Fast and Safe Trajectory Planning: Solving the Cobot Performance/Safety Trade-off in Human-Robot Shared Environments

Alessandro Palleschi[+], Mazin Hamad[*], Saeed Abdolshah[*], Manolo Garabini[+], Sami Haddadin[*], Lucia Pallottino[+]

Abstract— **The rise of collaborative robotics has offered new opportunities for integrating automation into the factories, allowing robots and humans to work side-by-side. However, this close physical coexistence inevitably brings new constraints for ensuring safe human-robot cooperation. The current paramount challenge is integrating human safety constraints without compromising the robotic performance goals, which require minimization of the task execution time alongside ensuring its accomplishment. This paper proposes a novel robot trajectory planning algorithm to produce minimum-time yet safe motion plans along specified paths in shared workspaces with humans. To this end, a safety module was used to evaluate the safety of a time-optimal trajectory iteratively. A safe replanning module was developed to optimally adapt the generated trajectory online whenever the optimal plan violates dynamically provided safety limits. In order to preserve performance, a recovery trajectory planning algorithm was included such that the robot is allowed to restore higher speed motions as soon as the safety concern has been resolved. The proposed solution's effectiveness was evaluated both in simulations and real experiments with a redundant robotic manipulator.**

I. Introduction

Recent years have seen an increased interest and effort in developing robotic automation solutions for new market areas, characterized by short product lifetimes and dynamic production changeovers. These requirements drive technology in the direction of scalable and flexible collaborative robots, mostly working in shared environments alongside humans [1]. Research on physical Human-Robot Interaction (pHRI) has also received growing attention recently [2], [3] to accelerate the employment of collaborative robots, especially in small and medium enterprises (SMEs). However, one of the key elements hindering full exploitation of these collaborative solutions is the difficulty of integrating human safety requirements with classical production constraints, e.g., high-speed operations, fast cycle time capabilities, and path constraints [4].

Typically, production constraints are addressed by planning path-constrained, time-optimal motions considering limits on actuator velocities, and accelerations using convex optimization techniques [5]. Such algorithms are mostly used as off-line batch methods for programming the robot to move at its full dynamic capabilities. However, these solutions cannot guarantee safety if the robot has to coexist with humans in unstructured, shared environments. Hence, for developing safe physical Human-Robot Collaboration (pHRC), beyond production constraints, it is of high importance to ensure human safety when robots operate in shared workspaces where collisions may happen.

This work was supported in part by the European Unions Horizon 2020 research and innovation program as part of the projects ILIAD (Grant no.732737), and in part by the Italian Ministry of Education and Research in the framework of the CrossLab project (Departments of Excellence).

[+]Research Center "Enrico Piaggio", Dipartimento di Ingegneria dell'Informazione, University of Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy

[*]Munich School of Robotics and Machine Intelligence, Technical University of Munich (TUM), 80797 Munich, Germany

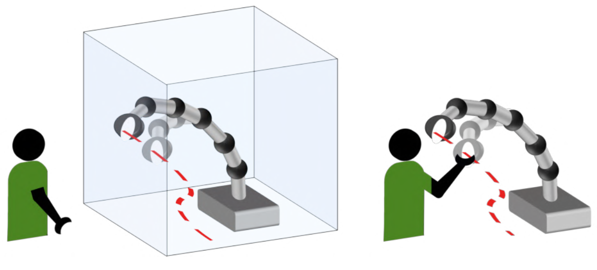
Fig. 1. Collaborative robotics has enabled robots to operate in human occupied workspaces without safety cages, bringing new challenges for developing safe yet productive motion planning strategies.

Human safety requirements during pHRC led to the development of two main paradigms [6]: Speed and Separation Monitoring (SSM) [7], and Power and Force Limiting (PFL) [8]. In the SSM, the relative distance and velocity between the human and the robot are the main factors influencing the robot speed limit, which should guarantee a feasible complete stop before an impact can occur. On the other side, the PFL allows non-zero velocity contacts between the robot and the operator, as long as the energy transferred by the impact is below a certain safety threshold. PFL methods, such as, e.g., the Safe Motion Unit (SMU) scheme proposed by Haddadin et al. [9], limit the robot velocity to a safe level based on the robot dynamic properties (i.e., reflected inertia, the geometry of potential collision points and velocity) in addition to human injury database, reducing the traumatic effects of possible contacts.

From the performance point of view, the SSM paradigm may cause the robot to move unnecessarily slow or even stay still when it drives closer to the human worker, thus compromising productivity. Nevertheless, it allows the robot to move at full speed for high separating distances. In contrast, since it does not consider the human-robot separating distance, PFL could reduce the performance when the human is far away from the robot or the robot moves away from the operator [10]. However, it would allow higher speed motions w.r.t. SSM methods in case of close proximity. While these

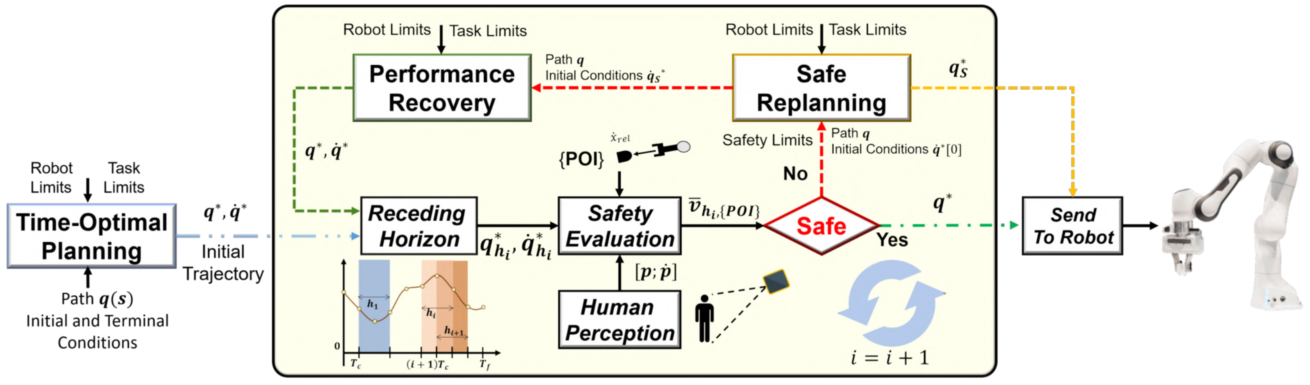

Fig. 2. Block diagram of the proposed discrete time fast and safe trajectory planning framework. At every cycle, based on the information about the human position and speed and the current plan, safety is possibly ensured by online replanning the time evolution of the robot motion along the path.

solutions can guarantee human safety, the development of a strategy able to ensure a decent level of performance for every possible situation is still an open problem. In this paper, we propose a novel trajectory planning algorithm capable of combining the benefits of both approaches to compute safe trajectories without compromising productivity.

The contributions of the paper are as follows. With the aim of maximizing the performance, we developed an optimization-based trajectory planning framework implemented in an iterative fashion to account for dynamic safety constraints generated online based on information on the human presence and motion provided by a perception system. A convex formulation for the path-constrained minimum-time trajectory planning problem with constraints on the robot velocities, accelerations, torques, and jerk is presented. This problem is solved to generate a reference minimum-time trajectory for the robot and replan online the temporal profile of the trajectory whenever the safety speed constraints are provided to the planner.

Compared to classical approaches that rely on finding a maximal scaling coefficient for the speed of the trajectory (see [10]), the proposed formulation allows to: i) enforce the safety constraints only for the part of the trajectory that is not safe; ii) adapt the temporal profile taking into account the dynamics of the robot and its actuation limits, generating a safe, feasible and fast trajectory. Furthermore, the module dedicated to the safety evaluation (SMU) has been implemented to take into account the separating distance between the human and the robot to allow higher speed motions whenever there is no close proximity.

The outline of the paper is as follows: Section II presents the planning strategy and the working principle of our proposed approach. The effectiveness of the proposed solution is evaluated through simulations and real experiments with a collaborative robotic manipulator in a shared human-robot environment in Sections III and IV, respectively. Section V concludes the paper highlighting future research directions.

II. Proposed Solution

Our proposed planning framework has been designed to comply at least with the following conditions: i) the robot has no prior information about the current human position and his intended motion trajectory; ii) if it is not endangering any human, the robot should move at full speed; iii) the robot motion should be adapted to respect rising safety constraints due to the current environmental conditions, taking into accounts the actuation capabilities and without deviating from its path; iv) the robot should resume moving at its maximum allowed speed as soon as the safety concerns are resolved. Therefore, an iterative planning scheme was developed as described by Algorithm 1, while a block scheme highlighting its working principle is depicted in Fig. 2.

Algorithm 1 Safe trajectory planning

```
    Inputs: path $q$, horizon $h$, points of interest $\{\mathbf{POI}\}$
 1: $\{q^*, \dot{q}^*\} = \mathbf{timeoptimalPlan}(q)$
 2: SAFE = TRUE
 3: repeat
 4:     $\{q^*_{h_i}, \dot{q}^*_{h}\} = \mathbf{recHorizon}(q^*, \dot{q}^*, h)$
 5:     $\{\text{SAFE}, \bar{v}_{h,\mathbf{POI}}\} = \mathbf{safetyEval}(q^*_h, \dot{q}^*_h, \{\mathbf{POI}\})$
 6:     if SAFE then
 7:         $\mathbf{sendtoRobot}(q^*)$
 8:     else
 9:         $\{q^*_S, \dot{q}^*_S\} = \mathbf{safeReplan}(q^*, \bar{v}_{h,\mathbf{POI}}, \{\mathbf{POI}\})$
10:         $\mathbf{sendtoRobot}(q^*_S)$
11:         $\{q^*, \dot{q}^*\} = \mathbf{perfRecover}(q^*_S)$
12:     h = moveHorizon()
13:     sleep until control period $T_c$ has elapsed
14: until End of Task
```

Given the desired path to be followed by the robot, first, an initial time-optimal trajectory, considering only the robot actuation limits and possible task-related constraints is generated (block *Time-Optimal Planning* in Fig. 2 and function **timeoptimalPlan** in Algorithm 1). After that, the safety of the generated time-optimal trajectory is assessed online by a *Safety Evaluation* module (block *Safety Evaluation* in Fig. 2 and function **safetyEval** in Algorithm 1) that, based on the robot planned trajectory and the human data from a visual perception system, provides speed constraints for given point of interests (POIs) along the robot structure. If the current time-optimal trajectory violates some of these constraints, the time evolution of the path is re-planned online by solving a minimum-time path-constrained problem that takes into account the safety constraints (block *Safe Replanning* in Fig. 2 and function **safeReplan** in Algorithm 1).

Finally, the new trajectory is sent to the robot, while a dedicated *Performance Recovery* module (function **perfRecover** in Algorithm 1) is used to generate a new time-optimal plan to be checked by the *Safety Evaluation* module, so as to restore higher speed motions as soon as the safety concerns are resolved.

In the following, a description of the design and working principles of the main blocks of our framework is presented.

A. Minimum-Time Trajectory Planning

The problem of generating optimal trajectories along a specified path can be efficiently tackled with state-of-the-art methods in case of limits on the joint velocities, accelerations, and torques. These methods translate the optimal control problem into a convex optimization problem [5]. However, in order to obtain smoother trajectories with less demanding actuator commands, we also include constraints on the jerk into the optimization problem[1]. The resulting optimization problem is described in the following.

Given a task and the related joint-space path $\boldsymbol{q} \in \mathbb{R}^n$, it can be typically represented by a function $\boldsymbol{\gamma}(s) \in \mathbb{R}^n$, where s is a monotonically increasing scalar parameter $s(t) \in [0, 1]$, and n is equal to the number of joints of the robot (see e.g. [12]). If we want to compute the trajectory that minimizes the total time T needed for following the given path, it is possible to write the following optimization problem

$$\begin{aligned} &\min_{T, s(\cdot)} T \\ &\{s(0), \dot{s}(0), s(T), \dot{s}(T)\} = \{s_0, \dot{s}_0, s_T, \dot{s}_T\} \\ &\dot{s}(t) \geq 0 \\ &\{\dot{\boldsymbol{q}}(t), \ddot{\boldsymbol{q}}(s(t)), \dddot{\boldsymbol{q}}(s(t)), \boldsymbol{\tau}(s(t))\} \in \boldsymbol{\mathcal{Q}}(s(t)) \\ &\text{for } t \in [0, T]. \end{aligned} \tag{1}$$

where $\boldsymbol{\mathcal{Q}}(s(t))$ is the constraint set for the joint velocities, accelerations, jerks, and torques[2]. However, w.r.t. classical torque-level formulations, by introducing jerk constraints, the optimization problem is no longer convex [12]. Solving a non-convex problem is usually computationally demanding, and the time needed for finding the optimal solution makes it practically unfeasible to use this approach for online trajectory planning. For these reasons, here we present a different formulation for (1), which considers the jerk constraints while still preserving the problem's convexity.

First, we reformulate Problem (1) defining the classic change of variables $b = \dot{s}^2$ and $a = b' = \partial b / \partial s$. Then, we use direct transcription to numerically solve it. Therefore, discretizing s with $K + 1$ grid points s_k, the problem of optimizing the motion of a robot moving along the discretized joint path can be written as

$$\min_{a_k, b_k} \quad \sum_{k=0}^{K-1} \frac{2\Delta s_k}{\sqrt{b_{k+1}} + \sqrt{b_k}} \tag{2a}$$

$$\begin{aligned} \text{s.t.} \quad & b_0 = 0 \text{ and } b_K = 0, && \text{(2b)} \\ & (b_{k+1} - b_k) = 2a_k \Delta s_k, && \text{(2c)} \\ & (b_K - b_{K-1}) = 2a_K \Delta s_K, && \text{(2d)} \\ & 0 \leq b_k \leq \bar{b}_k, && \text{(2e)} \\ & \underline{\boldsymbol{y}}_k \leq \boldsymbol{f}_k a_k + \boldsymbol{p}_k b_k \leq \bar{\boldsymbol{y}}_k, && \text{(2f)} \\ & \underline{\boldsymbol{y}}_K \leq \boldsymbol{f}_K a_K + \boldsymbol{p}_K b_K \leq \bar{\boldsymbol{y}}_K), && \text{(2g)} \\ & \underline{\dddot{\boldsymbol{q}}}_k \leq \dddot{\boldsymbol{q}}_k \leq \bar{\dddot{\boldsymbol{q}}}_k, && \text{(2h)} \\ & \text{for } k = 0, \ldots, K-1, && \text{(2i)} \end{aligned}$$

where we omit the dependency on s for the sake of clarity. We assumed b and a to be piecewise linear and piecewise constant, respectively, as in [5]. Additionally, $\boldsymbol{q}' = \partial \boldsymbol{q} / \partial s$, $\boldsymbol{q}'' = \partial^2 \boldsymbol{q} / \partial s^2$, $\Delta s_k = s_{k+1} - s_k$, $\bar{\dddot{\boldsymbol{q}}}$ and $\underline{\dddot{\boldsymbol{q}}}$ are the upper and lower bounds for the jerk, respectively, and

$$\boldsymbol{f}_k = \begin{bmatrix} \boldsymbol{m}_k{}^T & \boldsymbol{q}'^T_k \end{bmatrix}^T, \boldsymbol{p}_k = \begin{bmatrix} \boldsymbol{c}^T{}_k & \boldsymbol{q}''^T_k \end{bmatrix}^T,$$

$$\bar{\boldsymbol{y}}_k = \begin{bmatrix} (\bar{\boldsymbol{\tau}}_k - \boldsymbol{g}_k)^T & \bar{\ddot{\boldsymbol{q}}}_{\boldsymbol{k}}{}^T \end{bmatrix}^T, \underline{\boldsymbol{y}}_k = \begin{bmatrix} (\underline{\boldsymbol{\tau}}_k - \boldsymbol{g}_k)^T & \underline{\ddot{\boldsymbol{q}}}_k{}^T \end{bmatrix}^T,$$

where $\bar{\boldsymbol{\tau}}_k$, $\bar{\ddot{\boldsymbol{q}}}_k$, $\underline{\boldsymbol{\tau}}_k$, and $\underline{\ddot{\boldsymbol{q}}}_k$ are the upper and lower bounds for the joint torques and accelerations, respectively. For a formal definition of the vectors $\boldsymbol{m}$, $\boldsymbol{c}$, and $\boldsymbol{g}$ the interested reader can refer to [5]. From (2), the jerk $\dddot{\boldsymbol{q}}_k$ can be written using finite difference approximation

$$\dddot{\boldsymbol{q}}_k \approx \frac{\ddot{\boldsymbol{q}}_{k+1} - \ddot{\boldsymbol{q}}_k}{\Delta t_k} = (\sqrt{b_{k+1}} + \sqrt{b_k}) \frac{\ddot{\boldsymbol{q}}_{k+1} - \ddot{\boldsymbol{q}}_k}{2\Delta s^k}. \tag{3}$$

As expected, using this formulation constraints (2h) are non-convex. However, it is possible to substitute $\dddot{\boldsymbol{q}}_k$ with the following upper bound

$$\dddot{\boldsymbol{q}}_{ws,k} = \left(\sqrt{\bar{b}_{k+1}} + \sqrt{\bar{b}_k} \right) \frac{\ddot{\boldsymbol{q}}_{k+1} - \ddot{\boldsymbol{q}}_k}{2\Delta s_k} \tag{4}$$

where we replaced b_k and b_{k+1} with their upper bounds. In practice, we substituted Δt_k with its minimum value, constraining the worst-case jerk. Therefore, using this convex approximation for the constraints (2h), we obtain the convex problem

$$\min_{a_k, b_k} \quad \sum_{k=0}^{K-1} \frac{2\Delta s_k}{\sqrt{b_{k+1}} + \sqrt{b_k}} \tag{5a}$$

$$\begin{aligned} \text{s.t.} \quad & \text{(2b)} - \text{(2g)}, && \text{(5b)} \\ & \underline{\dddot{\boldsymbol{q}}}_k \leq \left(\sqrt{\bar{b}_{k+1}} + \sqrt{\bar{b}_k} \right) \frac{\ddot{\boldsymbol{q}}_{k+1} - \ddot{\boldsymbol{q}}_k}{2\Delta s_k} \leq \bar{\dddot{\boldsymbol{q}}}_k, && \text{(5c)} \\ & \text{for } k = 0, \ldots, K-1. && \text{(5d)} \end{aligned}$$

More details about the computational advantages and the optimality of the solution for this convex reformulation of the problem are reported in the Appendix.

[1]Smooth trajectories with reduced jerk also have the nice feature of increasing the operator's acceptance of pHRC [11].

[2]The formulation assumes that a joint space path has been determined. It can be obtained from the Cartesian path of the end-effector using any classical inverse kinematics algorithm. Therefore, any redundancy of the collaborative robot is not exploited while planning the time-optimal motion, but possibly during the inverse kinematics stage.

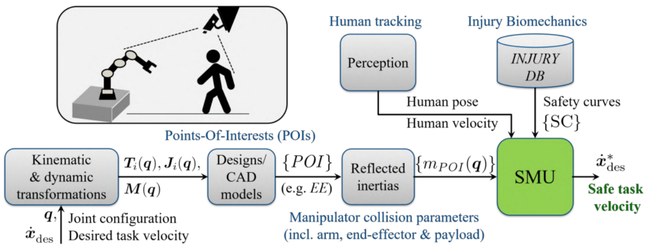
Human tracking
Injury Biomechanics
Perception
INJURY DB
Human pose
Human velocity
Safety curves
$\{SC\}$
Points-Of-Interests (POIs)
Kinematic & dynamic transformations
$\boldsymbol{T}_i(\boldsymbol{q}), \boldsymbol{J}_i(\boldsymbol{q}),$
$\boldsymbol{M}(\boldsymbol{q})$
Designs/ CAD models
$\{POI\}$
(e.g. EE)
Reflected inertias
$\{m_{POI}(\boldsymbol{q})\}$
SMU
$\dot{\boldsymbol{x}}^*_{\text{des}}$
Safe task velocity
$\boldsymbol{q}$, Joint configuration
$\dot{\boldsymbol{x}}_{\text{des}}$ Desired task velocity
Manipulator collision parameters
(incl. arm, end-effector & payload)

Fig. 3. The Safe Motion Unit (SMU) pipeline. This injury biomechanics-based approach ensures human safety for robot manipulators during pHRI.

B. Safety Evaluation

For the safety evaluation, we exploit a unified safety framework that relies on biomechanical injury information to ensure the human coworker safety in case of collision, the safe motion unit (SMU) [9]. The considered interaction/collision model, following the approach introduced in [9] and further elaborated in [13], focuses on capturing the instantaneous contact dynamics between two colliding objects. Using a scalar mass and its velocity (together with the curvature at impact/contact location), the energy exchange between the robot (possibly with additional payload) and an impacted human can be summarized in terms of the reflected robot dynamics along the chosen motion direction [14].

The SMU provides the velocity limits which guarantee that the traumatic effects of every possible contact phase may not exceed a certain level of minor injury (e. g. a bruise, contusion) or, more strictly, even no injury at all [9]. To achieve this, the robot reflected mass is evaluated in certain motion direction at a number of points of interest (POIs) along the robot structure. The curvature at these possible impact locations is encoded together with biomechanical injury information against the robot reflected mass and Cartesian velocity. Given the reflected mass in the current robot configuration and along its motion direction, a maximum safe relative speed can be evaluated from the corresponding safety curve. This upper relative speed limit can be calculated from a simple linear regression relationship

$$v_{\max}(m) = \operatorname{reg.lim}\left[c_1\left(i, \mathbf{p}_i\right) m + c_2\left(i, \mathbf{p}_i\right), v_1, v_2\right], \quad (6)$$

with $c_1(i, \boldsymbol{p}_i) < 0$ and $c_2(i, \boldsymbol{p}_i) > 0$ being the coefficients of the delimiting safety/injury curves for contact surface primitive i, m the reflected mass, and v_1, v_2 denoting the limits for cut-off minimum and maximum velocity. The contact primitive captures the geometrical information at the impact location, which is a key factor regarding how much energy is transferred to the impacted human tissue and the possible resulting injury. The complete detailed SMU algorithm can be found in [9], while a diagram summarizing the working principles of the SMU is reported in Fig.3. Some background information on the dynamic model used to characterize the collisions/interactions and the computation of the reflected mass are reported in the Appendix.

The SMU is the core of our **Safety Evaluation** module, implemented in a receding horizon fashion. The working principle of this module is described by Algorithm 2. Given

Algorithm 2 Safety Evaluation Module

```
safetyEval($q^*_{\text{h}}, \dot{\boldsymbol{q}}^*_{\text{h}}, \{\mathbf{POI}\}$)
1:  SAFE = TRUE
2:  $\bar{\boldsymbol{v}}_{\mathbf{POI}} = \emptyset$
3:  $\{\boldsymbol{p}, \dot{\boldsymbol{p}}\} = \mathbf{getPersonData}()$
4:  if $\boldsymbol{p} \neq \emptyset$ then
5:      return SAFE, $\bar{\boldsymbol{v}}_{\mathbf{POI}}$
6:  for POI $\in \{\mathbf{POI}\}$ do
7:      for $\{\boldsymbol{q}, \dot{\boldsymbol{q}}\} \in \{\boldsymbol{q}^*_h, \dot{\boldsymbol{q}}^*_h\}$ do
8:          $d = \mathbf{getDistance}(\boldsymbol{p}, \text{POI})$
9:          if $d \leq th$ then
10:             $\boldsymbol{v}_{\text{POI}} = \mathbf{getPOIvelocity}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \text{POI})$
11:             $m = \mathbf{getReflectedMass}(\boldsymbol{q}, \text{POI}, \boldsymbol{p}, \dot{\boldsymbol{p}})$
12:             $v_{max} = \mathbf{SMU}(m)$
13:             $\bar{\boldsymbol{v}}_{\text{POI}}.\mathbf{insert}(v_{max})$
14:             if $v_{max} < |\boldsymbol{v}_{\text{POI}}|$ then
15:                 SAFE = FALSE
16:     $\bar{\boldsymbol{v}}_{\{\mathbf{POI}\}}.\mathbf{insert}(\bar{\boldsymbol{v}}_{POI})$
17: return SAFE, $\bar{\boldsymbol{v}}_{\{\mathbf{POI}\}}$
```

the list of robot POIs $\{\mathbf{POI}\}$, at every control cycle, the SMU is used to evaluate the safety of the current planned trajectory over a user-defined monitoring time-horizon. By evaluating the safety of only a part of the planned trajectory, we reduce the computational burden of performing at each time step the evaluation for the complete path (even for points distant in time), given the current position of the person. On the other hand, deriving the safety constraints for the trajectory over a horizon instead of just the next point to send might possibly allow the robot to be more reactive, adapting its trajectory more smoothly and avoiding sudden (and possibly unfeasible) decelerations.

At each time step, the safety module receives the next joint positions and velocities of the trajectory within a horizon h. Then, it acquires information on the human presence from the perception system (function **getPersonData** in Algorithm 2). If no human is detected, the trajectory over the horizon is labelled as SAFE. If a person is detected, instead, it computes the maximum speed v_{max} for each trajectory point over the monitoring horizon and for each robot POI contained into the POI list $\{\mathbf{POI}\}$, using (6). To reduce the computational burden and avoid risks of unnecessary performance reductions, we introduced a safety threshold for the relative robot POI-human distance d. Indeed, if the person is not relatively close to the robot POI, it is not necessary to activate the SMU, and the maximum speed is set to the actual robot POI speed $|\boldsymbol{v}_{POI}|$. These safe values are then compared with the actual POI speed and, if for at least one robot POI and at least one trajectory point the safe value is below the actual robot POI speed, the planned trajectory is labelled as not safe, and sent to the **Safe Replanning** module for generating a new plan compliant with the new safety limits.

C. Time-Optimal Safe Replanning

The output of our safety module is a vector containing the maximum velocities for the robot POIs, computed on the given horizon h. In case the trajectory is not safe, it is necessary to adapt it to be compliant with the new limits. To do so, we designed a safe replanning module that, based on the same convex optimization approach presented in Section II-A, computes an optimal trajectory compliant with the safety limits, the robot actuation capabilities, and the task-related constraints. Thus, it is possible to replan the motion of the robot over the remaining part of the path so to be compliant with the safety speed constraints using the following modified version of Problem (5)

$$\begin{aligned}
\min_{a_k, b_k} \quad & \sum_{k=0}^{L-1} \frac{2\Delta s_k}{\sqrt{b_{k+1}} + \sqrt{b_k}} && \text{(7a)}\\
\text{s.t.} \quad & b_0 = \dot{s}_0^2 \text{ and } b_L = 0, && \text{(7b)}\\
& \text{(2c)} - \text{(2d)}, && \text{(7c)}\\
& 0 \leq b_k \leq \bar{b}_{\mathrm{S}}(s_k), && \text{(7d)}\\
& \text{(2f)} - \text{(2h)}, && \text{(7e)}\\
& \text{for } k = 0, \dots, L-1, && \text{(7f)}
\end{aligned}$$

where L is the length of the remaining part of the path $\boldsymbol{q}$, $\dot{s}_0$ is the current initial condition for the robot path velocity, and $\bar{b}_S$ are the safe-consistent speed constraints. The advantage of this formulation, w.r.t. less computational demanding velocity scaling techniques, is that it allows enforcing the safety constraints only over the unsafe part of the path, without reducing the performance unnecessarily. Furthermore, the computed optimal trajectory is compliant with the robot actuation capabilities and the task constraints.

It is worth noting that Problem (7) with the safety constraints is still convex, since the speed constraints are embedded into the upper bounds on the variable b. Indeed, for each value in $\bar{\boldsymbol{v}}_{h,POI}$, the constraints can be expressed as

$$\begin{aligned}
&|\boldsymbol{J}_v\dot{\boldsymbol{q}}| \leq |\bar{v}_{h,\mathrm{POI}}| \leftrightarrow (\boldsymbol{J}_v\dot{\boldsymbol{q}})^2 \leq (\bar{v}_{h,\mathrm{POI}})^2 \leftrightarrow \\
&\leftrightarrow (\boldsymbol{J}_v\boldsymbol{q}')^2\, b \leq (\bar{v}_{h,\mathrm{POI}})^2 \leftrightarrow b \leq \frac{(\bar{v}_{h,\mathrm{POI}})^2}{(\boldsymbol{J}_v\boldsymbol{q}')^2},
\end{aligned} \tag{8}$$

where $\boldsymbol{J}_v$ is the Jacobian matrix for linear motions associated with the robot POI, and where we exploited the equality $\dot{\boldsymbol{q}} = \boldsymbol{q}'\sqrt{b}$. Thus, the constraints on $v_{\mathrm{h}_i,\mathrm{POI}}$ can be rewritten into the form $b \leq \bar{b}_{\mathrm{S}}$, directly acting as upper bounds on b. Even for the case of multiple robot POIs, it is sufficient to consider only the most restrictive upper bound for each s over the monitoring horizon

$$\bar{b}_{\mathrm{S}}(s) = \min\left\{\bar{b}(s),\, \min_{\{\mathbf{POI}\}}\left(\frac{\bar{v}_{\mathrm{h}_i,\mathrm{POI}}}{\boldsymbol{J}_v\boldsymbol{q}'(s)}\right)^2\right\},$$

while $\bar{b}_{\mathrm{S}}(s) = \bar{b}(s)$ for the points outside the horizon.

D. Performance Recovery

Once the new plan has been obtained, the next point of the trajectory q_S^* is sent to the robot and the monitoring window is shifted ahead to the next point. However, before providing the **Safety Evaluation** module the next horizon, a **Performance Recovery** procedure is performed to compute a new optimal trajectory that does not consider safety speed constraints. This is achieved by solving an optimization problem with the same structure as Problem (5), optimizing over the path $\boldsymbol{q}_{\mathrm{S}}^*$, with consistent initial conditions. The new optimal trajectory is then provided as the input of the **Safety Module**. The aim of this recovery procedure is to give to the **Safety Evaluation** module the fastest dynamically consistent trajectory at every cycle. Therefore, we are able to restore high-speed motions as soon as the safety issue that triggered the safe replanning has been resolved, or to evaluate the safety limits over the fastest dynamically possible trajectory. The advantages given by this module will be more clear in the next section, where we will show with simulations that without the performance recovery, the planned trajectory might turn out to be unnecessarily conservative and the performance of the robotic system can be enhanced using the recovery action.

III. Simulations

In this section, the effectiveness of our proposed framework is assessed through simulations. The robot employed is a Franka Emika Panda manipulator[3] equipped with a Pisa/IIT SoftHand as end-effector. The task assigned to the robot is to follow a circular trajectory. We simulate the presence of a human operator in the robot workspace, following an assigned trajectory $\boldsymbol{p}(t)$. For the simulated case, only one robot POI, placed at the center of the SoftHand, is considered, while the safety curves used by the SMU encode exemplary injury/safety information of collision incidents against the human chest. The robot motion along the specified circular path is planned using four different approaches, whose performance in terms of safety and trajectory times are then compared and discussed. For all the experiments, the simulated human worker follows the same trajectory $\boldsymbol{p}(t)$. Note that the trajectory followed by the human is not known in advanced by the robot, but we assume that a perception system can provide the actual human data at every cycle. For the limits on the robot velocity, acceleration, and jerk, we used the nominal values provided by Franka[4].

In the first scenario, named ***Time-Optimal***, the motion of the robot is planned to fully exploit its actuation capabilities, solving the minimum-time optimization problem (5) presented in II-A. The safety module evaluates the safety of the optimal plan, neglecting the online safe-replanning. The proposed scenario has been chosen to highlight as the time-optimal trajectory can violate the safety constraints.

The second method, named ***Conservative***, solves the same minimum-time optimization problem as in the time-optimal approach, but the limits on the joint velocities have been scaled to the point at which the planned trajectory is considered safe by the safety module. This scenario is instrumental for comparison of a safe but conservative trajectory.

[3]https://www.franka.de

[4]https://frankaemika.github.io/docs/control_parameters.html

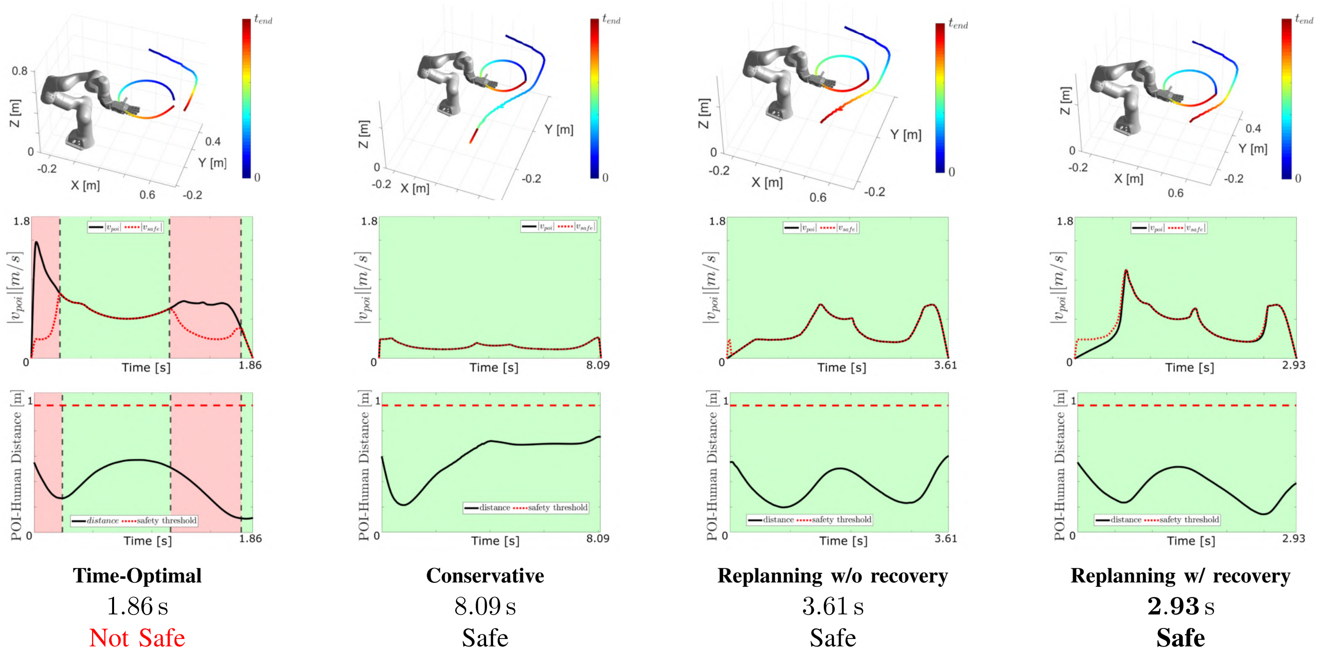

Fig. 4. Results of the simulation with different trajectory planning approaches. For each column, from top to bottom: visualization of the trajectory of both the POI and the person; plot of the planned module of the POI speed, solid line, and the corresponding safety limit computed by the SMU, dashed line (red area corresponds to violations of the safety constraint); time evolution of the relative POI-Human distance, solid line, and the corresponding safety threshold, dashed line.

The third method, named ***Replanning without Recovery***, uses our proposed approach, but without using the performance recovery step. In contrast, the fourth method, named ***Replanning with Recovery***, uses our proposed approach with the inclusion of the performance recovery step.

For the third and fourth methods, the planning loop runs at $25\,\mathrm{Hz}$, the monitoring horizon length has been set to $0.32\,\mathrm{s}$, and the safety threshold for the robot POI-human distance has been set to $1\,\mathrm{m}$. For all the four cases, the optimization problems presented in II have been implemented in C++ using CasADI [15] and the NLPs have been solved by the interior point solver IPOPT [16] with ma57-HSL linear solver[5]. The results of the simulations for the four scenarios, in terms of trajectory, speed of the robot POI, safe speed computed by the SMU, and relative robot POI-human distance are reported in Fig. 4. The total trajectory time and the safety of the planned trajectories as evaluated by the **Safety Module** are also shown. We recall that a plan is considered to be **unsafe** if, for at least one trajectory point, the speed of the robot POI is greater than the safe speed maximum limit.

As expected, the motion planned using the Time-Optimal approach is the fastest but turns out to be not safe since we do not consider the human presence. Indeed, there are two intervals, highlighted in red in the first column of Fig. 4, where the planned robot POI speed is greater than the safe one. It is worth noting that, even if the relative robot POI-human distance is always below the 1 meter safety threshold, there are intervals where the time-optimal trajectory is still safe (highlighted in green), due to the non-conflicting direction of motion between the robot and the human. On the other hand, the motion planned using the Conservative approach is able to ensure safety but at the expense of a larger total trajectory time (see second column of Fig. 4). Indeed, to obtain a safe trajectory without using any online replanning strategy, we needed to scale speed limits by six times. Our approach, instead, can overcome those issues, thanks to the safe replanning strategy implemented. With and without the recovery action, the total time is higher than the one obtained with the Time-Optimal strategy, but the planned trajectory is considerably faster than the one provided by the Conservative approach, since the robot is able to adapt online its motion to the presence of the human operator (see third and fourth column of Fig. 4). It is worth noting as the presence of the performance recovery stage allows the robot to increase the performance further, reducing the time needed to complete the task by about 18%, while ensuring safety. Clearly, this result comes at the expense of the computational effort since we need to solve two optimization problems at every cycle, and the safe replanning is expected to be activated more often. Indeed, with the Performance Recovery implemented, the safe replanning is activated 45 times against just the 23 of the replanning approach without the recovery step. Hence, without the performance recovery implemented, the system solves 23 optimization problems. In contrast, since the system executes the performance recovery action for each safe replanning, it is necessary to perform 90 optimizations when the performance recovery is implemented.

[5]"HSL. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/"

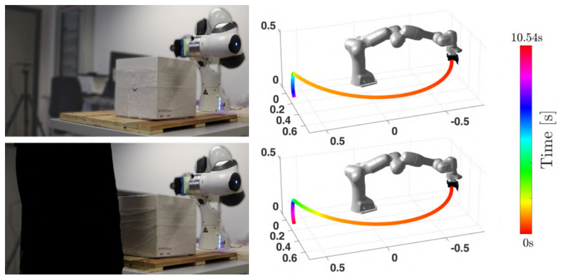

Fig. 5. Trajectory of the POI without (top) and with (bottom) the human operator.

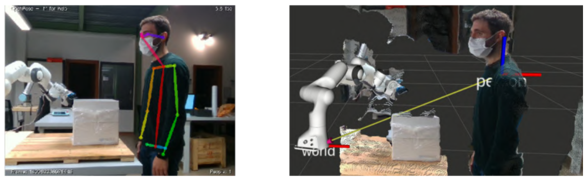

Fig. 6. Example of human body keypoints detection using OpenPose and corresponding 3D position of the chest.

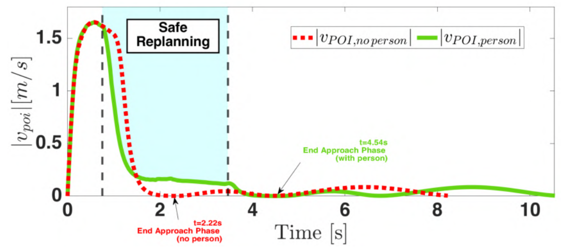

Fig. 7. Speed profile for the robot POI when the person is present (green solid line) and when is not present (red dashed line). The area highlighted represents the interval where the safe replanning procedure is performed.

TABLE I
TIME FOR EXECUTING THE WHOLE TRAJECTORY AND THE TWO PHASES OF THE TASK USING THE SAFE HUMAN-AWARE PLANNER.

	Total Time	Approach Time	Cutting Time
Without Person	8.22 s	2.22 s	6.00 s
With Person	10.54 s	4.54 s	6.00 s

IV. Experimental Validation

Our approach has also been evaluated experimentally on a scenario relevant to intralogistics, i.e., an autonomous single-object unwrapping task. Unwrapping, i.e., removing the plastic film enclosing the goods, is a crucial step of the intralogistic flow and is still mainly performed by human workers, with research focused on the development of autonomous robotic solutions [17]. Given the nature of the task, involving cutting tools and blades, it is of high importance to provide these robotic solutions with planning systems that guarantee human safety in shared environments.

We used a fixed-base autonomous unwrapping robot, composed of a Franka Emika Panda arm with a custom-designed cutting end-effector: a concealed round actuated blade, to avoid direct contact with the objects, ease the film's engagement, and provide a safer tool to be used in a human-robot shared environment, presented in [17]. The robot has to perform an unwrapping task on a single object brought to the cutting station by a human operator. The path that the robot has to follow is composed by two main parts. A first approach phase where the robot moves from the homing position, see Fig. 5, near the object to unwrap. The second phase is the actual cutting action, in which the robot should move slowly to avoid damaging the object and ensuring the film engagement.

We propose two different scenarios for the experimental validation. In the first scenario, the human operator places the object in a specified cutting position and then moves away from the robot. In the other case, instead, the human operator, after placing the object, stands still close to it. For the implementation of the **Safety Evaluation** module we considered only one robot POI, on the cutting end-effector, and the safety curves used by the SMU are again related to exemplary collision outcomes with the human chest. The human perception model used for the experiment is based on OpenPose [18]. OpenPose is a real-time multi-person visual perception software to detect human body, hand, facial, and foot keypoints. In our setup, we used OpenPose together with a depth camera, an Intel® Realsense™ D415, to extract the coordinates of the human keypoints and then of the chest[6]. An example of the human keypoints detected by OpenPose and the extracted 3D position of the chest of the human operator during one of the experiments is reported in Fig. 6. The planning algorithm runs at 25 Hz, with a monitoring horizon of 0.12 s and a safety threshold of 0.6 m. In Fig. 7 the speed profile for the robot POI for the two cases are reported, while Table I reports the time for the whole trajectory and the two task phases. As expected, the total time needed to complete the task increases if the person is present in the scene, indeed, as shown in Fig. 7 (area highlighted in cyan), the presence of the human operator triggers the activation of the safe replanning, reducing the robot POI speed and therefore increasing the total time needed to complete the approach phase. On the other hand, if there is no person our planning method allows maximizing the performance, thanks to the time-optimal planning module used to generate the initial plan. It is worth noting that the time for the cutting phase is not affected by the presence of the human operator since it is safe by design.

V. Conclusion

In this paper, we presented a trajectory planning algorithm to plan fast and safe motions for collaborative robots in shared environments. The algorithm is based on an iterative procedure that can ensure the nearby human safety by re-planning online the time evolution of the motion of the robot

[6]It is worth noting as the current setup for the perception system might allow retrieving the position of body parts other than the chest. Therefore, provided that suitable injury data for different body parts are available, the SMU could compute the safe speed based on the closest body part to the robot.

along the path based on the human data. The simulations and experimental validations have shown the effectiveness of the approach in producing safe but still efficient trajectories for the robot. Future works will focus on testing the performance in case multiple robot POIs and human body parts are used by the SMU when computing the safety constraints.

Appendix

A. Convex jerk-limited planning algorithm

We performed simulations solving both Problem (2) and (5) for a set of random trajectories using a Franka Emika Panda arm. Each trajectory has been discretized using 200 grid points for the path variable s. The problems have been implemented in Matlab using CasADI with the ma57-HSL linear solver for IPOPT, and with the same initial conditions. In the following table, we reported statistics on the solutions to the problems and the solving times

$\bar{\hat{\dddot{q}}}_{NC}$	$\bar{\hat{\dddot{q}}}_{C}$	$\Delta\bar{t}_{opt}$	$\bar{t}_{IPOPT}^{NC}$	$\bar{t}_{IPOPT}^{C}$
1.0000	0.9902	5.7170%	1128.4ms	82.235ms

The superscripts NC and C denote the results for problem (2) and (5), respectively, $\hat{\dddot{q}}$ is the normalized peak for the jerk signal, i.e., assuming symmetric bounds, $\hat{\dddot{q}} = \max_i\{|\dddot{\boldsymbol{q}}_i[t_k]|/\bar{\dddot{\boldsymbol{q}}}_i\}$, and $\bar{\hat{\dddot{q}}}$ represents the average over the different trajectories of the normalized peak for the jerk. The values $\bar{t}_{IPOPT}^{NC}$, $\bar{t}_{IPOPT}^{C}$, are the average time required by IPOPT to solve Problem (2) and (5), respectively, and $\Delta\bar{t}_{opt}$ is the average difference between the optimal times, computed as $t_{opt}^{C} - t_{opt}^{NC}$. As expected, the solutions found solving (5) are suboptimal. On average, the final time is 5.7% larger, and the jerk never completely saturates. However, problem (5) is much faster to be solved, with an average time-reduction of around 93%.

B. Backgrounds on the collision/interaction model

The rigid robot dynamic model can be expressed as [19]

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{ext}}, \tag{9}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ are the generalized link 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 torque 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[7]. The so-called reflected mass is the scalar mass perceived at the manipulator end-effector along certain motion direction during an impact with the robot [21]. It can be calculated in a normalized Cartesian direction $\boldsymbol{u} \in \mathbb{R}^3$ as

$$m_u(\boldsymbol{q}) = \left[\boldsymbol{u}^{\mathsf{T}}\boldsymbol{\Lambda}_v^{-1}(\boldsymbol{q})\boldsymbol{u}\right]^{-1},$$

[7]Most of the collaborative lightweight robots have flexible joint dynamics coupling the motor and link-side dynamics via elastic joint torque. Since the corresponding joint-side and link-side inertias are decoupled from each other during collisions, only the link-side inertia is required to determine the reflected mass [20].

where $\boldsymbol{\Lambda}_v^{-1}(\boldsymbol{q})$ is the translational pseudo kinetic energy matrix. It can be extracted from the partitioned inverse of the kinetic energy matrix [21]

$$\boldsymbol{\Lambda}^{-1}(\boldsymbol{q}) = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{J}^{\mathsf{T}}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{\Lambda}_v^{-1}(\boldsymbol{q}) & \overline{\boldsymbol{\Lambda}}_{v\omega}(\boldsymbol{q}) \\ \overline{\boldsymbol{\Lambda}}_{v\omega}^{\mathsf{T}}(\boldsymbol{q}) & \boldsymbol{\Lambda}_\omega^{-1}(\boldsymbol{q}) \end{bmatrix}. \tag{10}$$

References

[1] A. Cherubini *et al.*, "Collaborative manufacturing with physical human–robot interaction," *Robotics and Computer-Integrated Manufacturing*, vol. 40, pp. 1–13, 08 2016.

[2] P. Tsarouchi *et al.*, "Human–robot interaction review and challenges on task planning and programming," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 8, pp. 916–931, 2016.

[3] A. Ajoudani *et al.*, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, no. 5, pp. 957–975, 2018.

[4] A. M. Zanchettin *et al.*, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 882–893, April 2016.

[5] D. Verscheure *et al.*, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[6] V. Villani *et al.*, "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, vol. 55, pp. 248–266, 2018.

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

[8] P. Aivaliotis *et al.*, "Power and force limiting on industrial robots for human-robot collaboration," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 346–360, 2019.

[9] S. Haddadin *et al.*, "On making robots understand safety: Embedding injury knowledge into control," *The International Journal of Robotics Research*, vol. 31, no. 13, pp. 1578–1602, 2012.

[10] N. Lucci *et al.*, "Combining speed and separation monitoring with power and force limiting for safe collaborative robotics applications," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6121–6128, 2020.

[11] P. A. Lasota and J. A. Shah, "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration," *Human factors*, vol. 57, no. 1, pp. 21–33, 2015.

[12] A. Palleschi *et al.*, "Time-optimal path tracking for jerk controlled robots," *IEEE Robotics and Automation Lett.*, vol. 4, no. 4, pp. 3932–3939, 2019.

[13] N. Mansfeld *et al.*, "Safety map: A unified representation for biomechanics impact data and robot instantaneous dynamic properties," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1880–1887, Jul. 2018.

[14] M. Hamad *et al.*, "The role of robot payload in the safety map framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[15] J. A. E. Andersson *et al.*, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.

[16] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[17] C. Gabellieri *et al.*, "Towards an autonomous unwrapping system for intralogistics," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4603–4610, 2019.

[18] Z. Cao *et al.*, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[19] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[20] S. Haddadin *et al.*, "Requirements for safe robots: Measurements, analysis and new insights," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1507–1527, 2009.

[21] O. Khatib, "Inertial properties in robotics manipulation: An object-level framework," *The International Journal of Robotics Research*, vol. 14, no. 1, pp. 19–36, 1995.