Reactive Task Adaptation Based on Hierarchical Constraints Classification for Safe Industrial Robots

Nicola Maria Ceriani, Andrea Maria Zanchettin, *Member, IEEE*, Paolo Rocco, *Member, IEEE*, Andreas Stolt, *Student Member, IEEE*, and Anders Robertsson, *Senior Member, IEEE*

I. Introduction

HUMAN–ROBOT interaction and collaboration are some of the most promising opportunities for the development of industrial robots applications in the future. Such operational paradigms could allow robots to complement human work in scenarios where repetitive actions are combined with skilful dexterity-based operations. This is the case, for example, of electronic goods assembly lines where, in addition, frequent production line setup changes make manual assembly still one of the most effective solutions. The time and costs needed for the construction of an automated line are often uncompetitive with the less advanced manual solution. Flexibility enhancement coming from human–robot interaction could ease robots deployment and drive their penetration also in such kinds of scenarios. A crucial factor for the enhancement of robot flexibility and for a feasible interaction is the availability of safe collaborative robots. Implementation of safety systems to protect human workers from potential dangerous impacts with robots is indeed a very significant cost item [1] and usually the reason of a strict separation between robots and humans, thus completely preventing interaction. Ensuring safety in a collaborative scenario is, however, a difficult challenge, with different requirements to be satisfied, i.e., preventing harmful situations for the human and for the robot and, at the same time, preserving productivity. Such a problem entails investigation of both novel control strategies and safety systems to make the robot capable of coping with an unstructured and dynamic environment and requires adequate methodologies both from a robot programming [2] and design perspective (see [3] and [4]).

A. Control Strategies for Adaptation

Deployment of industrial robots in the aforementioned environments requires control strategies for task adaptation to unforeseen events such as obstacles moving in the workspace. To this purpose, strategies that perform collision avoidance at low control levels, therefore enabling a very fast reaction to an unexpected event, play an important role. In [5], the potential field approach was introduced as a real-time solution to obstacle avoidance, previously mostly managed at high-level planning. Virtual impedance is another common approach to the collision avoidance problem, where a detected object not in contact with the robot produces a virtual force which is applied in the impedance control law of the manipulator. First works on application of such an approach to manipulators are [6] and [7], where virtual forces were applied to the end-effector and to the whole manipulator structure, respectively. A relevant effort to connect global path planning and local real-time reaction to changes in the environment was proposed in [8]. Such contribution aimed at overcoming limitations of approaches based only on potential fields in achieving global goals. A decisive aspect for operation of robots in unstructured environments is capability of combining adaptation and consistency with task constraints. A significant contribution in this respect was given in [9] and expanded in [10] for the field of mobile robots: in these works, a framework was proposed for executing obstacle avoidance and other secondary tasks such as posture control without affecting enforcement of task constraints. Representation of task constraints also plays an important part in this context. A fundamental contribution to representation of task space constraints was given in [11] related to the field of compliance and force control. In the field of motion planning with constraints, Stilman [12] proposed a task space constraints representation for joint space planning algorithms. In [13], tolerance on constraints,

Manuscript received May 19, 2014; revised September 28, 2014 and December 29, 2014; accepted February 13, 2015. Date of publication May 6, 2015; date of current version October 21, 2015. Recommended by Technical Editor F. Aghili.

N. M. Ceriani is with Tenaris Dalmine, 24044 Dalmine, Italy (e-mail: nceriani@tenaris.com).

A. M. Zanchettin and P. Rocco are with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy (e-mail: andreamaria.zanchettin@polimi.it; paolo.rocco@polimi.it).

A. Stolt and A. Robertsson are with the Department of Automatic Control, the LCCC Linnaeus Center, and the eLLIIT Excellence Center Lund University, SE-22100 Lund, Sweden (e-mail: andreas.stolt@control.lth.se; anders.robertsson@control.lth.se).

Color versions of one or more of the figures in this paper are available online.

that is intervals of feasible values around the desired one, is considered in a motion planning algorithm.

When multiple constraints have to be enforced and the number of available degrees of freedom (DOF) is lower than the amount of constraints, or in case possible conflicting constraints have to be managed, the task priority approach can be adopted. In [14], a framework for enforcing an arbitrary set of equality constraints, each prioritized with respect to the next ones, was proposed. Inequality constraints were added to the framework in [15], and the new prioritized problem was solved as a sequence of quadratic programming problems. More recently, in [16], a more numerically efficient solution of the same problem was proposed.

B. Sensor Systems for Unstructured Environments

All the control strategies for adaptation to a dynamic setting need information about the environment that have to be retrieved by means of sensors. Different kinds of sensors, such as distance or vision ones, have been used for this purpose. RGB-D sensors have been widely adopted thanks to the availability of cheap implementations. In [17], for example, a Kinect is adopted to compute distances between a manipulator and an obstacle, and to compute repulsive forces that are used in a collision avoidance system. Measurements obtained from a laser scanner and a camera mounted on a mobile robot are used in a sensor fusion in [18] for path planning and collision avoidance. In [19], a system for safe human–robot interaction based on a sensor system composed of microwave sensors, infrared passive sensors, and cameras is proposed. An example of sensor fusion for human and robot localization based on pyroelectric infrared and RF sensors is given in [20]. A type of sensor that is particularly fit for collision avoidance and human–robot interaction applications is the distributed distance sensor, which is a sensor mounted directly on the robot and possibly covering its complete surface. This sensor can be assembled as a multitude of discrete sensors or as a continuous sensor and was first introduced in [21]. Distributed distance sensors based on other technologies and deployed as a continuous sensor have been investigated in [22], where a capacitive distributed distance sensor was adopted in a collision avoidance system. More recently, in [23], a more advanced capacitive sensor was again used in a collision avoidance system. The authors of the present work have contributed to the development of the distributed distance sensor, tackling the problem of sensor sizing and of optimal placement of a limited number of spots on a robot [24]. Moreover, in [25], they have presented a sensor prototype and developed a safety controller based on the sensor measurements.

In this paper, three main contributions are given. First, we propose a classification for constraints composing a preplanned task based on relevance for task execution. Such a classification defines how the task can be modified in reaction to unforeseen events. Classification of task constraints has been tackled in various works in the literature. The classification proposed in [11] leads to results similar to the ones of our classification, when motion restrictions imposed by the environment are considered. However, in this study, not only restrictions of motion but also possibilities and consequences of constraints relaxation are analyzed. In [26], as in this study, the category of soft constraints is introduced. However, the term "soft" is here used to identify constraints with the highest possibility of relaxation, while in [26], constraints defined with a tolerance are classified as "soft." In [27], the term "soft constraints" is given another significance, indicating constraints imposed by deformable objects to the robot motion, and an efficient cost function is introduced to assess and minimize objects deformation in path planning. The second contribution of this study is a task-consistent collision avoidance safety strategy, based on our constraints classification. The task is divided into subtasks, whose enforcement takes into account the environment state, similarly to what proposed in [28]. With this respect, an interesting software framework for the management of a decomposed task can be found in [29]. However, in this paper, we propose a general criterion for the division of the task into subtasks. Furthermore, the considered task is the preplanned trajectory generated by an industrial controller, which can be only locally modified or suspended. Finally, we propose a system for the integration of the safety strategy with an industrial controller, and we validate it experimentally on an assembly task. Compared to previous works concerning the modification of the set points computed by an industrial controller, such as [30], in this study, a deeper integration with the controller is achieved: we implement a system for the communication between the controller and the safety strategy, which gives the strategy the control of task execution and a connection with the user interface. In the experimental validation, a distributed proximity sensor for obstacle detection is used. The sensor, originally introduced in [24], is developed into a complete system, which is applied to a dual arm robot.

This paper extends the work presented in [31] adding the following contributions.

1) Constraints classification based on relevance for task completion is detailed and the procedure for its application is explained with concrete examples.
2) Integration of the collision avoidance system with an industrial controller is thoroughly described, analyzing the components of the integrated system.
3) A deeper insight at the experimental validation is given, describing the distributed distance sensor developed for the adopted manipulator.

The remainder of this paper is organized as follows. In Section II, a classification for instantaneous constraints composing a robotic task based on relevance for task completion is proposed. In Section III, a method for the decomposition a task according to the proposed classification is introduced. Section IV describes a task-consistent collision avoidance strategy exploiting constraints classification and assessment of danger in human–robot interaction. Section V analyzes the control system integrating the collision avoidance strategy with an industrial controller, and Section VI discusses its experimental application to an assembly task and the sensor system developed for such an application. Finally, Section VII discusses the presented results and analyzes possible future developments of the proposed system.

II. Relevance-Based Constraints Classification

Industrial robot programming languages typically allow users to define the task of the manipulator through a limited set of variables, such as final position and velocity or velocity and acceleration limits. The motion planner of the industrial controller then transforms the task specified in this way into a trajectory for the robot end-effector, or equivalently at the joint space. A set of instantaneous constraints for robot tool center point (TCP) translational and rotational velocities are thus enforced all along task execution.

Since all robot coordinates are typically constrained even in case of task redundancy, the task as it is defined by the robot controller may include unnecessary constraints. Such constraints restrict the set of possible executions of the task to a single one, even when alternative solutions are available. If adaptability of the task is pursued, redundancies should be identified and exploited. Moreover, when considering operation in an unstructured environment, unforeseen events occurring at execution time may cause new constraints to arise: for example, avoidance of obstacles may require a deviation from the planned trajectory. In case no redundancy is present in the preplanned task, or if task redundancy is not sufficient to effectively take into account the new constraint, a criterion is necessary to determine whether any of the constraints constituting the task can be temporarily relaxed. Finally, a strategy is needed to preserve the possibility of successful task completion in case relaxation of a constraint is performed.

Industrial controllers usually lack functionalities for adaptation of the task to a dynamic setting. Nonetheless, preplanned trajectory suspension is usually feasible and industrial and research tools are available for the modification of preplanned trajectories at execution time. In this framework, constraints relaxation is possible through proper modification of the preplanned trajectory, thus allowing task adaptation.

In the following, we propose a systematic classification of instantaneous velocity constraints based on their relevance for task execution, as the foundation for an adaptation strategy for preplanned trajectories. The expression "relevance" refers to the importance and to the necessity of enforcing a constraint to complete a task, and to the consequences of its relaxation for task execution. The less a constraint is relevant, the higher are the possibilities of its relaxation, and the lower are the consequences on task execution. As it will be explained later on in the paper, only velocity constraints are considered in this classification, since they are the only ones that can be taken into account in the modification of a preplanned task. The classification achieves three main objectives.

1) It defines a scale of priority for the constraints constituting a task, which can be adopted to determine which constraints to relax in case new constraints have to be enforced.
2) It defines how the different constraints can be relaxed.
3) It defines the actions to be performed to preserve task completion in case a constraint is relaxed.

In the remaining part of this paper, the term "task" will be used to refer to a complete operation performed by the robot, while the term "skill" will be used to express a single action. Different sequential skills, therefore, compose a task. The classification considers three types of constraints:

A) *Hard* constraints;
B) *Skill* constraints;
C) *Soft* constraints.

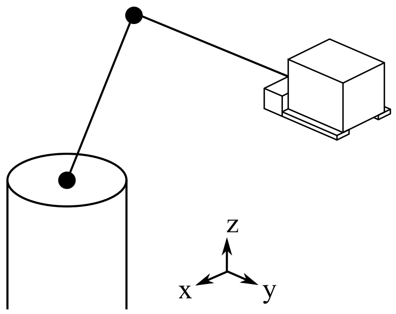


Fig. 1. Skill example: a palletizing operation. In order to prevent the object from falling, forks have to be kept on the horizontal plane.

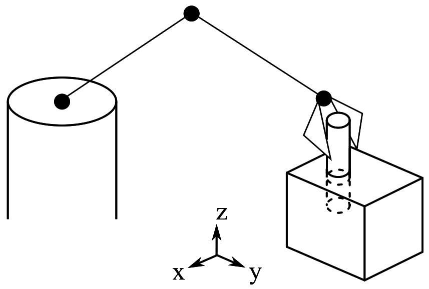


Fig. 2. Skill example: a peg-in-hole insertion operation. Orientation of the peg around the hole axis is not relevant during insertion.

A. Hard Constraints

Constraints whose relaxation would cause skill disruption are defined as *hard*. Let us consider, for example, a manipulator performing a palletizing operation with an object lying on lifting forks as shown in Fig. 1: in order to prevent the object from falling, the forks have to be kept parallel to the horizontal plane. Constraints defining the orientation around the *x*- and *y*-axes shown in Fig. 1 cannot, therefore, be relaxed and are classified as *hard*. Another example of *hard* constraints can be taken from the classical peg-in-hole skill depicted in Fig. 2. Reaction forces restrict translations along and rotations around the *x*- and *y*-axes. These four constraints are, therefore, classified as *hard* as their relaxation may cause reaction forces to arise with the risk of damaging the components involved. It is worth noticing that although in the latter example this class coincides with Mason's natural constraints class [11], the scope of the presented classification is broader. The criterion of relevance for skill execution in fact considers not only reaction forces but also motion inconsistency with the working scenario as a source of skill disruption. *Hard* constraints have the highest priority and thus cannot be relaxed.

B. Skill Constraints

Constraints for which relaxation is possible but implies the temporary suspension of skill execution are defined as *skill* constraints. Let us consider again the palletizing skill of Fig. 1: relaxation of the three constraints on translational velocities

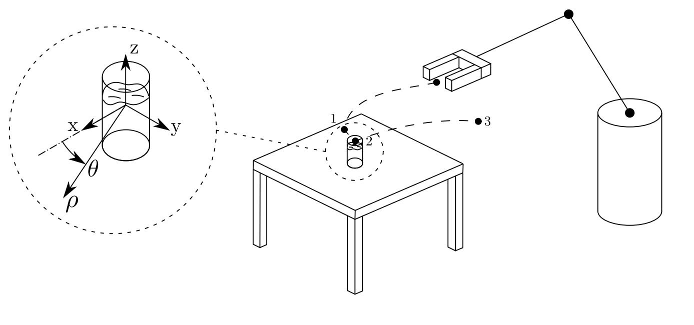


Fig. 3. Task that can be decomposed in three skills: the robot has to approach the liquid container, pick it, and then move it to another position.

would cause the suspension of the skill. However, such an action does not disrupt the skill as in the previous case: after a deviation from the preplanned trajectory, the manipulator may still complete the skill. A further example of *skill* constraints is given by the peg-in-hole skill: the translation along the hole axis can be temporarily relaxed, e.g., to retract the peg, keeping the possibility to accomplish the skill. *Skill* constraints are assigned the second highest level of priority and can be relaxed through a suspension of the preplanned trajectory execution.

C. Soft Constraints

The last group includes constraints whose relaxation has no effect on skill execution and thus correspond to task redundancies. As previously said, even if a skill can be accomplished by specifying less constraints than the number of robot DOF, industrial controllers typically impose the user to specify all of them. Let us consider once again the peg-in-hole skill of Fig. 2: orientation around the hole axis has no effect on skill completion and can, therefore, be classified as a *soft* constraint. *Soft* constraints are given the lowest level of priority among task constraints and can be relaxed without any consequence for task execution.

III. Decomposition of Task in Skills

Since, during the execution of a task, velocity constraints have different relevance, the set of DOF available for task modification changes along its advancement. In order to define a strategy for task adaptation, it is therefore convenient to divide the task into skills on the basis of how the relevance for task execution of the constraints changes. For the decomposition of a task into skills, the following procedure has to be followed.

1) For each of the instantaneous constraints enforced by the robot controller along task execution, relevance has to be defined according to the proposed classification. For the time being, such an operation has to be performed manually. In order to formalize the classification of constraints, the selection vectors $\mathbf{s}_{\text{soft}}$, $\mathbf{s}_{\text{skill}}$, and $\mathbf{s}_{\text{hard}}$ are introduced. These vectors are used to define which type of constraint is applied to each of the three TCP translational and rotational velocities. Let us consider, for example, the task shown in Fig. 3: a manipulator equipped with a gripper has to pick a liquid container and move it to position 3. The depicted reference system is adopted, where ρ is the radial distance from the origin of the coordinate axes and θ the angular coordinate. Let $\mathbf{v} = [v_\rho, v_\theta, v_z, \omega_x, \omega_y, \omega_z]$ be the vector of task velocities and define

$$s_{\text{soft,i}} = \begin{cases} 1, & \text{if } v_i \text{ is subject to a } soft \text{ constraint} \\ 0, & \text{otherwise} \end{cases}$$

$$s_{\text{skill,i}} = \begin{cases} 1, & \text{if } v_i \text{ is subject to a } skill \text{ constraint} \\ 0, & \text{otherwise} \end{cases}$$

$$s_{\text{hard,i}} = \begin{cases} 1, & \text{if } v_i \text{ is subject to a } hard \text{ constraint} \\ 0, & \text{otherwise.} \end{cases}$$

Each operational space velocity has to be assigned a constraint, so the three vectors sum up to a vector with all unitary elements.

2) Along the execution of the task, constraints are in general assigned different values of relevance, according to the operation performed by the robot: a new skill is defined at every point where relevance for task execution of a constraint changes.

In the considered task, during the approach motion, any TCP orientation will be acceptable, while TCP position imposed by the controller can be temporarily relaxed but has to be enforced to complete the operation. Position constraints will, therefore, be classified as *skill* while orientation ones as *soft*. During the grasping phase, the gripper fingers must be aligned with radius ρ and must lie in the horizontal plane. Orientation is, therefore, classified as a *hard* constraint, while position constraints are of *skill* type. Finally, after the robot grasps the container, orientation around the *x*- and *y*-axes cannot be relaxed to avoid spilling liquid and thus correspond to *hard* constraints. Orientation around the *z*-axis corresponds instead to a *soft* constraint. Finally, position constraints can be temporarily relaxed but have to be enforced to complete the motion and are therefore *skill* constraints. As the three phases present each a different relevance for constraints, a skill is defined for each of them. The transition between skills implies handling varying constraints applied to the task coordinates: details about such an issue will be given in Section V. Table I shows relevance of constraints for the mentioned skills.

TABLE I
Constraints of the Task in Fig. 3

	v_ρ	v_θ	v_z	ω_x	ω_y	ω_z
Skill 1 - *approach*	*skill*	*skill*	*skill*	*soft*	*soft*	*soft*
Skill 2 - *grasping*	*skill*	*skill*	*skill*	*hard*	*hard*	*hard*
Skill 3 - *moving*	*skill*	*skill*	*skill*	*hard*	*hard*	*soft*

IV. Task Consistent Collision Avoidance Strategy

One of the most important aspects of task adaptation to an unstructured environment is avoidance of obstacles appearing in the working scenario. Obstacle avoidance has the twofold goal of following a path, which is consistent with the environment and contributing to ensure safety for human workers interacting with the robot. However, if the task being performed by the manipulator constrains all of the available DOF, relaxation of at least a subset of constraints is necessary, in order to perform evasive motions. In the following, a strategy to dynamically relax constraints, maintaining consistency with the task, is proposed.

A. Danger Assessment

As previously introduced, if a task fully constrains the robot DOFs, collision avoidance can be performed only if some of the mentioned constraints are relaxed. In order to determine the set of constraints to be relaxed, a prioritization approach can be adopted. As the purpose of evasive motions is to avoid collisions with unpredicted obstacles and ensure safety for human workers, an assessment of danger in human–robot interaction can be used to define their level of priority. For this, the danger field [32] can be used. For a detailed explanation of such assessment, one can refer to [32] and [33], while, in this paper, only a brief overview will be given.

The kinetostatic danger field is a scalar field that expresses interaction danger for obstacles in proximity to the robot. Compared to the broadly adopted potential field approach [5] used for obstacle avoidance, it has two peculiarities. First, instead of considering the obstacle as the source of danger for the robot, it considers the robot as the source of danger for the obstacle. Second, it includes both distance and velocity between the robot and the obstacle in danger evaluation. More specifically, it takes into account not only the modulus of velocity but also the angle between robot velocity and the distance vector linking it to the obstacle. The danger field generated by an ideal pointlike robot at $\mathbf{r}_r$ moving with velocity $\mathbf{v}_r$ toward an obstacle located at $\mathbf{r}_o$ is defined as

$$DF(\mathbf{r}_r, \mathbf{v}_r, \mathbf{r}_o) = \frac{k_1}{\|\mathbf{r}_o - \mathbf{r}_r\|} + \frac{\|\mathbf{v}_r\|(1 + cos\angle(\mathbf{r}_o - \mathbf{r}_r, \mathbf{v}_r))k_2}{\|\mathbf{r}_o - \mathbf{r}_r\|^2}$$

where k_1 and k_2 are the weights of the static and kinematic parts of the danger field, respectively, which from now on will be simply expressed as $DF(\mathbf{r})$, with $\mathbf{r}$ indicating the obstacle position. Without going into further details, the danger field can be integrated along the robot structure obtaining its cumulative version $CDF(\mathbf{r})$, which allows us to take into account the robot size.

B. Evasive Motions

Computing the gradient of $CDF(\mathbf{r})$, a vector field $\nabla CDF(\mathbf{r})$ can be obtained. The field gradient $\nabla CDF(\mathbf{r})$ is used to obtain a virtual repulsive force, which pushes the robot in the direction of maximum danger decrease [33]. The vector $CDF(\mathbf{r}) \cdot \nabla CDF(\mathbf{r})/\|\nabla CDF(\mathbf{r})\|$ can, therefore, be interpreted as a repulsive force anchored in $\mathbf{r}$ with modulus the danger generated by the robot in $\mathbf{r}$. As it will be described in Section VI-A, multiple obstacles can be detected at once, say n_{obs}, so for the *i*th robot link, a cumulative virtual repulsive force can be computed:

$$\nabla CDF_i = \sum_{j=1}^{n_{\text{obs}}} CDF(\mathbf{r}_j)\frac{\nabla CDF(\mathbf{r}_j)}{\|\nabla CDF(\mathbf{r}_j)\|}.$$

As ∇CDF_i is the sum of all the repulsive forces for the n_{obs} obstacles, possible high-danger obstacles may have little influence on the overall virtual repulsive force. An insufficient repulsive force may, therefore, be obtained if many obstacles with low related danger are detected. A modified version of the cumulative danger field is, therefore, used in order to take into account more effectively possible high-danger obstacles. In order to ensure evasion even when a single obstacle has high related danger, a new virtual repulsive force can be defined as

$$\nabla CDF_i^* = \frac{\nabla CDF_i}{\|\nabla CDF_i\|} \cdot \max_j CDF(\mathbf{r}_j).$$

The direction of the force is, therefore, the average of the directions of the n_{obs} virtual forces related to the n_{obs} obstacles weighted on their modulus, while the modulus of the force is the highest of all the moduli of the n_{obs} contributions. Choosing the most dangerous obstacle to set the force modulus, a safety oriented assumption is made. Moreover, preserving the force direction as the weighted mean of all the forces directions, the computed force reflects obstacles disposition. The virtual force computed for the *i*th link is applied to the two link endpoints and then transformed into joint torques using the Jacobian transpose. The application of the force to both the link extremities reduces the possibility of configurations where the evasive force has zero lever-arm, and an evasive force yields no evasive movement. The contribution of the *i*th link to the evasive torques vector is thus

$$\mathbf{T}_i = \mathbf{J}_{i-1,v}^{\mathbf{T}}(\mathbf{q}) \cdot \nabla CDF_i^* + \mathbf{J}_{i,v}^{\mathbf{T}}(\mathbf{q}) \cdot \nabla CDF_i^*$$

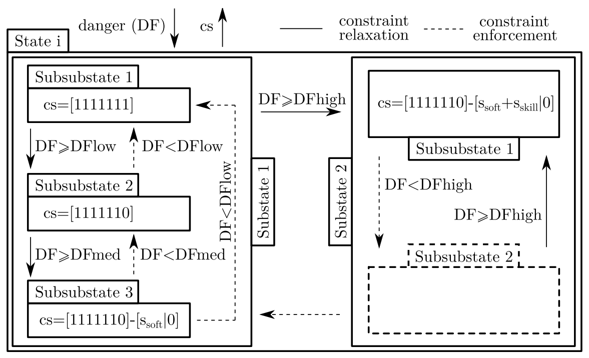
where $\mathbf{J}_{i,v}$ is the linear velocity Jacobian of the *i*th Denavit Hartenberg frame. Evasive joint velocities can finally be obtained by applying the torques to a mass–damper impedance filter:

$$\dot{\mathbf{q}}_{ev} = (\mathbf{M}s + \mathbf{D})^{-1} \cdot \mathbf{T}$$

where $\mathbf{T} = \sum_{i=1}^{n_{\text{link}}} \mathbf{T}_i$ and $\mathbf{M}$ and $\mathbf{D}$ are mass and damping matrices, respectively, imposing the desired behavior to the impedance filter, n_{link} being the number of joint space DOF. The impedance filter has, therefore, the purpose of transforming evasive torques into evasive velocities and smoothing the profile of evasive motions. By adequately tuning the filter parameters, accelerations, and therefore torques requested to the joints, can be limited.

C. Collision Avoidance Strategy

The previously proposed constraints classification and decomposition of tasks into skills can be used to assign priorities to each constraint, during the different phases of execution. Given such a priority classification, a framework is needed to automatically define the (sub)set of constraints to be enforced, and consequently, the DOF made available for execution of evasive motions, given the skill being executed and the current level of danger. For this purpose, the state machine shown



- **State i** - corresponding to skill i
  - **Sub-state 1** $DF < DF_{high}$ - skill execution
    - **Sub-sub-state 1** $DF < DF_{low}$ no constraint is relaxed
    - **Sub-sub-state 2** $DF_{low} \leq DF < DF_{med}$ kinematic redundancy is relaxed
    - **Sub-sub-state 3** $DF_{med} \leq DF < DF_{high}$ *soft* constraints are relaxed
  - **Sub-state 2** $DF \geq DF_{high}$ - skill suspension
    - **Sub-sub-state 1** $DF \geq DF_{high}$ *skill* constraints are relaxed
    - **Sub-sub-state 2** $DF < DF_{high}$ *skill* constraints are recovered to resume skill execution

Fig. 4. On the left, a state of the state machine, for a 7-DOF robot, is depicted. On the right, the strategy for state transition is described.

in Fig. 4 is used. Its states, each one corresponding to a skill, are given a common structure, which defines the general strategy for constraints management. In addition to the previously classified constraints, the ones corresponding to possible robot kinematic redundancy (e.g., the swivel angle for a 7-DOF robot [34]) are considered. Such constraints have no effect on task execution and are thus assigned the lowest priority. In order to define the set of constraints to be relaxed, three danger thresholds, DF_{low}, DF_{med} and DF_{high}, are introduced. If danger exceeds DF_{low}, evasive motions are given a higher priority than constraints corresponding to possible robot kinematic redundancy. If danger exceeds DF_{med} or DF_{high}, evasive velocities are assigned higher priority than *soft* and *skill* constraints, respectively. All constraints with lower priority than evasive motions are thus relaxed, and corresponding coordinates are used to perform evasion. The state of each instantaneous constraint (enforced/relaxed) is expressed by the state machine output $\mathbf{cs}$, which is a n_j-by-1 (where n_j is the number of robot operational space DOF) vector:

$$cs_i = \begin{cases} 1, & \text{if the constraint is enforced} \\ 0, & \text{if the constraint is relaxed.} \end{cases}$$

The first six elements of the vector correspond to the elements of $\mathbf{v}$. Possible further elements correspond to further instantaneous constraints that can be enforced on redundant manipulators, e.g., swivel angle velocity in a 7-DOF manipulator. As an example, a 7-element $\mathbf{cs}$ vector is considered. Vector $\mathbf{cs}$ is, therefore, determined according to the following rule:

$$\mathbf{cs} = \begin{cases} [\mathbf{s}_{\text{hard}},\ 0], \text{if } DF \geq DF_{\text{high}} \\ [\mathbf{s}_{\text{skill}} + \mathbf{s}_{\text{hard}},\ 0],\ \text{if } DF_{\text{med}} \leq DF < DF_{\text{high}} \\ [\mathbf{s}_{\text{soft}} + \mathbf{s}_{\text{skill}} + \mathbf{s}_{\text{hard}},\ 0],\ \text{if } DF_{\text{low}} \leq DF < DF_{\text{med}} \\ \mathbf{1}_{7\times 1}, \text{otherwise.} \end{cases}$$

The proposed strategy is parameterized on selection vectors $\mathbf{s}_{\text{soft}}$, $\mathbf{s}_{\text{skill}}$, and $\mathbf{s}_{\text{hard}}$, which define relevance of constraints for each skill, and is therefore automatically defined from constraints classification. Given the $\mathbf{cs}$ vector, a null-space projector can be defined to execute evasive velocities consistently with enforced constraints. We introduce the selected constraints Jacobian $\mathbf{J}_{cs}$:

$$\mathbf{J}_{cs}(\boldsymbol{q}) = \Sigma_{cs}\mathbf{J}(\boldsymbol{q}),$$

where $\mathbf{J}(\boldsymbol{q})$ is the Jacobian of the robot TCP pose, augmented to describe possible kinematic redundancies with appropriate operational space coordinates, and $\boldsymbol{\Sigma}_{cs}$ is the selection matrix extracting the Jacobian rows corresponding to nonzero elements of $\mathbf{cs}$. The null-space projector $\mathbf{N}_{cs}$:

$$\mathbf{N}_{cs}(\boldsymbol{q}) = \mathbf{I} - \mathbf{J}_{cs}(\boldsymbol{q})^{\dagger}\mathbf{J}_{cs}(\boldsymbol{q})$$

is then introduced, which projects evasive velocities in the null space of constrained coordinates velocities, $\mathbf{J}_{cs}(\boldsymbol{q})^{\dagger}$ being the pseudoinverse of $\mathbf{J}_{cs}(\boldsymbol{q})$. Projected evasive velocities are finally obtained as $\dot{\mathbf{q}}_{ev,cs} = \mathbf{N}_{cs}(\boldsymbol{q}) \cdot \dot{\mathbf{q}}_{ev}$

The steps for computation of evasive velocities $\dot{\mathbf{q}}_{ev,cs}$ are summarized in Algorithm 1.

V. Integration With an Industrial Controller

The constraints classification introduced in Section II allows adaptation to unforeseen events through modification of a preplanned task. Algorithm 1 defines the procedure for the computation of evasive velocities, consistently with the aforementioned classification and based on a measure of danger in human–robot interaction. In the integration of the collision avoidance strategy with an industrial controller, preplanned task modification is performed through modification of the position and velocity set points computed by the controller. A state machine, which is included in the safety system, ensures the bumpless commutation between the set points computed by the industrial controller and the modified ones. A hybrid system, mixing the continuous states of the collision avoidance strategy and the discrete states of the state machine, is therefore obtained. The system is schematically represented in Fig. 6.

In the following, integration of the collision avoidance strategy with an industrial controller is discussed. The integrated system, shown in detail in Fig. 5, is composed of four main parts, analyzed in the following.

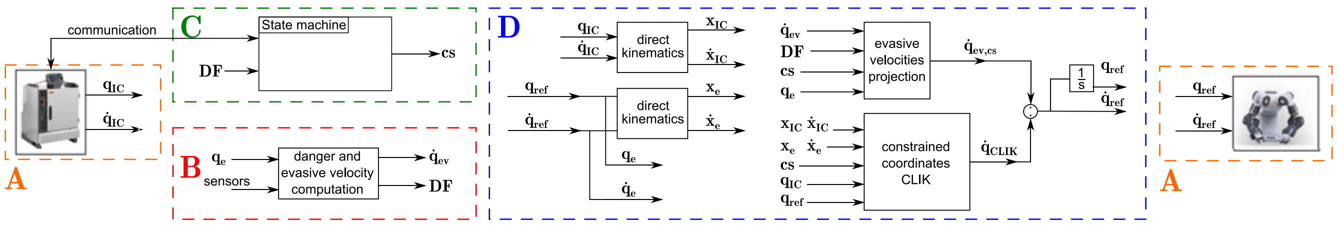


Fig. 5. Overall control system scheme: the orange section corresponds to the industrial controller and the robot, the red one to the obstacle detection and danger assessment system, the green one to the state machine and the blue one to the pose control system. The CLIK subscript refers to the closed loop inverse kinematic algorithm, which will be explained in Algorithm 2.

Algorithm 1 Computation of evasive velocities

```
 1: Measure positions $\mathbf{r}_j$ of the $n_{\text{obs}}$ obstacles
 2: $\mathbf{T} = \mathbf{0}_{n_j \times 1}$
 3: for $i$=1 to $n_{link}$ do
 4:     $\nabla CDF_i = [0\ 0\ 0]^T$
 5:     for $j$=1 to $n_{obs}$ do
 6:         $\nabla CDF_i = \nabla CDF_i + CDF(\mathbf{r}_j)\frac{\nabla CDF(\mathbf{r}_j)}{\|\nabla CDF(\mathbf{r}_j)\|}$
 7:     end for
 8:     $\nabla CDF_i^* = \frac{\nabla CDF_i}{\|\nabla CDF_i\|} \cdot \max_j CDF(\mathbf{r}_j)$
 9:     $\mathbf{T} = \mathbf{T} + \mathbf{J}_{i-1,v}^{\mathbf{T}}(\mathbf{q}) \cdot \nabla CDF_i^* + \mathbf{J}_{i,v}^{\mathbf{T}}(\mathbf{q}) \cdot \nabla CDF_i^*$
10: end for
11: $\dot{\mathbf{q}}_{ev} = (\mathbf{M}s + \mathbf{D})^{-1} \cdot \mathbf{T}$
12: $DF = \max_j CDF(\mathbf{r}_j)$
13: $\mathbf{cs} = statemachine(DF)$
14: $n_{cs} = \sum_{i=1}^{n_j} cs_i$
15: $\mathbf{\Sigma}_{cs} = \mathbf{0}_{n_{cs} \times n_j}$; $k = 1$
16: for $i$=1 to $n_j$ do
17:     if $cs_i$==1 then
18:         $\sigma_{k,i} = 1$; $k = k + 1$
19:     end if
20: end for
21: $\mathbf{J}_{cs}(\boldsymbol{q}) = \mathbf{\Sigma}_{cs}\mathbf{J}(\boldsymbol{q})$
22: $\mathbf{N}_{cs}(\boldsymbol{q}) = \mathbf{I} - \mathbf{J}_{cs}(\boldsymbol{q})^{\dagger}\mathbf{J}_{cs}(\boldsymbol{q})$
23: $\dot{\mathbf{q}}_{ev,cs} = \mathbf{N}_{cs}(\boldsymbol{q}) \cdot \dot{\mathbf{q}}_{ev}$
```

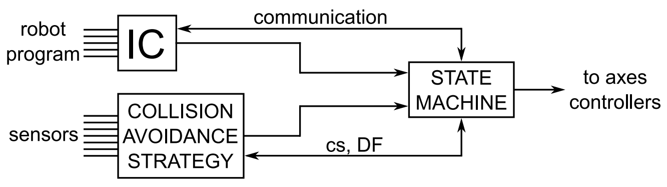


Fig. 6. Schematical representation of the safety system.

A. Industrial Controller

The task to be executed is programmed using the robot proprietary language and is executed by the industrial controller. The controller communicates with the state machine through a dedicated channel and produces the outputs $\mathbf{q}_{IC}$ and $\dot{\mathbf{q}}_{IC}$, the joint position and velocity set points, respectively. The rest of the system possibly modifies such set points, producing the new set points $\mathbf{q}_{\text{ref}}$ and $\dot{\mathbf{q}}_{\text{ref}}$.

B. Obstacle Detection and Danger Assessment System

In order to evaluate the danger field generated by the robot with respect to an obstacle, obstacle position has to be detected. The proposed system can be integrated with any sensor system capable of accomplishing such a task, and Section VI will demonstrate an implementation of the collision avoidance system using a distributed distance sensor. The obstacle detection and danger assessment system executes the following operations.

1) Sensor measurements are acquired and obstacles positions are computed.
2) Danger and evasive velocities are computed using obstacles and robot position.

C. State Machine

As described in detail in Section IV-C, the state machine defines constraints currently enforced. Moreover, it is assigned the task of communication with the industrial controller to perform task suspension and to control transition among skills. The state machine, therefore, performs the following actions:

1) It determines constraints to be enforced, based on the danger level computed from sensor measurements.
2) If danger exceeds the highest threshold, it suspends skill execution. Since suspension has to be performed on the industrial controller side, communication with the state machine is needed. Fig. 7 shows the communication protocol for skill suspension. The industrial controller performs a periodic polling and suspends skill execution according to the state machine answer. When danger decreases, the state machine activates a trajectory to make the robot return to the controller position reference (see *activate return trajectory* block in Fig. 7), as will be explained in Section V-D and sends the resumption message to the industrial controller, which then resumes skill execution.
3) When a skill is completed and transition to the next skill has to be performed, the state machine checks the robot state before allowing the transition. In a skill transition, relevance of some of the constraints may change, with *soft* constraints potentially becoming *skill* or *hard* constraints. Possible relaxed constraints may, therefore, need to be enforced in order to consistently start the next skill. Fig. 8 shows the protocol for skill transition.

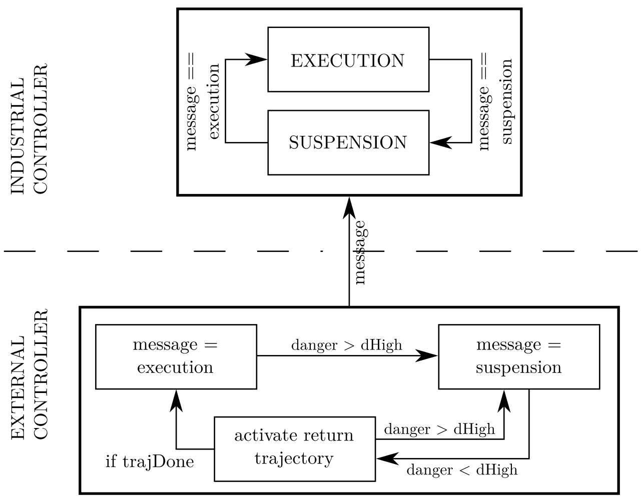


Fig. 7. Communication protocol used for skill suspension.

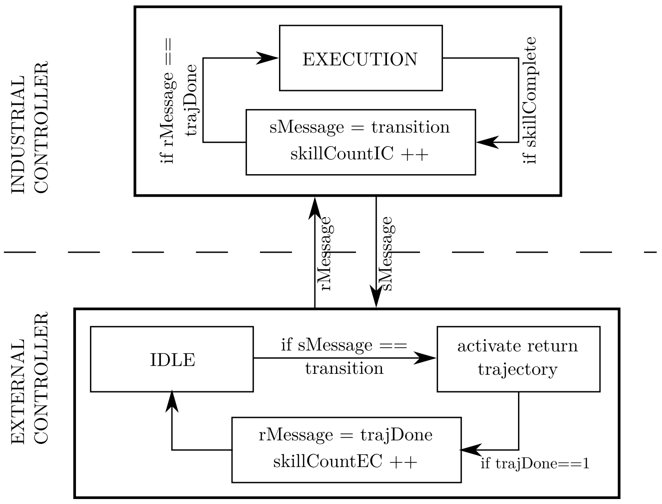


Fig. 8. State machine represents the protocol for skill transition.

D. Pose Control System

The last part of the control system manages modification of industrial controller joint velocity and position set points. It accomplishes the following functions.

1) It computes evasive velocities according to Algorithm 1 and projects them in the null space of constrained coordinates.
2) It computes robot joint position and velocity references, executing evasive velocities for unconstrained coordinates and tracking control reference for constrained coordinates.
3) When constraints are relaxed, corresponding robot operational space coordinates deviate from the industrial controller reference. If a constraint is enforced again, a return trajectory is computed to bring the corresponding robot operational space coordinate back to the controller reference. It is important to underline that returning to the point of trajectory suspension is the only possible strategy to complete the skill: the system used for trajectory modification cannot for example replan a trajectory from the current point to the skill endpoint, since, in the considered framework, it does not have access to trajectory parameters such as endpoints but only to the current set point computed by the controller.
As the controller set point is *a priori* not constant, the trajectory has to be updated at each sample time to cope with a possibly changing goal position. More specifically, a trapezoidal velocity profile trajectory is computed at every sample time, from the current pose and velocity to the reference pose. Algorithm 2 defines the steps to compute the robot joint velocity and position references.

Algorithm 2 Computation of Robot Position and Velocity Joint References for Time Instant $k+1$

```
Note : We introduce $\mathbf{k}$, the forward kinematics function,
PVTplan, which computes a trapezoidal velocity profile
trajectory from current pose and velocity to reference pose,
$\mathbf{K}_{CLIK}$, the closed loop inverse kinematic algorithm
(CLIK) gain matrix and the sampling time $dt$.
 1: for $i$=1 to $n_j$ do
 2:     if $cs_{i,k}$==1 and $cs_{i,k-1}$==0 then
 3:         $retTraj_i$=1
 4:     else if $cs_{i,k}$==0 and $cs_{i,k-1}$==1 then
 5:         $retTraj_i$=0
 6:     end if
 7: end for
 8: $\mathbf{x}_{IC,k} = \mathbf{k}(\mathbf{q}_{IC,k})$; $\mathbf{J}_{IC} = \mathbf{J}(\mathbf{q}_{IC,k})$;
    $\dot{\mathbf{x}}_{IC,k} = \mathbf{J}_{IC}\dot{\mathbf{q}}_{IC,k}$
 9: $\mathbf{x}_{e,k} = \mathbf{k}(\mathbf{q}_{e,k})$;   $\mathbf{J}_e = \mathbf{J}(\mathbf{q}_{e,k})$;   $\dot{\mathbf{x}}_{e,k} = \mathbf{J}_e\dot{\mathbf{q}}_{e,k}$
10: for $i$=1 to $n_j$ do
11:     if $x_{IC,k} - x_{e,k} < e_i$ then
12:         $retTraj_i = 0$
13:     end if
14:     if $retTraj_i == 1$ then
15:         $(x_{ret,i,k}, \dot{x}_{ret,i,k})$=PVTplan$(x_{e,i,k}, \dot{x}_{e,i,k}, x_{IC,i,k})$
16:         $\dot{x}_{ref,i,k} = \dot{x}_{ret,i,k}$
17:         $x_{ref,i,k} = x_{ret,i,k}$
18:     else
19:         $\dot{x}_{ref,i,k} = \dot{x}_{IC,i,k}$
20:         $x_{ref,i,k} = x_{IC,i,k}$
21:     end if
22: end for
23: $\dot{\mathbf{q}}_{CLIK,k+1} =$
    $(\mathbf{\Sigma}_{cs}\mathbf{J}_e)^{\dagger}\,\mathbf{\Sigma}_{cs}(\dot{\mathbf{x}}_{ref,k} + \mathbf{K}_{CLIK}(\mathbf{x}_{ref} - \mathbf{x}_{e,k}))$
24: $\dot{\mathbf{q}}_{ref,k+1} = \dot{\mathbf{q}}_{CLIK,k+1} + \dot{\mathbf{q}}_{ev,cs,k+1}$
25: $\mathbf{q}_{ref,k+1} = \mathbf{q}_{ref,k} + \dot{\mathbf{q}}_{ref,k+1} \cdot dt$
```

VI. Experimental Validation on an Assembly Task

The presented safety strategy has been applied to an assembly task performed by an ABB FRIDA[1] dual arm robot prototype. The assembly operation has been presented in [35]–[37] as an

[1]ABB FRIDA prototype has become the ABB YuMi robot, http://new.abb.com/products/robotics/yumi

TABLE II
MAIN FEATURES OF THE SENSOR SPOT

Sharp GP2Y0A21YK	
range [cm]	8–80
measurement frequency [Hz]	26
size L-W-H [mm]	29.5-18.9-15.5

application of force controlled assembly using either force measurements or sensorless force estimation. During the task, an emergency stop button consisting of several parts is assembled, through four main operations:

1) snap-fit of an electrical switch in the button base;
2) insertion of the button in the chassis hole;
3) screwing of a nut on the button thread;
4) insertion of the chassis in the button base.

Both position and force controlled skills compose the assembly task. The presented collision avoidance strategy has been applied only during position-controlled skills: although evasive motions are executed in the proper null space of the skill, they influence force measurements, due to an insufficient robot calibration accuracy.

A. Distributed Distance Sensor

A distributed distance sensor prototype has been designed and built for the ABB FRIDA robot. The distributed distance sensor is composed of a multitude of distance measuring spots mounted on the surface of the robot by means of housings. Some peculiarities make such a sensor well suited for usage in an unstructured environment and to be part of a safety system. First, as the sensor is mounted directly on the robot surface, risk of self-occlusion is eliminated and obstacles can be detected effectively. Second, the use of the sensor does not require modification of the working environment: therefore, unstructured environments can be effectively managed.

In the following, the main design problems are analyzed and a sensor description is given.

1) Sensing Spot: As previously said, the distributed sensor is composed of a multitude of spots. Various distance sensing technologies can be adopted for sensor implementation. Main requirements for the elementary spot are a package compact enough to be housed on the robot surface and a measuring range which is fit for detection of near obstacles. The device selected is a Sharp GP2Y0A21YK IR LED distance sensor, whose main features are shown in Table II.

2) Spots Placement: Sensor spots placement is a complex task, which embraces many problems. As a first step, possible areas for sensor placement on the robot surface have to be selected. The analysis to be performed for this purpose has to consider various criteria: areas should be large enough to house the sensors, they should have an effective orientation for obstacle detection, and they should not limit robot movements once sensors are mounted. Fig. 9 shows the possible areas identified on the FRIDA robot.

Once possible areas for sensor placement have been identified, geometric arrangement of sensors on such areas has to be

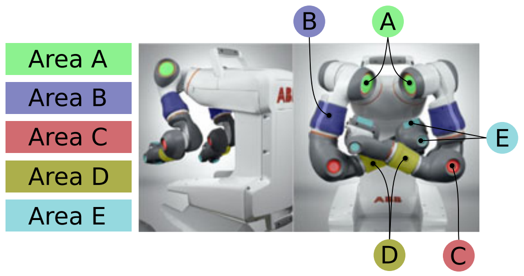


Fig. 9. Possible areas for sensor placement on the FRIDA robot surface are highlighted.

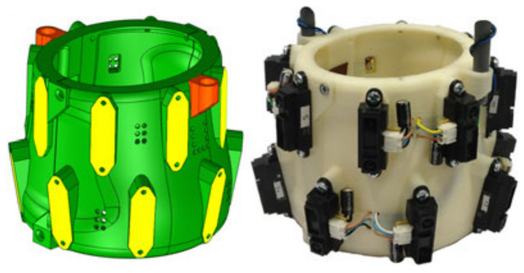


Fig. 10. CAD model of the robot upper arm sensor shell and the real sensor shell equipped with the spots.

defined. Areas B and D can be exploited with a ring arrangement of sensors around the arm axis, while on the other hand, areas A, C, and E are not suitable for such a pattern and can be used to host a single sensor each. The issue of spots placement for a cylindrical distributed sensor, to obtain a certain measurement resolution, has been addressed in [24] and is here briefly reviewed. Let us consider the cylinder-shaped sensor areas B and D and a spot pattern composed by two rings of spots with a relative angular displacement such that a spot is equidistant from the two adjacent spots of the other ring. Following equation defines the number of spots to be placed on the two rings comprehensively in order to detect an obstacle with at least one spot, given the obstacle diameter D_{obs} (under the hypothesis of a circular obstacle), the spot ring diameter D_s, and the obstacle distance R from the sensor:

$$N_{sC} = \pi / \arctan\left(\frac{D_{\text{obs}}}{2\left(\frac{D_s}{2} + R\right)}\right).$$

The sensor of area B was sized to detect an obstacle with $D_{\text{obs}}=$ 10 cm (approximately the width of an hand) at $R = 20$ cm and an obstacle with $D_{\text{obs}}=$ 30 cm (approximately the width of the torso) at $R = 70$ cm, which could be achieved using 16 spots. Area D could instead host only one ring of eight spots, which guarantees the detection of the biggest of the two obstacles at $R = 30$ cm.

Sensor spots occupying areas B and D have been mounted using special housings. Such housings, from now on called "shells," can be easily mounted on the robot surface allowing a good portability of the sensor system. Fig. 10 shows the CAD model of the shell, which has been designed starting from the robot CAD model, and the 3-D-printed shell with the distance sensors mounted.

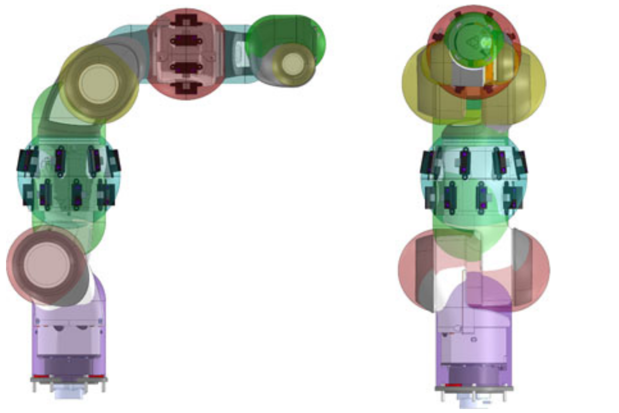


Fig. 11. Detailed triangular meshed model of one robot arm, colored in white and gray, is shown with the simplified model composed by seven capsules and two spheres superimposed, in clear colors. The simplified model is used for simulation of sensor measurements.

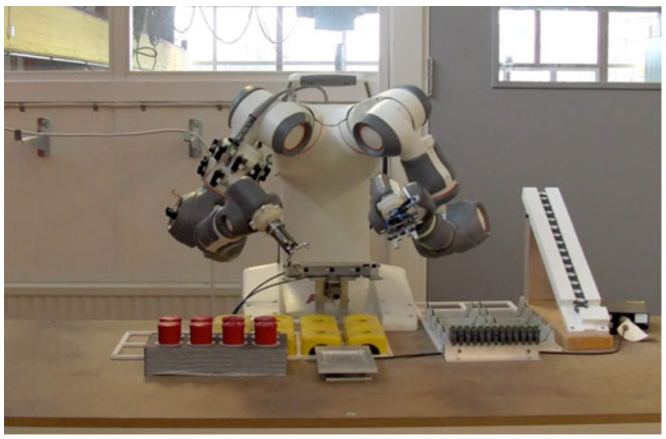


Fig. 12. Experimental setup: the robot with the distributed distance sensor on the right arm, the parts dispensers and, in the center, the metal assembly fixture with the force sensor underneath.

3) Self-Detection: The use of distance measurements in a collision avoidance system is subject to the problem of self-detection. In fact, the distributed sensor does not detect only obstacles present in the robot workspace but also parts of the robot itself. Such measurements lead to undesired evasive motions that make the robot evade from itself. To prevent such a behavior, a discrimination between detection of known objects and obstacles is needed. For this purpose, an online simulator of the distributed sensor and of the robot structure has been implemented. Given current robot joint values, the simulator computes the robot and the sensors pose. Possible sensor detections of the robot surface are then computed. Since this computation has to be performed in real time, an efficient representation of the robot structure is needed in order to reduce the heavy computational burden related to checking all possible intersections between sensor rays and robot surface. Standard triangular mesh models made available by robot manufacturers are not suitable for such a task, since they require the computation of intersections between rays and all the mesh triangles. The simulator, therefore, uses a 3-D model of the robot surface based on capsules and spheres, which greatly reduces the computational burden, while maintaining a satisfactory accuracy. Fig. 11 shows the complete triangular meshed model provided by the robot manufacturer, in white and gray, with the simplified model superimposed, in bright clear colors. At execution time, the simulator identifies those spots possibly detecting the robot surface. Measurements simulated for such spots are compared to real detected distances. If the two distances match, a self-detection has occurred and the measurement is not considered for evasive motions computation. Otherwise, an obstacle has been detected and the real measurement is taken into account.

B. Experimental Setup

The ABB FRIDA manipulator is composed of two 7-DOF lightweight arms. The robot is mounted on a work table where the button parts dispensers are located. Moreover, a force sensor with a fixture for the snap-fit assembly is located on the table. The experimental setting is shown in Fig. 12. The robot right arm is equipped with the distributed distance sensor, which has been implemented with 17 spots, 16 of which are mounted on the upper arm sensor shell and the last is mounted on the arm C area. An improved version of the sensor composed by 25 spots, eight of which are placed on the lower arm, has been later implemented but has not been adopted for this experiment. As previously introduced in Section V, the control system is based on the modification of the industrial controller position and velocity reference. Such action is performed using the External Control interface [38]. This system, developed by Lund University, allows an external real time PC to read and modify joint references computed by the industrial controller at a frequency of 250 Hz. It also allows communication with the controller through a dedicated channel. On the external PC, the desired control strategies are implemented using Simulink, which also allows external sensors acquisition. During position-controlled skills, both the industrial controller and the external PC are active and contribute to robot control, while during force controlled skills execution, the industrial controller is inactive and the external PC has the complete control of the manipulator.

The control system parameters have been experimentally tuned. Still, some tuning suggestions can be drawn from the performed experiments. Impedance filter parameters $\mathbf{M}$ and $\mathbf{D}$ define the dynamic behavior of the robot joints when virtual torques are applied. Diagonal matrices have been chosen for both, in order to decouple the joints response. The elements of $\mathbf{M}$ and $\mathbf{D}$ define the natural frequencies and the gains of the joint filters. Since the amplitude of the virtual torques depends on the danger field parameters, filters gain can be chosen arbitrarily (e.g., can be assigned to 1). The ratio between the diagonal elements of $\mathbf{D}$ and $\mathbf{M}$ instead defines the natural frequency. A value of 10 rad/s for every joint filter gave a satisfactory responsiveness of the robot evasion. Danger field thresholds DF_{low}, DF_{med}, and DF_{high} depend on the choice of the danger field parameters k_1 and k_2, for which [32] can be considered. Values of 8, 12, and 16 were, respectively, selected for three thresholds: such values were identified performing experiments with

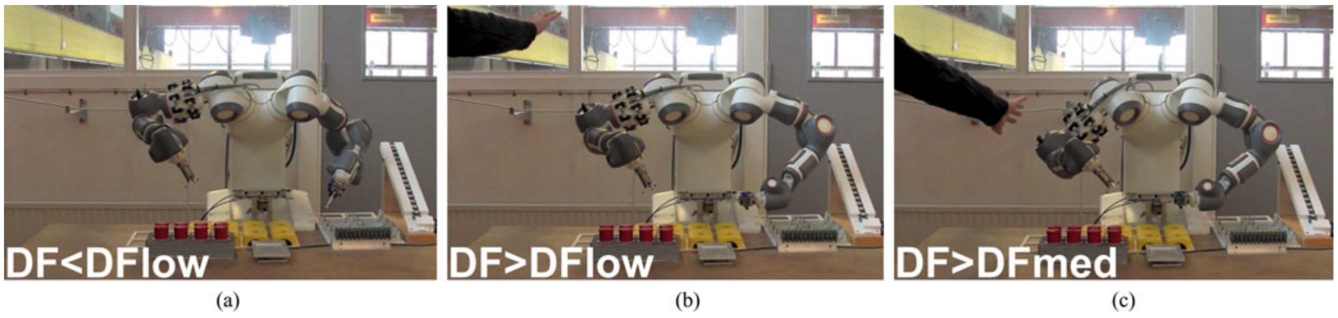


Fig. 13. Three phases of collision avoidance during free space movement. (a) Robot operating without obstacles. (b) Swivel angle is relaxed for evasion. (c) TCP orientation is relaxed.

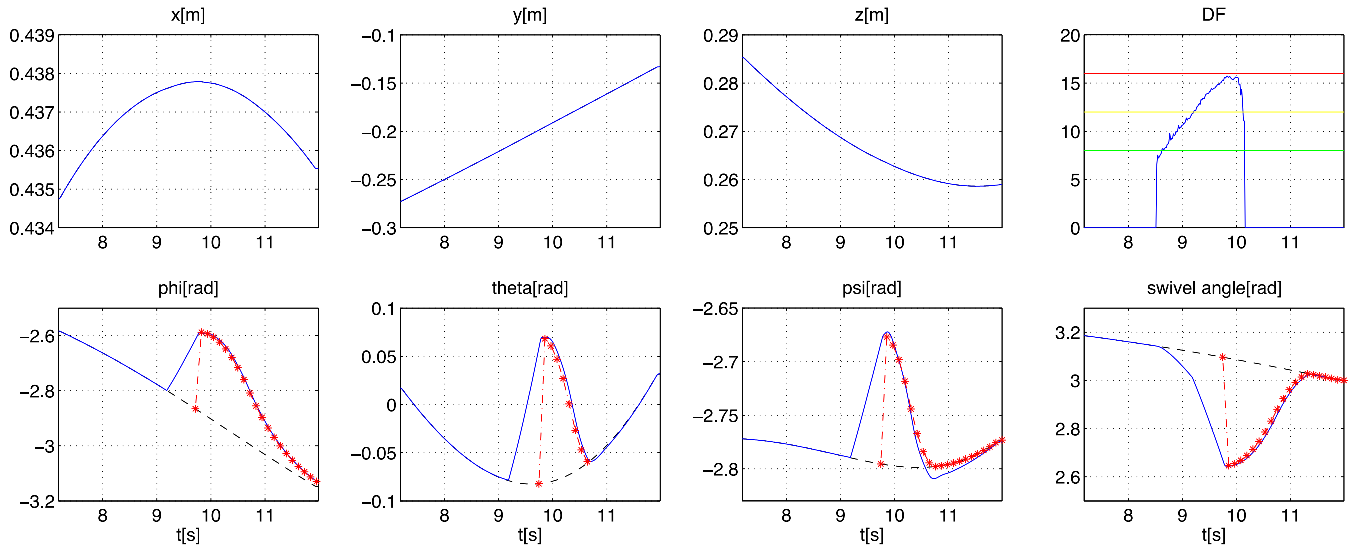


Fig. 14. Modification and return to robot controller set points during evasion. Solid blue lines represent current TCP coordinates and swivel angle. Black dashed lines show the industrial controller reference and red dash-dotted lines represent return trajectories. Top-right plot shows danger value.

a human standing still at the extremities of the sensor range, and the robot performing a sample task. DF_{low} and DF_{high} were assigned the minimum and the maximum measured danger and DF_{med} was assigned the mean of the two. $\mathbf{K}_{\mathrm{CLIK}}$ has been chosen as a diagonal matrix.

C. Skills

Eleven position-controlled skills compose the assembly task. A video of the experiment was published in [31] and can be accessed also in [39]. In the following, two skills are analyzed and safety system behavior during their execution is examined. Moreover, in order to clarify the collision avoidance strategy, an example of suspension of skill execution is given.

1) Button Approach: The first skill executed by the robot is the free space movement from the home position to the button dispenser, where the button is then picked. During this skill, no external part constrains the robot right arm end-effector, nor any constraint is applied by a manipulated object. In order to complete the skill, the robot end-effector has to reach the skill final position, so translational velocities are classified as *skill* and rotational velocities are classified as *soft*, while no constraint is classified as *hard*. Using three Cartesian axes to describe position and *XYZ* Euler angles to describe orientation, given the task velocities vector $\mathbf{v} = [v_x, v_y, v_z, \omega_\phi, \omega_\theta, \omega_\psi]$, the selection vectors are:

$$\mathbf{s}_{\mathrm{soft}} = [0\ 0\ 0\ 1\ 1\ 1], \quad \mathbf{s}_{\mathrm{skill}} = [1\ 1\ 1\ 0\ 0\ 0],$$
$$\mathbf{s}_{\mathrm{hard}} = [0\ 0\ 0\ 0\ 0\ 0].$$

Fig. 13 shows three snapshots of the skill execution during which an obstacle approaches the robot. The pictures correspond to increasing levels of danger and, therefore, to the progressive relaxation of constraints of increasing relevance. In Fig. 13(a), danger is lower than DF_{low}, so no constraint is relaxed. As danger exceeds such threshold, swivel angle is relaxed as shown in Fig. 13(b) and finally also TCP orientation is unconstrained, as shown in Fig. 13(c). Evolution of pose set points during such skill is depicted in Fig. 14. Industrial controller references (dashed magenta) are modified by evasive velocities to the actual positions (solid blue). When danger decreases below DF_{med} and then DF_{low}, a return trajectory is computed (starred red)

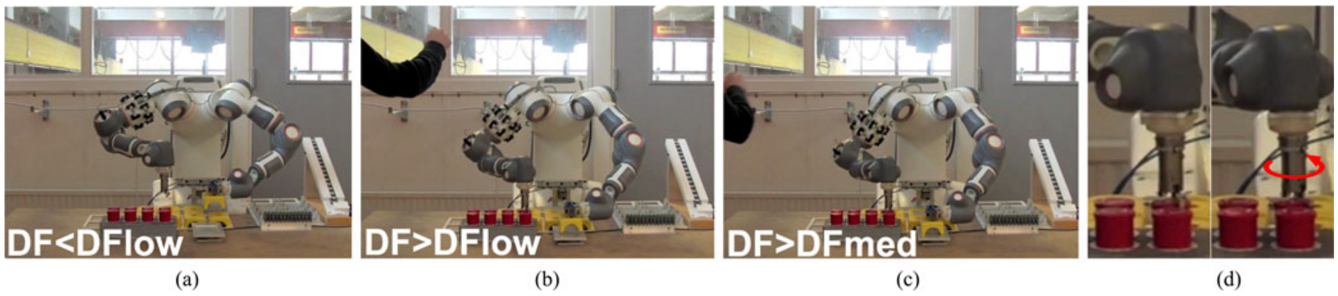


Fig. 15. Three phases of collision avoidance during button picking. Rotation in the button housing is highlighted in snapshot d. (a) Robot picking a button from the dispenser. (b) Evasion is performed relaxing the swivel angle. (c) Button orientation inside the dispenser hole is relaxed. (d) Closeup of the button rotation.

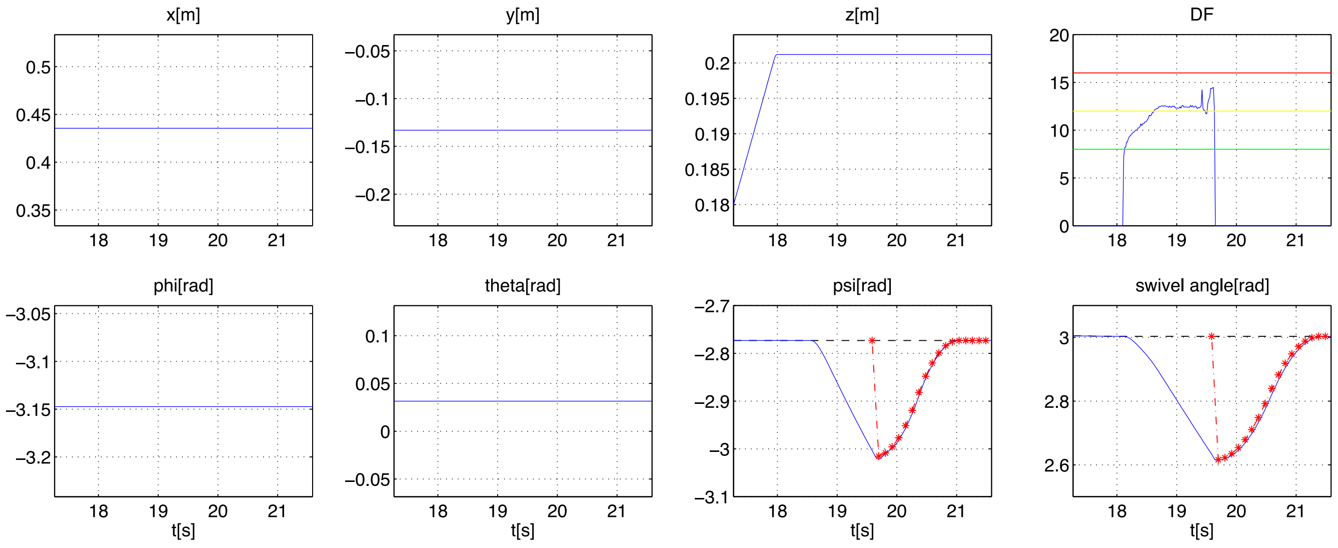


Fig. 16. Modification and return to robot controller set-points during the second experiment. Solid blue lines represent current TCP coordinates and swivel angle. Black dashed lines show the industrial controller reference and red dash-dotted lines represent return trajectories. Top-right plot shows danger value.

for TCP orientation and swivel angle to bring robot pose back to the controller reference.

2) Button Picking: The second skill considered is button picking, during which a push button is extracted from the part dispenser for assembly. The button and its housing constitute a cylindrical pair. Moreover button orientation around the housing axis is not relevant for skill completion, as the assembly procedure can cope with any orientation. The blind hole that houses the button unilaterally limits its motion: if translation along its axis were relaxed during an evasion, the button might hit the hole bottom. For this reason, a *hard* constraint is assigned to this coordinate. Assuming the z-axis coincides with the button axis and *XYZ* Euler angles are adopted, the selection vectors can be defined as

$$\begin{aligned}\mathbf{s}_{\text{soft}} &= [0\ 0\ 0\ 0\ 0\ 1],\ \mathbf{s}_{\text{skill}} = [0\ 0\ 0\ 0\ 0\ 0],\\ \mathbf{s}_{\text{hard}} &= [1\ 1\ 1\ 1\ 1\ 0].\end{aligned}$$

Orientation around the button axis is a *soft* constraint, since such coordinate does not affect task execution. All other coordinates are instead hard-constrained. Similarly to Fig. 13, Fig. 15 shows the different phases of evasion during skill execution. Fig. 15(d) highlights rotation around the button axis performed to decrease danger level. As in Fig. 14, Fig. 16 shows controller references modification and return trajectories computed when danger decreases.

3) Skill Suspension: In order to verify the behavior of the collision avoidance strategy in all possible conditions, it is important to validate it during skill suspension, when the strategy has to control the industrial controller operation. When the level of danger exceeds the highest threshold, the strategy commands to the industrial controller the suspension of skill execution, in order to exploit all possible DOF to evade from the detected obstacle. Fig. 17 shows the operation of the collision avoidance strategy in such a condition. A skill characterized by the following selection vectors has been considered:

$$\begin{aligned}\mathbf{s}_{\text{soft}} &= [0\ 0\ 0\ 0\ 0\ 1],\ \mathbf{s}_{\text{skill}} = [1\ 1\ 1\ 0\ 0\ 0],\\ \mathbf{s}_{\text{hard}} &= [0\ 0\ 0\ 1\ 1\ 0].\end{aligned}$$

When danger exceeds DF_{low} and DF_{med}, first the swivel angle, and then *soft* constraints, that is the angle ψ, are relaxed. When also DF_{high} is exceeded, position constraints, that is *skill* ones, are relaxed, and skill execution is suspended. *Hard*

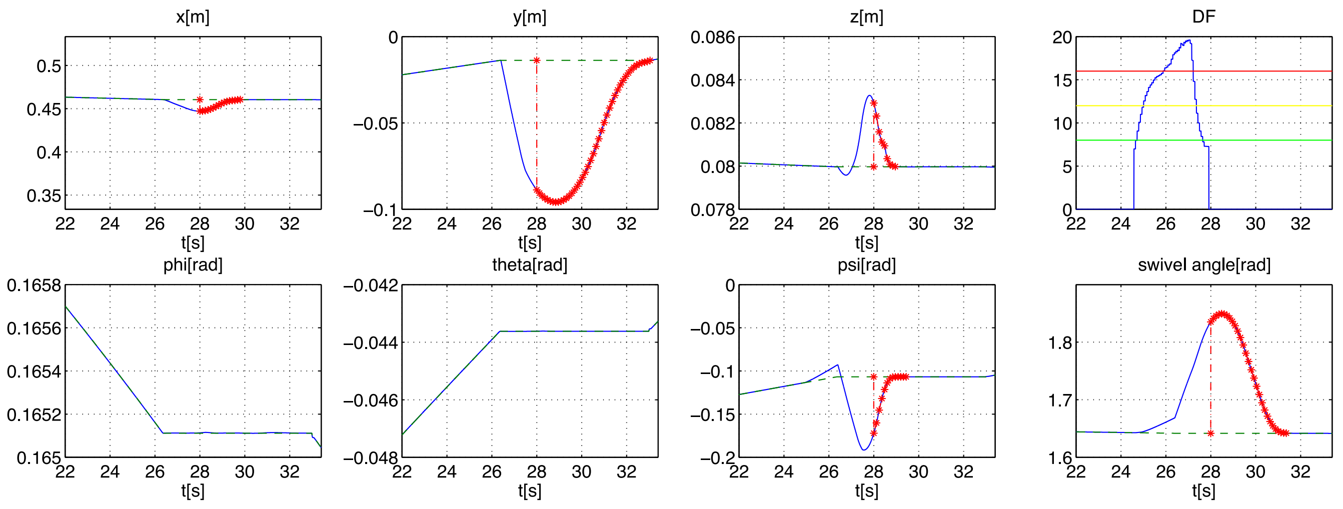


Fig. 17. Example of skill suspension. When danger exceeds the DF_{high} threshold, skill execution is suspended, as can be seen by the constant values of the plots of ϕ and θ. When danger decreases return trajectories are executed and skill execution is resumed.

constraints are thus not relaxed, and their values remain constant until skill resumption, as can be seen by the plots of ϕ and θ.

VII. Discussion

The projection of evasive velocities in the null space of the active operational space constraints allows adaptation to an unstructured environment and, at the same time, the successful completion of the task. However, the proposed strategy is not able to deal with the unilateral constraints that characterize real robots, such as joint limits and maximum actuators velocities. Solutions such as virtual impedance in the joint space can alleviate the problem of reaching the limits of the allowed robot state space, but the achievement of satisfactory performance greatly depends on impedance parameters tuning. A promising solution to manage operational space constraints and unilateral constraints in the joint space is the adoption of an optimization approach: execution of evasive motions is considered as a cost function to optimize, as long as the set of different constraints to be considered is respected.

The management of *hard* constraints presented in the paper apparently limits the actions available to ensure the worker's safety. In fact, when *skill* constraints are relaxed, no other possibilities are left to decrease the danger level. As a matter of fact a safety stop can be given when the danger level becomes unacceptable. In this way, the possibility of task completion is still preserved.

A further possible development of the proposed strategy could be its application to dual arm skills. When the same object is jointly manipulated by two arms, constraints between the two TCPs arise, and evasive motions have to be performed consistently with them. In order to apply the proposed strategy to such a scenario, a similar procedure to the one presented in this paper can be followed. Constraints between the two arms can be classified according to the proposed classification, and once the Jacobian of the arms relative coordinates is computed (see, e.g., [40] and [41]), evasive velocities can be projected in the null space of the active constraints, as done in the single arm case.

VIII. Conclusion

This paper presented a complete task-consistent collision avoidance system for safe human–robot interaction. The system includes both hardware and software components and is designed to be integrated with an industrial robot controller. Firstly, a classification for constraints constituting the robot task based on their relevance for task completion has been proposed. The classification also partitions the task into skills, that is elementary actions each characterized by different relevance of constraints. Then, based on this classification, a task-consistent collision avoidance strategy has been defined. The strategy sets the constraints to be relaxed to allow evasion from detected obstacles according to the assessed danger level and consistently with the skill. A distributed distance sensor, which has been designed and built for an ABB FRIDA dual-arm robot prototype, performs obstacle detection. The proposed safety strategy has then been integrated with industrial robot controller task execution. Finally, the system has been experimentally validated on the ABB FRIDA robot performing an emergency stop button assembly task. The robot equipped with the distributed distance sensor and the safety system was able to effectively combine obstacle collision avoidance and execution of the task, by performing evasive motions consistently with constrained DOFs.

References

[1] V. J. Lumelsky, M. Shur, and S. Wagner, "Sensitive skin," *IEEE Sens. J.*, vol. 1, no. 1, pp. 41–51, Jun. 2001.

[2] S. Lee, Y. Yamada, K. Ichikawa, O. Matsumoto, K. Homma, and E. Ono, "Safety-function design for the control system of a human-cooperative robot based on functional safety of hardware and software," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 2, pp. 719–729, Apr. 2014.

[3] P. Yadmellat, A. Shafer, and M. Kermani, "Design and development of a single-motor, two-dof, safe manipulator," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 4, pp. 1384–1391, Aug. 2014.

[4] A. Jafari, N. Tsagarakis, I. Sardellitti, and D. Caldwell, "A new actuator with adjustable stiffness based on a variable ratio lever mechanism," *IEEE/ASME Trans. Mechatronics*, vol. 19, no. 1, pp. 55–63, Feb. 2014.

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

[6] T. Tsuji, H. Akamatsu, and M. Kaneko, "Non-contact impedance control for redundant manipulators using visual information," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1997, vol. 3, pp. 2571–2576.

[7] T. Tsuji, H. Akamatsu, M. Hatagi, and M. Kaneko, "Vision-based impedance control for robot manipulators," in *Proc. IEEE/ASME Int. Conf. Adv. Int. Mechatron.*, Jun. 1997, p. 53.

[8] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1993, vol. 2, pp. 802–807.

[9] O. Brock and O. Khatib, "Elastic strips: Real-time path modification for mobile manipulation," in *Proc. Int. Sympos. Robot. Res.*, 1997, pp. 117–122.

[10] O. Brock and O. Khatib, "Elastic strips," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 1031–1052, 2002.

[11] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 6, pp. 418–432, Jun. 1981.

[12] M. Stilman, "Global manipulation planning in robot joint space with task constraints," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 576–584, Jun. 2010.

[13] T. Kunz and M. Stilman, "Manipulation planning with soft task constraints," in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Oct. 2012, pp. 1937–1942.

[14] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th Int. Conf. Adv. Robot.*, Jun. 1991, vol. 2, pp. 1211–1216.

[15] O. Kanoun, F. Lamiraux, and P. B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

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

[17] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 338–345.

[18] H. Baltzakis, A. Argyros, and P. Trahanias, "Fusion of laser and visual data for robot motion planning and collision avoidance," *Mach. Vision Appl.*, vol. 15, no. 2, pp. 92–100, 2003.

[19] Y. Lu, L. Zeng, and G. Bone, "Multisensor system for safer human-robot interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2005, pp. 1767–1772.

[20] R. Luo and O. Chen, "Wireless and pyroelectric sensory fusion system for indoor human/robot localization and monitoring," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 3, pp. 845–853, Jun. 2013.

[21] E. Cheung and V. J. Lumelsky, "Proximity sensing in robot manipulator motion planning: system and implementation issues," *IEEE Trans. Robot. Autom.*, vol. 5, no. 6, pp. 740–751, Dec 1989.

[22] J. L. Novak and J. Feddema, "A capacitance-based proximity sensor for whole arm obstacle avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1992, vol. 2, pp. 1307–1314.

[23] T. Schlegl, T. Kröger, A. Gaschler, O. Khatib, and H. Zangl, "Virtual whiskers—Highly responsive robot collision avoidance," in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Nov. 2013, pp. 5373–5379.

[24] N. M. Ceriani, G. Buizza Avanzini, A. M. Zanchettin, L. Bascetta, and P. Rocco, "Optimal placement of spots in distributed proximity sensors for safe human-robot interaction," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5858–5863.

[25] G. Buizza Avanzini, N. M. Ceriani, A. M. Zanchettin, P. Rocco, and L. Bascetta, "Safety control of industrial robots based on a distributed distance sensor," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 6, pp. 2127–2140, Nov. 2014.

[26] T. Kunz and M. Stilman, "Manipulation planning with soft task constraints," in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Oct. 2012, pp. 1937–1942.

[27] P.-G. C. and D. Berenson, "A representation of deformable objects for motion planning with no physical simulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2014, pp. 98–105.

[28] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 60–72, Feb. 2007.

[29] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *Proc. Int. Conf. Adv. Robot.*, Jun. 2009, pp. 1–6.

[30] 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 Robot. Autom.*, vol. 19, no. 1, pp. 162–167, Feb. 2003.

[31] N. M. Ceriani, A. M. Zanchettin, P. Rocco, A. Stolt, and A. Robertsson, "A constraint-based strategy for task-consistent safe human-robot interaction," in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Nov. 2013, pp. 4630–4635.

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

[33] B. Lacevic and P. Rocco, "Safety-oriented control of robotic manipulators—A kinematic approach," in *Proc. 18th IFAC World Cong.*, Aug./Sep. 2011, pp. 11508–11513.

[34] K. Kreutz-Delgado, M. Long, and H. Seraji, "Kinematic analysis of 7 dof anthropomorphic arms," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1990, vol. 2, pp. 824–830.

[35] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, "Force controlled robotic assembly without a force sensor," in *Proc. Robot. Autom. (ICRA), 2012 IEEE Int. Conf.*, May 2012, pp. 1538–1543.

[36] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansso, 'Force controlled assembly of emergency stop button," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3751–3756.

[37] M. Linderoth, A. Stolt, A. Robertsson, and R. Johansson, "Robotic force estimation using motor torques and modeling of low velocity friction disturbances," in *Proc. IEEE/RSJ Int. Conf. Int. Robot. Syst.*, Nov. 2013, pp. 3550–3556.

[38] A. Blomdell, G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. Wang, "Extending an industrial robot controller: implementation and applications of a fast open sensor interface," *IEEE Robot. Autom. Mag.*, vol. 12, no. 3, pp. 85–94, Sep. 2005.

[39] N.-M. Ceriani. (2014, Sep. 27). Task-consistent safe human-robot interaction. [Online]. Available: http://youtu.be/CYUNSksIXZU

[40] W. Owen, E. Croft, and B. Benhabib, "On-line trajectory resolution for two-armed systems with conflicting performance criteria," *Mech. Mach. Theory*, vol. 44, no. 5, pp. 949–965, 2009.

[41] J. Lee, P. Chang, and R. Jamisola, "Relative impedance control for dual-arm robots performing asymmetric bimanual tasks," *IEEE Trans. Ind. Electron.,* vol. 61, no. 7, pp. 3786–3796, Jul. 2014.