Task Priority Matrix at the Acceleration Level: Collision Avoidance Under Relaxed Constraints

Maram Khatib, Khaled Al Khudir, and Alessandro De Luca

***Abstract*—We propose a new approach for executing the main Cartesian tasks assigned to a robot while guaranteeing whole-body collision avoidance. The robot degrees of freedom are fully utilized by introducing relaxed constraints in the definition of operational and collision avoidance tasks. Desired priorities for each task are assigned using the so-called Task Priority Matrix (TPM) method [1], which is independent from the redundancy resolution law and handles efficiently switchings of priorities. To ensure smooth motion during such task reorderings, a control scheme with a suitable task allocation algorithm is developed at the acceleration level. The proposed approach is validated with MATLAB simulations and an experimental evaluation using the 7-dof KUKA LWR manipulator.**

***Index Terms*—Motion control, collision avoidance, redundant robots.**

I. Introduction

THE robot capability of handling simultaneously multiple tasks, while guaranteeing continuous avoidance of collisions in a human-shared workspace, is an essential feature in service applications of robotics [2], as well as one of the enabling technologies of Industry 4.0 [3], [4]. Collision avoidance should be obtained without giving up unnecessarily mobility and dexterity, as would happen when oversizing safety areas around the robot or by freezing some of the available degrees of freedom of the manipulator. To this end, three main subproblems have to be addressed.

The first problem is the reliable, online detection of obstacles in the robot workspace, which allows computing relative distances between the whole body of a robot in motion and dynamic objects present in the surveillance area. This typically requires the use of one or more exteroceptive sensors, e.g., cameras, depth sensors, or laser scanners. In [5], distances between the robot end-effector and nearby obstacles are computed using an on-board laser sensor. However, such equipment would be inefficient (or become too expensive) when collision avoidance with the whole robot body is to be considered. Resorting to special markers or inertial measurement units could be useful for the localization of a moving operator [6], [7], but other possibly dangerous obstacles in the workspace would then be neglected. On the other hand, an efficient robot-object distance computation has been introduced in [8]. The proposed approach works directly in the depth space of one or more RGB-D sensors, achieving detection and distance evaluation for any type of obstacles (including humans) present in the workspace, with strict real-time performance.

The second problem is the definition of an effective collision avoidance strategy, based on the computed distance information. Collision avoidance tasks are usually defined in terms of a number of selected robot control points in the 3D Cartesian space that are subject to the repulsive action of proximal obstacles according to any preferred artificial potential field method [9]. Thus, a generic repulsive vector associated to such a collision avoidance scheme will have dimension $m = 3$, and would then be transformed into the joint space and processed according to the chosen control scheme [5], [10]. In this framework, to reduce the number of dofs required to accomplish the local collision avoidance, it was proposed in [11] to project the Cartesian repulsive vector along the unit vector $\boldsymbol{d}_0$ that connects the closest points on the obstacle and on the robot, and to define accordingly an associated task Jacobian having just a single row ($m = 1$). In this way, while the control point moves away from the closest obstacle in any direction forming an acute angle with $\boldsymbol{d}_0$, more mobility is left to the robot for executing other tasks.

Exploiting the kinematic redundancy of the robot is the third and last problem to be considered here. In general, we would like to avoid any collision, while preserving as much as possible the correct execution of the main desired task(s) assigned to the robot. This can be done with the usual null-space projection, where the joint commands (usually velocities) related to collision avoidance tasks are cumulated algebraically and projected in the null space of the main task(s) [5]. Alternatively, collision avoidance commands can be converted into time-varying *hard* inequality bounds/constraints in the joint space, which should then be satisfied by any robot motion needed to handle multiple tasks with assigned priorities [10]. In these methods, a suitable scale factor is needed, either for switching priority between main tasks and/or collision avoidance tasks, or in order to modify the former according to new joint bounds arising from the latter. Collision avoidance can also be integrated in a unified optimization problem, as an inequality constraint defined in the joint or task space, and solved using the Hierarchical Complete Orthogonal Decomposition (HCOD) [12]. In the previous approaches, the process of each priority level starts only after obtaining the result of all higher priority tasks. Alternatively, a faster and simpler approach has been proposed by Fabrizio Flacco in [1], separating redundancy control from the need of frequent but efficient task priority resolution and its modifications. In this case, any change (swap/insertion/deletion) in the task

Manuscript received February 24, 2020; revised May 13, 2020; accepted June 13, 2020. This letter was recommended for publication by Editor P. Rocco upon evaluation of the Associate Editor and Reviewers' comments.

The authors are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Roma 00185, Italy (e-mail: khatib@diag.uniroma1.it; alkhudir@diag.uniroma1.it; deluca@diag.uniroma1.it).

Digital Object Identifier (DOI): see top of this page.

priority structure requires only to modify a so-called Task Priority Matrix (TPM) which contains a compact but complete information on the different task dependencies.

In this letter, we propose a new approach to handle the execution of multiple tasks while ensuring robot collision avoidance. To start with, the TPM method [1] is developed at the acceleration level, so as to *i)* eliminate undesired discontinuities in the joint velocity due to changes in the task priorities, and *ii)* allow for consideration of robot dynamics. Robot dexterity is better preserved by using an inequality constraint in place of an equality one for the relaxed pointing task proposed in [6], which requires the minimum number of dofs. Also, similar to [11], relaxed collision avoidance tasks are defined using multiple surveillance areas with different danger levels. The benefits of these choices are integrated in an efficient algorithm for task priority assignment.

The rest of the letter is organized as follows. Section II recalls the development of the TPM method and extends it to the acceleration level. Collision avoidance under relaxed constraints is discussed in Sec. III. Different comparative simulations and an experimental evaluation with a KUKA LWR IV robot are reported in Sec. IV. The obtained results are supported by an accompanying video and by a corresponding MATLAB simulation code for their reproducibility[1].

II. Task Priority Control

Let a set of l desired Cartesian tasks, to be achieved in a specific desired priority, be defined by the evolution of the task variables

$$\boldsymbol{p}_i = \boldsymbol{p}_i(t), \quad 0 < i \le l, \tag{1}$$

where $\boldsymbol{p}_i \in \mathbb{R}^{m_i}$. The i-th task has higher priority than the j-th task if $i < j$. The simplest command that tries to execute all the l Cartesian tasks without considering their priority order, is obtained using the first-order differential inverse kinematics as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{p}}, \tag{2}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ are the joint velocity commands and

$$\boldsymbol{J} = [\boldsymbol{J}_1^T \quad \dots \quad \boldsymbol{J}_i^T \quad \dots \quad \boldsymbol{J}_l^T]^T, \tag{3}$$

is the augmented matrix with the Jacobians $\boldsymbol{J}_i \in \mathbb{R}^{m_i \times n}$ related to each desired task, where $\sum_{i=1}^{l} m_i = m$ (in general, but not necessarily, with $m \le n$) and

$$\dot{\boldsymbol{p}} = [\dot{\boldsymbol{p}}_1^T \quad \dots \quad \dot{\boldsymbol{p}}_i^T \quad \dots \quad \dot{\boldsymbol{p}}_l^T]^T, \tag{4}$$

is the so-called Stack of Tasks (SoT). Using (2), the accuracy in the execution of each task will result according to the dependency between all the tasks. If the tasks are not in conflict, the robot will achieve all of them, otherwise the robot will move in response to the sum of each task contribution. Indeed, solution (2) provides a good shape for the inverse kinematics solution, in which the single task contributions and the null space for each task can be deduced easily. On the other hand, singularities may appear when there are linear dependencies between the tasks [13].

[1]The code is provided in the supplementary material and published on https://github.com/maram-khatib/task-priority-relaxed-constraints

The most common way to take into account the strict task priorities while executing the SoT (4), is to use the recursive solution (for $i = 1, \dots, l$) with null space projections as [14]

$$\dot{\boldsymbol{q}}_i = \dot{\boldsymbol{q}}_{i-1} + (\boldsymbol{J}_i\boldsymbol{P}_{i-1})^{\#}(\dot{\boldsymbol{p}}_i - \boldsymbol{J}_i\dot{\boldsymbol{q}}_{i-1}), \tag{5}$$

with $\dot{\boldsymbol{q}}_0 = \boldsymbol{0}$. The null space projector is given by

$$\boldsymbol{P}_i = \boldsymbol{P}_{i-1} - (\boldsymbol{J}_i\boldsymbol{P}_{i-1})^{\#}\boldsymbol{J}_i\boldsymbol{P}_{i-1}, \tag{6}$$

with $\boldsymbol{P}_0 = \boldsymbol{I}$. Using (5), the highest priority tasks are ideally no longer affected by the lower priority ones. On the other hand, when some of the tasks are linearly dependent, the contribution of a lower priority task cannot be inferred. This missing information is important to perform a smooth transition during any desired addition, deletion, or reordering operation on the SoT [15].

In order to combine the advantages of (2) and (5), a new simple approach that has a similar shape of (2) and returns exactly the solution obtained by (5), keeping the whole knowledge of the tasks, has been presented by [1]. The redundancy resolution is independent from enforcing the desired task priority. This is obtained using the so-called Task Priority Matrix (TPM) as follows

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}\dot{\boldsymbol{p}}, \tag{7}$$

where $\boldsymbol{F} \in \mathbb{R}^{m \times m}$ is a square matrix to be computed in a specific way to impose the desired task priority. The key tool of TPM calculation is to use the Gauss-Jordan elimination method [16], which is commonly used to compute the reduced row echelon form, considering a pivot square matrix with dimension $(m_i \times m_i)$ for each corresponding desired task instead of pivot elements.

Let the QR decomposition for the transpose of the augmented Jacobian (3) be given by

$$\boldsymbol{J}^T = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{O} \end{bmatrix}, \tag{8}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{R}^{m \times m}$ is an upper triangular matrix. The diagonal block matrices $\boldsymbol{R}_{ii} \in \mathbb{R}^{m_i \times m_i}$ correspond to the desired tasks and will be used as pivot matrices during TPM calculations. Note that, the upper triangular pivot matrix $\boldsymbol{R}_{ii}$ is nonsingular if the task i is nonsingular and linearly independent from all tasks with higher priority.

The algorithm to compute the TPM is as follows.

Step 1: Initialize TPM as

$$\bar{\boldsymbol{F}} = \boldsymbol{R}. \tag{9}$$

Step 2: Let $m_{h_i} = \sum_{k=1}^{i-1} m_k$ and $m_{h_1} = 0$. Multiply the block row $\bar{\boldsymbol{F}}(m_{h_i} + 1 : m_{h_i} + m_i, 1 : m)$, i.e. the blue blocks in Fig. 1, with the pseudoinverse of its pivot matrix $\boldsymbol{R}_{ii}^{\#}$. As a result, the i-th diagonal block will be identity if $\boldsymbol{R}_{ii}$ is not singular.

Step 3: For $i > 1$, multiply the block $\bar{\boldsymbol{F}}(1 : m_{h_i}, m_{h_i}+1 : m_{h_i}+m_i)$, i.e. the red blocks in Fig. 1, by the row block $\bar{\boldsymbol{F}}(m_{h_i} + 1 : m_{h_i} + m_i, 1 : m)$, i.e. the blue blocks in Fig. 1, and subtract it from the block row $\bar{\boldsymbol{F}}(1 : m_{h_i}, 1 : m)$, i.e. the green blocks in Fig. 1.

Step 4: Apply steps 2 and 3 in place for all block rows related to all tasks, i.e., for $0 < i \leq l$.

Step 5: Get the transpose of the modified matrix where

$$\boldsymbol{F} = \bar{\boldsymbol{F}}^T. \tag{10}$$

Note that, if $n \geq m$, the QR decomposition (8) can be also used to compute the pseudoinverse $\boldsymbol{J}^{\#}$ in (7) as

$$\boldsymbol{J}^{\#} = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{R}^{-T} \\ \boldsymbol{O} \end{bmatrix}. \tag{11}$$

In [1], the solution (7) using TPM is proved to be exactly equal to the standard null space projection (5). By using (7), however, the redundancy resolution is totally separated from controlling the SoT in a desired priority order. This gives more flexibility for applying online any desired change in the task priorities. In this case, the only part to be recomputed in (7) is the $\boldsymbol{F}$ matrix according to the new order of tasks. Also, to boost the computational efficiency, parallel computing can be used for the QR decomposition in (8). These features cannot be obtained using other methods, such as with (5) or HCOD [12], where all computations should be repeated sequentially since the control of low-priority tasks has to wait for the outcome of all higher priority tasks. Furthermore, since the solutions (5) and (7) are defined at the velocity level, they suffer from joint velocity discontinuity in correspondence of any change in either the desired SoT or in the dependencies among the tasks. Although a scaling factor can be used in (5) to achieve soft transition [15], the advantage of taking into account the robot dynamics is still missing.

In the following, we extend the solution (7) in order to get the following improvements.

- Eliminate discontinuities in the joint velocity by defining the control law (7) at the acceleration level as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}(\ddot{\boldsymbol{p}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}), \tag{12}$$

  where $\boldsymbol{F}$ is exactly the same as in (7), since its computation depends only on the augmented Jacobian matrix. In this case, when the joint acceleration (12) is integrated once, the obtained joint velocities are automatically ensured to be continuous [17]. Note that solution (12) returns exactly the one obtained by recursion as

$$\ddot{\boldsymbol{q}}_i = \ddot{\boldsymbol{q}}_{i-1} + (\boldsymbol{J}_i\boldsymbol{P}_{i-1})^{\#}(\ddot{\boldsymbol{p}}_i - \dot{\boldsymbol{J}}_i\dot{\boldsymbol{q}}_i - \boldsymbol{J}_i\ddot{\boldsymbol{q}}_{i-1}). \tag{13}$$

- Allow the consideration of robot dynamics or of any other additional task at the joint level, by including a null-space term as

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\boldsymbol{F}(\ddot{\boldsymbol{p}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) + (\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J})\ddot{\boldsymbol{q}}_0. \tag{14}$$

For example, in order to get stable and smooth joint trajectories, the joint acceleration $\ddot{\boldsymbol{q}}_0$ can be computed as

$$\ddot{\boldsymbol{q}}_0 = -\boldsymbol{D}\dot{\boldsymbol{q}}, \tag{15}$$

where $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ is a positive definite matrix, e.g., damping the momentum using the robot mass matrix as $\boldsymbol{D}(\boldsymbol{q})$. By projecting $\ddot{\boldsymbol{q}}_0$ in the null space of the augmented Jacobian (3), the additional joint-space task will always be at the lowest control priority, without any influence on the higher priority Cartesian tasks.

$$\bar{\boldsymbol{F}} = \begin{array}{c|cccc} & m_1 & m_2 & & m_l \\ m_1 & \boldsymbol{R}_{11} & \star & \dots & \star \\ m_2 & \boldsymbol{0} & \boldsymbol{R}_{22} & \dots & \star \\ & \boldsymbol{0} & \boldsymbol{0} & \dots & \star \\ m_l & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{R}_{ll} \end{array}$$

$$\bar{\boldsymbol{F}} = \begin{array}{c|cccc} & m_1 & m_2 & & m_l \\ m_1 & \boldsymbol{R}_{11} & \star & \dots & \star \\ m_2 & \boldsymbol{0} & \boldsymbol{R}_{22} & \dots & \star \\ & \boldsymbol{0} & \boldsymbol{0} & \dots & \star \\ m_l & \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{R}_{ll} \end{array}$$

Fig. 1. The structure of the temporary matrix $\bar{\boldsymbol{F}}$. The highlighted components refer to the involved block matrices during the operations of computing the TPM: [top] for $i = 2$ and [bottom] for $i = l$.

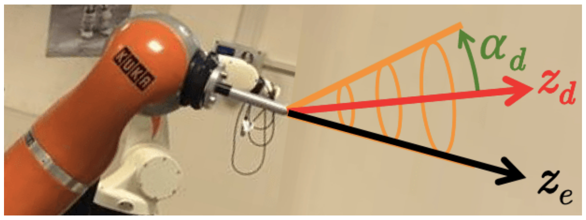


Fig. 2. Reference frames and definition of parameters for the desired relaxed pointing task.

III. Collision Avoidance Under Relaxed Constraints

A. Relaxed Pointing Task

Usually, the robot end effector (EE) Cartesian tasks include following a desired time-varying position $\boldsymbol{p}_{ee}$ ($m_i = 3$) and/or orientation ($m_i = 3$). In this case, the robot needs at least $n = 6$ dofs. The number of required joints reduces/prevents the robot capability to fulfill the desired tasks while optimizing the motion w.r.t. other desirable criteria, e.g., avoiding any robot body collision in the work space. In this case, decreasing the actual dimension of the desired task is preferred.

In [6], we proposed to use a relaxed pointing task ($m_i = 1$) instead of a classical and complete orientation definition. The desired pointing task is defined as a cone surface, and an EE unit axis, e.g., $\boldsymbol{z}_e(\boldsymbol{q})$, should point toward any Cartesian point belonging to this surface. The cone has its apex located at the EE position, and is defined by a unit axis $\boldsymbol{z}_d(t)$ with a desired orientation and by an apex angle $\alpha = \alpha_d > 0$ (see Fig. 2). In this case, the desired Cartesian task can be defined as

$$p_{rp} = \boldsymbol{z}_d^T\boldsymbol{z}_e(\boldsymbol{q}) = \cos\alpha, \tag{16}$$

where $\alpha = \alpha_d$. Then, the associated Jacobian is

$$\boldsymbol{J}_{rp} = \frac{\partial p_{rp}}{\partial \boldsymbol{q}} = \frac{\partial \boldsymbol{z}_e}{\partial \boldsymbol{q}}\boldsymbol{z}_d. \tag{17}$$

From [18],

$$\frac{d\boldsymbol{z}_e}{dt} = \frac{\partial \boldsymbol{z}_e}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{S}^T(\boldsymbol{z}_e)\boldsymbol{\omega} = \boldsymbol{S}^T(\boldsymbol{z}_e)\boldsymbol{J}_O\dot{\boldsymbol{q}}, \tag{18}$$

where the matrix $\boldsymbol{S} \in \mathbb{R}^{3\times 3}$ is skew-symmetric and the Jacobian $\boldsymbol{J}_O \in \mathbb{R}^{3\times n}$ maps the joint velocity $\dot{\boldsymbol{q}}$ to the EE angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$. From (18) and (17),

$$\boldsymbol{J}_{rp} = \boldsymbol{z}_d^T \boldsymbol{S}^T(\boldsymbol{z}_e)\boldsymbol{J}_O. \tag{19}$$

In this work, instead of using the equality constraint (16), we propose further relaxation by defining the desired pointing task using the inequality

$$0 \leq \alpha \leq \alpha_d. \tag{20}$$

Then, the desired task (16) can be written as

$$\cos\alpha_d \leq p_{rp} \leq 1, \tag{21}$$

where the unit axis $\boldsymbol{z}_e(\boldsymbol{q})$ is allowed to point toward any Cartesian point inside or on the surface of the cone. In this case, the pointing task control will be activated, i.e., be included in the SoT, for a period of time which is less or equal to the case of equality condition (16). This leaves one more dof to the robot, to be exploited for other tasks.

B. Relaxed Collision Avoidance Task

Collision avoidance is the most important Cartesian task that the robot should handle, regardless of other desired tasks. However, if there are enough dofs, it is always preferred to achieve the desired tasks while escaping any collision in the work space. This issue can be separated into two subproblems. The avoidance task definition in the Cartesian space, and the control of this task simultaneously with other robot tasks.

The artificial potential field approach is a common way to deal with the collision avoidance task [9]. The key tool is how the repulsive vector $\boldsymbol{r}(\boldsymbol{p}_{\boldsymbol{c}_k})$ between each robot control point $\boldsymbol{p}_{\boldsymbol{c}_k}$ and all obstacles will be defined, see for example [5] and [10]. Let the minimum Cartesian distances between each point $\boldsymbol{p}_{\boldsymbol{c}_k}$ and all obstacles in the surveillance area be given by

$$D_{min}(\boldsymbol{p}_{\boldsymbol{c}_k}) = \min_{j=1}^{h} D(\boldsymbol{p}_{\boldsymbol{c}_k}, \boldsymbol{o}_j), \quad k = 1, \ldots, k_t, \tag{22}$$

where $D(\boldsymbol{p}_{\boldsymbol{c}_k}, \boldsymbol{o}_j)$ is the distance between the control point $\boldsymbol{p}_{\boldsymbol{c}_k}$ and an obstacle $\boldsymbol{o}_j$. The total number of robot control points and of obstacles is k_t and h, respectively. Then, the magnitude r of the repulsive vector can be treated as [8]

$$r(\boldsymbol{p}_{\boldsymbol{c}_k}) = \frac{r_{max}}{1 + e^{(D_{min}(\boldsymbol{p}_{\boldsymbol{c}_k})(2/\rho)-1)\gamma}}, \tag{23}$$

being r_{max} the maximum admissible magnitude and $\gamma > 0$ a shape factor. Using (23), when $\gamma \to \inf$, the magnitude value equals

$$\begin{cases} r(\boldsymbol{p}_{\boldsymbol{c}_k}) = r_{max}, & \text{for } D_{min}(\boldsymbol{p}_{\boldsymbol{c}_k}) = 0, \\ r(\boldsymbol{p}_{\boldsymbol{c}_k}) \approx 0, & \text{for } D_{min}(\boldsymbol{p}_{\boldsymbol{c}_k}) = \rho, \end{cases} \tag{24}$$

while $r(\boldsymbol{p}_{\boldsymbol{c}_k})$ is not defined beyond ρ.

The desired direction of the repulsive vector can be computed from the mean of the unit distance vectors between the control point and all objects in the surveillance area where

$$\boldsymbol{r}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k}) = \frac{1}{h}\sum_{j=1}^{h} \frac{\boldsymbol{p}_{\boldsymbol{c}_k} - \boldsymbol{o}_j}{\|\boldsymbol{p}_{\boldsymbol{c}_k} - \boldsymbol{o}_j\|}, \tag{25}$$

$$\hat{\boldsymbol{r}}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k}) = \frac{\boldsymbol{r}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k})}{\|\boldsymbol{r}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k})\|}. \tag{26}$$

Using (23) and (26), the repulsive vector corresponding to each control point is defined as

$$\boldsymbol{r}(\boldsymbol{p}_{\boldsymbol{c}_k}) = r(\boldsymbol{p}_{\boldsymbol{c}_k})\hat{\boldsymbol{r}}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k}). \tag{27}$$

The response intensity of the control point considers the nearest object only. While the direction takes into account all objects in the surveillance area. This hybrid reaction prevents any possible undesirable behavior (e.g., continuous direction switching among two close objects) resulting from the topology of obstacles in the surveillance area [8].

In this work, we consider each repulsive vector (27) as a desired Cartesian acceleration where

$$\begin{aligned} \ddot{\boldsymbol{p}}_{c_k} &= \boldsymbol{r}(\boldsymbol{p}_{\boldsymbol{c}_k}), \\ \boldsymbol{J}_{c_k} &= \frac{\partial \boldsymbol{p}_{\boldsymbol{c}_k}}{\partial \boldsymbol{q}}. \end{aligned} \tag{28}$$

Indeed, the collision avoidance task (28) is overdetermined and requires at least $n = 3$ dofs to be handled. Therefore, similar to [11], we relax each avoidance task constraint to a dimension $m_k = 1$, for each control point along the robot body. This is done by projecting the desired task (28) along its corresponding direction as

$$\begin{aligned} \ddot{p}_{n_k} &= \hat{\boldsymbol{r}}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k})^T \ddot{\boldsymbol{p}}_{c_k} = r(\boldsymbol{p}_{\boldsymbol{c}_k}), \\ \boldsymbol{J}_{n_k} &= \hat{\boldsymbol{r}}_{mean}(\boldsymbol{p}_{\boldsymbol{c}_k})^T \boldsymbol{J}_{c_k}, \end{aligned} \tag{29}$$

where $\ddot{p}_{n_k} \in \mathbb{R}$ and $\boldsymbol{J}_{n_k} \in \mathbb{R}^{1\times n}$. Using (29), the avoidance task drives the control point toward any point of a (sufficiently away) plane orthogonal to its repulsive direction. When the robot control point of interest is located at the end-effector ($k = ee$), there is no way to preserve the desired EE task $\ddot{\boldsymbol{p}}_{ee}$ while avoiding collision. Then, to obtain a smooth avoidance motion, the EE collision avoidance task $\ddot{\boldsymbol{p}}_{c_{ee}}$ should be added directly to the main EE positional task as

$$\ddot{\boldsymbol{p}} = \ddot{\boldsymbol{p}}_{ee} + \ddot{\boldsymbol{p}}_{c_{ee}}. \tag{30}$$

C. The Complete Approach

In order to control the robot EE to achieve prioritized desired Cartesian tasks while avoiding any collision efficiently, we summarize our complete approach as follows.

- Define the desired orientation using the soft pointing task (21).
- Provide the control point at the EE with a danger threshold distance ϵ_{ee}. When $D_{min}(\boldsymbol{p}_{\boldsymbol{c}_{ee}}) \leq \epsilon_{ee}$, the repulsive vector $\ddot{\boldsymbol{p}}_{c_{ee}}$ is computed according to (28) and added to the main EE desired task $\ddot{\boldsymbol{p}}_{ee}$ as in (30).
- Provide the other control points along the robot body with two danger threshold distances, $\epsilon_2 > \epsilon_1 > 0$,

Algorithm 1 Task priority assignment using TPM method

```
$\ddot{p} = \ddot{p}_{ee}, \quad J = J_{ee}, \quad \dot{J} = \dot{J}_{ee}$
if $D_{min}(p_{c_{ee}}) \leq \epsilon_{ee}$ then
    $\ddot{p} = \ddot{p}_{ee} + \ddot{p}_{c_{ee}}$
end if
if $D_{minimal} \leq \epsilon_1$ then
    if $D_{minimal} < D_{min}(p_{c_{ee}})$ then
        $\ddot{p} = [\ddot{p}_{n_k}^T \quad \ddot{p}^T]^T$
        $J = [J_{n_k}^T \quad J^T]^T$
        $\dot{J} = [\dot{J}_{n_k}^T \quad \dot{J}^T]^T$
    else
        $\ddot{p} = [\ddot{p}^T \quad \ddot{p}_{n_k}^T]^T$
        $J = [J^T \quad J_{n_k}^T]^T$
        $\dot{J} = [\dot{J}^T \quad \dot{J}_{n_k}^T]^T$
    end if
end if
if $\cos\alpha_d - \cos\alpha > 0$ then
    $\ddot{p} = [\ddot{p}^T \quad \ddot{p}_{rp}^T]^T$
    $J = [J^T \quad J_{rp}^T]^T$
    $\dot{J} = [\dot{J}^T \quad \dot{J}_{rp}^T]^T$
end if
if $\epsilon_1 < D_{minimal} \leq \epsilon_2$ then
    $\ddot{p} = [\ddot{p}^T \quad \ddot{p}_{n_k}^T]^T$
    $J = [J^T \quad J_{n_k}^T]^T$
    $\dot{J} = [\dot{J}^T \quad \dot{J}_{n_k}^T]^T$
end if
```

in the surveillance area. Only one control point will then be taken into account in the SoT using the relaxed definition (29). The considered control point corresponds to the minimal distance

$$D_{minimal} = \min_{k=1,\ldots,k_t} D_{min}(\boldsymbol{p}_{\boldsymbol{c}_k}) \leq \epsilon_2, \qquad (31)$$

between all obstacles and each control point. Note that $\boldsymbol{p}_{c_{ee}}$ is not considered in (31).

- Use Algorithm 1 to set the proper priority for each task.
- Use the TPM control law (14) at the acceleration level with the null space damping (15), through a closed-loop inverse kinematics algorithm to stabilize the Cartesian error dynamics [19].

In Algorithm 1, the first priority is always to avoid the collision with the closest object to the whole robot. Moreover, when $\epsilon_1 < D_{minimal} \leq \epsilon_2$, a robot body collision avoidance task will be added as the lowest Cartesian task priority. In this case, if there are enough dofs, the control scheme will prevent the robot nearing to the higher danger region. Note that this algorithm can be easily extended/modified in order to apply other control strategies (e.g., considering more control points on the robot body for collision avoidance).

Using the proposed approach, the robot needs less dofs to fulfill the desired tasks than with the previous approaches in [5] and [10]. Differently from [10], the EE collision avoidance can be set at the first or at any other desired priority. Also, when using (14), the joint acceleration reflects the desired robot body collision avoidance task more accurately than when using the approximate solution in [5] or [10]. Moreover, the special features of the $\boldsymbol{F}$ matrix used at the acceleration level make it simpler and smoother to enforce any desired change in the SoT, without using any additional scaling parameter.

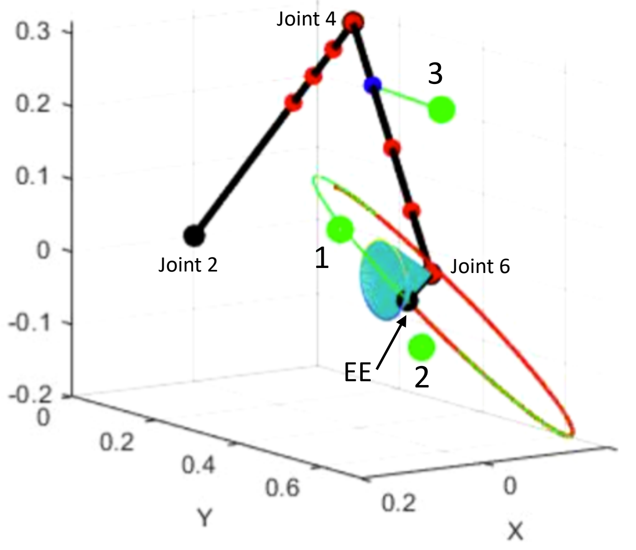


Fig. 3. Simulation environment with a snapshot during robot motion. Only the DH frame origins placed at robot joints and EE, assigned according to [20], are drawn (black spheres). Black lines represent the corresponding robot links. Red points represent the control points along the robot body. During motion, the control point corresponding to (31) is changed to blue color and linked to the nearest obstacle with a green line. The green ellipse is the desired path, while the red curve shows the actual robot EE position. The cyan cone represents the relaxed pointing task for $\alpha_d = 35°$.

IV. Results

A. Simulations

We present here different simulation scenarios and comparisons. First, we compare the use of TPM at the velocity level (7) with the proposed scheme at the acceleration level (14). Second, the robot behavior using the classical 3-dimensional avoidance task (28) is compared with the 1-dimensional relaxation (29). Third, we show the different performance obtained when specifying the pointing task by an equality (16) or with the proposed inequality (21).

The evaluations are done in MATLAB using the 7-dof KUKA LWR IV robot. The desired tasks are to follow an elliptic path in the Cartesian space for three turns, while pointing to a direction parallel to the positive $\boldsymbol{x}$-axis of the world frame. During the motion, the robot should avoid any collision with three static obstacles located in the robot work space. For this, $k_t = 8$ control points along the robot body were considered, see Fig. 3. To have a challenging task, the first obstacle is placed across the path. The desired priority for each of the robot tasks is assigned according to Algorithm 1, where the danger thresholds distances (in meters) are $\epsilon_1 = 0.05, \epsilon_2 = 0.15$, and $\epsilon_{ee} = 0.1$. In all simulations, the value of r_{max} in (23) has been set to 0.15 [m/s^2] for the robot body control points, and to 5 [m/s^2] for the EE control point. The robot behavior during the various simulations is shown in the accompanying video.

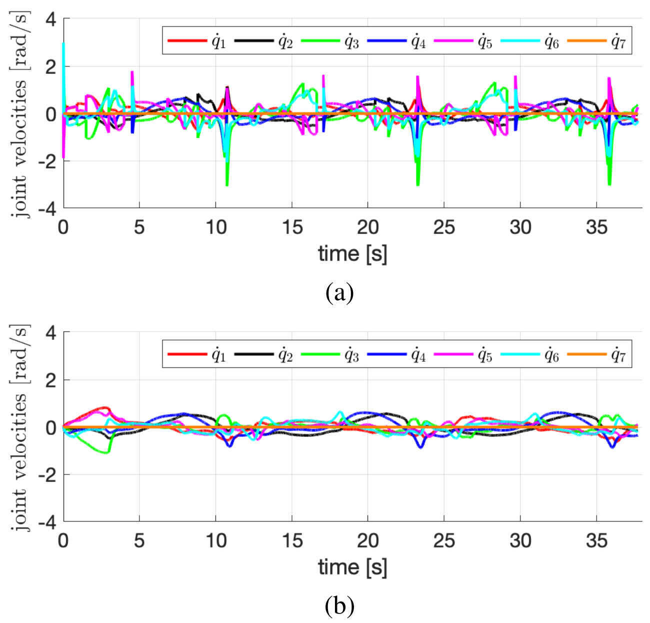


Fig. 4. Joint velocities for the first comparison. (a) TPM at the velocity level. (b) TPM at the acceleration level.

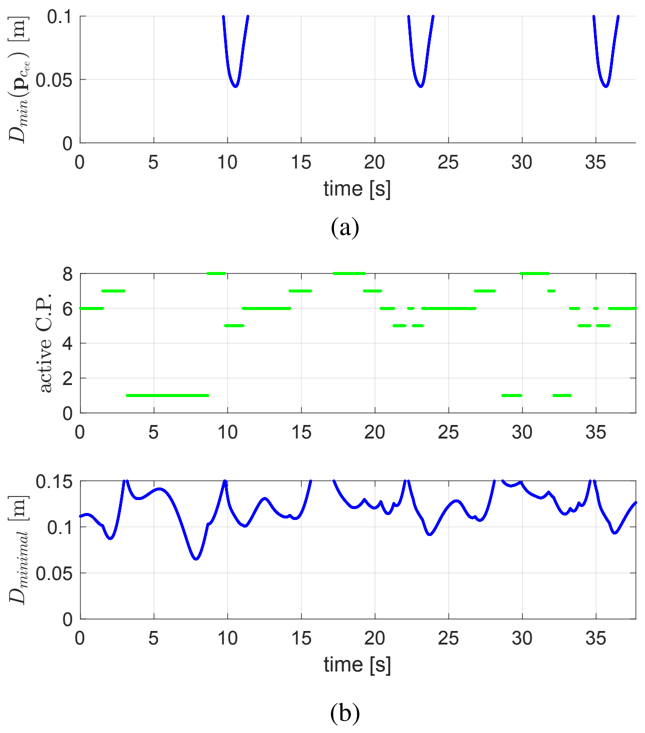


Fig. 5. Simulation at the acceleration level: (a) $D_{min}(\boldsymbol{p}_{\boldsymbol{c}_{ee}})$ where $\epsilon_{ee} = 0.1$ [m]. (b) Active robot control point [above] and corresponding $D_{minimal}$ [bottom], with $\epsilon_1 = 0.05$ [m] and $\epsilon_2 = 0.15$ [m].

1) Velocity level vs. acceleration level: In this comparison, the robot performs a positional task and an inequality pointing task with $\alpha_d = 5°$. For the collision avoidance of the control points, the 1-dimensional definition is used. First, the tasks are controlled at the velocity level using TPM (7) through a similar approach of Sec. III-C. Then, the same Cartesian tasks are controlled using the proposed scheme at the acceleration level (14). When using (7), Fig. 4(a) shows discontinuities in the joint velocity in correspondence of any change in either the desired SoT or in the dependencies among the tasks. In contrast, when using (14), the joint velocity in Fig. 4(b) are smooth and without any high or sudden peaks. In Fig. 5, it is clear how the use of (29) is sufficient to keep the robot body far from any obstacle. According to Algorithm 1, this is done while the corresponding avoidance task has the lowest Cartesian task priority, since $D_{minimal} > \epsilon_1$ during the whole simulation.

2) 3-dimensional collision avoidance task: The robot should achieve the aforementioned EE Cartesian tasks at the acceleration level using the 3-dimensional definition (28) for the robot body collision avoidance. As shown in Fig. 6(b), the robot comes closer to the obstacles more frequently and with less $D_{minimal}$ values than when using the relaxed definition (29). This is because the 3-dimensional avoidance task pushes the robot strongly to move away from the current closest obstacle. As a result, the robot comes near to the other obstacles repeating again a similar reaction. Moreover, in this case, there will be not enough dofs available for both the pointing task and the joint-space damping, causing the oscillations in the joint velocities —see Fig. 6(a).

3) Equality vs. inequality pointing task constraint: For a desired pointing angle task of $\alpha_d = 35°$, two simulations are done using TPM at the acceleration level with 1-dimensional task for robot body collision avoidance and two different

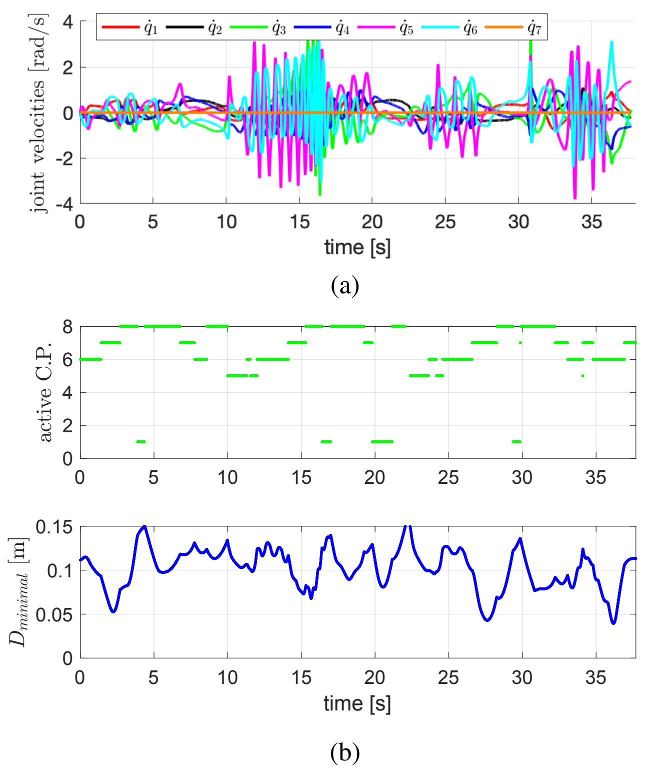
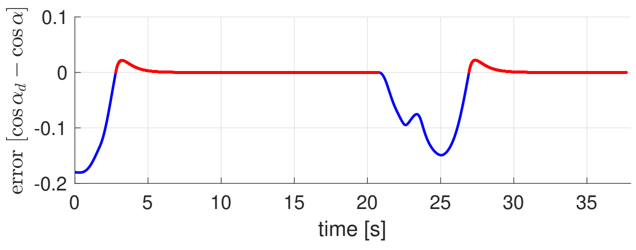
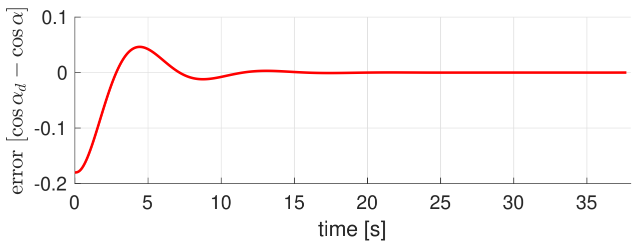


Fig. 6. Simulation at the acceleration level using the 3-dimensional avoidance task definition (28).



(a)



(b)

Fig. 7. EE pointing task error: (a) Using inequality constraint. (b) Using equality constraint. Red traces indicate when the pointing task is active (i.e., included in the SoT).

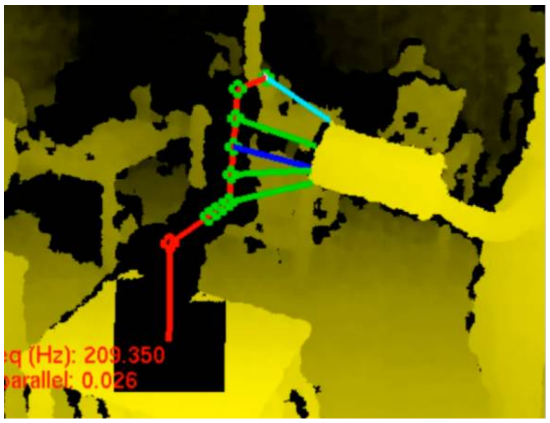
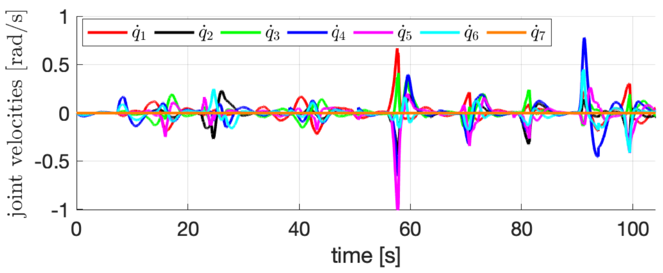
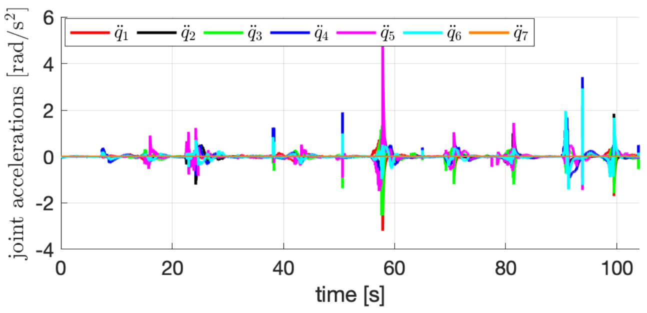


Fig. 8. Snapshot of a depth image superposed with the computed distances between control points (green circles) on the robot and a close object in the surveillance area. The blue line refers to $D_{minimal}$. The cyan line shows the computed distance between the EE and the nearest point of the obstacle.



(a)



(b)

Fig. 9. Joint velocity (a) and acceleration (b) during the experiment with our approach on the KUKA LWR IV robot.

constraints for the pointing task. In Fig. 7, using the equality constraint, the pointing task is always included in the SoT and the error should converge to zero if possible. Using the inequality constraint, the task is not included in the SoT while the pointing error is equal or less than zero, i.e., when the EE points on the cone surface or inside it. In this case, the robot has one extra dof to be exploited for achieving other tasks.

B. *Experiments*

Our proposed approach is implemented using C++ through the ROS 2 middleware and evaluated in an experimental setup using KUKA LWR IV robot. The robot is commanded using the position control mode through the Fast Research Interface (FRI) library [21], with a control cycle of 5 [ms]. The distances between the robot and any obstacle in the surveillance area are evaluated directly in the depth space using one Kinect camera as in [8], see Fig. 8. The desired task is to regulate the robot EE to a fixed prescribed Cartesian point while pointing to a cone with $\alpha_d = 5°$. In this case, only one surveillance area is considered for the robot collision avoidance, with $\epsilon_1 = \epsilon_{ee} = 0.4$ [m].

During the experiment, the hand of an operator and/or an object will be moved frequently close to the robot. As shown in Fig. 10, the proposed approach is efficient in keeping the robot far enough from any obstacle while exploiting the available redundancy to minimize the error of the desired Cartesian tasks —see the accompanying video for a complete understanding. Approximately, for the first 8 seconds, the EE regulates the desired position and pointing tasks by minimizing the Cartesian errors. Then, an obstacle moves close to the EE which reacts accordingly. Later on, the obstacle moves near the robot body and the robot responds depending to the closest control point to the obstacle. Since the desired pointing has the lowest Cartesian task priority, the related error is high when collision avoidance is activated. In Fig. 9, the corresponding joint velocity and acceleration are reported. Discontinuities will appear in the joint acceleration at every change in the SoT. However, joint velocity is smooth along the whole trajectory, with high peak values when the obstacle is too close to the robot.

V. Conclusions

In this letter, the problem of controlling the execution of multiple tasks with different priorities, including the robot collision avoidance task, has been considered. Robot redundancy is exploited efficiently using relaxed (low-dimensional) constraints for both pointing and collision avoidance tasks. We defined the pointing task as an inequality constraint, where the robot EE can point toward any point belonging to a predefined cone region. The collision avoidance task was relaxed by pushing away the robot control point which is closest to the nearby obstacles toward any point in a plane orthogonal to the predefined repulsion direction. Furthermore, we adopted the Task Priority Matrix (TPM) method [1] to deal separately with redundancy resolution control and handling of task priorities. The TPM method was extended and implemented at the acceleration level, eliminating in this way discontinuities in the joint velocity due to task switchings and including the consideration of robot dynamics for joint motion damping. A simple algorithm for efficient task priority management was also presented, where the collision avoidance task is defined as task space inequality constraint on the highest priority level.

The TPM approach is faster and simpler than previous task priority approaches. However, joint inequality constraints have not been considered at present (as instead, e.g., in the HCOD method). This issue will be investigated in the future.

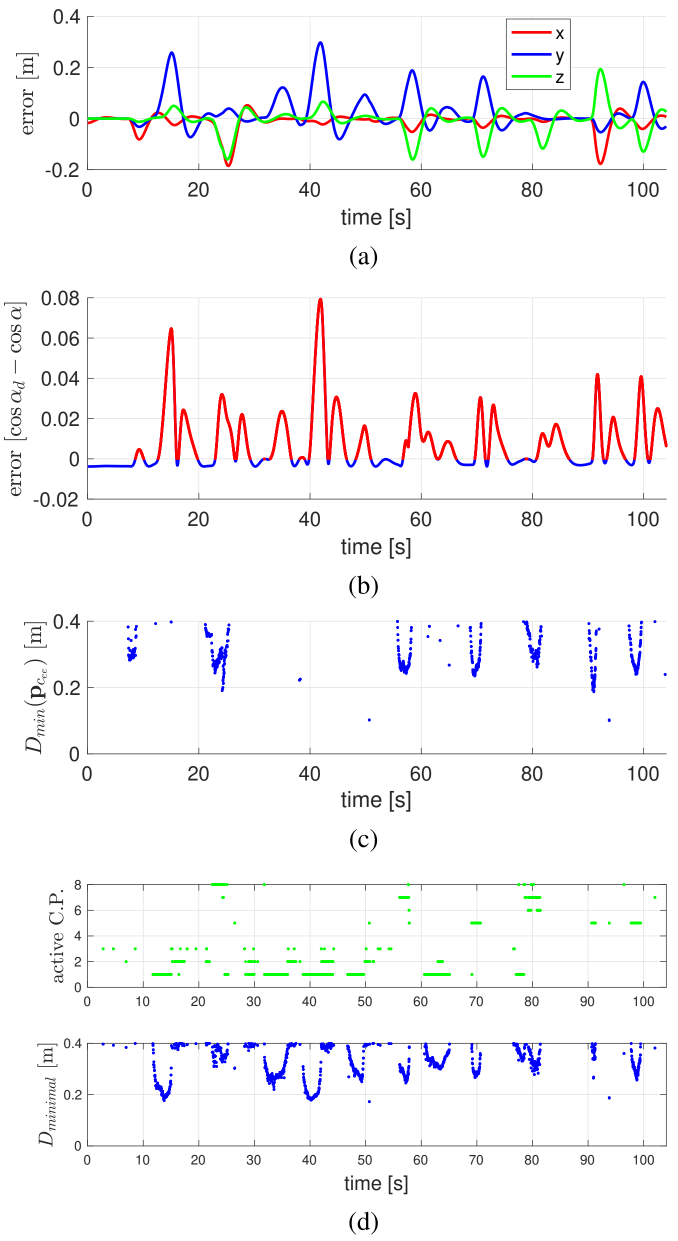


Fig. 10. An experiment using the KUKA LWR IV robot with our approach: (a) Cartesian EE position error. (b) Inequality pointing error. Red traces indicate when the pointing task is active. (c) $D_{min}(\boldsymbol{p}_{\boldsymbol{c}_{ee}})$, with $\epsilon_{ee} = 0.4$ [m] (the several outliers in the shape of single points are due to depth sensor accuracy). (d) Active robot control point [above] and corresponding $D_{minimal}$ [below], with $\epsilon_1 = 0.4$ [m].

References

[1] F. Flacco, "The tasks priority matrix: A new tool for hierarchical redundancy resolution," in *16th IEEE-RAS Int. Conf. on Humanoid Robots*, 2016, pp. 1–7.

[2] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[3] M. Hägele, K. Nilsson, J. Pires, and R. Bischoff, "Industrial robotics," in *Handbook of Robotics (2nd Ed.)*, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 1385–1421.

[4] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," *J. of Industrial Information Integration*, vol. 6, pp. 1–10, 2017.

[5] B. Lacevic, P. Rocco, and A. M. Zanchettin, "Safety assessment and control of robotic manipulators using danger field," *IEEE Trans. on Robotics*, vol. 29, no. 5, pp. 1257–1270, 2013.

[6] M. Khatib, K. Al Khudir, and A. De Luca, "Visual coordination task for human-robot collaboration," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 3762–3768.

[7] M. Safeea and P. Neto, "Minimum distance calculation using laser scanner and IMUs for safe human-robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 58, pp. 33–42, 2019.

[8] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach for evaluating distance to objects – with application to human-robot collision avoidance," *J. of Intelligent & Robotic Systems*, vol. 80, Suppl. 1, pp. 7–22, 2015.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[10] F. Flacco, A. De 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.

[11] L. Zlajpah and B. Nemec, "Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, pp. 1898–1903.

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

[13] S. Chiaverini, "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators," *IEEE Trans. on Robotics and Automation*, vol. 13, no. 3, pp. 398–410, 1997.

[14] J.-J. Slotine and B. Siciliano, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. on Advanced Robotics*, 1991, pp. 1211–1216.

[15] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *IEEE Trans. on Robotics*, vol. 28, no. 6, pp. 1260–1277, 2012.

[16] K. E. Atkinson, *An Introduction to Numerical Analysis*. John Wiley, 2008.

[17] A. Reiter, A. Müller, and H. Gattringer, "On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators," *IEEE Trans. on Industrial Informatics*, vol. 14, no. 4, pp. 1681–1690, 2018.

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[19] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *Int. J. of Robotics Research*, vol. 10, no. 4, pp. 410–425, 1991.

[20] C. Gaz, F. Flacco, and A. De Luca, "Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2016, pp. 2075–2081.

[21] KUKA Systems Technology, *KUKA.FastResearchInterface 1.0*, D-86165 Augsburg, Germany, 2011, version 2.