A Safety Kinodynamic Planning Framework for Human-Robot Collaboration

Andrea Pupa[1], Mohammad Arrfou[2], Gildo Andreoni[2] and Cristian Secchi[1]

***Abstract*— In recent years, the safety standards have been updated in order to regulate the new work environments, characterized by the presence of humans and robots that collaborate. The simple application of these standards translates in great reduction of the robot speed. In order to ensure safety for the human operator and allow the robot a more efficient behavior, a two layer architecture is for trajectory planning and scaling is presented. The first layer is responsible of planning a collision-free trajectory, adapting it in real-time. The second layer, instead, explicitly consider the safety standards to ensure a safe robot velocity.**

I. Introduction

The diffusion of collaborative robotics in the industrial settings has created the need update the safety standards in order to address the new collaborative scenarios [1]. Specifically, the ISO 10218-1 and the ISO 10218-2 [2], [3] standards define four different collaborative modes: *safety-rated monitored stop* (SMS), *hand guiding* (HG), *speed and separation monitoring* (SSM) and *power and force limiting* (PFL). Moreover, the technical specification ISO/TS 15066 [4] provides further information to assess the risk for each collaboration mode. In industrial scenarios, the SSM is typically implemented. In this collaborative mode the speed of the robot is reduced according to the relative human-robot velocity and position. However, this approach is overly conservative, since the robot speed should not be limited if its motion is directed away from the human.

Different approaches were presented in the literature to deal with human safety and collision avoidance in a human-robot collaboration (HRC) scenario. In [5] the authors propose a real-time solution to evaluate the future human occupancy and scale the robot speed accordingly, ensuring safety. In [6] an optimization that proportional reduces the robot speed while ensuring safety is presented.

Reducing the speed of the robot is not always the best solution. Sometimes it might be better to change the pre-planned path. In [7] the authors exploit the concept of static and kinetostatic danger field on a mobile robot in order to avoid collision. In [8] a strategy based on virtual fixture, which combine attractive and repulsive potential field, in a teleoperated environment has been implemented. However, potential fields can easily cause the system to be stuck in local minima.

For this reason, optimization-based algorithms have been exploited to ensure safety by applying the minimum correction to the desired path. In [9] an optimization problem is solved in real-time in order to force the robot to stay inside a safe set, evaluating the variation of a safety index. In [10], the authors propose the use of control barrier functions around the robot body to maintain a collision-free trajectory while fulfilling the ISO/TS 15066.

Solving an optimization algorithm could be computationally challenging, especially in a real industrial scenario where the number of obstacles to be considered is very high. In [11], [12] the authors use kinodynamic rapidly-exploring random tree (RRT) to plan collision free trajectory under kinodynamic constraints. However, these solutions are not suitable for constraints that change in real time based, as the safety constraints.

In this paper we propose a two-layers framework for trajectory planning and velocity scaling for HRC scenario that ensures safety for the human operator by explicitly considering safety regulations. Given a desired configuration to reach, a trajectory planner layer computes and adapts online the trajectory that the robot has to follow. The trajectory scaling layer scales the robot velocity ensuring safety for the human operator. Moreover, in order to avoid drastic drops of the robot velocity, the trajectory scaling can request for a replan of a new trajectory, increasing the robot performances.

The main contributions of this paper are:

- A novel adaptive framework for trajectory planning and scaling that takes into account the high dynamicity of the environment, adapting in real-time the trajectory.
- A strategy for trajectory scaling that is computationally cheap, i.e. suitable for real industrial application, and that explicitly considers the kindoynamic safety constraint.

Further detailed on this work can be found in [13].

II. Problem Statement

We consider a HRC application where a robot manipulator with n joints has to move from an initial configuration $q(t_i) = q_i \in \mathbb{R}^n$ to a desired final configuration $q(t_f) = q_f \in \mathbb{R}^n$ in order to execute a task. The trajectory $q(t) \in \mathbb{R}^n$ that the robot has to perform can be decomposed with a path-velocity decomposition:

$$q(t) = q(s(t)) \tag{1}$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 818087 (ROSSINI).

[1] Andrea Pupa and Cristian Secchi are with the Department of Science and Method of Engineering, University of Modena and Reggio Emilia, Italy. E-mail: {andrea.pupa, cristian.secchi}@unimore.it

[2] Mohammad Arrfou and Gildo Andreoni are with Datalogic S.p.A., Italy. E-mail: {mohammad.arrfou, gildo.andreoni}@datalogic.com

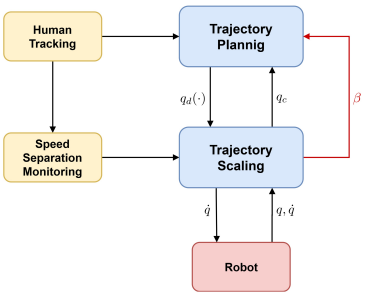

Fig. 1. The overall architecture. The blue blocks represent the two layers. The yellow blocks, instead, symbolize the strategies implemented to provide richer information to the layers. The red block represents the agent.
The black lines symbolize the data exchange, while the red one constitutes the signal that request for a replan of a new trajectory.

where s is the curvilinear abscissa that parametrizes the geometrical path $q(s)$. The variation of s represents the time law of the desired path (i.e. the velocity profile).

Differentiating (1) we obtain:

$$\dot{q}(t) = q'(s)\dot{s} \tag{2}$$

where $q'(s)$ is the vector tangent to the desired path, while $\dot{s}$ constitutes the magnitude of the joint velocity.

The shared workspace is equipped with a monitoring system that allows to track the human movements and estimate the human speed.

In this work, we aim at designing a safety kinodynamic architecture that:

- Computes a nominal trajectory that is always collision-free, replanning a new trajectory when the actual trajectory becomes infeasible.
- Starting from the nominal trajectory, scales the robot velocity according to the safety standards.

III. Safety Kinodynamic Architecture

The proposed dynamic trajectory planning and scaling strategy can be represented by the architecture in Fig. 1, where two main layers can be distinguished:

1) **The trajectory planning layer**. It is responsible of generating the initial nominal trajectory that the robot can execute at maximum speed. Subsequently, it continuously adapts this trajectory exploiting the human tracking information.
2) **The trajectory scaling layer**. It is responsible of scaling the robot velocity along the planned path, fulfilling the safety standards.

Once the trajectory planning computes the initial nominal trajectory, it sends it to the trajectory scaling and it remains active until the robot reaches the desired final configuration q_f. The trajectory planning layers does not take into account the safety regulation, i.e. it computes a trajectory that the robot could ideally execute at maximum speed.

The trajectory scaling firstly applies a path-velocity decomposition to the desired trajectory as shown in (1) and (2). Subsequently, it computes online the optimal scaled velocities in order to satisfy the constraint imposed by ISO/TS 15066.

During the execution of the motion, mutual communication between the two layers is enabled. The trajectory planning exploits the human tracking information and replans a new trajectory when the previous one becomes infeasible, as explained in Sec. III-A. The trajectory scaling immediately parameterizes the new trajectory and starts following the new path. At each iteration, it returns to the trajectory planning the actual state of the trajectory. Moreover, when the scaling factor decreases too much, the trajectory scaling sends a signal to the trajectory planning requesting for a new trajectory to be planned, as it is becoming inefficient, see Sec. III-B.

A. Trajectory Planning

The role of this layer is to find a trajectory $q_d(t)$ for the robot that is collision-free and that the robot can execute at maximum speed. Moreover, the trajectory planning aims at continuously maintaining a collision-free trajectory, adapting it online when required.

The trajectory planning is implemented according to the pseudo-code reported in Alg. 1.

Algorithm 1 TrajcetoryPlanning()

```
1:  Require: q_i, q_f, n
2:  q_d(·) ← plan(q_i, q_f)
3:  send(q_d(·))
4:  q_c ← q_i
5:  while q_c ≠ q_f do
6:      h ← horizon(q_c, n)
7:      for i = 1 : n do
8:          if not_feasible(h(i)) then
9:              q_d(·) ← replan(q_d(·), q_d(h(i − 1)), q_f)
10:             break
11:         end if
12:     end for
13:     if β then
14:         q_d(·) ← replan(q_d(·), q_c, q_f)
15:     end if
16:     update_q_c()
17: end while
```

The trajectory planning needs as input the initial and the final configuration, respectively q_i and q_f, and the length of the horizon trajectory that will be checked n (Line 1). It immediately plans the maximum speed trajectory $q_d(\cdot)$ that the robot could perform, sending it to the trajectory scaling layer, and it initialize the current state q_c as q_i (Lines 2-4). From this point the algorithm starts to loop until the entire trajectory has been executed (Line 5). In the loop, the dynamic planner first creates the horizon h, which represents the set of the future configuration that are analyzed to check the feasibility of the trajectory (Lines 6-8). In case an infeasible configuration is found a new feasible trajectory is planned through the function replan (Line 9). Subsequently, the dynamic planning algorithm reads if the trajectory scaling

layer has requested to replan a new trajectory, as described in Sec. III-B. If that is the case, a new trajectory starting from the actual configuration is computed (Line 14). Lastly, the actual configuration is updated exploiting the information coming from the trajectory scaling in (6) (Line 16).

The replan algorithm is presented in Alg. 2.

Algorithm 2 replan()

1: **Require:** $q_d(\cdot), q_{rp}, q_f$
2: $q_{new}(\cdot) \leftarrow \mathsf{plan}(q_{rp}, q_f)$
3: $q_d(\cdot) \leftarrow \mathsf{merge}(q_d(\cdot), q_{new}(\cdot))$
4: $\mathsf{send}(q_d(\cdot))$
5: $\mathsf{return}(q_d(\cdot))$

The algorithm takes as input the actual planned trajectory $q_d(\cdot)$, the starting configuration of the new trajectory q_{rp} and the final desired configuration q_f (Line 1). It plans a new trajectory $q_{new}(\cdot)$ that goes from the starting configuration q_{rp} to the desired goal q_f and it merges the new trajectory with the old one (Lines 2-3). Lastly, the merged trajectory $q_d(\cdot)$ is sent to the trajectory scaling and returned to the dynamic planner (Lines 4-5).

B. Trajectory Scaling

Starting from the output of the dynamic planner, the goal of the trajectory scaling is to regulate the robot velocity without violating the safety standards. The trajectory scaling aims at scaling only the magnitude of the velocity $\dot{s}$, following exactly the collision-free path planned.

This is achieved in two steps. Firstly, by applying the path-velocity decomposition as shown in (1)–(2). Secondly, by solving the following optimization problem:

$$\begin{aligned} & min - \alpha \\ & \text{subject to} \\ & J_{r_i}(q) q'_d(s) \dot{s} \alpha \leq V_{max} \quad \forall i \in \{1, \dots, n\} \\ & \dot{q}_{min} \leq q'_d(s) \dot{s} \alpha \leq \dot{q}_{max} \\ & \ddot{q}_{min} \leq \tfrac{q'_d(s)\dot{s}\alpha - \dot{q}}{T_r} \leq \ddot{q}_{max} \\ & 0 \leq \alpha \leq 1 \end{aligned} \tag{3}$$

$\alpha \in \big[0, 1\big]$ is the optimization variable and represents the scaling factor. $J_{r_i}(q) \in \mathbb{R}^{1 \times n}$ is a *modified jacobian* that takes into account only the scalar velocity towards the human operator of the i-th link. This modified version of the jacobian is required as the velocity constraint imposed by the ISO/TS 15066 limits only the velocity that reduce the human-robot distance. V_{max} is the velocity limit imposed by the ISO/TS 15066. $\dot{q}_{min} \in \mathbb{R}^n$ and $\dot{q}_{max} \in \mathbb{R}^n$ are the joint velocity lower bounds and the joint velocity upper bounds, respectively. While $\ddot{q}_{min} \in \mathbb{R}^n$ and $\ddot{q}_{max} \in \mathbb{R}^n$ are the acceleration limits. $\dot{q} \in \mathbb{R}^n$ is the actual robot velocity and T_r is the robot execution time.

The modified jacobian $J_{r_i}(q)$ is expressed as:

$$J_{r_i}(q) = \vec{n}_i^T \begin{bmatrix} J_i(q) & \bar{0} \end{bmatrix} \tag{4}$$

where $\vec{n}_i = \{n_{x_i}, n_{y_i}, n_{z_i}, 0, 0, 0\}$ is the versor representing the direction that goes from the i-th robot link to the human. $J_i(q) \in \mathbb{R}^{6 \times i}$ is the i-th jacobian, i.e. the jacobian matrix that relates the firsts i joint velocity to the linear and angular velocity of the i-th link, and $\bar{0} \in \mathbb{R}^{6 \times (n-i)}$ is a matrix with all zero elements.

The optimization problem (3) is a convex problem and computationally cheap. Moreover the problem has always a feasible solution. When the human operator is very far from the robot, the robot is allowed to move at the desired speed, i.e. $\alpha = 1$ that is the maximum speed as seen in Sec. III-A. When the human approaches the robot, the safety standards require to decrease the velocity until, in the worst case, stopping the robot. This is guaranteed by the solution $\alpha = 0$.

The output of the trajectory scaling is then used to send the desired velocity to the robot:

$$\dot{q} = q'_d(s) \dot{s} \alpha \tag{5}$$

while the new robot configuration that is sent to the dynamic planner (see Alg.1, Line 16) will be:

$$q_c = q_c + q'_d(s) \dot{s} \alpha T_r \tag{6}$$

However, greatly reduce the robot velocity is a very conservative strategy and it strictly decreases the overall efficiency. Sometimes it could be more convenient for the robot to move away from the human and execute another trajectory. For this reason it has been implemented a step signal that requests to the dynamic planner the replan of new trajectory:

$$\beta = \begin{cases} 1 & \alpha \leq \alpha_{min} \\ 0 & otherwise \end{cases} \tag{7}$$

where α_{min} is a predefined threshold and represents the lower desired bound for the scaling factor.

When β is high a replan request is sent to the trajectory planning and a new trajectory is planned (see Alg. 1, Line 13).

IV. Experiments

The proposed two-layers framework has been experimentally validated on a Pilz PRBT, a 6-DoF manipulator for industrial application. We decided to exploit six OptiTrack Prime^x cameras to track the movements of the human right arm. A complete setup of the experiments is shown in Fig. 2. All the software components were developed using ROS Melodic Morenia. The dynamic planner layer is based on the RRT-Connect algorithm, while the trajectory scaling layer exploits the C code generated by CVXGEN. For simplicity, the modified jacobian J_{r_i} is applied only to the end-effector and the versor $\vec{n}_i$ is the versor of minimum distance between the $i-th$ robot link and the human operator arm.

In the experiment the robot has to go continually from the start configuration