An MPC Framework for Online Motion Planning in Human-Robot Collaborative Tasks

Marco Faroni, Manuel Beschi, Nicola Pedrocchi

Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato
Consiglio Nazionale delle Ricerche – Via Alfonso Corti 12, Milano, Italy
Email: {marco.faroni; manuel.beschi; nicola.pedrocchi}@stiima.cnr.it

***Abstract*—Human robot collaboration requires new planning strategies to guarantee an efficient and safe coexistence of robots and humans in the workspace. We propose a framework based on a model predictive control approach to trajectory scaling and inverse kinematics. The online modification of the velocity override slows down the task to ensure safety and the redundancy of the system is exploited to maximize the distance from the operator. Experimental results on a 7-degree-of-freedom robotic system prove the effectiveness of the method.**

I. Introduction

Human Robot Collaboration (HRC) is a trending topic nowadays, both for the academic and the industrial communities. Collaboration between robots and humans in the execution of tasks combines the strengths of both, increasing the flexibility and the throughput of the process. However, this combination also brings about safety issues, which translate into new planning requirements. Safe motion planning means, for example, avoidance or energy limitation of the collisions and requires an intensive use of runtime modifications of the motion and re-scheduling of tasks. Industrial applications have often taken the easy way out: robots are still fenced in cells and static environments and the presence of operators causes the stop of the robot or the reduction of the speed of execution of the task. Although safety is guaranteed, the result is an alleged collaboration that does not exploit the full potential of the parts, leaving the wish of a holistic human-robot combination unattended [1]. Better solutions should be able to react more dynamically to the presence of humans and changes in the environment. New strategies tend to modify the robot nominal tasks based on current information about the system. A typical approach consists in applying virtual repulsive forces to the robot to move it away from the operator [2], [3]. A trajectory scaling method for safe HRC is proposed in [4]: given a Cartesian trajectory, the timing law is modified online to match safety requirements. Similarly, [5] uses trajectory scaling for safe HRC in case of multi-robot systems.

In general, trajectory scaling consists in modifying the timing law of a given desired trajectory. It is typically adopted to preserve the geometrical path in case the nominal trajectory exceeds the constraints of the system. Relevant examples can be found in [6], [7], [8]. In particular, [8] solves the inverse kinematics of redundant manipulators as a Quadratic Program (QP) where a scalar variable, which multiplies the current task velocity, acts as a scaling factor.

Recently, [9] improved task scaling for redundant manipulators by means of a Model Predictive Control (MPC) approach: the scaling principle described in [8] is applied along the predictive horizon and this gives significant improvements of the average scaling and the deformation of the path.

In this work, we exploit the predictive method [9] to build an HRC-MPC framework. Such framework lies between the planning and the control layers most robot architectures realize. Given that trajectory planners typically output a collision-free joint-space trajectory expressed as a sequence of positions and velocities with the corresponding timestamps (instead of parametric curves and timing laws), an interpolator is here implemented to provide a parametrized desired motion, which is then converted into the Cartesian space. The nominal Cartesian motion is then given as input to the MPC algorithm [9], which computes the scaled inverse kinematics and gives the result as reference to the joint controller. The MPC method implements a Stack of Tasks (SoT) (see also [10], [11]) such that the robot should, in order of priority: i) follow the nominal Cartesian task (decomposed in its geometrical and timing part thanks to the scaling); ii) maximize the distance from the operator when closer than a certain threshold; iii) follow the original joint-space trajectory (which is collision-free with respect to static obstacles). Furthermore, the original scaling method is slightly modified: the scaling reference is not constant anymore, but changes based on a safety function in the human-robot relative distance and velocity. In this way, the speed of execution automatically decreases proportionally to the risk of a collision. Preliminary experiments on a 7-degree-of-freedom platform prove the validness of the approach.

II. Preliminaries

We briefly recall the predictive inverse kinematics and task scaling method [9], which is the core of the proposed framework. Consider an n-degree-of-freedom manipulator and denote with q, $\dot{q}$, $\ddot{q} \in \mathbb{R}^n$ its joint configuration, velocity, and acceleration vectors respectively. The robot is required to perform a primary task $x_d(t) \in \mathbb{R}^m$, $m < n$, in the Cartesian space, such that $f_0(q(t)) = x_d(t)\ \forall\, t$, where f_0 is the forward kinematic function of the task. The task is considered at differential level and implements a Closed Loop Inverse Kinematics (CLIK) scheme. A task scaling factor $\dot{s} \in [0, 1]$ is introduced to slow down the execution of the task if needed (*e.g.*, when the desired motion is too demanding for

the physical capability of the robot). The primary task can be therefore written as:

$$J_0(q)\,\dot{q} + Ke_p - x'(s)\dot{s} = 0 \tag{1}$$

where $J_0 \in \mathbb{R}^{m\times n}$ is the task Jacobian, $x' := dx/ds$, $K \in \mathbb{R}^m$ is a positive-definite (usually diagonal) gain matrix, and the task position error e_p id defined as $e_p := x_d(s) - f_0(q)$ (computation of e_p with different orientation descriptions is detailed in [12], [13]). Scaling variable $\dot{s}$ allows the decomposition of the geometrical and the timing parts of the task. Indeed, (1) represents the imposition of the desired geometrical path, while the timing law performed by the robot resembles the nominal one as much as $\dot{s}$ is close to 1.

Consider now a sequence of secondary tasks in the form $E_i\,\dot{q} + g_i = 0$. The planner should compute the optimal joint variables to: i) minimize the path error of the Cartesian task; ii) minimize the deviation with respect to the nominal timing law; and iii) minimize the deviation with respect to the secondary tasks (according to their priority). The method in [9] does so by considering the system and the tasks at p time instants along a given predictive horizon. At each time k, this results in a lexicographic optimization problem as follows:

$$\begin{aligned}
\underset{\ddot{q},\dot{s}}{\operatorname{lex\,min}} \quad & \{w_1, \dots, w_b\} \\
\text{subject to} \quad & \underline{q} \le q_{k+i} \le \bar{q} \;\; \forall i = 1,\dots,p \\
& \underline{\dot{q}} \le \dot{q}_{k+i} \le \bar{\dot{q}} \;\; \forall i = 1,\dots,p \\
& \underline{\ddot{q}} \le \ddot{q}_{k+i} \le \bar{\ddot{q}} \;\; \forall i = 0,\dots,p-1 \\
& q_{k+1} = q_k + T\dot{q}_k + 0.5T^2\ddot{q}_k \\
& \dot{q}_{k+1} = \dot{q}_k + T\ddot{q}_k
\end{aligned} \tag{2}$$

where $\underline{q}$, $\underline{\dot{q}}$, $\underline{\ddot{q}} \in \mathbb{R}^n$ and $\bar{q}$, $\bar{\dot{q}}$, $\bar{\ddot{q}} \in \mathbb{R}^n$ are the lower and upper limits of the joint configuration, velocity, and acceleration respectively, and $w_1, \dots, w_b$, $b \in \mathbb{N}$, are the desired tasks (in order of decreasing priority), which are equal to:

$$w_1 = \sum_{i=1}^{p} \left\| J_0(q_{k+i})\dot{q}_{k+i} + Ke_{p,k} - \dot{s}_{k+i}x'(s_{k+i}) \right\|^2, \tag{3}$$

$$w_2 = \sum_{i=1}^{p} \left\| \dot{s}_{\text{ref}} - \dot{s}_{k+i} \right\|^2 \;\text{ with }\; \dot{s}_{\text{ref}} = 1, \tag{4}$$

$$w_j = \sum_{i=1}^{p} \left\| E_j\dot{q}_{k+i} + g_j \right\|^2 \;\; \forall\, j = 3,\dots,b. \tag{5}$$

Feasibility of (2) with simultaneous constraints on position, velocity, acceleration and/or torque is treated in details in [14]. In brief, (2) is guaranteed to be feasible for all possible states if viability constraints are added to (2). The derivation of such constraints in terms of linear inequalities is also given in [14].

Problem (2) is solved at each cycle time and the joint variables corresponding to the first time instant are given as reference to the robot controller and the procedure is then repeated. It is possible to notice that w_1 is nonlinear in the optimization vector as the Jacobians nonlinearly depend on the joint variables. The method in [9] linearizes the task at each cycle k by using the solution obtained at the previous cycle $k-1$ to compute a prediction of the joint configurations q_{k+i}. This prediction is then used to compute the Jacobians at future time instants, and, in this way, w_1 becomes a quadratic cost function. This strategy can be adopted for nonlinear secondary tasks as well. In this way, Problem (2) becomes a hierarchical QP, which can be solved in real time. Finally, input blocking is adopted to reduce the computational burden of the online optimization (see also [15] for details).

III. Proposed Framework

The proposed framework exploits the features of the method in [9] to cope with the presence of an operator in the robot workspace. The aim is to perform a given primary task at best, exploiting the redundancy to maximize the distance between the robot and the operator and slowing down the execution of the task depending on such distance. As shown in Figure 1, the planner is inserted as an intermediate layer between an existing offline trajectory planner and the robot controller. The inputs to the planner consists in: i) a nominal trajectory, coming from the higher-level offline planner; and ii) information about the human position in the workspace (typically provided by a vision system). The outputs are the joint position/velocity references q, $\dot{q}$ to be given to the robot controller. The offline planner is typically in charge of finding a collision-free trajectory by taking into account the static obstacles in the workspace. From a practical perspective, the output of the offline trajectory planner consists in a sequence of joint position/velocities, due to the collision-check procedure, which is carried out in the robot configuration space. We therefore denote with $q_d(s) \in \mathbb{R}^n$ the nominal trajectory. This trajectory is collision-free with respect to the static obstacles, but does not take into account the possible human presence (which will be handled by the online MPC planner).

A. Stack of tasks for human-robot distance maximization

The proposed framework maximizes the distance between the human and the robot, while preserving the robot primary task. This is possible thanks to a smart exploitation of the robot kinematic redundancy. Moreover, deviating the least from the original joint-space trajectory is a desirable behavior, as such trajectory is known to be collision-free with respect to static obstacles. Assume the human position provided by the vision system refers to the robot base frame and is denoted with $x_{\text{obs}} \in \mathbb{R}^3$, and l points of interest on the robot are considered, and denote with $f_r(q)$ and $J_r(q)\,\forall\, r = 1,\dots,l$ the forward kinematic functions and the linear part of the Jacobians of those points with respect to the robot base. Referring to (2), the stack of tasks is composed of:

- The primary Cartesian task w_1, obtained by mapping the joint-space desired motion as:

$$x_d(s) = f_0(q_d(s)), \tag{6}$$

$$x'_d(s) = J_0(q_d(s))\,q'_d(s). \tag{7}$$

(Notice that it is possible to select the task axis of interest by selecting just the corresponding rows in f_0 and J_0.)

- A clearance task, defined as:

$$w_3 = \sum_{i=1}^{p} \sum_{r=1}^{l} \left\| J_r(q_{k+i})\dot{q}_{k+i} - \kappa_r \left(f_r(q_{k+i}) - x_{\text{obs}} \right) \right\|^2 \tag{8}$$

where $\kappa_r \in \mathbb{R}^+$ are positive gains. This task is activated only when $\min_r \| f_r(q_k) - x_{\text{obs},k} \| < d_{\text{safe}}$, where $d_{\text{safe}} \in \mathbb{R}^+$ is a safety distance threshold.

- A joint-space trajectory-following task given by:

$$w_4 = \sum_{i=1}^{p} \left\| q_{k+i} - q_d(s_{k+i}) \right\|^2. \tag{9}$$

The resulting planner therefore aims to: i) follow at best the desired Cartesian task; ii) maximize the human-robot distance if the operator is closer than a given threshold; iii) following the original joint-space motion (which is known to be collision-free with respect to static obstacles). Notice that in case the human is not within d_{safe}, the inverse kinematics solution is forced to the original joint-space trajectory, whereas, in case the human-robot distance is smaller than d_{safe}, priority is given to avoiding the operator.

B. Safety velocity override through modification of the scaling reference

The second feature of the MPC framework is the possibility of slowing down the task execution according to the human-robot relative distance and/or velocity. This is straightforwardly implementable by exploiting the structure of the MPC planner. As a matter of fact, the value of $\dot{s}_{\text{ref}}$ in (4) represents the percentage of the nominal speed at which the task should be executed. In [9], $\dot{s}_{\text{ref}}$ is always equal to 1, in order to follow the nominal timing law at best. However, by modifying $\dot{s}_{\text{ref}}$ based on a safety rationale, it is possible to slow down the robot without losing the path tracking of the primary task. In particular, $\dot{s}_{\text{ref}}$ is modified according to the following function:

$$\dot{s}_{\text{ref}} = \begin{cases} h(d, \dot{d}\,) & \text{if } d < d_s \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

where $d = \| f_0(q_k) - x_{\text{obs}} \|$, h is a monotonic decreasing function such that $h(d_{\text{stop}}, \dot{d}\,) = 0$, $h(d_s, \dot{d}\,) = 1$, $d_s \in \mathbb{R}^+$ is the safety-scaling activation distance, and $d_{\text{stop}} \in \mathbb{R}^+$ is the distance that requires the stop of the robot. The overall working scheme of the method is shown in Figure 1.

IV. Software Implementation

The proposed method has been implemented in ROS (Robot Operating System): the MPC planner has been implemented in ROS-control as an intermediate controller between *MoveIt!* and the PID-based controller of the robot joints. All *MoveIt!* planners can be therefore used interchangeably to generate the nominal trajectory, which then feeds the MPC planner. In these experiments, STOMP [16] has been used as high-level planner.

It is worth pointing out that the MPC planner is based on the assumption that the desired trajectory is parametrized with respect to the curvilinear abscissa s that represents the time of the nominal trajectory. However, most state-of-the-art

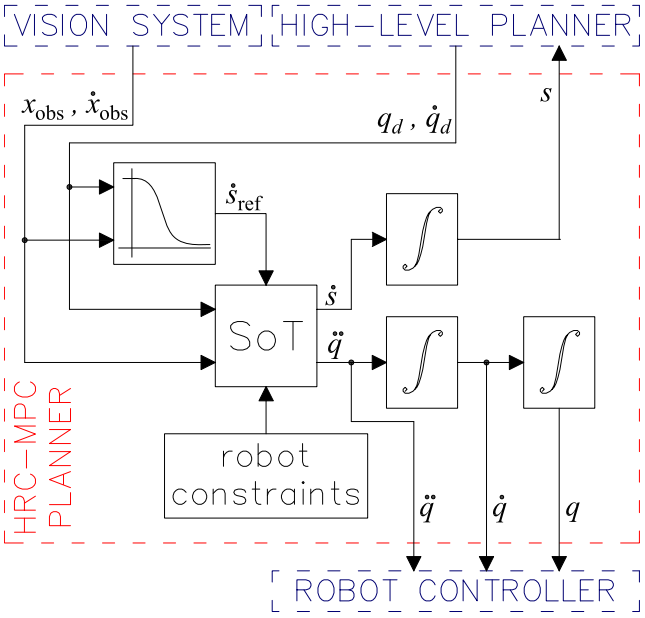

Fig. 1. Illustrative scheme of the proposed HRC-MPC planner.

planners generate as output a list of timed waypoints instead of a parametric curve. To cope with this, an interpolator has been devised. The algorithm works as follows: at each cycle, it searches for the interval of the nominal time within which the current values of $s_{k+i}, i = 1, \ldots, p$, lie; then, it performs a polynomial interpolation of the joint positions and velocities of the nominal trajectory and passes them to the MPC planner. The MPC planner, in turn, returns to the interpolator the updated value of the curvilinear abscissa.

The low-level controller of the joints consists in a cascade of a proportional position controller implemented in ROS as primary controller, and the manufacturer velocity built-in controller. The MPC planner gives as output the reference joint position which feeds the position controller and the reference joint velocity, which is given as a feed-forward term.

V. Results

The proposed framework has been tested on a 7-degree-of-freedom experimental platform composed of a 6-degree-of-freedom Universal Robot UR5 manipulator mounted on a linear guide. The limits of the system are set as follows:

$$\bar{q} = (\ 1.75,\ \pi,\ \pi,\ \pi,\ \pi,\ \pi,\ \pi\ ) \text{ rad},$$
$$\underline{q} = (\ 0.0,\ -\pi,\ -\pi,\ -\pi,\ -\pi,\ -\pi,\ -\pi\ ) \text{ rad},$$
$$\bar{\dot{q}} = -\underline{\dot{q}} = (\ 0.5,\ \pi,\ \pi,\ \pi,\ 3.2,\ 3.2,\ 3.2\ ) \text{ rad/s},$$
$$\bar{\ddot{q}} = -\underline{\ddot{q}} = (\ 2.0,\ 2.0,\ 2.0,\ 2.0,\ 2.0,\ 2.0,\ 2.0\ ) \text{ rad/s}^2.$$

Clearance task (8) considers the robot elbow as point of interest. A vision system composed of four Mesa Swissranger 4000 Time-of-Fly cameras provides the position information of the human in the workspace. Function h in (10) is defined as a third-order polynomial in d such that $h(d_s) = 1$, $h(d_{\text{stop}}) = 0$, $\frac{\mathrm{d}h}{\mathrm{d}d}(d_s) = \frac{\mathrm{d}h}{\mathrm{d}d}(d_{\text{stop}}) = 0$, $d_s = 1.4$ m, $d_{\text{stop}} = 0.3$ m.

The sampling period of the MPC planner is equal to $T = 8$ ms; the predictive horizon is equal to 0.5 s and the number of time-instants considered in the horizon is $p = 5$. As a

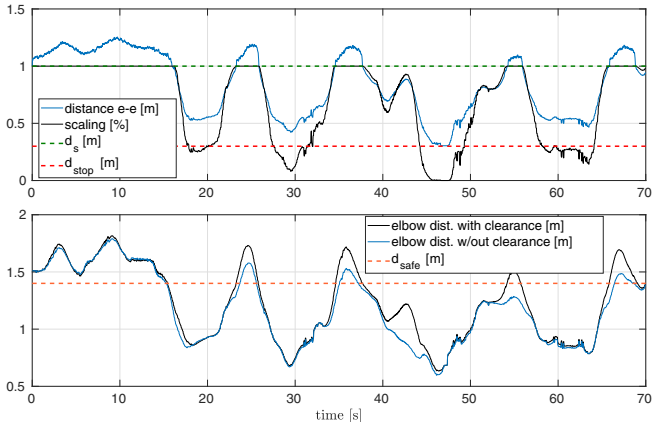

Fig. 2. Experimental results for a pick-and-place task.

preliminary approach, the human position is assumed to be constant along the predictive horizon, that is, at each time k, $x_{\text{obs}}(k+i) = x_{\text{obs}}(k)\,\forall i = 1, ..., p$. Further developments should take into account a model to predict the human motion during the predictive horizon as, for example, in [17], [18].

As an illustrative example, we consider a repetitive pick-and-place task back and forth from a point P_A to a point P_B corresponding respectively to the joint configurations $q_A = (0.2, -1.6, -1.9, -1.5, -1.3, 1.6, 0.0)$ rad and $q_B = (0.6, -2.0, -2.0, -1.4, -1.3, 1.6, 0.0)$ rad. The task considers the Cartesian position of the end-effector and its rotations around the x and y axis, while the rotation around the z axis (orthogonal to the ground) is let free. The primary task has therefore $m = 5$, and the redundant degrees of freedom of the system are equal to $n - m = 2$. The experiment consists in the robot executing the task repetitively while a person moves in its proximity. The video of the test execution is available at the following link[1]. Figure 2 shows the results for 70 seconds of task execution. The top plot shows the distance between the robot end-effector and the human (blue line) and the scaling factor $\dot{s}$ (black line). When the distance is below the threshold d_s (dashed green line), the scaling factor decreases to reduce the speed of the robot. If the distance reaches the safety value d_{stop} the robot should stop and, as a matter of fact, $\dot{s}$ decreases to zero (at about 45 seconds in the plot). The bottom plot shows the distance between the robot point of interest (elbow) and the human, in the case that the clearance strategy is activated (black line) and in the case it is not (blue line). Dotted orange line is the clearance activation threshold d_{safe}. Results show that the distance from the operator is always larger when the clearance strategy is used. This is visible especially when the scaling is not particularly heavy: when $\dot{s}$ decreases significantly, then the redundancy is exploited mainly to accomplish the primary task and the difference between the two cases (in terms of distance from the robot elbow to the human) decreases.

VI. Conclusions

We have proposed the proof of concept of an MPC framework for HRC and validated it through experiments on an industrial manipulator. Future works will focus on the use of models for the prediction of the human motion and the derivation of a general rationale to derive the safety scaling function and thresholds.

[1] youtu.be/JiQ9yobm1Is

Acknowledgment

The research leading to these results was partially funded by the European Union H2020-ICT-2017-1 – Pickplace: Flexible, safe and dependable robotic part handling in industrial environment (grant agreement: 780488).

References

[1] M. Koppenborg, P. Nickel, B. Naber, A. Lungfiel, and M. Huelke, "Effects of movement speed and predictability in human–robot collaboration," *Human Factors and Ergonomics in Manufacturing & Service Industries*, vol. 27, pp. 197–209, 2017.

[2] D. Kulić and E. A. Croft, "Real-time safety for human–robot interaction," *Robot. Auton. Syst.*, vol. 54, no. 1, pp. 1–12, 2006.

[3] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. IEEE ICRA*, Saint Paul (USA), 2012, pp. 338–345.

[4] A. M. Zanchettin, N. M. Ceriani, P. Rocco, H. Ding, and B. Matthias, "Safety in human-robot collaborative manufacturing environments: Metrics and control," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 882–893, 2016.

[5] M. Lippi and A. Marino, "Safety in human-multi robot collaborative scenarios: a trajectory scaling approach," in *Proc. IFAC SYROCO*, Budapest (Hungary), 2018.

[6] O. Dahl and L. Nielsen, "Torque-limited path following by online trajectory time scaling," *IEEE Trans. Rob. Autom.*, vol. 6, pp. 554–561, 1990.

[7] G. Antonelli, S. Chiaverini, and G. Fusco, "A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits," *IEEE Trans. Rob. Autom.*, vol. 19, pp. 162–167, 2003.

[8] F. Flacco, A. De Luca, and O. Khatib, "Control of redundant robots under hard joint constraints: Saturation in the null space," *IEEE Trans. Rob.*, vol. 31, pp. 637–654, 2015.

[9] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, "Predictive inverse kinematics for redundant manipulators with task scaling and kinematic constraints," *IEEE Trans. Rob.*, vol. 35, pp. 278–285, 2019.

[10] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The Int. J. of Robot. Res.*, vol. 33, pp. 1006–1028, 2014.

[11] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: generalizing the task priority framework to inequality tasks," *IEEE Trans. Rob.*, vol. 27, pp. 785–792, 2011.

[12] F. Caccavale, B. Siciliano, and L. Villani, "The role of Euler parameters in robot control," *Asian J. of Control*, vol. 1, pp. 25–34, 1999.

[13] L. Roveda, N. Pedrocchi, M. Beschi, and L. Molinari Tosatti, "High-accuracy robotized industrial assembly task control schema with force overshoots avoidance," *Cont. Eng. Prac.*, vol. 71, pp. 142–153, 2018.

[14] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, "Viability and feasibility of constrained kinematic control of manipulators," *Robotics*, vol. 7, no. 3, pp. 1–19, 2018.

[15] M. Faroni, M. Beschi, M. Berenguel, and A. Visioli, "Fast MPC with staircase parametrization of the inputs: Continuous Input Blocking," in *Proc. IEEE ETFA*, Limassol (Cyprus), 2017, pp. 1–8.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE ICRA*, Shanghai (China), 2011, pp. 4569–4574.

[17] J. Mainprice, R. Hayne, and D. Berenson, "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces," *IEEE Trans. Rob.*, vol. 32, pp. 897–908, 2016.

[18] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "A scalable framework for real-time multi-robot, multi-human collision avoidance," *arXiv preprint arXiv:1811.05929*, 2018.