\boldsymbol{x}}$

$$\dot{s} = \boldsymbol{c}^T \dot{\boldsymbol{x}} = \boldsymbol{c}^T J(\boldsymbol{q}_k) \dot{\boldsymbol{q}} = \boldsymbol{c}^T J(\boldsymbol{q}_k) \dot{\boldsymbol{q}} + \boldsymbol{c}^T \boldsymbol{b}_v \tag{18}$$

where $\boldsymbol{b}_v$ is zero vector. Substituting this relationship into the LP formulation (17) yields

$$\begin{aligned} \dot{s}_{max} = &\max_{\dot{\boldsymbol{q}}} \boldsymbol{c}^T J(\boldsymbol{q}_k) \dot{\boldsymbol{q}} + \boldsymbol{c}^T \boldsymbol{b}_v \\ &s.t. \quad V_2^T J(\boldsymbol{q}_k) \dot{\boldsymbol{q}} = -V_2^T \boldsymbol{b}_v, \\ &\qquad \dot{\boldsymbol{q}} \in [\dot{\boldsymbol{q}}_{min}, \dot{\boldsymbol{q}}_{max}] \end{aligned} \tag{19}$$

The equivalent LP expressions for finding the maximal acceleration $\ddot{s}$ and jerk $\dddot{s}$ are obtained by substituting $\dot{\boldsymbol{q}}$ and $\boldsymbol{b}_v$ with $\ddot{\boldsymbol{q}}$, $\boldsymbol{b}_a$ and $\dddot{\boldsymbol{q}}$, $\boldsymbol{b}_j$. The minimal values are found by minimising the same problem.

B. Scaling robot's Cartesian limits

When it comes to planning robot trajectories, it is a common practice to consider only a part (certain percentage) of the specified robot limits. This is partially due to the fact that more dynamic robot movements require higher actuation capacity and leave less margin to account for potential tracking error, in many cases resulting in impaired tracking performance. Additionally, robot movement with higher velocity is associated with longer stepping time and higher potential impact momentum [22], therefore scaling strategies can be used to adapt robot's velocity in order to improve the safety.

If the robot's CS kinematic limits are considered constant and if they are specified as (8), the simplest form of scaling can be done by multiplying the specified limits with scalar factors $\alpha_v, \alpha_a, \alpha_j \in [0, 1]$.

$$\begin{aligned} \dot{\boldsymbol{x}} \in [\alpha_v \dot{\boldsymbol{x}}_{min}, \alpha_v \dot{\boldsymbol{x}}_{max}], \ \ddot{\boldsymbol{x}} \in [\alpha_a \ddot{\boldsymbol{x}}_{min}, \alpha_a \ddot{\boldsymbol{x}}_{max}], \\ \dddot{\boldsymbol{x}} \in [\alpha_j \dddot{\boldsymbol{x}}_{min}, \alpha_j \dddot{\boldsymbol{x}}_{max}] \end{aligned} \tag{20}$$

allowing the robots velocity $\dot{\boldsymbol{x}}$, acceleration $\ddot{\boldsymbol{x}}$ and jerk $\dddot{\boldsymbol{x}}$ not to exceed certain percentage of the specified limits.

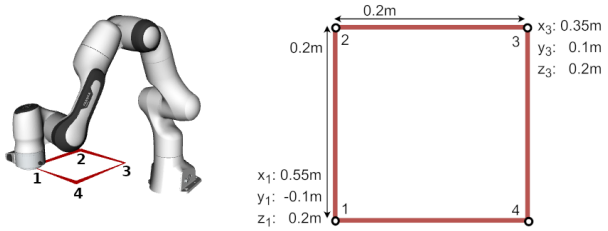

Fig. 4. Benchmarking trajectory in the robot's workspace in a form of 20cm×20cm square placed in the robots ISO cube. The path is defined in the robot's base frame.

However, if the robot's CS kinematic capacity is not considered constant but a result of the robot's current configuration $\boldsymbol{q}$ and its JS kinematic limits, then the scaling using the scalars $\alpha_v, \alpha_a, \alpha_j \in [0, 1]$ can done in the joint space

$$\begin{aligned} \dot{\boldsymbol{q}} \in [\alpha_v \dot{\boldsymbol{q}}_{min}, \alpha_v \dot{\boldsymbol{q}}_{max}], \ddot{\boldsymbol{q}} \in [\alpha_a \ddot{\boldsymbol{q}}_{min}, \alpha_a \ddot{\boldsymbol{q}}_{max}], \\ \dddot{\boldsymbol{q}} \in [\alpha_j \dddot{\boldsymbol{q}}_{min}, \alpha_j \dddot{\boldsymbol{q}}_{max}] \end{aligned} \tag{21}$$

These new modulated JS limits can then be used to calculate the polytopes $\mathcal{P}_v$,$\mathcal{P}_a$ and $\mathcal{P}_j$ using equations (12-13), and in term scale the resulting Cartesian robot limits (15).

IV. Experimental setup

This section presents a benchmarking experiment designed to quantify the performance benefits of the proposed trajectory planning method capable of accounting for the real-time evolution of the robot's capacity. The proposed method's execution time and tracking error performance are compared to the standard off-line approach for trajectory generation considering fixed robot's CS kinematic limits.

The experiments are conducted on a Franka Emika Panda robot and all the robot's kinematic limits, both in joint space and the Cartesian space are obtained from Franka Emika's official datasheet [2].

A. Benchmarking experiment

The proposed adaptive approach is compared to the standard TAP trajectory planning using fixed Cartesian kinematic limits from the standard datasheet

$$\begin{aligned} &|\dot{\boldsymbol{x}}_t| \leq 1.7 m/s, \ |\ddot{\boldsymbol{x}}_t| \leq 13 m/s^2, \ |\dddot{\boldsymbol{x}}_t| \leq 6500 m/s^3, \\ &|\dot{\boldsymbol{x}}_o| \leq 2.5 s^{-1}, \ \ |\ddot{\boldsymbol{x}}_o| \leq 25 s^{-2}, \quad |\dddot{\boldsymbol{x}}_o| \leq 12500 s^{-3} \end{aligned} \tag{22}$$

where $\dot{\boldsymbol{x}}_t$,$\ddot{\boldsymbol{x}}_t$ and $\dddot{\boldsymbol{x}}_t$ are CS translation velocity, acceleration and jerk, whereas $\dot{\boldsymbol{x}}_o$,$\ddot{\boldsymbol{x}}_o$ and $\dddot{\boldsymbol{x}}_o$ are these variables corresponding the the CS orientation.

The geometric path chosen for the experiments is a horizontal 20cm×20cm square path in the ISO cube [23] of the robotic manipulator. The path is defined in the base frame of the robot and is placed at the height of 20cm from the tabletop, placed in the x-y plane. In each experiment the path is executed 5 times.

The performance of the two approaches is compared for different scaling levels α, starting at 10% (α=0.1) of robot's capacity and going to either 100% or until the maximal tracking position error exceeds 1cm. The scaling strategy is chosen to be equal for velocity, acceleration and jerk $\alpha = \alpha_v = \alpha_a = \alpha_j$.

The implementation of the TAP trajectory generator for both approaches is done using the open-source library ruckig [13], while the efficient LP solver used for real-time CS limit calculation is GLPK [24].

B. Robot control architecture

As shown on the schema from the figure 2, robot control strategy for real-time Cartesian trajectory following is formulated as a Quadratic Program (QP) and solved in each control loop. As a secondary (regularization) task of the QP, the robot's redundant degrees of freedom are used to dampen the movement in the trajectory *null-space* and keep the robot away from its joint limits.

$$\begin{aligned} \ddot{\boldsymbol{q}}_{opt} = \arg\max_{\ddot{\boldsymbol{q}}} \quad & ||\dot{\boldsymbol{v}} - J_k\ddot{\boldsymbol{q}} - \dot{J}_k\dot{\boldsymbol{q}}||^2 + \omega_r||\ddot{\boldsymbol{q}}_r - \ddot{\boldsymbol{q}}||^2 \\ \text{s.t.} \quad & \ddot{\boldsymbol{q}} \in [\ddot{\boldsymbol{q}}_{ub},\ \ddot{\boldsymbol{q}}_{lb}] \end{aligned} \tag{23}$$

where the trajectory following control law is formulated as a PD controller with a feed-forward term through the desired Cartesian acceleration $\dot{\boldsymbol{v}}$

$$\begin{aligned} \dot{\boldsymbol{v}} &= K_p\boldsymbol{e} + K_d(\dot{\boldsymbol{x}}^*_{k+1} - \dot{\boldsymbol{x}}_k) + \ddot{\boldsymbol{x}}^*_{k+1} \\ \boldsymbol{e} &= Adj(X_k)\log(X_k^{-1}X^*_{,k+1}) \end{aligned} \tag{24}$$

X^*_{k+1},$\dot{\boldsymbol{x}}^*_{k+1}$,$\ddot{\boldsymbol{x}}^*_{k+1}$ are the desired Cartesian pose, velocity and acceleration in the next step $k+1$, X_k and $\dot{\boldsymbol{x}}_k$ are the measured Cartesian pose and velocity in current step k. $K_p, K_d \in \mathbf{R}^m$ are diagonal matrices containing the proportional and derivative gains. Vector $\boldsymbol{e}$ is the Cartesian pose error expressed in the world frame. The regularization task is expressed through the joint acceleration $\ddot{\boldsymbol{q}}_r$

$$\ddot{\boldsymbol{q}}_r = k_{rp}(\boldsymbol{q}_r - \boldsymbol{q}) - k_{rd}\dot{\boldsymbol{q}} \tag{25}$$

where k_{rp} and k_{rd} are scalar gains and $\boldsymbol{q}_r$ is the initial robot pose close to the center of all the joint ranges. The bounds of each of joint accelerations $\ddot{q}_{i,ub}$, $\ddot{q}_{i,lb}$ are calculated in a way to guarantee that the joint jerk $\dddot{\boldsymbol{q}}$, velocity $\dot{\boldsymbol{q}}$ and position $\boldsymbol{q}$ in the horizon δt respect their limits.

$$\begin{aligned} \ddot{q}_{i,ub} = \min\Big\{ & \ddot{q}_{i,k} + \dddot{q}_{i,max}\delta t, \ddot{q}_{i,max},\ .. \\ & \frac{1}{\delta t}(\dot{q}_{i,max} - \dot{q}_{i,k}), \frac{2}{\delta t^2}(q_{i,max} - q_{i,k} - \dot{q}_{i,k}\delta t)\Big\} \end{aligned} \tag{26}$$

where the horizon δt has to be chosen long enough to ensure constraints compatibility without leading to conservative behaviour [25]. Equation (26) shows the upper bound expression, the lower bound calculation is equivalent, and is obtained by substituting *min* for *max* and maximizing instead of minimizing.

Robot is controlled using the joint velocity commands which are calculated using an Euler backward numerical integration

$$\dot{\boldsymbol{q}}^*_{k+1} = \dot{\boldsymbol{q}}^*_k + \ddot{\boldsymbol{q}}_{opt}\Delta t \tag{27}$$

In the experiments, the PD controller gains used are $K_p = \text{diag}([70.0, 70.0, 70.0, 50.0, 50.0, 50.0])$ and $K_d = \text{diag}([50.0, 50.0, 50.0, 30.0, 30.0, 30.0])$. The secondary task gains used are $k_{rp} = 5s^{-2}$ and $k_{rd} = 2\sqrt{k_{rd}}s^{-1}$, while secondary task weight iss $\omega_r = 1e^{-5}$. The horizon δt is chosen to be 15ms. The robot control architecture is implemented using Robot Operating System (ROS) framework and run in real-time at frequency of 1kHz.

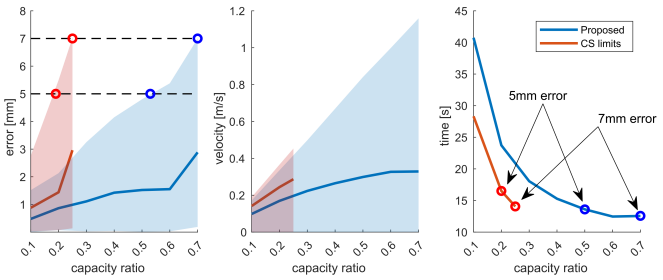

Fig. 5. Result of the benchmarking experiment comparing the fixed CS planning approach (red) to the proposed method (blue). The graph on the left shows the evolution of the tracking error with respect to the ratio of the capacity α used. The dotted lines show the 5mm and 7mm error for easier visibility. The middle graph shows the average velocity during the trajectory execution, while the right graph shows the execution time of the methods. The red and blue circles show the moments in time where each of the approaches reached 5mm and 7mm maximal tracking error. On the first two graphs, the solid lines show mean value evolution, while the evolution of the area between minimal and maximal values is shown in shaded colors.

V. Results

The results of the benchmarking experiment are shown on figure 5. In the experiments, the fixed Cartesian limits approach surpassed the maximum following error level of 1cm at $\alpha = 0.3$ or 30% of robot's CS capacity, while the proposed approach surpassed this error level at $\alpha = 0.8$ or 80% percent of the robot's JS capacity. Only the valid runs, with the maximum error under 1cm, are shown on the graphs. For the sake of conciseness, focus is placed here on the position tracking error only but a similar analysis can be performed at the orientation level. This error is calculated as $\boldsymbol{e}_k = ||(X^*_k)^t - X^t_k||_2$.

From the error graph on the left of Figure 5, it can be seen that the proposed approach has much lower error for the same values of scaling factor α. Furthermore the fixed CS approach reached the maximal error of 5mm and 7mm at $\alpha = 0.2$ and $\alpha = 0.25$ while the proposed approach reached it at $\alpha = 0.5$ and $\alpha = 0.7$ respectively.

The middle graph of Figure 5 shows that the two approaches have very similar maximum velocity for the same values of α, while the mean velocity value is significantly higher for the fixed CS approach. This is due to the fact that this approach considers the same kinematic capacity in all directions, producing the trajectory with constant CS velocity, while the proposed approach will result with different velocities in different directions. However, it can be seen that the proposed approach is able to reach much higher velocities (1.1m/s) than the fixed CS approach (0.45m/s), before reaching the 1cm tracking error.

From the trajectory execution time graph, right image of Figure 5, it can be concluded that the proposed approach is able to execute the trajectory faster than the fixed CS approach for the same levels of accuracy. The fixed CS approach reached the maximum error of 5mm with the time 16.5s and the proposed approach with the time of 13.6s. Furthermore, the fastest run of the fixed CS approach is 14.5s at $\alpha = 0.25$, while the proposed approach's fastest run is at 12.5s at $\alpha = 0.7$, both having maximum error of 7mm.

Figure 6, shows the normalised time evolution of the x-axis

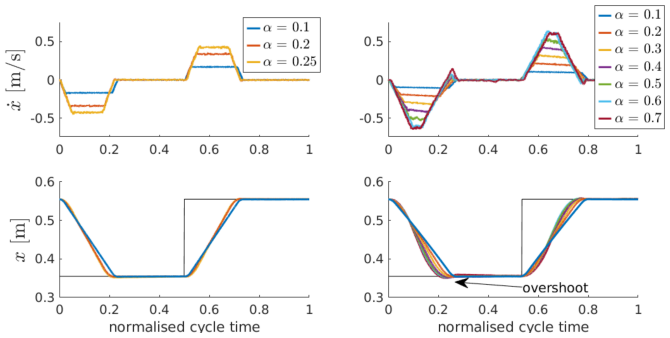

Fig. 6. Recorded evolution of the x-axis position and velocity $\dot{x}$ over the course of one path cycle for different values of capacity scaling factor α. Scaled fixed CS limits approach is shown on the left and the proposed approach on the right. All the plots have been normalised in time of the one cycle.

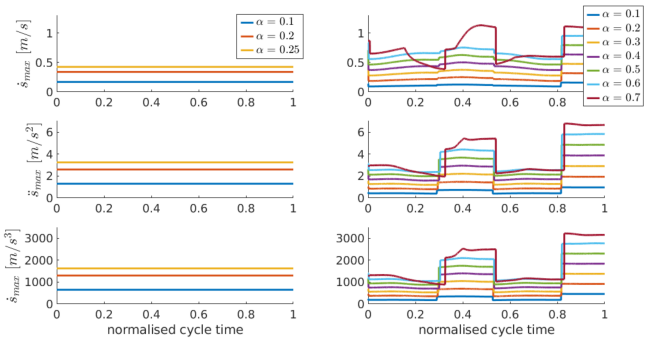

Fig. 7. The evolution of the jerk $\dddot{s}_{max}$, acceleration $\ddot{s}_{max}$ and velocity $\dot{s}_{max}$ upper limits over the course of one path cycle for different values of capacity scaling factor α. Scaled fixed CS limits approach is shown on the left and the proposed approach on the right. All the plots have been normalised with respect to the considered cycle time.

position and velocity, of the robot, recorded in the experiments. The figure confirms that the maximal velocity reached by the robot is higher for the proposed method ($\dot{x}$=0.63m/s) than the fixed CS limits approach ($\dot{x}$=0.43m/s). However, the figure shows that in the experiments, for scaling values $\alpha \geq 0.5$, the robot's position x has an overshoot. This overshoot is a result of the real-time updating of the robot's limits rather than of lower-level control tracking issues. Overshoot can happen if the robot's kinematic capacity drops significantly towards the end of the trajectory, where the robot is no longer able to stop in time given its diminishing capacity. This behavior is an inherent limitation of the proposed approach.

Figure 7, shows the normalised time evolution of the upper bounds on trajectory velocity $\dot{s}_{max}$, acceleration $\ddot{s}_{max}$ and jerk $\dddot{s}_{max}$ during one cycle of the square path. The time evolution of the limits confirms that robot's capacity changes not just in time but with respect to the trajectory direction as well. Additionally it can be seen that, for the same values of α, the fixed CS limits are much higher than the kinematic limits calculated with the proposed approach. It can also be seen that for scaling factor value of $\alpha = 0.7$ the robot's capacity evolution has a significantly different form than for the lower values of α, which is the result of a different evolution of the robot's redundant degrees of freedom.

This experiment is illustrated in the accompanying video, as well as an additional experiment combining the position and orientation motion in the same trajectory.

VI. Discussion

As shown in the result section, the proposed trajectory planning approach has many benefits over the classic fixed CS limits approach. It exploits better robot's true kinematic capacity and generate faster robot trajectories while having comparable accuracy level. However, the proposed approach has several limitations.

The robot's joint space kinematic limits (1) are considered constant in time, which is generally not the case. These limits will depend highly on the robot's actuation limits $\boldsymbol{\tau}$ and different dynamical and gravitational effects acting on the robot, as well of the robot's joint state $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\}$. Therefore, for highly dynamical robot's movements, the available actuation capacity of the robot might reduce certain of the limits (1). The experiments in this paper show that, for the trajectory in question, robot is not able to follow the trajectories planned with more than 70% of its joint space kinematic limits. However, this value might change for different areas of robot's workspace and for different trajectories. Therefore, the integration of the robot's actuation limits could make the proposed approach more robust and it is a promising direction for future research.

Furthermore, as shown on Figure 6, the proposed approach, in some cases, results in an overshoot in robot's position due to the diminishing robot's kinematic capacity towards the end of the trajectory. In order to overcome this issue, instead of considering only instantaneous robot's capacity for each trajectory generation execution, it would be necessary to predict the robot's kinematic capacity along the remaining trajectory. This is a challenging research topic which results might be applied not just in TAP planning techniques but also to the optimal control methods such as *model predictive control*.

VII. Conclusion

This paper proposes a trajectory planning approach for adapting to the real-time evolution of the robot's Cartesian space kinematic capacity along the trajectory. The approach is based on an efficient method for calculating the robot's velocity, acceleration and jerk limits in the trajectory direction, deriving them directly form the robot's kinematic joint limits using the convex polytope algebra. Based on the robot's configuration and the desired geometric path, the method first determines its instantaneous kinematics limits in the trajectory direction and uses them, in the real-time, to plan the optimal trajectory based on the time optimal trapezoidal acceleration profiles.

In order to analyse the performance of the proposed method an experiment is conducted using a Franka Emika Panda robot following a square geometric path. The method is compared to the classical approach considering fixed robot's Cartesian space kinematic capacity. The performance analysis has shown that the proposed approach reaches significantly higher velocities and has significantly lower execution times for the same accuracy level, indicating better exploitation of the robot's capacity.

REFERENCES

[1] Universal Robots, *Universal Robots e-Series User Manual*, 2022.

[2] Franka Emika, "Panda robot datasheet - franka control interface." https://frankaemika.github.io/docs/control_parameters.html, 2022.

[3] T. Chettibi, M. Haddad, H. E. Lehtihet, and W. Khalil, "Suboptimal trajectory generation for industrial robots using trapezoidal velocity profiles," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 729–735, 2006.

[4] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-optimal control of robotic manipulators along specified paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[5] N. J. M. van Dijk, N. van de Wouw, H. Nijmeijer, and W. C. M. Pancras, "Path-constrained motion planning for robotics based on kinematic constraints," vol. Volume 8: 31st Mechanisms and Robotics Conference, Parts A and B of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1071–1080, 09 2007.

[6] S. Singh and M. C. Leu, "Optimal Trajectory Generation for Robotic Manipulators Using Dynamic Programming," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, pp. 88–96, 06 1987.

[7] Y. Fang, J. Qi, J. Hu, W. Wang, and Y. Peng, "An approach for jerk-continuous trajectory generation of robotic manipulators with kinematical constraints," *Mechanism and Machine Theory*, vol. 153, p. 103957, 2020.

[8] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Trajectory planning in robotics," *Mathematics in Computer Science*, vol. 6, pp. 269–279, Sep 2012.

[9] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. USA: Cambridge University Press, 1st ed., 2017.

[10] P. Meckl and P. Arestides, "Optimized s-curve motion profiles for minimum residual vibration," in *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, vol. 5, pp. 2627–2631 vol.5, 1998.

[11] J. R. García Martínez, J. Rodríguez Reséndiz, M. Martínez Prado, and E. E. Cruz Miguel, "Assessment of jerk performance s-curve and trapezoidal velocity profiles," in *2017 XIII International Engineering Congress (CONIIN)*, pp. 1–7, 2017.

[12] K. D. Nguyen, T.-C. Ng, and I.-M. Chen, "On algorithms for planning s-curve motion profiles," *International Journal of Advanced Robotic Systems*, vol. 5, no. 1, p. 11, 2008.

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

[14] A. D. Dragan, K. C. Lee, and S. S. Srinivasa, "Legibility and predictability of robot motion," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 301–308, 2013.

[15] R. Finotello, T. Grasso, G. Rossi, and A. Terribile, "Computation of kinetostatic performances of robot manipulators with polytopes," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, pp. 3241–3246 vol.4, 1998.

[16] P. Chiacchio, "A new dynamic manipulability ellipsoid for redundant manipulators," *Robotica*, vol. 18, no. 4, p. 381–387, 2000.

[17] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.

[18] F. Pfeiffer and R. Johanni, "A concept for manipulator trajectory planning," *IEEE Journal on Robotics and Automation*, vol. 3, no. 2, pp. 115–123, 1987.

[19] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on Automatic Control*, vol. 25, pp. 164–176, Apr. 1980.

[20] S. Vajda and S. I. Gass, "Linear programming. methods and applications," *OR*, vol. 15, no. 4, p. 412, 1964.

[21] A. Skuric, V. Padois, N. Rezzoug, and D. Daney, "On-line feasible wrench polytope evaluation based on human musculoskeletal models: An iterative convex hull method," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5206–5213, 2022.

[22] N. Mansfeld, B. Djellab, J. R. Veuthey, F. Beck, C. Ott, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5390–5397, 2017.

[23] I. Kuric, V. Tlach, Z. Ságová, M. Císar, and I. Gritsuk, "Measurement of industrial robot pose repeatability," in *MATEC web of conferences*, vol. 244, p. 01015, EDP Sciences, 2018.

[24] A. O. Makhorin, "Glpk - gnu linear programming kit, version 4.0." Online: http://www.gnu.org/software/glpk/, 2022.

[25] A. D. Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 281–288, 2018.