POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

A TASK-CONSISTENT SAFETY SYSTEM FOR CLOSE COOPERATION BETWEEN HUMAN WORKERS AND DUAL ARM ROBOTIC MANIPULATORS

Doctoral Dissertation of:
Nicola Maria Ceriani

Supervisor:
Prof. Paolo Rocco

Tutor:
Prof. Marco Lovera

The Chair of the Doctoral Program:
Prof. Carlo Fiorini

2014 – XXVII Cycle

A Valentina
e ai miei Genitori

«A thousand policemen directing the traffic
Cannot tell you why you come or where you go.

[...]

When the Stranger says: “What is the meaning of this city?
Do you huddle close together because you love each other?”
What will you answer? “We all dwell together
To make money from each other”? or “This is a community”?»

(T.S. Eliot, “Choruses from The Rock")

«O brave new world, that hath such people in it!»

(A. Huxley, “Brave New World")

Acknowledgements

First of all, I would like to thank Prof. Paolo Rocco for the experience that I have had the opportunity to enjoy in these three years of PhD program. Working in his research group I have been introduced to the world of research, and I have been challenged with the hard task of giving a contribution to technology.

I want to thank Prof. Gianantonio Magnani for the opportunity of working with him and knowing him.

I would like to thank Dr. Andrea Zanchettin for the huge amount of practical help, interesting discussions and suggestions.

I thank all the members of the Merlin group: Prof. Luca Bascetta, with whom I have collaborated working on my first research project, and all my office mates, Amir Ghalamzan, Giovanni Buizza Avanzini, Matteo Ragaglia, Antonio Valsecchi and Roberto Rossi, that have been good friends in this journey.

I thank Anders Robertsson, Andreas Stolt, Magnus Linderoth, Anders Nilsson and Eva Westin for the warm welcoming they have given me during my research stay in Lund and for the fruitful collaboration I have had with them.

I finally thank Prof. Gianluca Antonelli, the reviewer of this thesis, for the useful suggestions to improve this work.

Abstract

Despite the availability of robotic manipulators with high levels of productivity, precision and dexterity, large sectors of the manufacturing industry still rely on manual assembly for mass production of goods. Flexibility needed to keep up with rapidly changing products and production rates is often provided by human workforce, which is employed according to production demand, usually to perform repetitive and tiring operations. Human-robot collaboration is a promising solution to the problem: in fact, on one hand, robot indefatigability may relieve human workers from repetitive operations and, on the other hand, human work could increase the flexibility and the adaptability of the automated solution. Moreover, the increasing availability of dual arm, redundant, and human like, industrial robots is offering important tools to develop the human-robot collaboration production setup. However, such a scenario opens new challenges and requires novel methodologies to preserve workers' safety and robots productivity.

The goal of this thesis is to develop a system for safe human-robot interaction, which avoids collisions between humans and robots, while guaranteeing completion of the task. In order to preserve productivity, the constraints composing the production task have to be identified and taken into account. For this, a classification of constraints defining a preplanned robot task, based on their relevance for task execution, is proposed, and a task-consistent sensor based collision avoidance strategy is presented. Moreover, this research aims at integrating the safety system with an industrial controller, in order to extend its functionalities with capabilities for adaptation to unstructured environments. By doing so, functionalities currently offered by industrial controllers are preserved, and the application of safety strategies to real world scenarios is simplified. All the results of this thesis are validated on the ABB dual arm concept robot (FRIDA).

Riassunto

Nonostante la disponibilità di robot dotati di elevata produttività, precisione e destrezza, in settori significativi dell'industria manifatturiera la produzione di massa di beni è ancora affidata alla lavorazione manuale. La flessibilità necessaria per realizzare prodotti che cambiano continuamente nei modelli e nei volumi di produzione, è ancora spesso ottenuta facendo ricorso alla manodopera umana, che viene impiegata in dipendenza della domanda, in lavori logoranti e ripetitivi. La collaborazione uomo-robot si offre come una soluzione promettente a questa situazione: infatti, se da un lato i robot possono sollevare i lavoratori umani dagli impieghi più ripetitivi, dall'altro l'uomo può accrescere la flessibilità e la capacità di adattamento dei robot grazie alla sua abilità. Inoltre, la crescente disponibilità di manipolatori industriali a due braccia, cinematicamente ridondanti e caratterizzati da un aspetto simile a quello umano, rappresenta un importante strumento per lo sviluppo dell'interazione uomo-robot. Questa prospettiva apre però nuove sfide tecniche e richiede nuovi metodi per garantire al contempo la produttività dei robot e la sicurezza degli operatori.

Lo scopo di questa tesi è lo sviluppo di un sistema per l'interazione sicura uomo-robot, che sia in grado di evitare le collisioni tra i due soggetti, garantendo però il completamento delle lavorazioni. Per preservare la produttività del robot, i vincoli che caratterizzano l'operazione compiuta devono essere identificati e adeguatamente considerati. A tal scopo, in questa tesi, viene proposta una classificazione dei vincoli che compongono un'operazione pre pianificata, in funzione della loro rilevanza per il completamento dell'operazione stessa. Viene poi introdotta una strategia anti collisione basata su tale classificazione e sull'utilizzo di sensori per il rilevamento di ostacoli. Inoltre, questa ricerca si pone l'obbiettivo di integrare il sistema di sicurezza con un controllore industriale, dotan-

dolo così della capacità di adattamento ad ambienti non strutturati. In questo modo vengono conservate le funzionalità originali del controllore e viene semplificata l'applicazione a casi applicativi reali delle strategie di sicurezza. I risultati ottenuti in questa tesi sono stati verificati sperimentalmente utilizzando un prototipo di robot a due braccia, l'ABB FRIDA.

Outline

INDUSTRIAL manipulators represent a significant element for the automation of some industry sectors, such as cars manufacturing, or for machines tending and parts movement. However, their diffusion in productive settings such as consumer electronics industry is hampered by an insufficient flexibility, or by an excessive cost for their application. Human robot interaction is a promising solution to such a problem, as cooperation between robots and workers could greatly increase robots flexibility, and, at the same time, the adoption of manipulators that are safe for human robot interaction would reduce the costs related to environment structuring.

The deployment of industrial robots in human robot collaboration scenarios poses new challenges for robot manufacturers: guaranteeing safety for human operators cooperating with robots, while achieving productivity in unstructured environments. Robots should appear friendly to workers, avoid collisions and reduce the risk of consequent injuries. At the same time, the pursue of safety must not diminish robots productivity, nor should it disrupt the possibility of task completion or generate a risk of damages for the manipulator or the production setup.

Industrial robots specifically designed for human robot interaction are becoming available in the market, and are characterised by human like dual arm configurations, lightweight or soft padded structures and sensing systems for force and torque control specifically designed for physical interaction. However, industrial robot controllers currently lack the features needed to ensure safe and productive human robot interaction. Nonetheless, the significant amount of control functionalities already offered by such devices make an extension of their functionalities more desirable than a complete redesign.

This thesis aims at extending an industrial controller functionalities with a collision avoidance system, in order to allow the robot operation in unstructured environments and in close cooperation with humans. This research therefore contributes to the use of industrial robots in new production scenarios and proposes a safety system which can be implemented adopting existing technologies, shortening the gap between research and application.

Thesis contributions and organization

In this thesis, the following main contributions are given:

1. a classification of constraints that form a preplanned task, based on their relevance for task execution, is introduced;
2. a task-consistent collision avoidance strategy, which acts modifying a preplanned trajectory at execution time, based on the above mentioned constraints classification, is proposed;
3. a system for the extension of an industrial robot controller functionalities with the proposed collision avoidance strategy is designed and implemented.

The thesis is organised as follows:

In **Chapter 2** a classification of constraints forming a preplanned task, based on their relevance for task completion, is presented. Such a classification can be used to identify the feasible modifications of a preplanned trajectory, and thus to adapt it to unforeseen events, while preserving the possibility of its completion.

In **Chapter 3** a task consistent collision avoidance strategy, based on the previously introduced constraints classification, is proposed. The purpose of the strategy is to select which constraints to relax, in order to make some of the robot degrees of freedom available for the evasion of the robot from obstacles. For this purpose, an assessment of danger generated by the robot towards surrounding obstacles is adopted. As danger level increases, constraints of increasing relevance are relaxed, progressively dedicating more robot degrees of freedom to collision avoidance. Moreover, evasive motions are computed using such an assessment.

In **Chapter 4** a communication system for the integration of the proposed collision avoidance strategy with an industrial controller is introduced. Such a system allows the extension of the industrial controller functionalities with the mentioned strategy for adaptation

to unforeseen events. Moreover, it allows the robot programmer to control such functionalities by means of the standard controller interface, hence allowing a true integration of them.

In **Chapters 5 and 6**, two different implementations of the collision avoidance strategy are proposed, which progressively take into account a more complete set of constraints. Only task constraints are considered at first, and robot kinematic limitations are then added. Experimental validation on a robot with two kinematically redundant arms is performed, demonstrating the effectiveness of the proposed strategy on two different tasks. Moreover, a distributed distance sensor prototype specifically designed for collision avoidance applications is proposed. Finally, limitations for the real time implementation of a further version of the strategy, exploiting both robot arms for evasion, are discussed.

Publications

This thesis is based on the following publications:

1. G. Buizza Avanzini, N.M. Ceriani, A.M. Zanchettin, P. Rocco, L. Bascetta - "Safety control of industrial robots based on a distributed distance sensor", IEEE Transactions on Control System Technology, (published online, doi: 10.1109/TCST.2014.2300696).
2. N.M. Ceriani, A.M. Zanchettin, P. Rocco - "Collision Avoidance with Task Constraints and Kinematic Limitations for Dual Arm Robots", International Conference on Intelligent Autonomous Systems, IAS 2014, Padua/Venice (Italy), July 2014, 15th-18th (accepted).
3. L. Bascetta, G. Buizza Avanzini, N.M. Ceriani, M. Ragaglia, P. Rocco, A.M. Zanchettin - "Interazione sicura uomo-robot in ambiente industriale", Motion Control 2013, Cinisello Balsamo (Italy), November 2013, 20th-21st.
4. N.M. Ceriani, A.M. Zanchettin, P. Rocco, A. Stolt, A. Robertsson - "A constraint-based strategy for task-consistent safe human-robot interaction", IEEE/RSJ International Conference on Robots and Intelligent Systems, IROS 2013, Tokyo (Japan), November 2013, 3rd-7th.
5. N.M. Ceriani, A.M. Zanchettin, P. Rocco - "Task constraints classification and exploitation for safe and productive human-robot interaction", HFR2013, 6th International Workshop on Human-Friendly Robotics, Rome (Italy), September 2013, 25th-26th.
6. G. Buizza Avanzini, N.M. Ceriani, A.M. Zanchettin, L. Bascetta, P. Rocco - "A distributed proximity sensor for safe human-robot interaction", HFR2013, 6th Interna-

tional Workshop on Human-Friendly Robotics, Rome (Italy), September 2013, 25th-26th.

7. A.M. Zanchettin, N.M. Ceriani, P. Rocco - "Reactive motion planning for robotic manipulators: problem setting and challenges", Automatica.it 2013, Palermo (Italy), September 2013, 16th-18th.

8. N.M. Ceriani, G. Buizza Avanzini, A.M. Zanchettin, L. Bascetta, P. Rocco - "Optimal placement of spots in distributed proximity sensors for safe human-robot interaction", IEEE International Conference on Robotics and Automation, ICRA 2013, Karlsruhe (Germany), May 2013, 6th-10th.

and on the following submitted material:

1. N.M. Ceriani, A. M. Zanchettin, P. Rocco, A. Stolt, A. Robertsson - "Reactive task adaptation based on hierarchical constraints classification for safe industrial robots", IEEE/ASME Transactions on Mechatronics (under review).

2. A.M. Zanchettin, N.M. Ceriani, P. Rocco, H. Ding, B. Matthias, - "Safety in human-robot collaborative manufacturing environments: metrics and control", IEEE Transactions on Automation Science and Engineering, (under review).

Cooperation with other institutions

The experimental validation of the strategy presented in Chapter 5 has been carried out in the robotic laboratory of the Department of Automatic Control (Reglerteknik) at Lund University.

Contents

CHAPTER 1

Introduction and state of the art

Human-robot interaction is one of the most interesting and promising current areas of technology development. In fact, its goal is making automation reach not only new sectors of industry, but also important parts of everyday life, where human labour is still the predominant factor for the accomplishment of most of the tasks. Industry has obviously a strong interest in the development of such a technology, since the demand for a higher level of automation of the productive processes is growing. Even though emerging countries are offering low cost workforce, the development of such countries has the positive effect of improving working conditions and salaries [24], hence slowly decreasing the benefit of cheap labour. At the same time, developed countries are seeking new automation solutions to counterbalance the disadvantage of their high labour cost. Human-robot interaction can play a central role in solving the mentioned problems. In fact, the collaboration between the two players combines human adaptability and problem solving capability with robot precision and indefatigability, and makes the deployment of robots feasible also in sectors where manual work is still predominant. In a human-robot collaboration scenario, the combination of the two types of workers reduces the need of auxiliary automation for robot tending, relieves human workers from repetitive work, decreases the working environment structuring [68] and increases the automated system flexibility.
However, such a scenario calls for new robotised systems capable of managing the chal-

lenges implied by this production setup, that can be mainly divided in safety and adaptability. Safety issues arise from the idea itself of human-robot collaboration: industrial robots normally operate in closed-ended environments (cages), where human-robot interaction takes place only when the robot is not operating, since a moving robot represents a source of danger for a human. If the two are to collaborate in close proximity, systems are therefore needed to ensure human safety. Such a problem has been tackled from many different perspectives: robot design for safety, with focus on the robot structure, actuating system and sensing capability, robot control, focusing on performing harmless motions, avoiding collisions or ensuring safe ones, and, finally, sensing systems, which constitute an important element for all safety strategies.
At the same time, operating in an unstructured environment requires the capability of adapting a robot task to unexpected events and changing conditions. Various strategies can contribute to cope with such a problem: modification of preplanned trajectories, reactive motion planning, path planning, collision avoidance strategies and once again the use of ad hoc sensor systems.
Various overviews of the issues and of the main research directions related to human-robot interaction can be found in the literature. In [1], an analysis of metrics used to assess robot suitability for physical human-robot interaction is given. Different approaches to mechanical design, robot actuation and control are evaluated based on their safety and dependability. Finally, standards regulating human-robot interaction are reviewed. In [78] a state of the art of works contributing to development of physical human-robot interaction is offered. Three main areas, interaction safety assessment, interaction safety through robot design, and interaction safety through planning and control, are identified as the main research directions, and current open issues are outlined. Finally, in [29] an evolution of [1] is proposed, focusing once again on the metrics for the assessment of robots suitability for physical interaction with humans and suggesting possible new metrics for the development of such application.
In the following, the main contributions to the design of a robotic system for human-robot interaction are reviewed.

1.1 Intrinsically safe robots

One important approach to the problem of ensuring safety in human-robot interaction is the design of intrinsically safe robot. The goal of such a methodology is to create manipulators whose operation is harmless to humans, independently from the task being performed. As a first step for the design of an intrinsically safe robot, safety requirements have to be identified and methods for their verification have to be proposed. In [45] human-care robots are considered as the case study and a method is proposed to evaluate their safety. Impact force and stress are considered as metrics of danger, and are used to evaluate the effectiveness of safety systems, and their compliance to safety requirements. Another

contribution to the evaluation of compliance of robots to safety requirements is given in [40]. A systematic method for safety evaluation in physical human-robot interaction is described, analysing the main injury mechanisms. Moreover, safety tests are carried out on several industrial robots. In the following, some relevant contributions to two important approaches to the design of safe robots are reviewed.

1.1.1 Safe actuators

A possible method to design a safe robot is to create a mechanical structure which is intrinsically harmless. In case of an unexpected impact between the robot and a human operator, if the robot structure absorbs the kinetic energy inside elastic elements, the danger can be significantly reduced. Such an approach has the clear advantage of creating a system which keeps its safety properties even when control is not present. For this reason, the use of compliant actuating systems for safety has been deeply investigated.
In [8] an early contribution to the development of intrinsically safe actuators was proposed. Actuators with programmable compliance were proposed, and the control problems generated by such a configuration were analysed. At the end, a robot prototype with controllable compliance was demonstrated. In [9] a complete analysis of the problem of obtaining high performances while employing safe actuation principles was proposed. The tradeoff between safety and performance was quantitatively analysed and some possible solutions were proposed. Moreover, a method for performance evaluation was introduced. A novel actuator concept, aiming at reducing the effective manipulator inertia, and thus the impact forces and the risk related to a possible collision between a human operator and the manipulator, was proposed in [103] and [102]. In [103] a detailed analysis of the actuator structure was given: such an actuator uses two different motors, one with high mass and designed to generate low-frequency torques, which is located at the base of the robot, and another one characterised by low mass and better high-frequency performances, which is located inside the robot arm. In [102] such an actuator was compared with other actuating principles for safe human-robot interaction.
Actuation concepts introduced in [9] were implemented and presented in [94]: a VSA (variable stiffness actuator) was proposed, based on a belt transmission system and on controllable mechanical loadings to variate its stiffness. The actuator and its ad-hoc control system were also experimentally validated. Another concept of a variable stiffness mechanism can be found in [44], where variable stiffness is achieved by means of a special mechanical cam profile. Among alternative actuating principles for safe human-robot interaction, pneumatic actuators have a relevant role, thanks to their very low weight and friction and high compliance. In [23] a review of most common pneumatic actuators can be found. An interesting hybrid electric-pneumatic actuator was proposed in [87], aiming at combining intrinsic safety of pneumatic actuators with performance of electric motors.

1.1.2 Collision detection and reaction

Another relevant method for the design of manipulators that are safe for human-robot interaction is actively controlling the robot impact behaviour, so as to guarantee a safe collision for the human operator, in case one occurs. This methodology focuses on robot control instead of robot design, even though it often requires adequate robot sensing capabilities for the detection of impacts.
First research works on robot collisions, such as [96], [71] and [95], mainly focused on the manipulator alone, instead of on the collision between the robot and a human. In [96] impact dynamics were analysed and classified in order to build a base for robot control strategies capable to cope with such events. In [95], the risk for the robot involved in the collision was considered, and metrics for the evaluation of danger were proposed, in order to assess the robot safety. Moreover, risk dependence on robot configuration was analysed, and most dangerous configurations were identified. In [98] a human-robot interaction system was proposed, where physical contact between the two operators was controlled in order to comply with the human pain tolerance limit. After having set such a limit, a robot concept for safe-human robot interaction was proposed. The robot was covered with soft contact sensing material, with the twofold purpose of reducing collision risk and assessing the severity of the contact. Finally, a control strategy was proposed to keep contact forces below the pain tolerance limit.
Two contributions on the topic of sensing collisions between robot and obstacles, without resorting to dedicated sensors, can be found in [27] and [37]. In [27] a simulation study was conducted on the evaluation of contact forces for a planar manipulator, without a force sensing system. A fault diagnosis scheme was used to identify contact forces through residual signals. The link of the robot where the collision occurred could be identified and force intensity and location could be estimated. Finally, a controller of the contact forces, based on the interaction force estimate, was proposed. In [37] a control system for physical human-robot interaction and collaboration, based on a standard industrial robot, was proposed. No force or torque sensors were used, and contact between the robot and the environment was identified from motor currents and joint positions. Finally, different collaborative contact-based control schemes were proposed. In [25] the concept developed in [27] was extended with strategies for collision reaction. Moreover, collision detection was improved, enabling the evaluation of contact direction. Finally, the collision detection and reaction strategy was experimentally validated on a lightweight robot endowed with torque sensors, which enable the application of the proposed strategy using only proprioceptive sensors. A complete overview of the manipulator adopted in the previously cited work, the DLR LWR, can be found in [2]. Such a manipulator was designed for physical human-robot interaction, favouring reduced robot mass instead of creating a high stiffness structure allowing high absolute accuracy, and hence giving intrinsic safety characteristics to the robot. Moreover, torque sensing was embedded in the joint design, providing the

measurements needed for collision detection and reaction, compliant robot behaviour and force/torque control. In [39] the same robot was used as the testbed of collision detection and reaction strategies, the validity of which was experimentally assessed in collision tests with humans. The possibility of guaranteeing safe collisions even for high operating velocities was demonstrated, and intentional contacts could be distinguished from undesired collisions, thus allowing a human operator to control the robot by touching it. Finally, in [26] a collision reaction strategy for a variable stiffness actuator was proposed. First, a controller for independent position and stiffness profiles tracking was designed. Then, a sensorless collision detection strategy was proposed, and a reaction strategy based on stiffness reduction and evasion from the obstacle was introduced.

1.2 Collision avoidance strategies

A major role in safety strategies is played by collision avoidance: avoiding contact with humans is the most basic method to ensure their safety, as injury risk derives from impact forces and/or dangerous contact configurations such as clamping. Moreover, the presence of human operators in the robot workspace implies a continuous modification of the environment, and consequently calls for strategies to adapt the robot trajectory to the changing conditions, in order to successfully accomplish it. In the following, the main collision avoidance strategies that have been progressively proposed in the literature are reviewed.
A fundamental contribution to the design of real time collision avoidance strategies was proposed in [48], where the artificial potential field approach was applied to manipulators. Virtual repulsive forces applied at the operational space level were adopted to guide a robot between obstacles, and at the joint level to avoid kinematic limitations. Repulsive forces were used at low control level in order to give a high level task plan the capability of adaptation to unforeseen obstacles. In [12] virtual forces were combined with certainty grids for obstacles representation in mobile robots navigation. A mobile robot was equipped with ultrasonic sensors for obstacle detection and certainty grids were used to build a real time map of the environment. Attractive and repulsive forces were used to guide the robot, and different strategies were evaluated to ensure successful reaching of the goal. The same authors of [12] proposed in [13] a different real time obstacle avoidance method based on a two dimensional histogram model of environment occupation. The robot sensing system updated the environment model in real time, which was afterwards transformed into a polar map around the current robot position, and finally used to find an obstacle free motion direction. Aiming at combining real time reaction to obstacles with global path achievement, in [81] the approach of elastic bands was proposed. In such an approach the global path is treated as an elastic which can be deformed by obstacles. By doing so, the global path is preserved, and adaptation to the presence of obstacles is possible.
Kinematically redundant robotic arms constitute particularly fit platforms for collision avoidance, as the extra degrees of freedom can be used to combine task execution with

evasion from obstacles. An example of a collision avoidance strategy for a kinematically redundant 7 dof arm was proposed in [38]. Obstacle avoidance was formulated as a set of inequality constraints, and manipulator redundancy was exploited to avoid obstacles while performing the task. In [85] a collision avoidance strategy for standard and kinematically redundant position controlled robot arms was proposed. The strategy was based on the definition of a protective area around obstacles, and on the computation of virtual repulsive forces corresponding to the intrusion of the robot in such an area. The collision avoidance system modifies the robot position in order to nullify such forces, acting on the robot end effector, if a standard six degrees of freedom manipulator is controlled, or on the closest point to the obstacle between the end effector and the robot elbow, if a redundant manipulator is used. Also arm self motions can be exploited for evasion. In [22] a metric for obstacle avoidance was introduced, which evaluates the risk of collision taking into account both distance between the robot and the obstacles and the direction of motion of robot links. Such a metric is then minimised using null space motions.
Considering also humanoids and mobile robots, in [14] a framework for the enforcement of collision avoidance and posture constraints, consistently with task ones, was proposed. Such a framework exploits the null space of the task to execute additional constraints, and in case a conflict between them occurs, suspends task execution in order to fulfil the other ones. The use of online trajectory generation algorithms for the design of reactive control strategies such as collision avoidance has recently gained significant attention. In [36] an obstacle avoidance strategy based on measurements of a depth sensor, which generates repulsive forces from robot-obstacles distances, and computes evasive motions using the online trajectory generation algorithms presented in [51] and [52], was proposed. The same algorithms were exploited for collision avoidance in [84], where a capacitive distance sensor was mounted on a robot surface, and evasive motions were generated from the sensor measurements.

1.2.1 Self collision avoidance

When facing the problem of avoiding unpredicted obstacles, and in general when the robot motion is not planned and verified offline, but has to be planned online, possibly reacting to unforeseen events, one of the issues that have to be taken into account is the possibility of robot self collisions. Such a problem is obviously more pressing when multi arm or complex robotic structures are considered, as the possibilities of self collisions increase. Different approaches to the problem have been proposed in the literature. In [53] an algorithm for online trajectory checking against self collisions, specially designed for humanoids robots, was presented. The strategy adopts an efficient method for the computation of distances between the robot limbs, and discards those trajectories which lead to a collision. Another possible approach is to modify a task currently being executed, in order to avoid the collision between different parts of the robot. In [28], an example of the

application of such an approach to a dual arm robot was given. A skeleton model of the robot structure composed of multiple segments was adopted, and an algorithm to identify the pair of closest points of the structure segments was used. Repulsive forces to avoid the collision between such points were computed, and forces were then transformed into torques using the collision points Jacobians transpose. Finally, such torques were added to the task ones, thus enforcing the self collision avoidance behaviour. In [89] a control strategy for collision and self collision avoidance was proposed, which embeds such objectives in a hierarchy of tasks approach. Collision and self collision avoidance were pursued by means of a cost function to be optimised, maintaining compliance with task constraints. In [30] an evolution of the strategy presented in [31] for the integration of self collision avoidance in a task hierarchy was proposed. The strategy, which was applied to a complex humanoid robot, uses the computation of distances between the robot limbs to derive repulsive forces to avoid collisions. In order to adequately dissipate the energy introduced by such forces, a damping factor which takes into account the robot configuration was adopted. Finally, continuity in the control actions, when transition between self collision avoidance activation and deactivation is performed, was guaranteed.

1.3 Danger, safety and comfort assessments

An important element in the control of human-robot interaction is the formulation of danger assessments to encompass all of the aspects of human safety, both physical and psychological, and the design of figures of merit for the maximisation of robots acceptability for human operators. Such assessments are needed to design effective safety control strategies, and to make human-robot cooperation fully acceptable for human operators. In [55] two different danger assessment criteria, which consider robot inertia and distance from obstacles, were adopted for safe path planning. Using such assessments, a cost function including three terms, related to danger, obstacle avoidance and goal reaching was formulated, and the path optimising such function was chosen. In [56] the same authors used one of the previously introduced danger indices to design a real time collision avoidance strategy. As in many other collision avoidance approaches, such index was used to compute repulsive forces to move the robot away from the obstacle. Finally, in [57], the same authors introduced the element of psychological comfort in human-robot interaction. Physiological signals were used to develop a model of human emotional state, which could then be used to control a robot action, in order to ensure the psychological wellness of the human operator engaged in the interaction. Contributions given in [55], [56] and [57] were integrated in a single system for the control of human-robot interaction in [58].
A recent contribution in the field of danger assessment is the *danger field* metric, originally proposed in [61] and further developed in [62] and [63]. Such an assessment, adopted in this thesis, considers both robot velocity and distance from the obstacle in the computation of danger, and, moreover, is able to take into account the whole robot structure as a

source of danger. Other metrics instead consider only some points of the robot in danger assessment, and are hence less effective in capturing the robot motion. The availability of kinematically redundant robot arms has provided further tools for the development of control strategies to increase robots acceptability. In [100], a study has been proposed on the exploitation of the kinematic redundancy of a robot arm to achieve human-like motions. Experiments have been carried out to identify a correlation between the human arm swivel angle and the hand position and orientation. Then, in order to perform human-like motions on robotic arms, a closed form function describing such a correlation has been formulated and adopted as a redundancy resolution criterion. The acceptability of the robot motions obtained with the proposed redundancy resolution criterion has been validated in [99], where several physiological signals were used to asses the level of emotional stress induced by a dual arm robot motion on a human operator working on its side. A lower emotional arousal was obtained when a human-like redundancy resolution criterion was adopted, compared to the one caused by a standard resolution.
Knowledge of injury dynamics can be exploited in robot design and embedded in robot control in order to guarantee safe collaboration with humans. In [77] an injury criterion specifically designed for service robots, and an impact model to analyse such event, were proposed. Impact simulations were performed, and suggestions for the design of safe robots were drawn. In [80] a study of the impact between a small industrial robot and a human operator was conducted. A passive mechanical arm was adopted as the test bench of collisions, as a replacement of real human arms. A comparison was then carried out between collisions with human volunteers and the ones with the model arm, in order to validate the model. Injury dynamics comprise many different aspects and injury types, so specific studies are needed to find adequate safety strategies for every type of possible injury. In [76] an analysis of the correlation between robot coverings features and skin injuries was conducted. Using experiments of collision with real human bodies, a skin injury criterion was proposed, and robot design principles to reduce the possibility of skin injuries were defined. In [41], a motion supervisor for the generation of safe motion profiles was proposed. Such a system adopts a model of the correlation between robot inertia, velocity and impact zone geometry and the injury type and severity to set a velocity limit for robot motion, which guarantees an upper limit to the severity of possible injuries.

1.4 Sensor systems for human-robot interaction

The deployment of robots in unstructured scenarios and in close interaction with humans calls for the adoption of sensing systems to provide the robot with information about the environment state. Peculiar features are required, such as the capability to avoid self occlusions, a complete coverage of the robot workspace and, to make the human-robot interaction configuration feasible, a reduced cost. Many different sensing principles have been investigated in the literature for this purpose. In [11], a mobile service robot was

equipped with a set of ultrasonic range finders to be used for obstacle avoidance and navigation. A thorough analysis of the sensors characteristics and limitations was performed, highlighting in particular the problems due to the detected object surface orientation and structure, and to the amplitude of the emission cone. In [49] infrared distance sensors were considered for a mobile robot navigation. A comparative analysis of different sensing solutions, based on their cost and performance, was done, and a sensor system was proposed. In [72] a sensing system for collision avoidance, based on multiple cameras installed on the manipulator surface was proposed. A real time collision avoidance algorithm was designed based on such a sensor system, which is particularly fit for fast collision avoidance. In [47] a fusion of measurements coming from different types of sensors was used to design a system for safe human-robot interaction. Ultrasound sensors, capacitive sensors and light curtains were used to determine the position of a human with respect to an industrial robot, and to limit its maximum speed accordingly in order to control the interaction risk level. Another example of sensor fusion for the control of safety can be found in [65]. Passive infrared sensors, cameras and microwave sensors were used to identify the human operator position in a work cell, which was used as the input for a safety system.
A sensor concept which is particularly fit for human-robot interaction, and can be implemented adopting any sensing technology, as shown in [68], is the distributed distance sensor. In such a concept, the sensor is installed on the robot surface, and virtually completely covers it. The main advantages of such a sensor are the complete prevention of sensor occlusions and the possibility of its adoption without any environment structuring. The first introduction and adoption of such a sensor on an industrial robot was presented in [19]. The sensor was then developed in [20] and applied to motion planning, teleoperation and real time collision avoidance in [21], [66] and [67] respectively, by the same authors. Infrared proximity sensors were adopted for the first implementation, but other implementations based for example on capacitance sensors have been proposed, for example in [75] and more recently in [84]. The distributed distance sensor concept is adopted in this thesis to create a sensing system for a dual arm robot prototype. The author has already contributed to the development of such a system in [18] and [15], tackling the problem of sensor sizing and placement.

1.5 Constraints management

When dealing with multiple constraints of different nature, such as task completion, safe operation and obstacle avoidance, an approach to transparently and systematically manage them is needed, in order to efficiently exploit all of the robot degrees of freedom. Such a problem has been tackled in many aspects of robot control, from offline and online path planning to real time kinematic control. Considering path planning, many efforts have been done on formalising and efficiently dealing with the constraints characterising the task to be planned. An interesting contribution to the problem of path planning in

the joint space, under task space constraints, was proposed in [91]. The completion of a task requires the satisfaction of operational space constraints, but at the same time robot kinematic limitations have to be taken into account. Moreover, possible task redundancies and robot self motions may be exploited to satisfy additional constraints, such as collision avoidance. In [91] a joint space planner able to satisfy task space constraints was proposed, and representations for common operational space constraints were introduced. The work presented in [91] was extended in [90] with a theoretical analysis of the path planner properties. In [59] a path planner capable of managing constraints tolerances was proposed: the concept of "soft constraints" was introduced to describe intervals of feasible values, with a favoured exact one, for robot operational space coordinates. A planner capable of choosing the coordinate value closest to the favoured one was proposed, and adequate constraints representations for the automatic execution of such a choice were introduced. In this thesis, "soft constraints" are the ones applied to coordinates corresponding to task redundancies. Such constraints can therefore be relaxed without influencing task execution. The problem of constraints management in path planning is particularly relevant when redundant robots are considered, since the extra degrees of freedom can be used to perform additional tasks. In [101] an algorithm for path planning specifically designed for redundant robots was proposed: such an algorithm explores the task space for feasible paths that comply with the task constraints, and then searches the robot configuration space only for paths tracking the task space ones. By doing so, infeasible regions of the configuration space are avoided, decreasing the computational burden and improving the path quality.
A fundamental contribution to the problem of multiple constraints management is given by the task priority framework. Such an approach has been widely adopted as a tool to exploit the degrees of freedom left available by each constraint, in order to execute the ones having lower priority. First proposed in [42], this framework was further developed by the same authors in [74]. The core concept of the approach is to define different levels of priority for the constraints to be enforced in robot control, and then to enforce each constraint in the null space of the one with higher priority. In [88] the original approach was modified, solving the robot inverse kinematics for joint velocities instead of joint accelerations, with the purpose of avoiding possible unstable behaviours caused by the original formulation. The task priority framework is well suited for the control of kinematically redundant robots, since it offers a systematic method to take advantage of their redundancy, in order to enforce a stack of constraints, with different relevance levels. An example of the application of the task priority framework to a kinematically redundant robot can be found in [4]. An underwater vehicle equipped with a planar manipulator was considered, and two case studies with different top priority tasks were simulated. Energy saving and then path tracking were considered as primary tasks, while keeping robot position constant was considered as the secondary goal. Another example of the application of task prioritisation to redundant robots can be found in [7], where such an approach was applied to the control of a humanoid robot. An improvement in the computational efficiency of the

method, which is of particular relevance when considering highly redundant structures, was introduced, and the management of multiple constraints was validated in simulation on a human like figure.
A relevant problem in the adoption of the task priority framework is guaranteeing the absence of conflicts between tasks with different priority levels, and, moreover, ensuring the stability of the adopted algorithms. A discussion on the stability problem of some relevant task priority algorithms was presented in [3]. Conditions for their asymptotic stability were formulated, thus allowing to verify the possibility of simultaneously executing the different tasks. The task priority framework was recently extended to consider unilateral constraints in [46], and in [32] a more efficient numerical solution of the same problem, which also overcomes some drawbacks of the previous methodology, was proposed. In this thesis, a priority is defined for the constraints constituting a preplanned task, but no conflict is present between them, so projection of lower priority constraints in the null space of higher level ones is not necessary. A preplanned task is decomposed into subtasks, which are enforced based on an assessment of danger generated by the robot. Such a strategy has analogies to what proposed in [69], but in this research a general criterion for the decomposition of a task into subtasks is proposed, and, moreover, a particular task type is considered, that is a preplanned trajectory which can be only locally modified or suspended.

1.6 Commercially available human like dual arm robots

A relevant element for the development of human-robot interaction applications is the increasing availability of dual arm, human like, kinematically redundant industrial robots. Fig. 1.1 shows some of the latest systems. Such robots are fostering the interest for human-robot interaction solutions and are improving the economical feasibility of such configurations.

(a) ABB FRIDA dual arm robot prototype.

(b) Nachi MZ07 dual arm robot.

(c) EPSON autonomous dual arm robot.

(d) Kawada Nextage robot.

(e) Rethink Robotics Baxter robot.

(f) Motoman SDA20 robot.

Figure 1.1: *Some of the latest commercially available dual arm, human like and kinematically redundant industrial robots.*

CHAPTER 2

Instantaneous constraints classification based on their relevance for task execution

2.1 Introduction

Industrial robot controllers available today adopt imperative languages for programming robot tasks. Robot programmers can use libraries of motion primitives to compose the desired task, by choosing those best fitting their application. Motion primitives usually define the profile of the trajectory to be followed by the robot, e.g. linear or circular. By setting the primitives parameters, some trajectory properties, such as velocity or acceleration limits and final or intermediate poses and velocities, can be chosen. Once the task has been specified, the motion planner generates a trajectory for the robot TCP and for possible kinematic redundancy coordinates, or equivalently for the robot joints. Such a procedure, conceptually represented in Fig. 2.1, always leads to constraining all the robot degrees of freedom along the complete task trajectory.

However, the task to be executed by the robot may not require to constrain the whole set of robot degrees of freedom during its complete duration: robot kinematic redundancy or task intrinsic redundancy may allow an infinite number of possible solutions for task execution, and this could be exploited to take into account other, unpredictable at programming stage, constraints. For example, in a pure positioning task of the TCP, the orientation might be

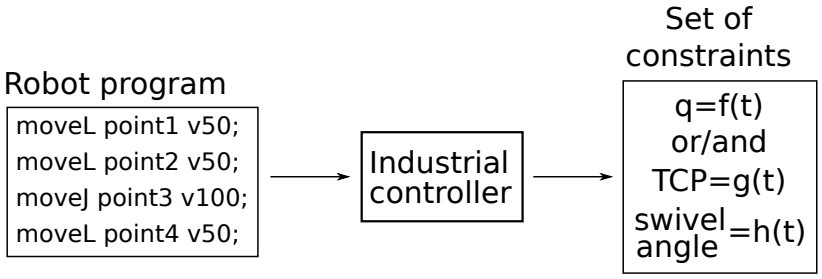


Figure 2.1: *The steps leading from the robot program to the set of constraints are schematically represented. On the left, an excerpt of an ABB* RAPID *robot programming code is shown. The industrial controller processes the code and generates a set of constraints for the robot joints or equivalently for the robot end effector and possible coordinates expressing degrees of kinematic redundancy.*

exploited to satisfy other constraints.
Unnecessary constraints are therefore imposed by the current implementation of industrial controllers when the task trajectory is generated. Since all degrees of freedom are constrained, and industrial robot controllers normally do not allow replanning of the task at execution time to take into account possible modifications of the environment, adaptation of the preplanned task to unforeseen events is completely prevented. If adaptation capability is indispensable, e.g. when a robot operates in an unstructured environment, industrial controllers fall short in achieving this goal. Nonetheless, the amount of functionalities already offered by commercial industrial controllers make an extension of such functionalities to cope with unforeseen events more convenient than a complete controller redesign. Such an extension is thus an interesting option.
Tools are available for the modification of a preplanned trajectory at execution time, such as [10] and [54]. Such tools can be used to adapt a preplanned task to an unforeseen event, without the need for a complete replanning. The adaptation of the task can be based on the relaxation of constraints constituting the preplanned trajectory, to take into account new ones arising at execution time. As already mentioned, some of the constraints enforced by the industrial controller may be unnecessary for task completion. Moreover, other constraints constituting the task may allow a temporary relaxation, without compromising the final completion. In this chapter, a classification of the instantaneous velocity constraints that form a task is proposed, to serve as the basis of a strategy for adaptation to unforeseen events.

2.1.1 Main approaches to task constraints representation

Representation of constraints constituting a robot task has been investigated in various fields of robotic research. A fundamental contribution to the modelling of restrictions im-

posed by the environment, for the field of hybrid control, was given in [70], where the class of *natural constraints*, that is restrictions to the force and positional degrees of freedom of a robot, was introduced. The classification introduced in [70] supports the definition of the appropriate control strategy for each direction in the operational space, that is the definition of the set of *artificial constraints*.
In the field of path planning in unstructured environments, constraints modelling plays an important role in reducing the computational burden of planning. The term *artificial constraint* was adopted in [91] to define the limitations to possible reallocations of obstacles in a movable obstacles scenario, due to the movement of a robot in the workspace. Such constraints were used to reduce the number of possible paths to be explored, in order to speed up path planning. In [59], *soft constraints* in the task space, that is constraints with intervals of feasible values around a desired one, were introduced in a path planning problem.
Another approach to modelling of constraints composing a task is the decomposition of the task into elementary operations. Such an approach allows to specify the robot task by means of elementary actions, instead of programming the robot explicitly. The elementary actions, which represent the constraints to be enforced, are converted by a control system into motions, increasing the robot capability for adaptation. In [73] a method for the decomposition of assembly operations into *skill primitives*, that is elementary movements and actions, was proposed. The proper primitive is then selected by the control system, based on the state of the environment and of the assembly task, adapting the assembly procedure to unpredicted events occurring during execution. In [35], a complete system for robot control based on *manipulation primitives*, was proposed.
The classification proposed in the following aims at grouping the constraints composing a preplanned task, based on their relevance for task execution. Such a classification criterion defines for each constraint the implications of its relaxation, and consequently the possibilities for its modification. The possible actions to adapt a preplanned trajectory to unforeseen events are thus defined.

2.2 Classification of constraints imposed by the environment

Restrictions to the motion of bodies in the space have been extensively discussed in the field of rational mechanics. When multiple rigid bodies are connected, such that each one moves with respect to another one, a mechanism is obtained. Depending on the type of connection between the rigid bodies, and thus the motion restriction applied, different types of mechanisms are created. A complete analysis of mechanisms, and of the restrictions they impose to movement, can be found in [43]. There, the description of mechanisms is based on the classification introduced in [83]. Such a classification establishes the class of *lower pairs*, that is, elementary mechanisms which impose elementary restrictions to the motion of bodies. As this classification is the basis of the following discussion, it is

here briefly summarised. Six elementary pairs are introduced:

1. **revolute pairs**, which allow only a rotation between the two bodies;
2. **prismatic pairs**, which allow a single relative translation between the connected bodies;
3. **cylindric pairs**, which merge the two previous pairs, allowing the rotation and the translation with respect to an axis;
4. **screw connections**, where the translation along an axis and the rotation around the same axis are related by a constant (the screw lead);
5. **planar connections**, where one of the two bodies is allowed to slide on a plane, which is rigidly linked with the other body. Two translations and one rotation are therefore allowed;
6. **spherical pairs**, which allow only rotations between the two connected bodies, thus giving three degrees of freedom.

Simple forms of construction for such pairs are shown in Fig. 2.2.

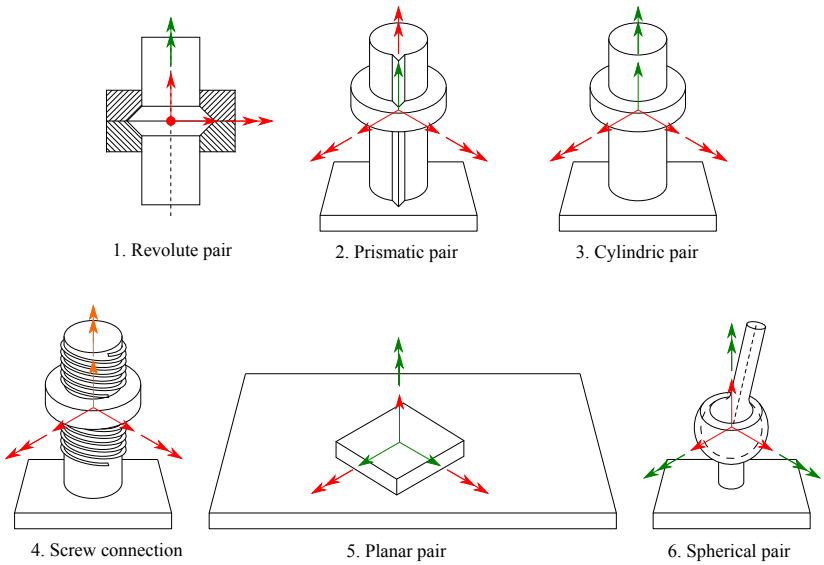


Figure 2.2: *Examples of construction of the lower pairs. Red arrows correspond to restricted motions, while green ones correspond to available degrees of freedom. Orange arrows are used for mutually constrained degrees of freedom.*

When interacting with the environment, a robotic manipulator or its tool are usually subject to motion restrictions. Such restrictions can be modelled using the introduced lower

pairs, or a combination of them for more complex environment configurations. The kinematic pairs models are a useful tool for programming a robot task, as they support the identification of motion restrictions imposed by the environment.
The natural constraints class introduced in [70] completes the modelling of constraints imposed by the environment, identifying the restrictions on both positional and force degrees of freedom. Positional degrees of freedom correspond to the motions allowed by the kinematic pair, while force degrees of freedom correspond to restricted motion directions, thus obtaining two complementary sets.
When a position controlled task is programmed, restrictions to robot movements imposed by the environment are normally taken into account avoiding motions which could lead to the generation of reaction forces. Some of the constraints composing a robot program thus directly correspond to motion restrictions imposed by the environment (e.g. the task would not include motions along restricted directions). Other constraints instead depend on the specific operation that has to be performed.
The classifications reviewed above identify constraints imposed by the environment and derive the available degrees of freedom. A different classification can be made considering constraints imposed by the current task execution and consequently identifying the types of restrictions it enforces on the directions of motion.

2.3 Instantaneous constraints classification based on constraints relevance for task execution

Constraints constituting a task can be classified based on their relevance for the completion of the task itself. Relevance can be assessed considering the consequences of the relaxation of a constraint, and can be used to define which constraint to relax in case new ones have to be taken into account. The classification proposed in the following achieves three main objectives:

1. it defines a scale of priority for the constraints constituting a task, that 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.

The classification introduces four types of constraints:

- *Hard* constraints
- *Skill* constraints
- *Soft* constraints
- *Null* constraints

2.3.1 Hard constraints

Hard constraints are defined as the ones whose relaxation would cause an incorrect execution of the task. Mason's natural/artificial constraints classification [70] can support the identification of such a set of constraints. A first part of the *hard* constraints set is composed by the ones classified as natural velocity constraints in [70], i.e. restrictions of the robot TCP movement imposed by the environment. The portion of the environment applying a restriction to the robot TCP motion can be modelled as a kinematic pair, and restricted directions can be easily identified as exemplified in Fig. 2.2. The second part of the set consists in restrictions of the robot TCP motion imposed by the task. The specific operation being executed by the robot may in fact require the restriction of some velocities to properly complete the operation or to avoid damages to the involved components, even if the environment does not impose such restrictions. Let us consider for example a manipulator holding a liquid container with its gripper, as shown in Fig. 2.3. To avoid spilling the liquid, the container should be kept in the upright position. Rotations around the coordinate axes perpendicular to the vertical one should therefore be restricted.

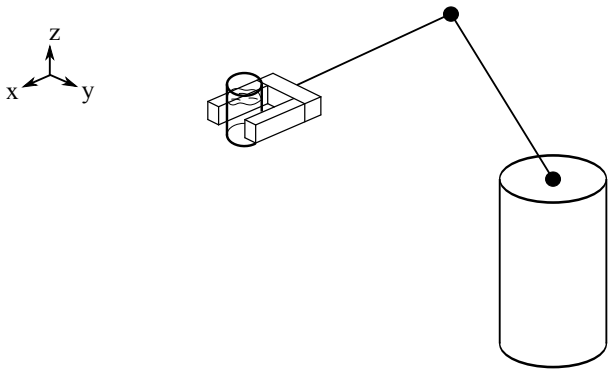


Figure 2.3: *Example task: a robot holding a liquid container with its gripper.*

In order to support a systematic classification, constraints can still be identified by modelling the restrictions imposed by the task with a kinematic pair. The constraints imposed e.g. by the task represented in Fig. 2.4 correspond to a planar pair, with a vertical translational degree of freedom added.

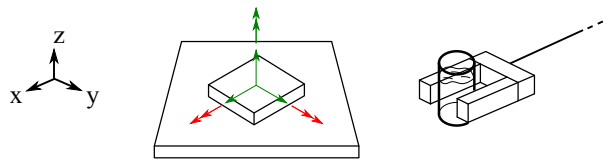


Figure 2.4: *The lower pair corresponding to the constraints imposed by the task. The vertical translational degree of freedom is added to a planar pair.*

Finally, restrictions to TCP movements imposed by *hard* constraints include not only constraints imposing null velocity along a direction, but also trajectories imposed to the TCP coordinates. Let us consider a robot performing a glueing task, as depicted in Fig. 2.5: the TCP has to follow the glueing profile and keep the glue dispenser perpendicular to the object tangent plane. A deviation from the glueing path or a modification of the correct tool orientation would cause an erroneous glue deposition and thus the disruption of the task. The three constraints corresponding to TCP position and the two constraints corresponding to the orientation with respect to the object perpendicular axis are therefore of *hard* type.

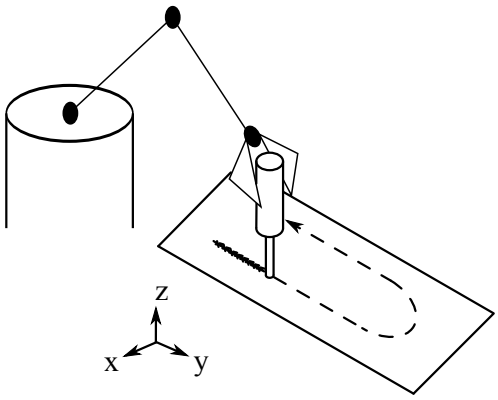


Figure 2.5: *Example task: a robot performing a glueing operation.*

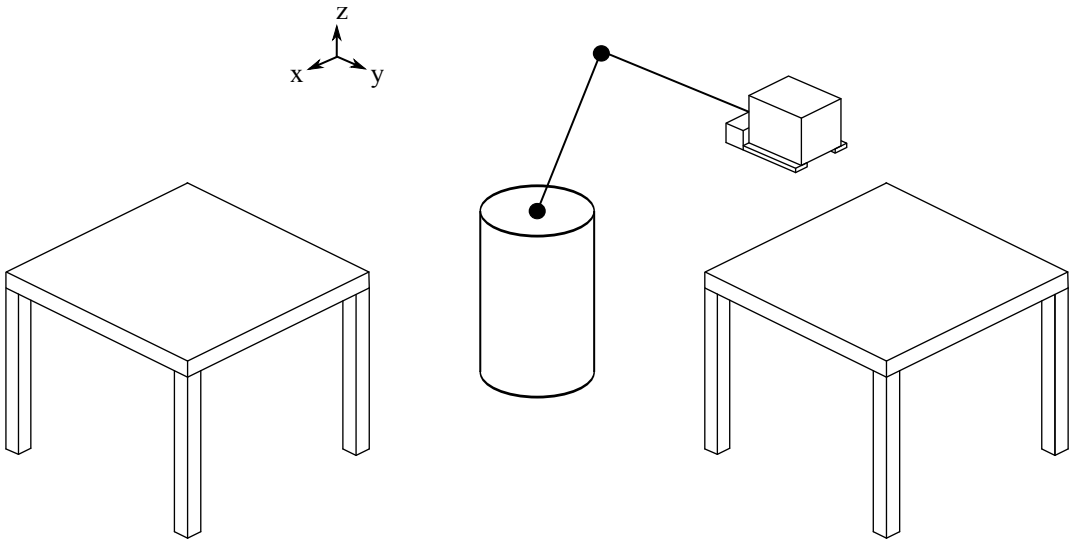


Figure 2.6: *Example task: a palletising operation.*

2.3.2 Skill constraints

The previously introduced set of *hard* constraints includes natural velocity constraints and part of the natural force constraints [70]. Among the remaining directions of the operational space, two more classes of constraints can be identified. The first one, defined in this section, is the class of *skill* constraints. The constraints belonging to this class are the ones whose relaxation is possible, but implies suspension of the skill being performed.
Let us consider for example the palletising task depicted in Fig. 2.6, where an object has to be displaced from one support to another. Constraints on robot TCP position determine the displacement of the object. Relaxation of such constraints therefore causes the suspension of task execution. However, if constraints are enforced once again after their relaxation, the task can still be completed. Such constraints are therefore assigned the *skill* type. A second example can be drawn from an assembly operation during which a robot has to mount a screw, see Fig. 2.7.

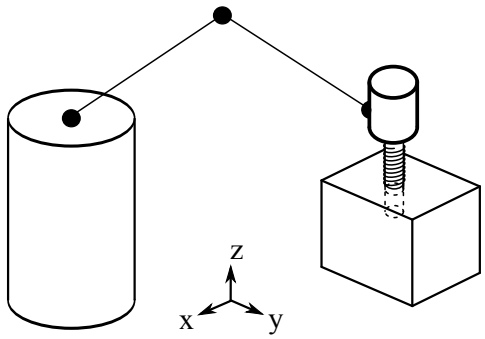


Figure 2.7: *Example task: a robot performing a screwing operation with a dedicated tool.*

The screw assembly can be suspended and resumed without any consequence on the final task completion. Translation along, and rotation around the screw axis are therefore *skill* constraints.

2.3.3 Soft constraints

Constraints applied to the remaining operational space coordinates are the ones whose suspension has no effect on task execution, and thus correspond to task redundancies. Constraints on such coordinates are defined as *soft* ones. Let us consider the peg in hole task depicted in Fig. 2.8: orientation of the peg around the hole axis is irrelevant for the completion of the task, and can be therefore classified as a *soft* constraint.

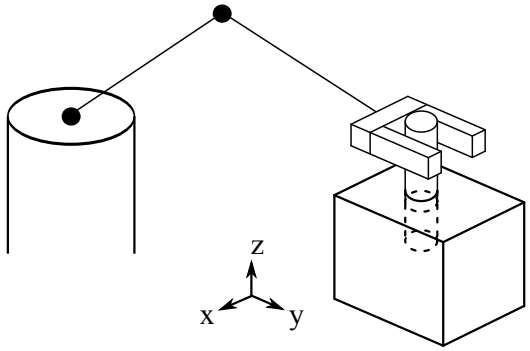


Figure 2.8: *Example task: the classical peg in hole insertion operation.*

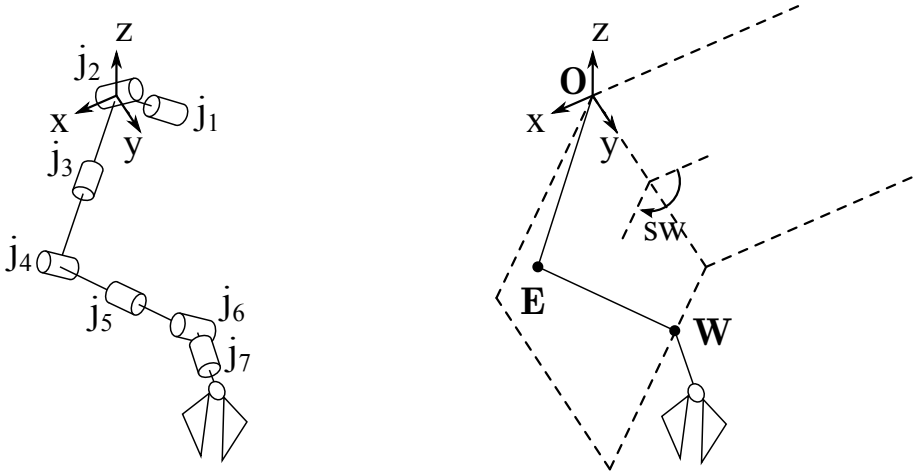


Figure 2.9: *On the left, a 7 degrees of freedom manipulator, and on the right the geometric model for the definition of the swivel angle sw.*

2.3.4 Null constraints

The last class of constraints includes the ones applied to coordinates representing possible kinematic redundancies of the manipulator. Such coordinates lie in the null space of the robot end effector pose, and their variation has by definition no effect on the TCP pose, and thus on task execution. Considering for example a 7 degrees of freedom manipulator, having therefore one degree of kinematic redundancy, a convenient coordinate for the representation of such a redundancy is the swivel angle sw, introduced in [50]. Considering Fig. 2.9, the swivel angle can be defined as follows. Let us define the vectors $\mathbf{e} = (\mathbf{E} - \mathbf{O})$ and $\mathbf{w} = (\mathbf{W} - \mathbf{O})/\|\mathbf{W} - \mathbf{O}\|$. We introduce the vector $\mathbf{p} = (\mathbf{I} - \mathbf{w}\mathbf{w}^T)\mathbf{e}$, and we define the swivel angle as:

$$sw = atan2(\mathbf{w}(\mathbf{z}^T \times \mathbf{p}), -\mathbf{z}^T\mathbf{p}) + \pi/2 \tag{2.1}$$

Considering the arm depicted in Fig. 2.9, if the robot wrist lies on the y axis, and the robot elbow lies on the x-y plane, a swivel angle of 0 is obtained if the elbow lies in the negative x semi axis, while a swivel angle of π is obtained if the elbow lies in the positive x semi axis. The variation of the swivel angle has no effect on the TCP pose and thus on task execution. The constraints applied to such a coordinate is therefore a *null* one.

Table 2.1 summarises the classification of the constraints.

Class	Relaxable	Relaxation Consequences	Coordinate space	Constraint imposed by the environment
Hard	No	-	Operational Space	Natural Velocity Constraints
Skill	Yes	Task suspension		Natural Force Constraints
Soft	Yes	None		
Null	Yes	None	Null Space	-

Table 2.1: *A summary of the presented classification. For each type of constraint, relaxation possibilities and consequences are defined. The space corresponding to the constrained coordinate is shown and the corresponding type of constraint imposed by the environment is identified.*

2.4 Decomposition of a task into skills

During the execution of a task, as different operations are performed, constraints applied to the robot TCP coordinates are characterised by different relevance. In order to assess the possibilities for task modification during each of its phases, it is convenient to divide it according to the variation of constraints relevance: the task will therefore be decomposed into sections, each characterised by constraints belonging to different classes.
In order to operate such decomposition, the following two steps have to be performed.

2.4.1 Identification of constraints type

In order to decompose a task into parts according to constraints relevance, the first step is to identify which class they belong to. The identification can be performed according to

the following steps:

1. Restrictions to robot motion directions imposed by the environment have to be identified. The introduced lower pairs support such an operation, offering a library of elements to compare the environment configuration with. Each lower pair allows one or more degrees of freedom: the remaining translations or rotations, that are restricted by the pair, correspond to *hard* constraints.
2. In addition to restrictions imposed by the environment, the ones imposed by the task have to be identified. Constraints that cannot be relaxed in order to successfully complete the task must therefore be determined. Lower pairs can once again support the identification of such restrictions, which belong to the *hard* constraints class too. Identifying the lower pair that models the motion restriction imposed by the task, *hard* constraints can be easily mapped.
3. Among the possible motion directions, some are applied constraints that can be temporarily relaxed without compromising the completion of the task. Such constraints belong to the *skill* class.
4. The remaining motion directions represent task redundancies, and the constraints applied to them belong to the *soft* class.

In order to formalise constraints classification, the selection vectors $\mathbf{s}_{hard}$, $\mathbf{s}_{skill}$, $\mathbf{s}_{soft}$ and $\mathbf{s}_{null}$ are introduced. Such vectors define which type of constraint is applied to the coordinates describing the robot pose. Let $\mathbf{v} = [v_x, v_y, v_z, \omega_x, \omega_y, \omega_z, \dot{sw}]$ be a possible vector of robot velocities, composed by the TCP ones and the swivel angle time derivative, we define

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

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

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

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

Each coordinate has to be assigned to a class, such that the selection vectors sum up to a vector with all unitary elements.

2.4.2 Decomposition of the task

Once the constraints have been classified, the task can be divided into parts, named *skills*. Skills are the elementary operations composing the robot task. A new skill is defined at every point where relevance of a constraint changes.

2.5 Application example

In the following, a complete example of the application of the constraints classification and of the division of a task in skills is presented. Let us consider a palletising task, during which a manipulator has to lift an object using a pair of lifting forks, and then move it to another support. At first, the robot TCP moves from the initial pose to the forks insertion pose, then, the forks are inserted under the load and such load is displaced from the initial pose to the final configuration. Finally, the forks are de-inserted from the object and the task final pose is reached. The task is depicted in Fig. 2.10.

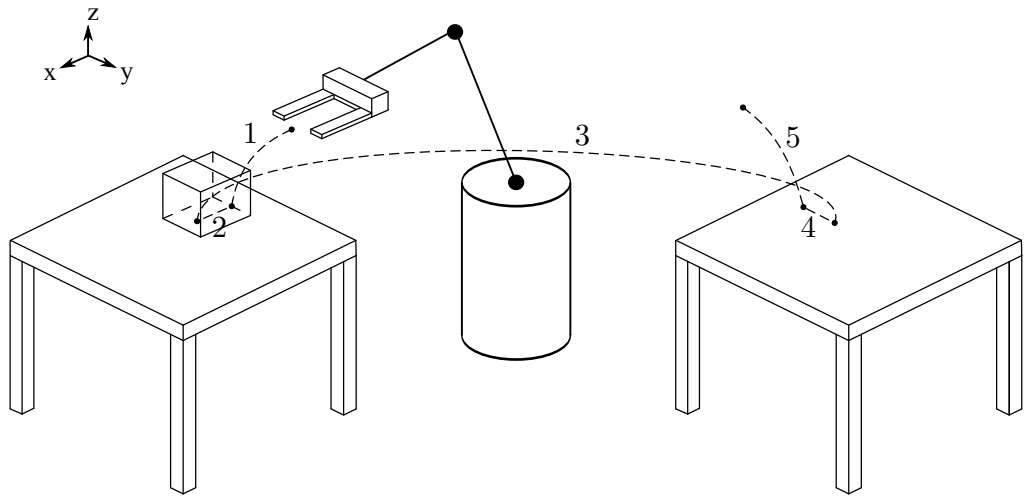
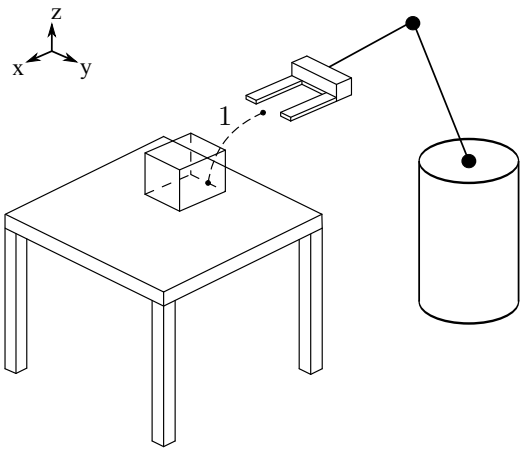


Figure 2.10: *A palletising task. The robot approaches the table, lifts the object with the lifting forks and then moves it to another table. Then, the object is left on the second table and the robot moves to the home position.*

2.5.1 Classification of constraints

The constraints composing the palletising task can now be classified according to the proposed scheme. During the approach the environment does not apply any limitation on robot velocities. Likewise, the task does not require any restriction to the TCP velocities to guarantee its completion. Such operation, shown in Fig. 2.11, has therefore no *hard*

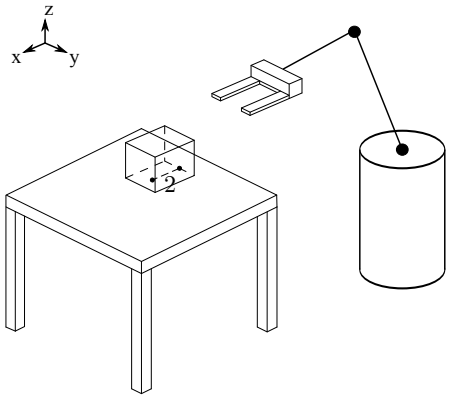
constraints. Possible constraints whose relaxation implies the suspension of the task have now to be identified. The relaxation of constraints on robot TCP position implies the suspension of the approaching motion, which can be completed anyway after the suspension. Such constraints therefore belong to the *skill* class. The remaining constraints on the TCP pose, that is constraints on orientation, are *soft*, as their relaxation has no effect on the successful completion of the task.



	constraint type
v_x	skill
v_y	skill
v_z	skill
ω_x	soft
ω_y	soft
ω_z	soft

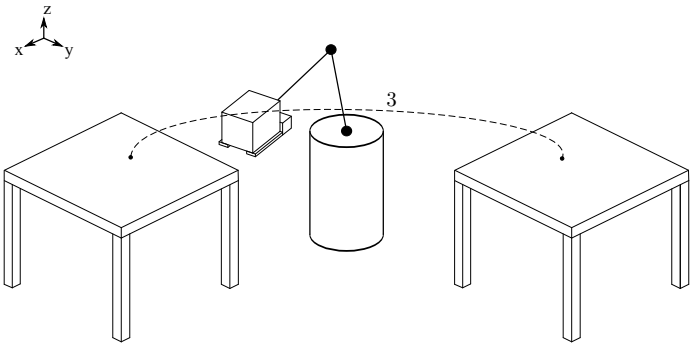
Figure 2.11: *The robot approaching motion.*

During the forks insertion, shown in Fig. 2.12, the environment restrictions on robot velocities correspond to those of a prismatic lower pair: when the forks are inserted under the object, any rotation of the robot TCP and any translation perpendicular to the insertion axis would cause reaction forces between the robot tool and the environment to rise. Such constraints therefore belong to the *hard* class. The remaining constraint on translation along the insertion axis can be easily identified as a *skill* constraint: in fact, its relaxation implies the suspension of the insertion operation, which could then be resumed and completed.
The displacement of the load between the two tables, depicted in Fig. 2.13, is a free space movement. However, the task being performed imposes some restrictions to the robot TCP velocities. In fact, while the absence of contact with the environment implies no related limitations of the robot movements, the presence of the object on the lifting forks restricts feasible end effector orientations. To prevent the object from falling, the plane of contact with the tool must be kept parallel to the horizontal plane. Rotations around the x and y axes are thus *hard* constraints, as their relaxation would cause the disruption of the skill. Similarly to the approaching phase, constraints on TCP position belong to the *skill* class, as their relaxation implies the suspension of the operation. The remaining constraint on orientation around the vertical axis is a *soft* constraint, as its relaxation has no effect on task completion.



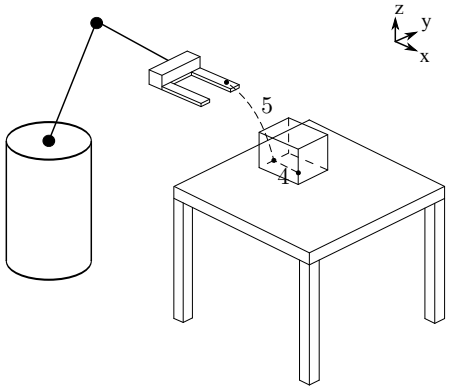
	constraint type
v_x	skill
v_y	hard
v_z	hard
ω_x	hard
ω_y	hard
ω_z	hard

Figure 2.12: *The operation of insertion of the lifting forks.*



	constraint type
v_x	skill
v_y	skill
v_z	skill
ω_x	hard
ω_y	hard
ω_z	soft

Figure 2.13: *The displacement of the object between the two tables.*



4	constraint type
v_x	skill
v_y	hard
v_z	hard
ω_x	hard
ω_y	hard
ω_z	hard

5	constraint type
v_x	skill
v_y	skill
v_z	skill
ω_x	soft
ω_y	soft
ω_z	soft

Figure 2.14: *The de-insertion of the lifting forks and the final movement to the home position.*

The de-insertion of the lifting forks and the movement to the final pose, represented in Fig. 2.14 are characterised by the same constraints as the insertion and the initial motion, respectively.

2.5.2 Decomposition of the task into skills

As previously mentioned, once constraints are classified, the task performed by the robot can be decomposed into *skills*. A new skill is defined at each point of the task where relevance of a constraint changes, so, referring to the considered example, five skills are present. The task can therefore be defined as:

$$\mathbf{task} = \{\mathbf{skill}_1\ ,\ \mathbf{skill}_2\ ,\ \mathbf{skill}_3\ ,\ \mathbf{skill}_4\ ,\ \mathbf{skill}_5\}$$

$\mathbf{skill}_1$	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{soft}$	0	0	0	1	1	1
$\mathbf{s}_{skill}$	1	1	1	0	0	0
$\mathbf{s}_{hard}$	0	0	0	0	0	0

$\mathbf{skill}_2$	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{soft}$	0	0	0	0	0	0
$\mathbf{s}_{skill}$	1	0	0	0	0	0
$\mathbf{s}_{hard}$	0	1	1	1	1	1

$\mathbf{skill}_3$	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{soft}$	0	0	0	0	0	1
$\mathbf{s}_{skill}$	1	1	1	0	0	0
$\mathbf{s}_{hard}$	0	0	0	1	1	0

$\mathbf{skill}_4$	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{soft}$	0	0	0	0	0	0
$\mathbf{s}_{skill}$	1	0	0	0	0	0
$\mathbf{s}_{hard}$	0	1	1	1	1	1

$\mathbf{skill}_5$	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{soft}$	0	0	0	1	1	1
$\mathbf{s}_{skill}$	1	1	1	0	0	0
$\mathbf{s}_{hard}$	0	0	0	0	0	0

In case the considered manipulator is kinematically redundant, the selection vector $\mathbf{s}_{null}$ is added, to identify the self motion coordinate.

2.6 Summary

In this chapter, a classification of instantaneous constraints composing a preplanned task, based on constraints relevance for task execution, has been presented. The classification defines four classes of constraints, *null*, *soft*, *skill* and *hard*, which correspond to increasing levels of relevance. For each class of constraints, the possibilities for relaxation have been identified. Such a classification thus defines the possibilities for the modification of a preplanned task. Then, based on the proposed classification, a procedure for the decomposition of the task into elementary operations, named skills, has been introduced. The classification can be employed as the basis of a strategy for adaptation to unforeseen events for preplanned tasks.

CHAPTER 3

Task-consistent collision avoidance strategy

3.1 Introduction

Guaranteeing safety for human operators is one of the crucial requirements that industrial robots, in order to be deployed in a human-robot interaction scenario, have to accomplish. As already mentioned in Chapter 1, different approaches can contribute to the achievement of safety, ranging from safe robot design to the development of safety-oriented control strategies. In this chapter, the problem of safety in human-robot interaction is tackled using the approach of collision avoidance, and in particular building on the potential field framework. Originally introduced in [48], the potential field approach allows the computation of virtual repulsive forces, which can be used to ideally push a robot away from an obstacle. Such a method is an important tool for the application of collision avoidance in real time.

In the following, a collision avoidance strategy based on the constraints classification introduced in Chapter 2 is proposed. The goal of the strategy is to allow a robotic manipulator, which is performing a preplanned task, avoid unforeseen obstacles. As already introduced, preplanned tasks constrain all robot degrees of freedom. The execution of evasive motions thus requires the relaxation of at least one constraint. In this scenario, our constraints classification is used to identify possible constraints to be relaxed and a danger assessment, named the *danger field*, is adopted in order to decide which constraints to relax. Such an

assessment, originally proposed in [61] and further developed in [62] and [63], is characterised by the consideration of the robot as the source of danger for the obstacle, and captures the contribution of both robot velocity and distance from the obstacle. Moreover, it considers the complete robot structure for danger evaluation, differently from previous approaches which take into account only some points.
The goal of the strategy is thus to reduce danger by exploiting the degrees of freedom made available by relaxation, in order to enforce evasive motions. The proposed collision avoidance strategy is formalised using a finite state machine, which is parameterised on constraints classification and serves as a general template for the strategy definition.

3.2 Danger assessment and evasive velocities computation

The goal of the collision avoidance strategy is to allow the robot to be safely operated in an unstructured environment, enabling the close cooperation with human workers and the adaptation to a dynamic working scenario. For this, the robot must be able to react to the presence of obstacles while it is performing its task, in such a way that safety of the obstacle, which could potentially be a human worker, is maximised. For this purpose, a danger assessment, named the danger field, which measures the level of danger generated by the robot with respect to the surrounding obstacles, is adopted. Such an assessment can also be exploited to determine which direction of robot motion yields the maximum decrease of danger, and thus to derive a motion which maximises safety for the surrounding obstacles. Such a motion can then be used in a safety strategy, as a reaction to the presence of obstacles.
In this section, the danger assessment is briefly reviewed and the computation of evasive motions for a robot is discussed.

3.2.1 The danger field

The danger field assessment, proposed in [63], describes the level of danger associated with the robot motion for a point in the surrounding space. Such an assessment is composed by two parts, which describe two main sources of danger for the obstacles surrounding the robot. Let us start considering a pointlike robot located at $\mathbf{r}_r$, moving with velocity $\mathbf{v}_r$ and an obstacle located at $\mathbf{r}_o$. The first part of the assessment is the static danger field SDF:

$$SDF = \frac{1}{\|\mathbf{r}_o - \mathbf{r}_r\|} \tag{3.1}$$

which describes the part of danger associated with the distance between the robot and the obstacle. As distance decreases, the static danger field increases, obviously indicating the greater probability of collision between the robot and the obstacle. The second part of the

assessment is the kinetic danger field KDF:

$$KDF = \frac{\|\mathbf{v}_r\|(1 + cos\angle(\mathbf{r}_o - \mathbf{r}_r, \mathbf{v}_r))}{\|\mathbf{r}_o - \mathbf{r}_r\|^2}. \tag{3.2}$$

Such a term captures the influence on danger of the robot velocity $\mathbf{v}_r$ and of the angle between the velocity and the vector connecting the robot and the obstacle. The danger assessment can therefore discriminate between conditions where the robot is approaching the obstacle and where the robot is moving away from that.
The sum of the two elements is the kineto-static danger field (from now on simply *danger field*), which is defined as:

$$DF(\mathbf{r}_r, \mathbf{v}_r, \mathbf{r}_o) = \frac{k_1}{\|\mathbf{r}_o - \mathbf{r}_r\|} + \frac{\|\mathbf{v}_r\|(1 + cos\angle(\mathbf{r}_o - \mathbf{r}_r, \mathbf{v}_r))(1 - k_1)}{\|\mathbf{r}_o - \mathbf{r}_r\|^2} \tag{3.3}$$

where k_1 is a tuning parameter.
When moving from the toy example of a pointlike robot to a real robotic structure, it is important to consider how the robot shape and the different velocities along its structure influence the generated danger. For this purpose, the danger field DF can be integrated along the robot, which can be modelled as a series of segments. The cumulative danger for the robot i-th link CDF_i is therefore introduced:

$$CDF_i = \int_{link\, i} DF_i = \int_0^1 DF_i(\mathbf{r}_i, \mathbf{v}_i, \mathbf{r}_o)\, \mathrm{d}s \tag{3.4}$$

where $\mathbf{r}_i$ and $\mathbf{v}_i$ are the position and the velocity along the i-th link, and are expressed as functions of the link coordinate s:

$$\begin{aligned} \mathbf{r}_i &= \mathbf{r}_{i,s} + s(\mathbf{r}_{i,e} - \mathbf{r}_{i,s}) \\ \mathbf{v}_i &= \mathbf{v}_{i,s} + s(\mathbf{v}_{i,e} - \mathbf{v}_{i,s}) \qquad \text{with} \qquad s \in [0, 1], \end{aligned}$$

where $\mathbf{r}_{i,s}$, $\mathbf{r}_{i,e}$ and $\mathbf{v}_{i,s}$, $\mathbf{v}_{i,e}$ are the positions and the velocities of the i-th link tips respectively. Fig. 3.1 shows some examples of the cumulative danger field induced by a single link robot. On one hand, the danger field is integrated along the robot structure, in order to capture the effect of the robot shape and movement on the induced danger. On the other hand, danger is computed for a single point in the workspace, so obstacles are consequently considered as point-like entities. Integration of the danger field on the obstacle surface is a problematic task, due to the absence of a general closed form solution. The obstacles shape is therefore considered resorting to multiple points extracted from their surface, and making the assumption that a sufficient resolution is available, so as to compute a cumulative danger field which captures the obstacle shape satisfactorily.

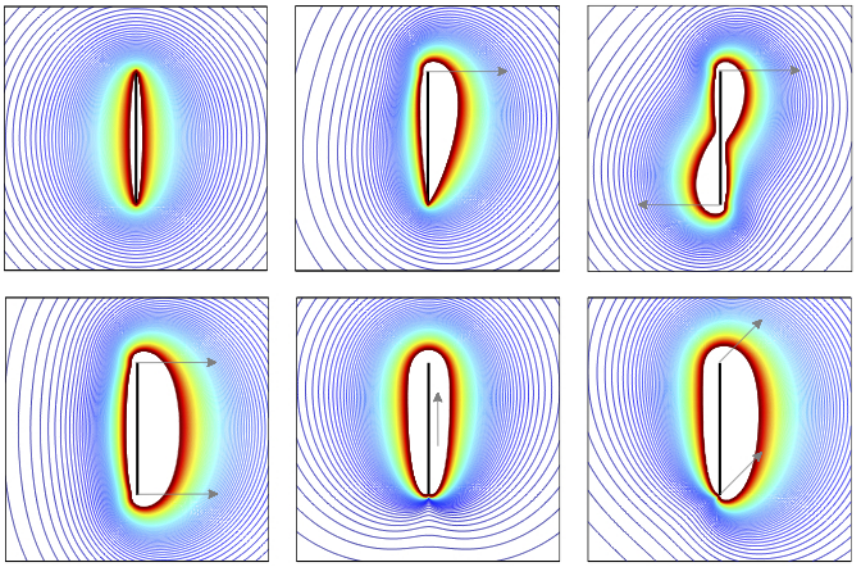


Figure 3.1: *The cumulative danger field generated by a single link robot, moving in different directions. Taken from [60].*

3.2.2 Virtual repulsive force field

The danger field is a scalar quantity which defines the level of danger for each point in the robot workspace. If, for safety reasons, the level of danger induced by the robot on an obstacle has to be reduced, a scalar assessment of danger is not sufficient. In fact, in order to move the source of danger in the direction of maximum danger decrease, knowledge of the variation of the danger level in the space is necessary.
The danger field gradient offers such an information, and can be used to define a vector field which is directed as the maximum variation of danger. The vector field $\overrightarrow{CDF}_i(\mathbf{r})$, associated with the i-th link, is introduced:

$$\overrightarrow{CDF}_i(\mathbf{r}) = CDF_i(\mathbf{r}) \frac{\boldsymbol{\nabla} CDF_i(\mathbf{r})}{\|\boldsymbol{\nabla} CDF_i(\mathbf{r})\|} \tag{3.5}$$

such a field is directed as the danger field gradient, and its amplitude is defined as the danger field evaluated at $\mathbf{r}$.
Following the widely adopted potential field approach [48], $\overrightarrow{CDF}_i(\mathbf{r})$ can be used as a virtual repulsive force field which pushes the robot in the direction of maximum danger decrease. The vector $\overrightarrow{CDF}_i(\mathbf{r})$ can therefore be interpreted as a repulsive force anchored in $\mathbf{r}$, whose amplitude is the danger generated by the robot in $\mathbf{r}$. An evasive motion can finally be derived from the computed repulsive force, in order to avoid the obstacle and to

maximise its safety.

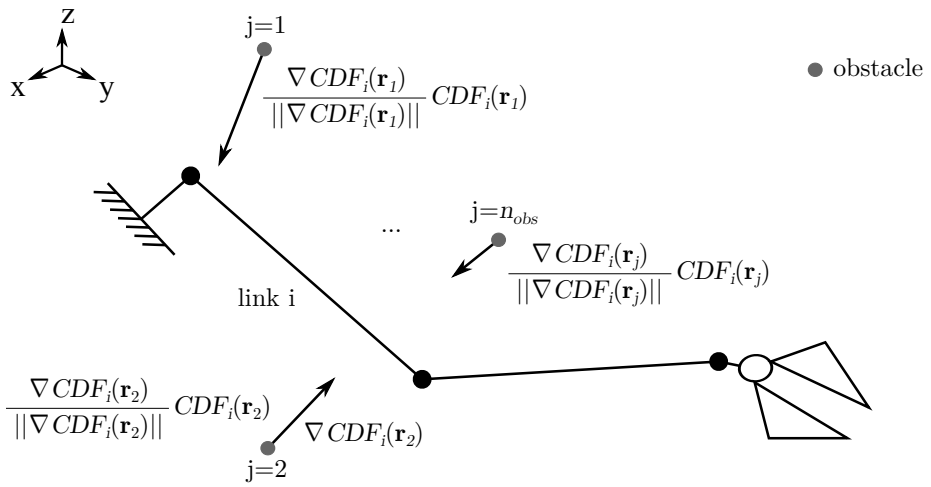


Figure 3.2: *The virtual repulsive forces computed for different obstacles surrounding the robot.*

A cumulative version of the vector field $\overrightarrow{CDF}_i(\mathbf{r})$, to capture the contribution of multiple obstacle points, can be obtained. Given n_{obs} point-like obstacles, as shown in Fig. 3.2, for the i-th robot link a cumulative virtual repulsive force can be computed:

$$\overrightarrow{CDF}_{s,\,i} = \sum_{j=1}^{n_{obs}} CDF_i(\mathbf{r}_j) \frac{\boldsymbol{\nabla} CDF_i(\mathbf{r}_j)}{\|\boldsymbol{\nabla} CDF_i(\mathbf{r}_j)\|} \tag{3.6}$$

As $\overrightarrow{CDF}_{s,\,i}(\mathbf{r})$ 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 thus used, so as 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:

$$\overrightarrow{CDF}^{*}_{s,\,i} = \frac{\overrightarrow{CDF}_{s,\,i}}{\|\overrightarrow{CDF}_{s,\,i}\|} \cdot \max_j CDF_i(\mathbf{r}_j) \tag{3.7}$$

The direction of the force is therefore given by the vector sum of the virtual forces related to the n_{obs} obstacles, 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 average of all the forces directions, the computed force reflects obstacles disposition.

3.2.3 Evasive velocities computation

The introduced virtual force field conceptually pushes the robot away from the obstacle, causing the decrease of the level of danger induced on it. Such a force has to be transformed into a controlled evasive motion to be performed by the manipulator, displacing the robot along the force direction and proportionally to the force modulus. For this purpose, the repulsive force, which is computed in the obstacle position, can be virtually applied on the robot surface. As the robot is controlled at the joint level, it is convenient to transform the virtual repulsive force into virtual repulsive torques at the robot joints. Given a generic robot structure, the joint torques corresponding to a force applied on the robot are defined as:

$$\boldsymbol{\tau} = \mathbf{J}_p(\boldsymbol{q})^{\mathbf{T}} \cdot \mathbf{F} \tag{3.8}$$

where $\boldsymbol{\tau}$ is the joint torques vector, $\mathbf{J}_p(\boldsymbol{q})$ is the Jacobian matrix of the force application point and $\mathbf{F}$ is the vector of force / torques applied to the robot in p.

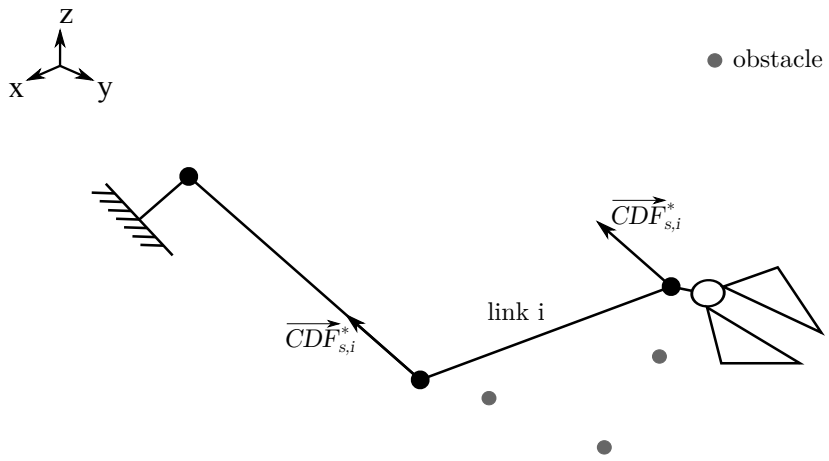


Figure 3.3: *The application of the cumulative virtual repulsive forces to the robot link.*

Since the danger field is induced by the whole robot link on the obstacles, a criterion is needed to determine on which point of the link the associated virtual repulsive force should be applied. Depending on the robot configuration, a force applied on the robot structure yields different torques at the joints, including the case when external forces are balanced by reaction forces, and zero torques are obtained. Since the virtual force goal is to guarantee the evasion from the obstacle, such an occurrence has to be avoided. In order to reduce the computations needed for the application of the forces, a reduced number of application points should be chosen, hence limiting the number of Jacobian

matrices to be computed. However, if a single point of application for each force is used, configurations where a force yields zero torques at the joints can easily happen. The virtual force associated with the i-th link can thus be applied at the two link endpoints in order to reduce the occurrences of such configurations, and to limit the amount of related computations. The torques obtained for the i-th link are therefore:

$$\boldsymbol{\tau}_i = \mathbf{J}_{i-1,v}(\boldsymbol{q})^{\mathbf{T}}(\mathbf{q}) \cdot \overrightarrow{CDF}^*_{s,\,i} + \mathbf{J}_{i,v}(\boldsymbol{q})^{\mathbf{T}}(\mathbf{q}) \cdot \overrightarrow{CDF}^*_{s,\,i} \tag{3.9}$$

where $\mathbf{J}_{i,v}(\boldsymbol{q})$ is the linear velocity Jacobian of the i-th Denavit Hartenberg frame. Fig. 3.3 shows an example of the effect of the application of the virtual force at the two link endpoints. In such a configuration, the application of the force to a single point, e.g. the first tip of the i-th link, would have led to zero evasive torques, preventing evasion. The computed evasive torques depend on the obstacles positions, which can vary abruptly and unpredictably. To avoid feeding the robot with such abrupt commands, evasive torques can be filtered in order to attenuate high frequency harmonics. Moreover, for reasons that concern the practical application of such evasive actions, transforming the torques into joint velocities is more convenient. A mass-damper impedance filter is thus used to obtain joint velocities from the evasive torques:

$$\dot{\mathbf{q}}_{ev} = (\mathbf{M}s + \mathbf{D})^{-1} \cdot \boldsymbol{\tau} \quad \text{where} \quad \boldsymbol{\tau} = \sum_{i=1}^{n_{link}} \boldsymbol{\tau}_i, \tag{3.10}$$

$\mathbf{M}$ and $\mathbf{D}$ are mass and damping matrices respectively, imposing the dynamic behaviour to the impedance filter. In order to decouple the joints response, the two matrices can be selected as diagonal. The ratio between the damping and mass value chosen for each joint define the joint filter pole frequency.

3.3 Constraints relaxation strategy

In Chapter 2 a classification has been presented to define relevance of constraints composing a preplanned task. Adopting such a classification, the possibilities for modification of the preplanned task are identified and, more specifically, the implications of relaxation of the different constraints are defined. Exploiting these possibilities of task modification, unforeseen events generating new constraints, arising during the execution of the preplanned task, can be taken into account. For example, if a high level of danger is induced on an obstacle and an evasive movement has to be performed, the relaxation of task constraints can make some robot degrees of freedom available for the execution of the evasion.
A strategy for the relaxation of constraints composing a preplanned task is proposed in the following. The strategy directly derives from the introduced constraints classification and can be automatically designed once the classification has been performed.

3.3.1 Finite state machine for constraints relaxation

The classification introduced in Chapter 2 defines four classes of constraints, with the following associated possibilities for relaxation:

1) *Null* constraints: relaxation is possible without consequences on the robot TCP pose

2) *Soft* constraints: relaxation is possible without consequences on task execution

3) *Skill* constraints: relaxation is possible but implies suspension of task execution

4) *Hard* constraints: relaxation is not possible

In order to face unforeseen events and then to enforce new constraints, the constraints composing the task can be relaxed, according to the possibilities defined by the classification. More specifically, constraints of increasing relevance can be gradually relaxed as unforeseen ones become more compelling. The latter are thus assigned a priority, based on which the set of constraints to be relaxed is defined. Fig. 3.4 shows a pictorial representation of the stack of constraints composing the task and of the other constraints arising at execution time, whose priority changes according to the environment conditions.

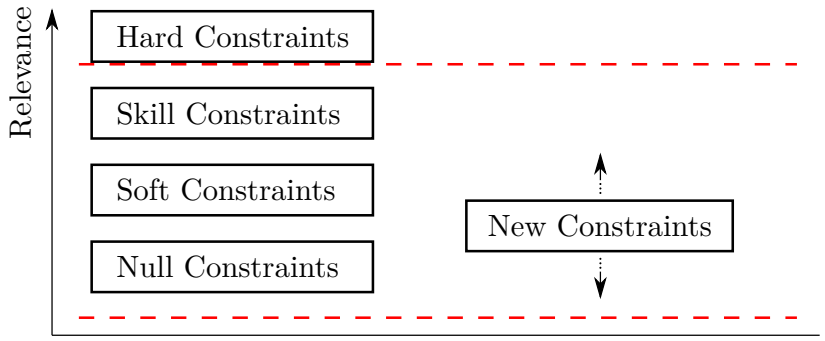


Figure 3.4: *Pictorial representation of the stack of task constraints and of constraints arising at execution time.*

In order to determine the groups of constraints to be relaxed, the relative priority of the unforeseen constraints with respect to task ones has to be set. Considering the case where the further constraint to be enforced is the evasive motion, the current level of danger induced by the robot towards an obstacle can be used as the constraint priority level. Three danger thresholds, CDF_{low}, CDF_{med} and CDF_{high}, corresponding to the relaxation of *null*, *soft* and *skill* constraints, respectively, are thus used. Fig. 3.5 shows the set of applied constraints as a function of danger level.
As danger exceeds one of the three thresholds, the corresponding constraints are relaxed, freeing degrees of freedom to execute evasive motions.

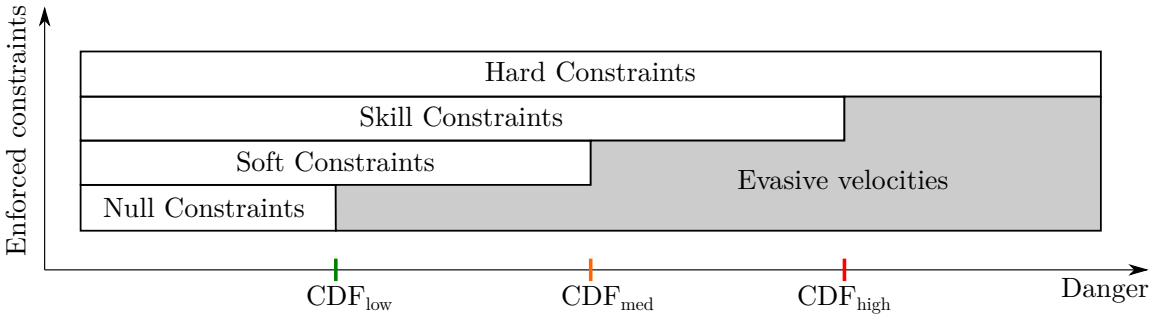


Figure 3.5: *The set of enforced constraints is represented as a function of danger level. As danger grows, an increasing set of constraints are relaxed.*

A finite state machine, from now on *constraints state machine* or *CSM*, is adopted to select the set of constraints to be performed. The *CSM* output is the vector $\mathbf{cs}$ which determines the constraints status:

$$cs_i = \begin{cases} 1, & \text{if the i-th constraint is enforced} \\ 0, & \text{if the i-th constraint is relaxed} \end{cases} \tag{3.11}$$

The state machine structure reproduces the structure the task: for each skill, a state is assigned, whose inner structure reflects the collision avoidance strategy. The $\mathbf{cs}$ vector is determined using the $\mathbf{s}_{null}$, $\mathbf{s}_{soft}$, $\mathbf{s}_{skill}$ and $\mathbf{s}_{hard}$ vectors, introduced in Chapter 2, such that the state machine is directly derived from the classification. Fig. 3.6 shows the structure of one sample state.

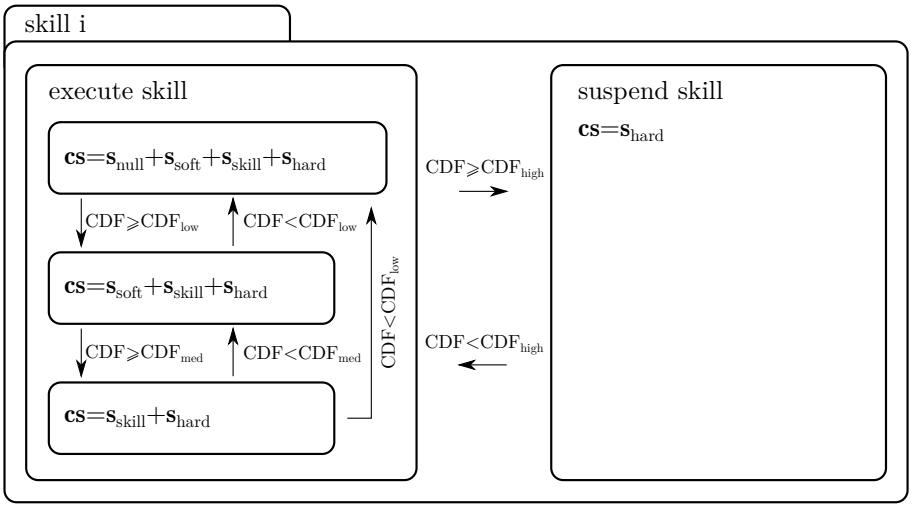


Figure 3.6: *Structure of a sample state of the constraints state machine.*

Each state is divided into two main substates, corresponding to skill execution and skill suspension. The latter is executed when *skill* constraints are relaxed: in fact, according to the classification introduced in Chapter 2, *skill* constraints relaxation implies skill suspension. As already mentioned, the strategy is designed to be applied to preplanned tasks. During skill suspension, the preplanned trajectory is stopped, and is resumed as soon as the danger level falls below CDF_{high}, as shown in Fig. 3.7.

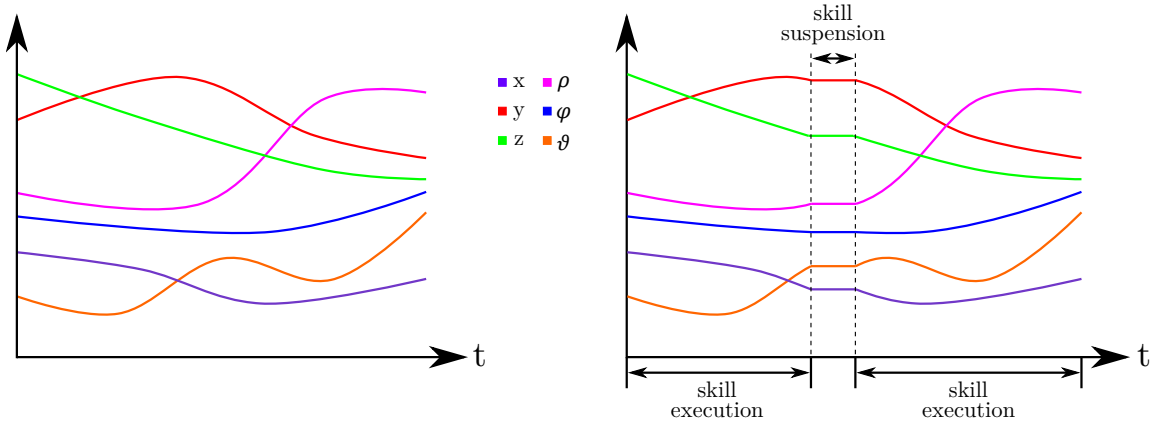


Figure 3.7: *Pictorial representation of skill suspension. As the skill is suspended, the preplanned trajectory execution is stopped.*

3.3.2 Transition between constraints

Given the set of enforced constraints, defined by the vector $\mathbf{cs}$, evasive motions can be performed exploiting the remaining degrees of freedom. Chapters 5 and 6 will introduce two different control strategies for the execution of evasive velocities, taking into account increasingly complex constraints. In this chapter, the general strategy adopted by the introduced state machine, for management of evasive velocities, is detailed.
Generally speaking, evasive velocities computed according to (3.10) are performed taking into account constraints related to the skill being executed. Such velocities make the robot evade from a detected obstacle, and cause the robot pose to drift from the preplanned trajectory. As danger induced by the robot on surrounding obstacles changes, the set of relaxed constraints changes accordingly. If a constraint is relaxed and then enforced once again, discontinuities in the commanded joints positions and velocities may occur, due to the difference between the actual robot pose and that commanded by the preplanned task. Fig. 3.8 shows an example of the drift induced by evasive velocities, with respect to the preplanned trajectory: at first, only soft constraints are relaxed (represented by the grey solid line), and the corresponding coordinates start to drift from the reference (represented by the dashed line). Then, trajectory is suspended and skill constraints are

relaxed (represented by the orange and blue lines). When the constraints are enforced once again, an offset is present between the current coordinates values and the preplanned ones, which would lead to a discontinuity in the commands.

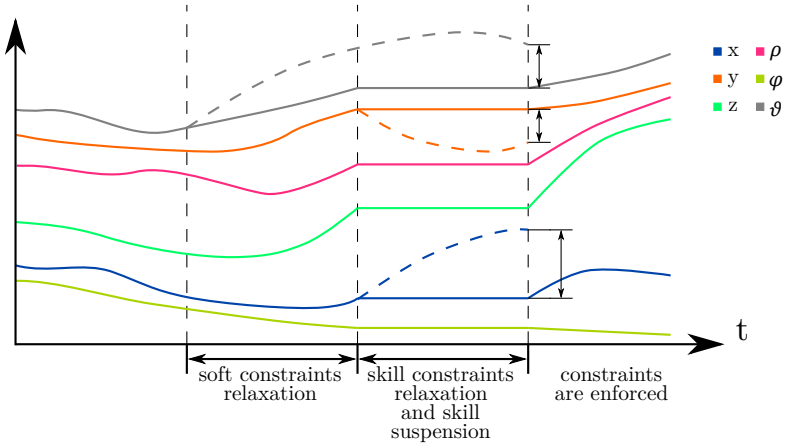


Figure 3.8: *Relaxation of constraints. Soft constraints are relaxed at first, and the corresponding coordinates drift from the planned values. Skill execution is then suspended, and skill constraints are relaxed. When skill execution is resumed, the drift from the planned trajectory has to be compensated. The preplanned trajectory is represented by solid lines, while modified trajectories are represented by dashed lines.*

In order to avoid discontinuities in the control law, a trajectory smoothly connecting the current robot pose to the preplanned trajectory, from now on *return* trajectory, is adopted. Therefore, when an element of $\mathbf{cs}$ changes from 0 to 1, a return trajectory is activated for the corresponding coordinate. It has to be noted that, when considering a *skill* constraint, resumption of skill execution can be done upon completion of the return trajectory. In fact, by definition, *skill* constraints relaxation implies the suspension of skill execution, which can be resumed only after such constraints have been enforced once again. Such policy is described in Fig. 3.9, where the state machine presented in Fig. 3.6 is detailed including the strategy for resumption of skill execution. When a *null* or *soft* constraint is enforced, once again after relaxation, the corresponding coordinate simply follows the return trajectory, until the preplanned task reference is reached. The algorithms adopted for the planning of the return trajectories will be introduced in the following chapters. For the purpose of describing the adopted finite state machine, pictorial examples, shown in Figs. 3.10-3.11, are sufficient. Fig. 3.10 shows the relaxation and the succeeding enforcement of a *soft* constraint. It has to be noted that the preplanned trajectory is not suspended, and the return one is able to connect the previously relaxed coordinate with the changing reference. Fig. 3.11 depicts the relaxation and the enforcement of a *skill* constraint: the preplanned trajectory is suspended until the return one has reached it.

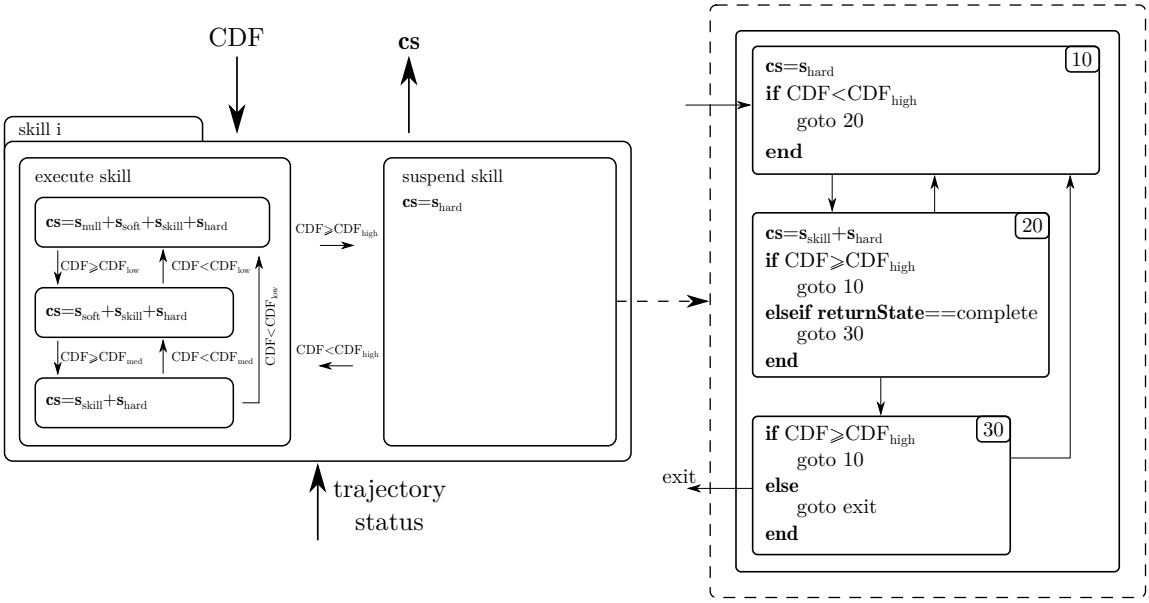


Figure 3.9: *The part of the state machine in charge of trajectory suspension and resumption.*

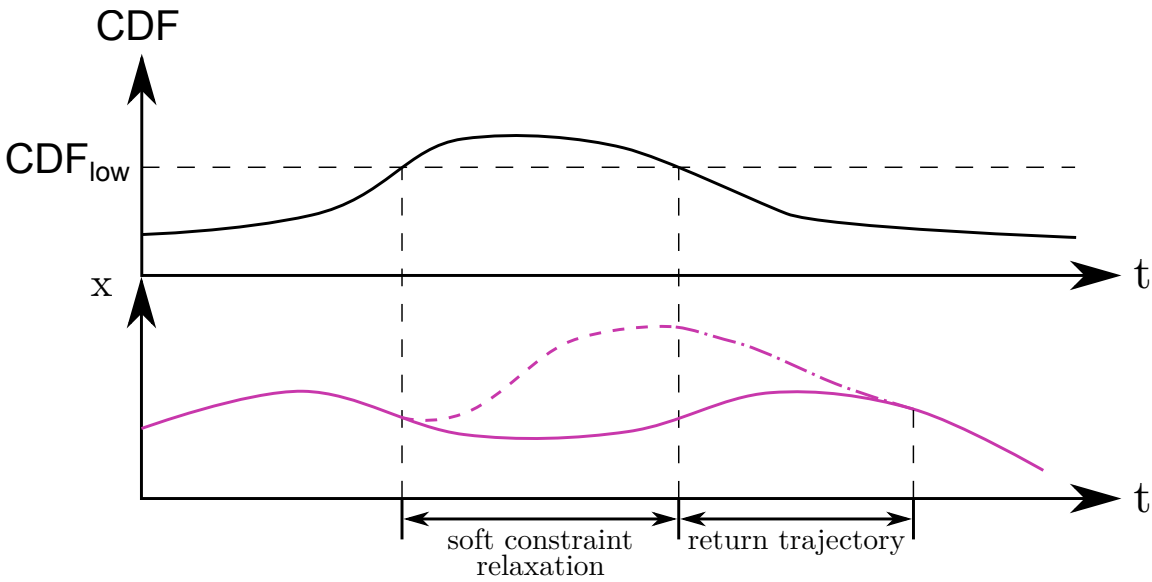


Figure 3.10: *Activation of a return trajectory, represented by the dash-dotted line, for a soft constraint.*

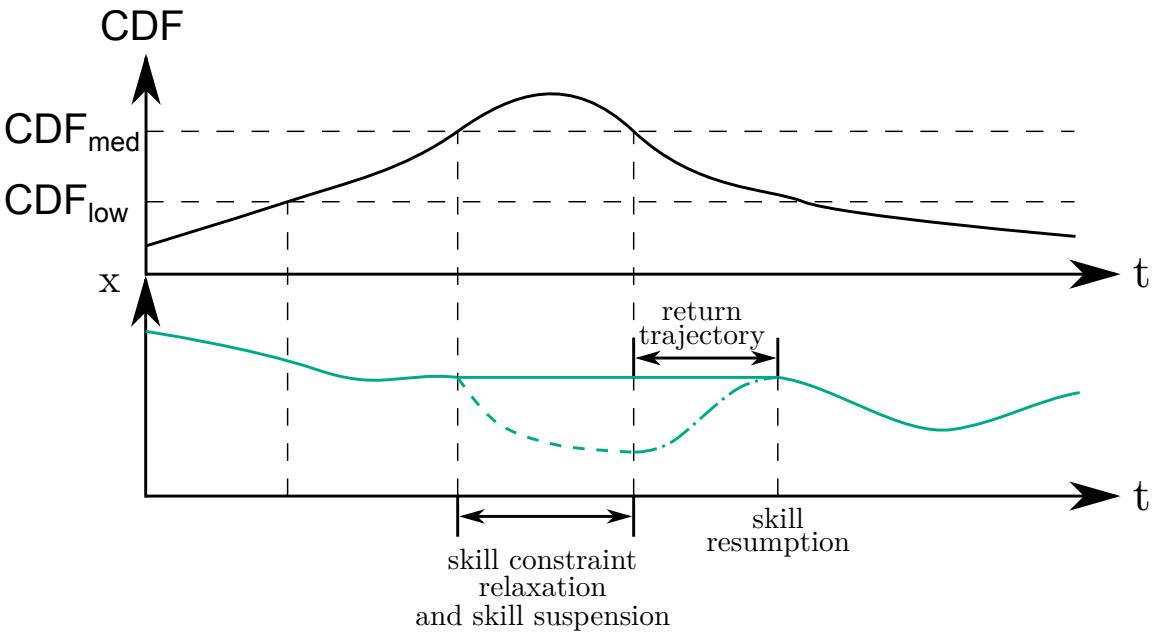


Figure 3.11: *Activation of a return trajectory, represented by the dash-dotted line, for a skill constraint.*

3.4 Summary

In this chapter, a strategy for the avoidance of unforeseen obstacles during the execution of preplanned tasks has been presented. The strategy is based on the relaxation of constraints constituting a skill, in order to execute evasive motions. An assessment of danger induced by the robot on surrounding obstacles, the danger field, has been used to compute a virtual repulsive forces field, from which evasive velocities, causing the robot to move away from the obstacle, have been derived. A finite state machine for the management of constraints constituting a skill has been proposed. Based on the current level of danger, the finite state machine determines which set of constraints, categorised according to the classification proposed in Chapter 2, has to be relaxed. Finally, the state machine manages transition between enforcement of different sets of constraints, guaranteeing the bumpless commutation between them.

CHAPTER 4

Integration of the collision avoidance strategy with an industrial controller

4.1 Introduction

Currently available commercial industrial robot controllers are not designed to allow adaptation of a robot task to unforeseen events. In fact, the robot programmer has usually to create a fully specified task, which completely constrains the robot degrees of freedom and does not give any possibility for modification at execution time. The only way to obtain some adaptation capability is to predefine different execution paths, that are executed based on external signals or environment measurements. Moreover, industrial controllers usually only allow high-level robot programming and do not grant access to information about the robot state, that is joints positions and velocities, impeding the design of complex control strategies. Such devices thus lack the tools needed to enable adaptation capability. Nonetheless, research tools for the real time modification of the set points computed by industrial controllers are available: such tools can be adopted to design reactive control strategies, to cope with unforeseen events. For example, the robot manufacturer KUKA developed an interface, named FRI (Fast Research Interface), which provides access to position, velocity and acceleration set points of an industrial robot and allows their modification at frequency of 1 kHz [54]. An interesting example of a collision avoidance control

system, implemented with such an interface, can be found in [36]. The same manufacturer introduced a lower performance interface, which provides access to and modification of the robot state, at a frequency of approximately 80 Hz. A collision detection system, based on such an interface, was presented in [37]. An interesting contribution on the real time modification of a preplanned trajectory, generated by an industrial controller, was presented in [5].
In this chapter, the previously presented collision avoidance strategy is integrated with an industrial controller, thus extending its functionalities with capability of adaptation to an unstructured environment. For this purpose, a software architecture for the communication and modification of the controller set points is proposed. More specifically, such an architecture allows an external controller to modify the industrial controller task trajectory, in order to evade from a detected obstacle, leaving the user interface and the normal programming procedure unchanged. Additional control functions are added to the robot proprietary language, using which the user can specify the type of constraints which characterise a task. Using ad-hoc communication functions, the external controller receives information regarding the constraints composing the task, and modifies the industrial controller trajectory accordingly, in real-time. The communication and interaction between the industrial controller and the external controller is allowed by an open version of the industrial controller software, and by a special controller interface. The actual implementation of the communication system is here described, detailing the adopted communication protocols, the principles followed in the design of the system and its performance limitations. The purpose of the system analysis and description is to offer concrete guidelines for the expansion of industrial controllers functionalities, and, moreover, demonstrating the feasibility of such an extension within the current framework of industrial controllers.

4.2 Open controller architecture

The control architecture detailed in this chapter has been designed to be integrated with an ABB IRC5 industrial robot controller, shown in Fig. 4.1.
In its normal configuration, the controller allows the programmer to specify the robot trajectory by means of high-level trajectory planning functions. The user is not granted direct access to the computed trajectory, nor can the planned trajectory be modified at execution time. In order to enable such functionalities, a particular version of the robot controller software and an interface for the communication with external computers, developed by Lund University [10], is used. Such an interface was originally developed to perform sensor based control on industrial controllers, and allows to control industrial robots with externally executed control laws.
The default industrial controller architecture can be decomposed into two main elements:

- the **main computer**, which is in charge of planning the robot trajectory and comput-

Figure 4.1: *The ABB IRC5 industrial robot controller.*

ing the joints references, and includes the user interface;

- the **axes computer**, which is not accessible to the user and executes the low level control loops on the robot joints.

In the normal controller configuration, depicted in Fig. 4.2, the main computer provides the set points to the axes computer. The communication channels between the two are not accessible.

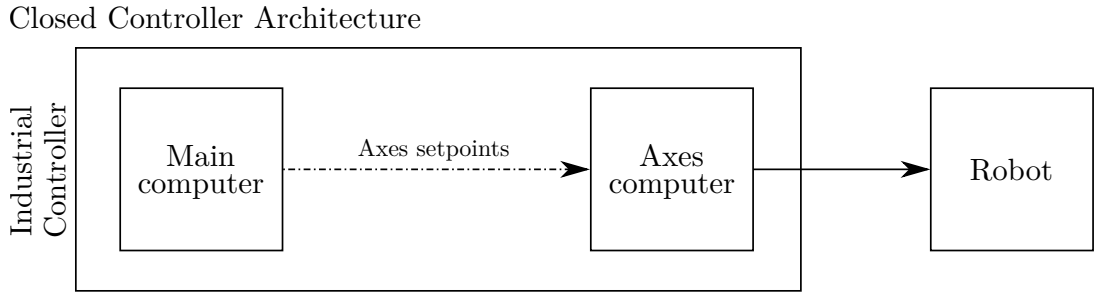


Figure 4.2: *A schematic representation of the closed controller architecture.*

In the open controller architecture, the communication channels between the main computer and the axes computer are made accessible, allowing the user to conceptually insert an external computer between the two. Moreover, a bidirectional communication channel

between the main computer and the external computer is created. Such a channel allows an external control law to interact with the high level controller functionalities, and thus to provide the user with a direct interface with the external control system. The open controller architecture is depicted in Fig. 4.3.

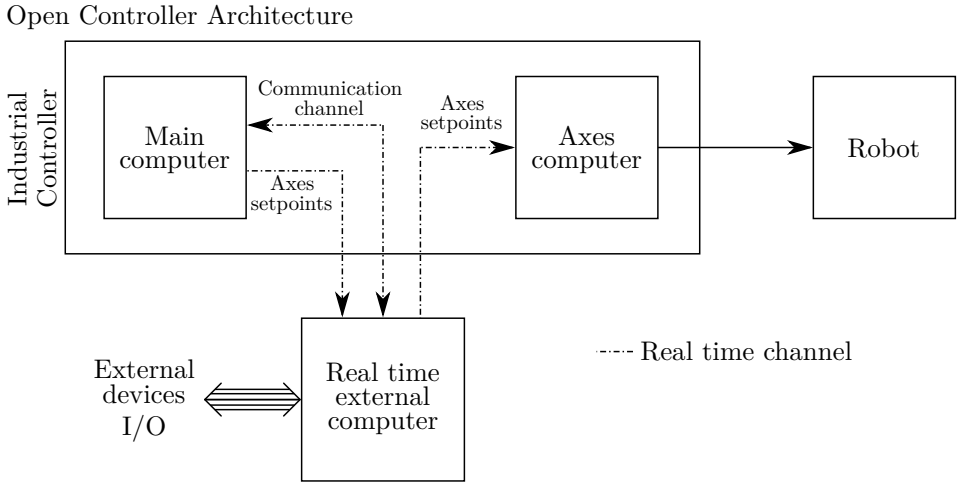


Figure 4.3: *The open controller architecture, with the external computer connected to the controller.*

The external computer opens a further channel for the interaction between the industrial controller and external devices such as sensors. Such a channel is crucial to create sensor based and reactive control strategies.
In the following, the features of the mentioned channels are detailed.

4.2.1 Available communication channels

The channels created by the open controller architecture are divided into two main groups: mono directional, from the main computer to the axes computer, passing from the external one, and bidirectional, between the main computer and the external one.
The mono directional channels include:

- robot joints position references, from now on $\mathbf{q}_{ref,ic}$;
- robot joints velocity references, from now on $\dot{\mathbf{q}}_{ref,ic}$.

The bidirectional channels include:

- a channel to specify the type of communication between the controller and the robot (send/receive), from now on **Instruction**;
- a channel to communicate up to six integer values, from now on **value**;

- a channel for the communication of up to two strings, from now on **string**.

The use of the communication channels requires compliance to hard real time constraints in order to guarantee the functionality of the control system. More specifically, the external computer has to generate a reference for the axis computer every 4 ms.

4.3 Communication protocol

Two state machines on the two sides handle the communication between the industrial controller and the external computer. Due to implementation constraints, the communication system is configured according to a client-server architecture, with the industrial controller working as a client and the external computer as a server. Communication therefore takes place only as the industrial controller takes the initiative.

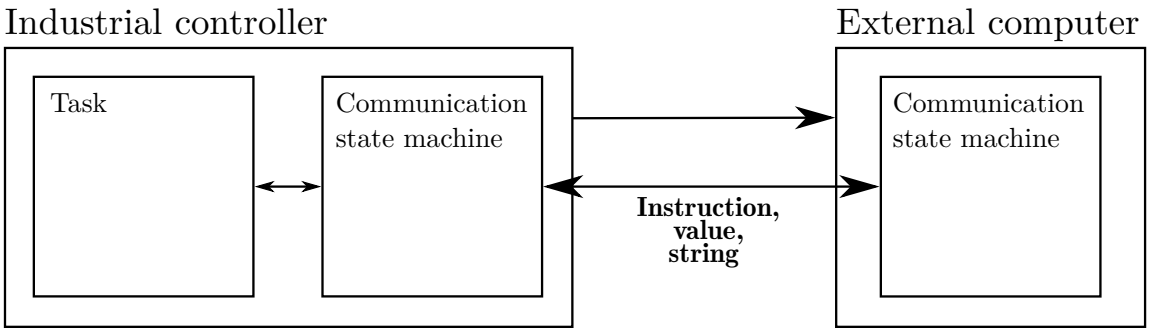


Figure 4.4: *The communication state machines on the two sides of the communication system.*

The industrial controller and the external computer are characterised by two different approaches in the execution of their programs, which determine also the method for the communication between them: on the external computer, the control algorithm is executed cyclically, while, on the industrial controller, the task program is executed sequentially. However, as will be explained in the following sections, the communication requires the execution of cyclic operations: for this reason, a timed interrupt is used in the task program in order to activate the communication state machine. This configuration determines the operation of the industrial controller as the client, and determines some system limitations, which will be detailed later on in this chapter. The task program is thus composed of the sequential robot instructions and by a cyclic part, which is used for communication. Such a configuration is depicted in Fig. 4.4.
The communication state machines perform various operations, however, a common protocol is adopted for the exchange of messages. Fig. 4.5 shows the general protocol adopted for such a communication. The following steps are taken during the communication of a message:

1. upon the activation of an interrupt or a flag, the communication process is activated;
2. a message is sent to the external computer by calling the communication function and by specifying the communication direction through the **Instruction** variable;
3. the communication process waits for a fixed amount of time;
4. a message is received from the external computer, calling the communication function once again in the receiving mode;
5. depending on the received message, an action is taken.

It is worth noticing that, while the sending/receiving operations on the external computer side can be performed at every execution step, on the industrial controller side such operations are performed only upon the call of a dedicated function. Such a call can take place at most at the timed interrupt activation frequency, which, being notably lower than the external code execution frequency, constitutes the bottleneck of the communication system. Such a limitation will be discussed in more details in Section 4.5.

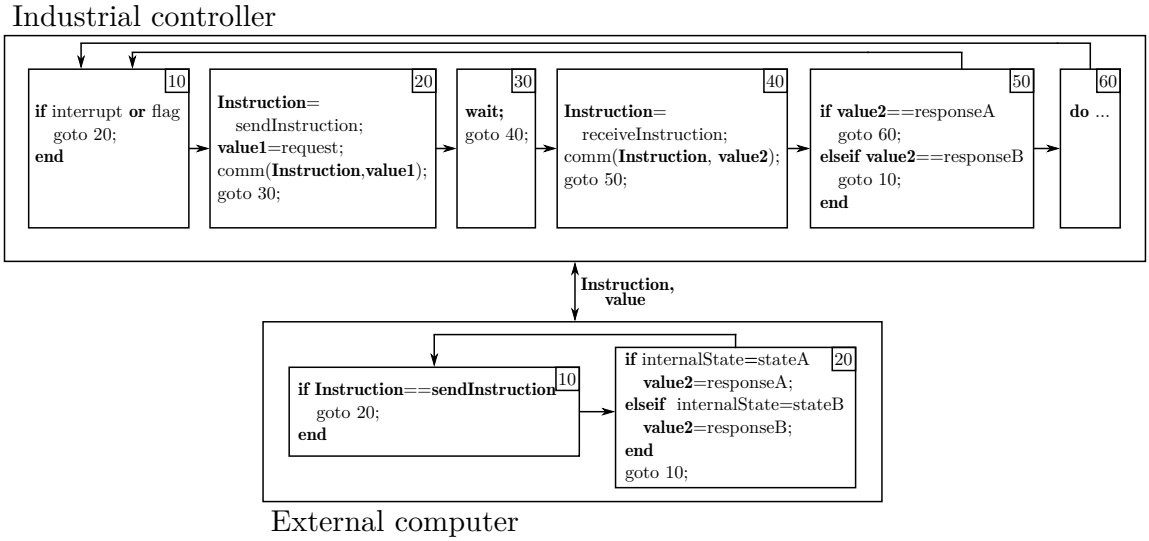


Figure 4.5: *The state machine representing the general protocol for the communication of messages between the industrial controller and the external computer.*

4.4 Functionalities of the communication system

Using the introduced communication system, an industrial controller and an external computer can exchange messages during the execution of a robot task. In the considered scenario, the communication system is used to control the collision avoidance strategy

presented in Chapter 3, which is executed on the external computer. Thanks to ad-hoc communication functions, the robot program running on the industrial controller can ensure consistency of the strategy with the task being performed. The following control actions are carried out through the communication channels:

- **activation** of the collision avoidance system;
- **transition** between skills;
- **suspension** and **resumption** of skill execution;
- **deactivation** of the collision avoidance system.

4.4.1 Example task

As already mentioned, the communication system allows the robot programmer to control the collision avoidance system using the normal programming interface. The introduced communication functionalities can be activated simply by calling specific functions or through activation flags. The functions and the flags for management of the collision avoidance strategy are the following ones:

- **activateCollAvSystem**: by calling this function, activation of the collision avoidance system is performed;
- **robTransReq**: using this flag, transition to the next skill is initiated;
- **robTransAll**: using such a flag, the communication system enables transition to the next skill;
- **robTaskCompl**: when the robot task is completed, such a flag is used to request the deactivation of the collision avoidance system;
- **constType**: when the collision avoidance system is activated, or a skill transition is performed, the constraints types for the new skill have to be communicated to the collision avoidance system. This string variable is adopted to specify the constraints types, using one string character for each coordinate. Characters “3”, “2” and “1” are used to define *hard*, *skill* and *soft* constraints respectively.

In the following, a simple example of a complete robot program composed of two skills, coded with the the ABB RAPID programming language, is sketched:

```
constType := "222111";
activateCollAvSystem;
WaitUntil taskStart;
MoveL point1, v100, z15, tool0;

constType := "222331";
robTransReq := TRUE;
WaitUntil robTransAll;
robTransAll := FALSE;
MoveL point2, v100, z15, tool0;

robTaskCompl := TRUE;
```

The robot program is made of the above sequential part and by a cyclic routine, which is executed upon activation of an interrupt. The flags adopted to control the collision avoidance system are used in the cyclic routine, which is also in charge of skill suspension and resumption.

In the next sections, the functions employed to perform the introduced control actions on the collision avoidance system are detailed.

4.4.2 Activation of the collision avoidance system

The first action that can be performed on the collision avoidance system, by means of the communication channels, is the activation of the system itself. The initial state of the CSM presented in Chapter 3 is an idle one, during which no constraint is relaxed. No evasion is thus performed, and the set points received from the main computer are sent to the axes computer without any modification.
When the collision avoidance system is activated, the following operations are thus performed:

- the constraints of the first skill are communicated to the system;
- the CSM advances to the first skill state;
- computation of obstacles positions and danger field starts;
- modification of robot set points starts.

Fig. 4.6 shows the protocol for the state machine activation. The **string** channel is used to communicate the types of constraints characterising the first skill. As the constraints

types are communicated by the industrial controller, no information about the task has to be programmed in the collision avoidance system, thus increasing its flexibility.

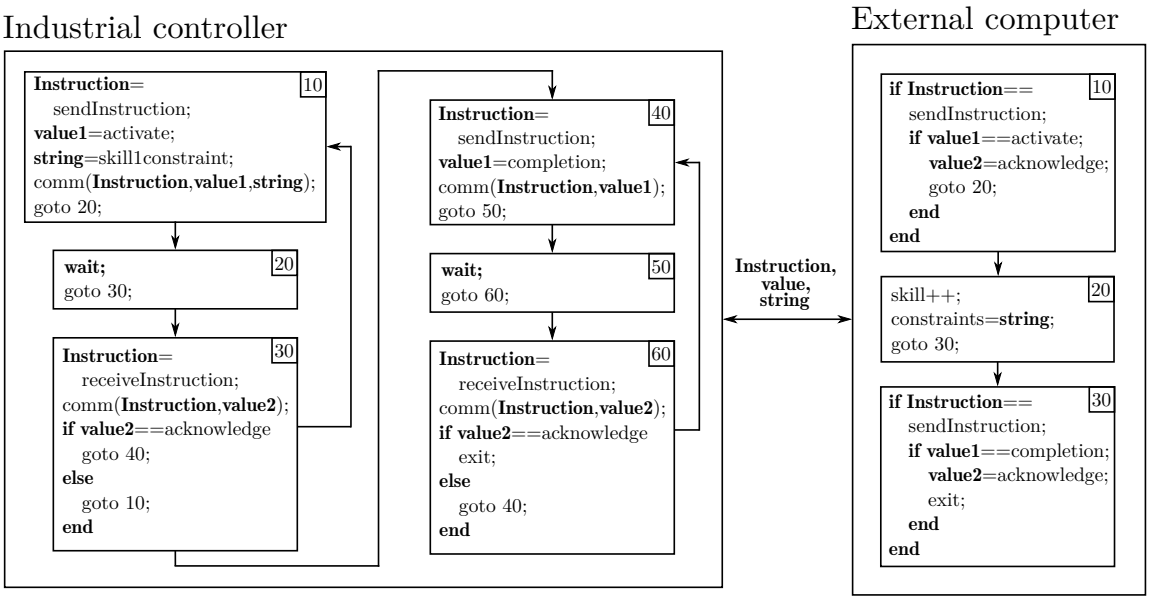


Figure 4.6: *The state machine representing the protocol for activation of the collision avoidance system.*

The implementation in the industrial controller proprietary RAPID language is shown in the next code excerpt.

```
PROC activateSafety()
   mcgdata.value6 := activateCall;
   mcgdata.string1 := constType;

   WHILE TRUE DO
      MocGenInstr mcgInstrCall, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO ACTIVATION;
      ENDIF
   ENDWHILE
```

```
   ACTIVATION:

   mcgdata.value6 := waitCall;
   WHILE TRUE DO
      MocGenInstr mcgInstrWait, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO WAITDONE;
      ENDIF
   ENDWHILE

   WAITDONE;
   taskStart := TRUE;
ENDPROC
```

The **MocGenInstr** function is used to send and receive messages to and from the external computer. The first function argument is used to define whether a message has to be sent or received, while the second argument is the message to be sent or received. The first argument is communicated over the **Instruction** channel, while the second argument is sent or received over the **value** and **string** channels.

4.4.3 Transition between skills

Once the collision avoidance system has been activated, when a new skill has to be started, an action is requested to such a system. As constraints relevance may change between skills, constraints that have been relaxed may no longer be relaxable in the next skill. A return trajectory has hence to be performed in order to bring robot coordinates back to the industrial controller reference. The industrial controller has thus to communicate to such a system the completion of skills and the need to start the following ones. During skill transition, evasive velocities are deactivated, so as to allow the execution of the return trajectory. In order to limit the possible consequences on safety, such an operation is performed only if CDF<CDF_{high}.
The following operations are hence performed during skill transition:

- new skill constraints types are sent to the collision avoidance system;
- return trajectories are activated, in order to prepare the robot for skill transition;

- the task state machine proceeds to the following skill.

As already mentioned, the communication of the types of constraints characterising each skill avoids the need to reprogram the collision avoidance system as the considered task changes. Fig. 4.7 shows the protocol for skill transition. On the external computer side, the communication state machine interacts with the task state machine, determining the skill transition, and receives information from the return trajectory generation about the state of the return trajectory.

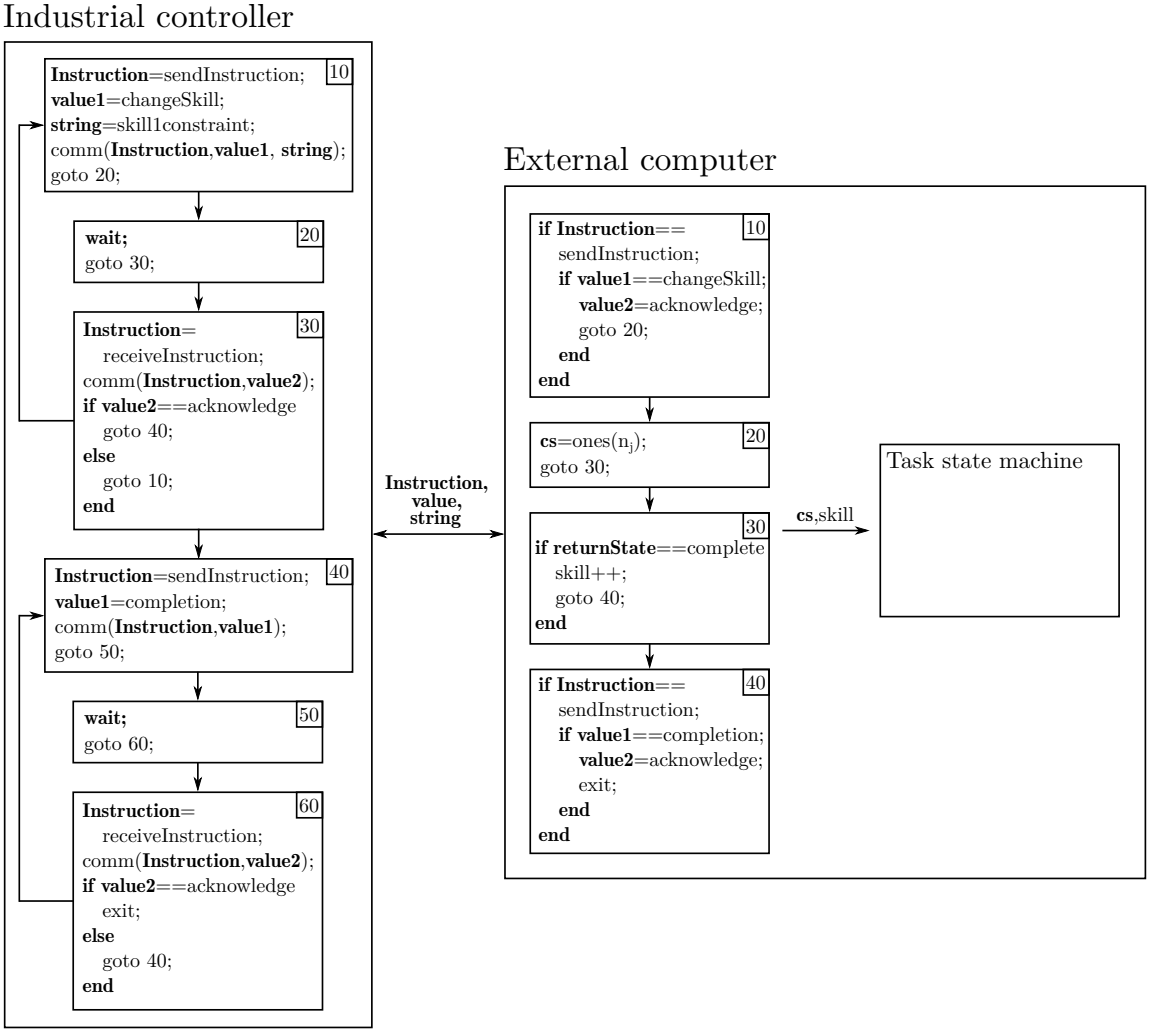


Figure 4.7: *The state machine representing the protocol for skill transition.*

In the following, the RAPID implementation of the skill transition protocol is shown.

```
IF robTransReq THEN
   mcgdata.value6 := changeCall;
   mcgdata.string1 := constType;

   WHILE TRUE DO
      MocGenInstr mcgInstrCall, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO TRANSITION;
      ENDIF
   ENDWHILE

   TRANSITION:

   mcgdata.value6 := waitCall;
   WHILE TRUE DO
      MocGenInstr mcgInstrWait, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrWaitReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO WAITDONE;
      ENDIF
   ENDWHILE

   WAITDONE:

   robTransAll := TRUE;
   robTransReq := FALSE;

   transArm := 0.0;
ENDIF
```

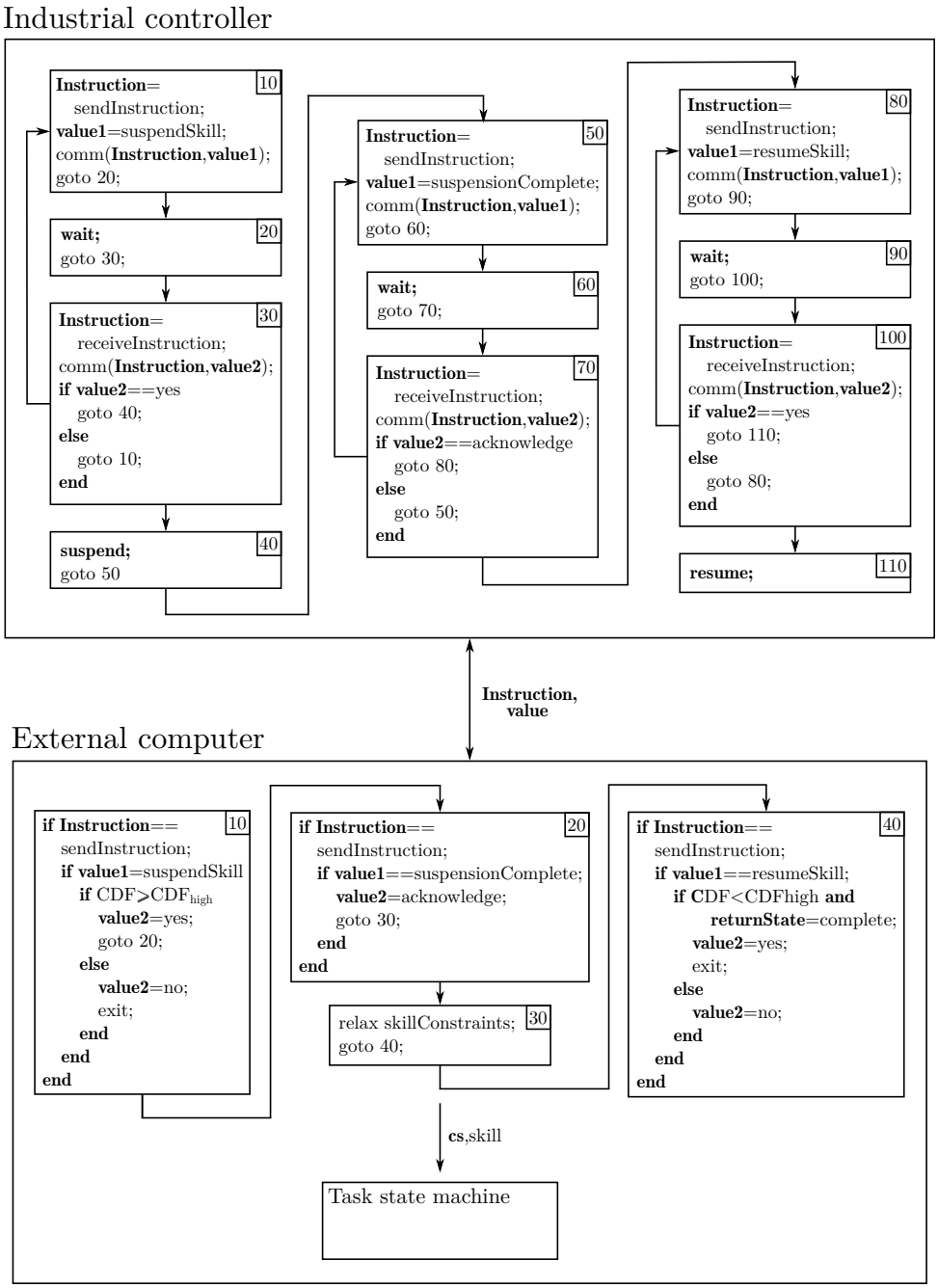


Figure 4.8: *The state machine representing the protocol for skill suspension and resumption*

4.4.4 Skill suspension and resumption

If the level of measured danger exceeds CDF_{high}, *skill* constraints, if any, are relaxed, and the skill execution has to be suspended. As the task trajectory is generated by the industrial controller, skill suspension has to be performed on the industrial controller side. However, danger is evaluated by the collision avoidance strategy on the external computer. In order to determine whether the skill has to be suspended, the industrial controller thus executes a cyclic polling of the collision avoidance system. If danger exceeds the highest threshold, skill execution is suspended and *skill* constraints are relaxed.
The following operations are therefore performed for skill suspension:

- the industrial controller cyclically polls the collision avoidance strategy, asking whether the skill has to be suspended;
- upon positive answer of the collision avoidance system, skill execution is suspended;
- after skill suspension, the collision avoidance system is allowed to relax *skill* constraints.

As danger level decreases below CDF_{high}, resumption of skill execution can take place. In order to perform such an operation, the industrial controller polls the external computer and resumes skill execution depending on the safety system answer.
Skill resumption takes place according to the following steps:

- as danger falls below CDF_{high}, the collision avoidance system activates the return trajectories to enforce *skill* constraints;
- when return trajectories for the *skill* constraints have been completed, the external computer communicates the possibility to resume execution to the industrial controller;
- skill execution is resumed.

The implementation of the protocol in RAPID is shown in the following section:

```
mcgdata.value6 := pollingCall;
MocGenInstr mcgInstrCall, mcgdata; WaitTime 0.02;
MocGenInstr mcgInstrReceive, mcgdata;

IF mcgdata.value5 = 1.0 THEN
   SpeedRefresh 20;
   StopMove;
```

```
   StorePath;
   ClearPath;
   StartMove;
   WHILE TRUE DO
      mcgdata.value6 := stopCall;
      MocGenInstr mcgInstrCall, mcgdata; WaitTime 0.02;
      MocGenInstr mcgInstrCallReceive, mcgdata;
      IF mcgdata.value5 = 1.0 THEN
         Wait;
         GOTO RESUME;
      ENDIF
   ENDWHILE
   RESUME:
   RestoPath;
   SpeedRefresh 100;
   StartMove;
   WHILE TRUE DO
      mcgdata.value6 := 5.0;
      MocGenInstr mcgInstrCall, mcgdata; WaitTime 0.02;
      MocGenInstr mcgInstrCallReceive, mcgdata;
      GOTO ENDSUSP;
   ENDWHILE
   ENDSUSP:
ENDIF
```

Fig. 4.8 shows the protocol for skill suspension and resumption.

4.4.5 Deactivation of the collision avoidance system

The last functionality included in the communication system is the deactivation of the collision avoidance system. When a task has been completed, safety functionalities are deactivated and the external control system stops the modification of industrial controller set points.
The following operations are therefore performed:

- evasive velocities are deactivated and return trajectories are started;
- when return trajectories have been completed, the collision avoidance system is deactivated and industrial controller set points are no longer modified.

Once deactivation has been completed, the industrial controller starts operating in a configuration which is equivalent to the closed controller architecture. Fig. 4.9 shows the protocol for deactivation of the collision avoidance strategy. In the following, RAPID implementation of the deactivation functionality is shown.

```
IF robTaskCompl THEN
mcgdata.value6 := deactivateCall;
   WHILE TRUE DO
      MocGenInstr mcgInstrCall, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO DEACTIVATION;
      ENDIF
   ENDWHILE

   DEACTIVATION:

   mcgdata.value6 := waitCall;
   WHILE TRUE DO
      MocGenInstr mcgInstrWait, mcgdata;
      WaitTime 0.02;
      MocGenInstr mcgInstrWaitReceive, mcgdata;

      IF mcgdata.value5 = 1.0 THEN
         GOTO WAITDONE;
      ENDIF
   ENDWHILE

   WAITDONE:

taskStart := FALSE;
ENDIF
```

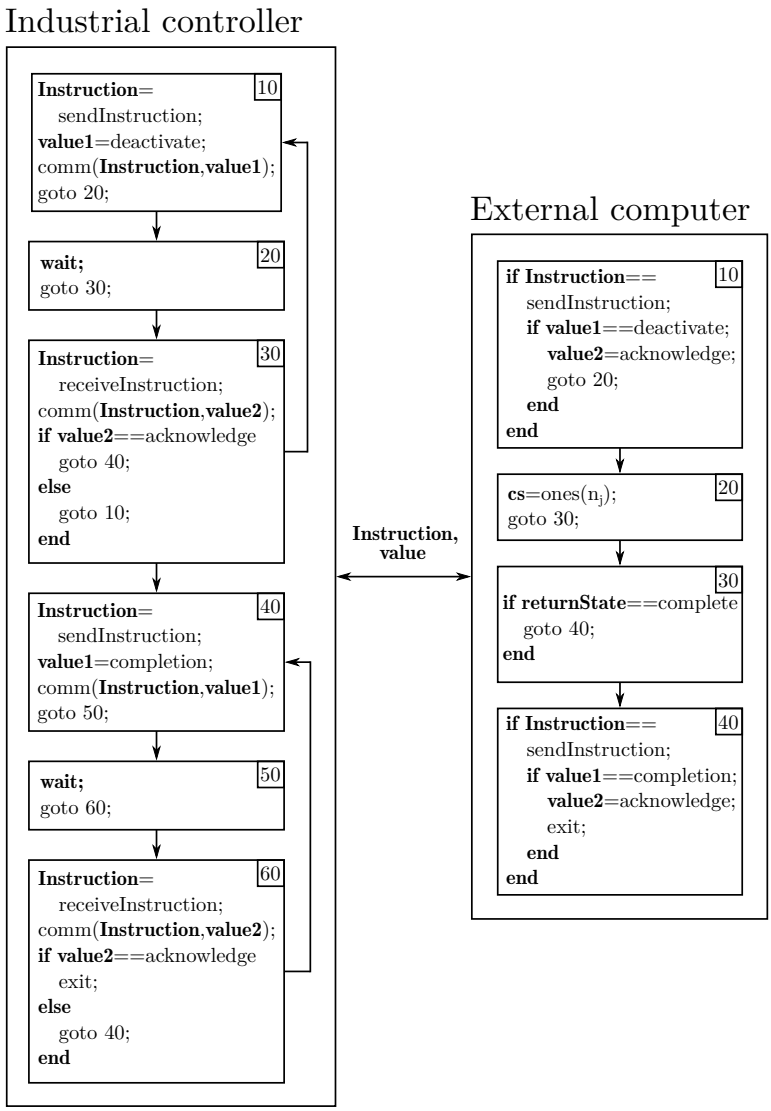


Figure 4.9: *The state machine representing the protocol for the collision avoidance system deactivation.*

4.5 System performance and limitations

The open controller architecture establishes communication channels, between the industrial controller and the external computer, whose update frequency is 250 Hz. Such a frequency is sufficient to provide robot axes set points to the axes computer. Nonetheless, the structure of the RAPID language introduces a significant limitation to the communication frequency. In fact, the concurrent execution of a motion instruction and of the commu-

nication with the external computer, can be performed only using an interrupt-activated subroutine. While in the main routine the motion instruction is being executed, upon activation of the interrupt, the subroutine and thus communication are performed. However, as the RAPID language limits interrupt-activated subroutines execution frequency to 10 Hz, polling of the external computer is limited to such a frequency.

4.6 Summary

In this chapter, the architecture for the integration of the collision avoidance system presented in Chapter 3 with an industrial controller has been presented. The industrial controller adopted for the system and the open controller architecture used for the integration have been introduced. Then, the general communication protocol has been presented, and the specific communication functions, adopted for the control of the collision avoidance system, have been detailed. The actual implementation of such functions has been shown, and exemplified by means of a simple task. Finally, system limitations have been analysed.

CHAPTER 5

Collision avoidance strategy based on projection in the null space of the task

5.1 Introduction

In Chapter 3 a strategy for the execution of evasive velocities consistent with task constraints has been presented. Such a strategy defines a general method for the relaxation of constraints forming a preplanned task and for management of the transition between different enforced constraints. No details have been given so far neither on the execution of evasive velocities, exploiting relaxed constraints, nor on the planning of return trajectories. In this chapter, the first approach adopted to manage such aspects is analysed, detailing the following parts:

- execution of evasive velocities consistently with enforced constraints;
- computation of return trajectories;
- definition of the overall robot control strategy.

Moreover, such an approach is experimentally validated on a dual arm assembly task, introducing a distributed distance sensor for obstacle detection.

5.2 Collision avoidance system

5.2.1 Projection of evasive velocities in the null space of enforced constraints

The *CSM* defines the vector $\mathbf{cs}$, whose elements correspond to the status of each coordinate constraint. Given the set of coordinates which can be used to evade from detected obstacles, the evasive velocities $\dot{\mathbf{q}}_{ev}$ can be projected in the null space of the remaining enforced constraints. The $\mathbf{cs}$ vector is thus used to define the null space projector of the velocities. Given

$$n_c = \sum_{i=1}^{n_j} cs_i \tag{5.1}$$

the number of enforced constraints, we define the vector of the indexes of the enforced constraints $\mathbf{ci}$, with the following algorithm:

$\mathbf{ci} = \{ci_i : \ i = 1, ..., n_c\}$;
k=1;

for j=1 to n_j **do**
  if cs_j==1 **then**
    ci_k=j;
    k=k+1;
  end if
end for

The elements of $\mathbf{ci}$ are therefore the indexes of the elements of $\mathbf{cs}$, corresponding to enforced constraints. The selection matrix $\mathbf{\Sigma}_{cs}$ can now be defined:

$$\mathbf{\Sigma}_{cs} = \{\sigma_{i,j} : \ i = 1, ..., n_c, j = 1, ..., n_j\}, \text{where } \sigma_{i,j} = \begin{cases} 1, \text{ if } j = \mathbf{ci}(i) \\ 0, \text{ otherwise} \end{cases}, \tag{5.2}$$

which is used to extract the rows corresponding to constrained coordinates from the robot Jacobian. We can now introduce the selected constraints Jacobian:

$$\mathbf{J}_{cs}(\mathbf{q}) = \mathbf{\Sigma}_{cs}\mathbf{J}(\mathbf{q}), \tag{5.3}$$

where $\mathbf{J}(\mathbf{q})$ is the robot TCP Jacobian. The projector in the null space of the constrained coordinates $\mathbf{N}_{cs}(\mathbf{q})$ can finally be defined as:

$$\mathbf{N}_{cs}(\mathbf{q}) = \mathbf{I} - \mathbf{J}_{cs}(\mathbf{q})^{\dagger}\mathbf{J}_{cs}(\mathbf{q}) \tag{5.4}$$

where † denotes the Moore-Penrose pseudoinverse matrix. The projected evasive joint velocities are obtained:

$$\dot{\mathbf{q}}_{ev,cs} = \mathbf{N}_{cs}(\mathbf{q})\dot{\mathbf{q}}_{ev}. \tag{5.5}$$

The computed velocities can be added to those used to perform the task, without influencing the execution of enforced constraints. The projected evasive velocities will be used to compose the overall robot velocity set points, as will be detailed later on in the chapter.

5.2.2 Return trajectories computation

As introduced in the previous chapters, the relaxation of coordinates constraints, and the execution of evasive velocities, causes the robot pose to drift from the industrial controller reference. If a constraint is enforced after its relaxation, the respective coordinate value may differ from the robot controller reference one. To avoid discontinuities in the robot commands, in Chapter 3 the use of a return trajectory, to smoothly connect the current coordinate value and the reference one, has been proposed.

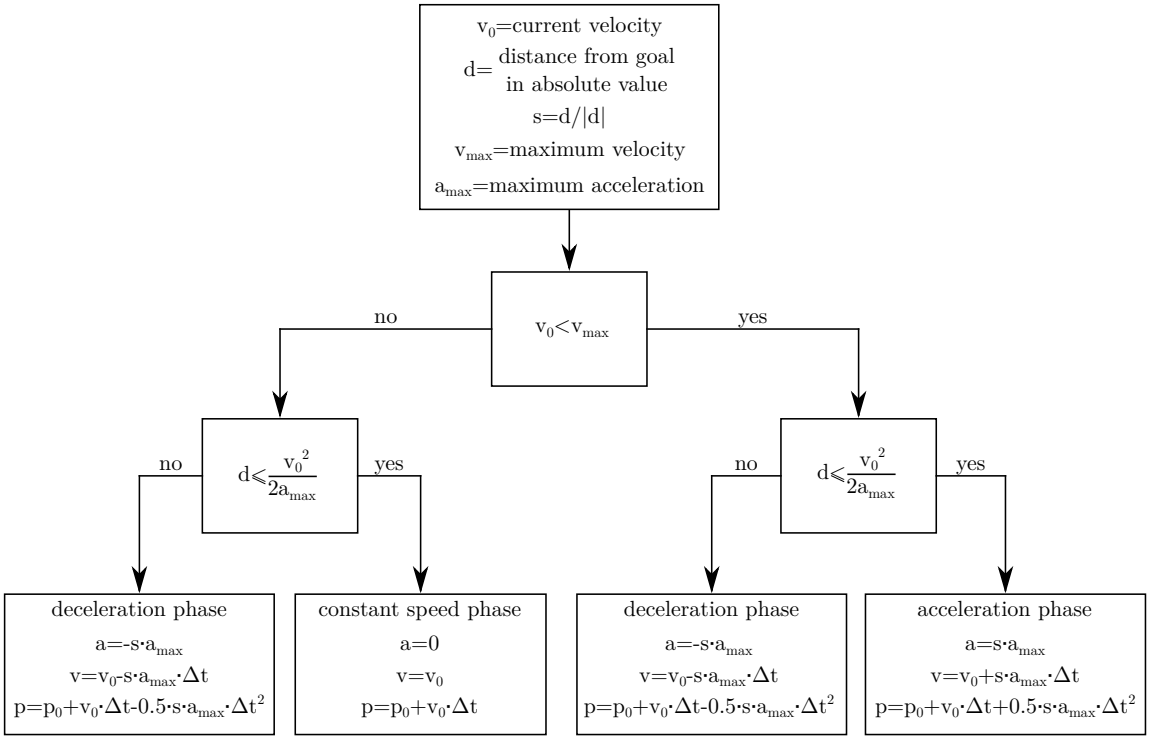


Figure 5.1: *The decision tree, to be executed at every time step, for the computation of the return trajectory.*

As already explained, the return trajectory goal is a priori not constant. In fact, the return trajectory tracks the industrial controller reference, which can evolve during trajectory execution. The return trajectory has therefore to be continuously updated to cope with the changing goal.
A simple solution for the generation of such a trajectory is to use a bang-bang profile for the acceleration, as it is done in trapezoidal and triangular velocity profiles. However, in this case, the changing trajectory reference will shape the trajectory profile. The trajectory generation can be based on the decision tree depicted in Fig. 5.1, where, at every time step, based on the current position and velocity and on the current goal, the maximum or minimum or null acceleration is applied. Fig. 5.2 shows an example of trajectory computation.

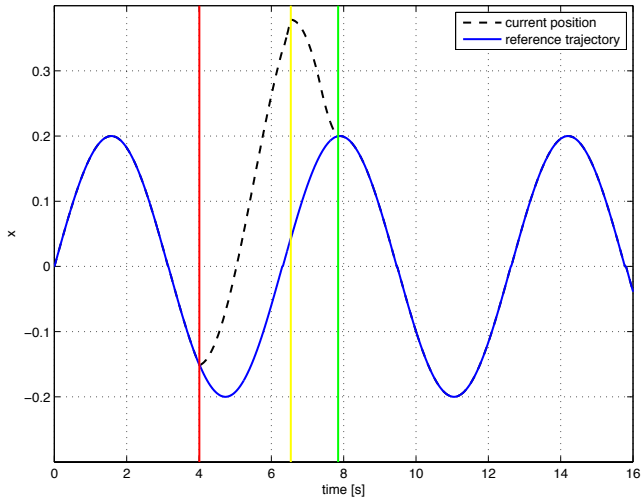


Figure 5.2: *An example of computation of a return trajectory with the presented algorithm. The solid blue curve represents the reference position, and the black dashed one the current position. The first vertical line from the left denotes the instant of the coordinate relaxation. The second vertical line denotes the instant of the coordinate enforcement, and the last vertical line to the right the completion of the return trajectory.*

A return trajectory is generated for each coordinate, and the maximum velocity and acceleration can be assigned independently for each of them. As the trajectories reference is a priori continuously changing, synchronisation of goal reaching for the different coordinates is not pursued, favouring the fastest possible completion of the single trajectories.

5.2.3 Overall control law

As already mentioned in the previous chapters, the collision avoidance system modifies the robot joints position and velocity set points, that are generated by the industrial controller

main computer. The introduced projected evasive velocities and return trajectories are used to compose the modified robot set points. Depending on the current state of the collision avoidance system, three different laws for the computation of robot set points can be identified.

No set points modification

When measured danger is below the CDF_{low} threshold, and in case no return trajectory is being executed, robot set points computed by the industrial controller are sent to the axes computer without modification. The following are thus assigned to the robot set points:

$$\dot{\mathbf{q}}_{ref}(t) = \dot{\mathbf{q}}_{ref,ic}(t), \tag{5.6}$$

$$\mathbf{q}_{ref}(t) = \mathbf{q}_{ref,ic}(t), \tag{5.7}$$

where $\mathbf{q}_{ref}$ and $\dot{\mathbf{q}}_{ref}$ are the robot joints position and velocity set points.

Execution of evasive velocities

The second possible case for the computation of robot joint set points is the execution of evasive actions. As already mentioned, when a group of constraints is relaxed, evasive velocities can be projected in the null space of remaining enforced constraints. Robot set points could therefore be computed as follows:

$$\dot{\mathbf{q}}_{ref}(t) = \dot{\mathbf{q}}_{ref,ic}(t) + \dot{\mathbf{q}}_{ev,cs}(t), \tag{5.8}$$

$$\mathbf{q}_{ref}(t) = \mathbf{q}_{t_0} + \int_{t_0}^{t} \dot{\mathbf{q}}_{ref}(\tau)\,\mathrm{d}\tau. \tag{5.9}$$

However, the industrial controller joint velocity references are computed to enforce all of the task constraints, even the relaxed ones. The obtained joint references thus enforce the sum of the evasive velocities and of the task velocities on the relaxed constraints. In order to properly execute the evasive velocities, the relaxed constraints contribution has to be removed from the industrial controller reference.
As the industrial controller references are available only in the joint space, in order to extract the part corresponding to the enforced constraints, a transformation in the operational space is necessary. We introduce:

$$\dot{\mathbf{x}}_{cs,ic}(t) = \mathbf{J}_{cs}(\mathbf{q})\dot{\mathbf{q}}_{ref,ic}(t), \tag{5.10}$$

the vector of the industrial controller reference for the constrained operational space coordinates. New robot set points can be computed as follows:

$$\dot{\mathbf{q}}_{ref}(t) = \mathbf{J}_{cs}(\mathbf{q})^{\dagger}\dot{\mathbf{x}}_{cs,ic}(t) + \dot{\mathbf{q}}_{ev,cs}(t), \tag{5.11}$$

$$\mathbf{q}_{ref}(t) = \mathbf{q}_{t_0} + \int_{t_0}^{t} \dot{\mathbf{q}}_{ref}(\tau)\,\mathrm{d}\tau. \tag{5.12}$$

The so obtained robot joints position and velocity references are derived from the part of the industrial controller references corresponding to enforced constraints, and from the projection of evasive velocities in the null space of such constraints. It has therefore to be noted that they differ from the ones computed by the external controller.
The robot set points have to be computed for discrete time steps and, moreover, the linearisation introduced in the Jacobian computation introduces an error, which has to be compensated by a closed loop algorithm. Considering a time step of size Δt, the following algorithm is thus adopted:

$$\dot{\mathbf{q}}_{ref,k+1} = \mathbf{J}_{cs}(\mathbf{q}_k)^{\dagger}\left(\dot{\mathbf{x}}_{cs,ic} + \mathbf{K}\mathbf{e}_{cs,ic,k}\right) + \dot{\mathbf{q}}_{ev,cs,k}, \tag{5.13}$$

$$\mathbf{q}_{ref,k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_{ref,k+1}\cdot\Delta t, \tag{5.14}$$

where $\mathbf{e}_{cs,ic,k}$ is the kinematic error with respect to the industrial controller reference for the constrained coordinates in the operational space, at time step k, and $\mathbf{K}$ is the error gain matrix.

Execution of return trajectories

The last case to be considered for the computation of robot joints set points, is the execution of return trajectories. In such a condition, constrained coordinates can be assigned the industrial controller reference or the return trajectory reference, based on the return trajectory status. Let us introduce the vector $\mathbf{rs}$, which determines the status of the return trajectories:

$$\begin{aligned}&\mathbf{rs} = \{rs_i :\ i = 1, ..., n_c\},\\ &\text{where } rs_i = \begin{cases}1, \text{ if a return trajectory is active on the i-th constrained coordinate}\\ 0, \text{ otherwise}\end{cases}.\end{aligned} \tag{5.15}$$

We also introduce the vector of return trajectories reference $\dot{\mathbf{x}}_{ret}$:

$$\dot{\mathbf{x}}_{ret} = \{\dot{x}_{ret,i} :\ i = 1, ..., n_j\}, \tag{5.16}$$

Whose elements are the return trajectories computed for the constrained coordinates. The reference for the constrained coordinates can then be defined as:

$$\dot{\mathbf{x}}_{cs} = \{\dot{x}_{cs,i} : \ i = 1, ..., n_c\}\text{, where } \dot{x}_{cs,i} = \begin{cases} \dot{x}_{cs,ic,i}\text{, if } rs_i\text{=0} \\ \dot{x}_{ret,i}\text{, otherwise} \end{cases},$$

and $\dot{x}_{cs,ic,i}$ is the i-th element of $\dot{\mathbf{x}}_{cs,ic}$. The robot set points can finally be defined as:

$$\dot{\mathbf{q}}_{ref}(t) = \mathbf{J}_{cs}(\mathbf{q})^{\dagger}\dot{\mathbf{x}}_{cs}(t) + \dot{\mathbf{q}}_{ev,cs}(t), \tag{5.17}$$

$$\mathbf{q}_{ref}(t) = \mathbf{q}_{t_0} + \int_{t_0}^{t} \dot{\mathbf{q}}_{ref}(\tau)\, \mathrm{d}\tau. \tag{5.18}$$

Similarly to the two previous cases, a discrete time closed loop algorithm is adopted to compute the robot commands:

$$\dot{\mathbf{q}}_{ref,k+1} = \mathbf{J}_{cs}(\mathbf{q}_k)^{\dagger}\left(\dot{\mathbf{x}}_{cs} + \mathbf{K}\mathbf{e}_{cs,k}\right) + \dot{\mathbf{q}}_{ev,cs,k}, \tag{5.19}$$

$$\mathbf{q}_{ref,k+1} = \mathbf{q}_k + \dot{\mathbf{q}}_{ref,k+1} \cdot \Delta t. \tag{5.20}$$

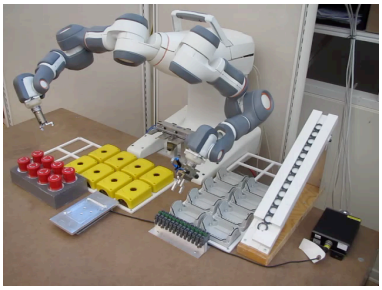
where $\mathbf{e}_{cs,k}$ is the new kinematic error.

5.3 Experimental validation

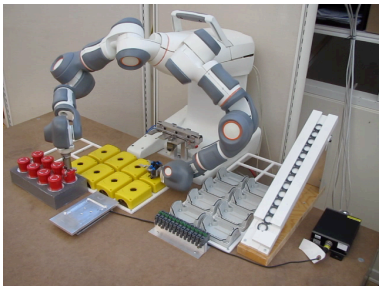
The collision avoidance strategy introduced in this chapter has been experimentally validated on a robotic assembly task. Such a task has been presented in [92], [93] and [64], and comprises both position and force controlled operations. During the assembly operation, an ABB FRIDA dual arm industrial robot prototype builds an emergency stop button, and exploits both force measurement and force estimation to perform the operations requiring force control. The assembly procedure is composed by four main phases:

- the button is inserted in the device main case;
- a nut is fastened on the button thread in order to secure it to the case;
- an electric switch is snapped on the button chassis;
- the button case is mounted on the chassis.

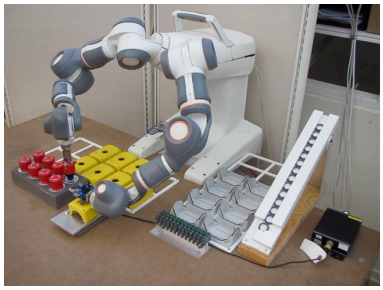
Snapshots of the assembly operation are shown in Fig. 5.3.



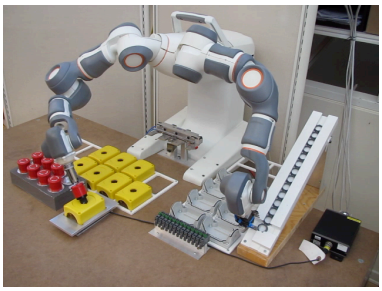
(a) The task initial state.



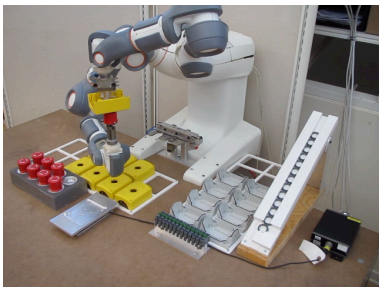
(b) Red button and yellow case picking.



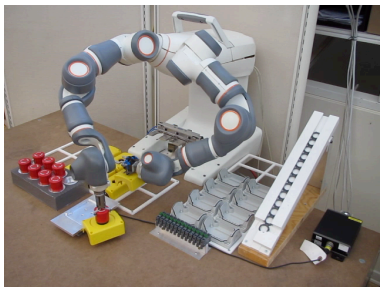
(c) Positioning of the yellow case on the assembly fixture. A force sensor placed underneath the fixture is used for the operation.



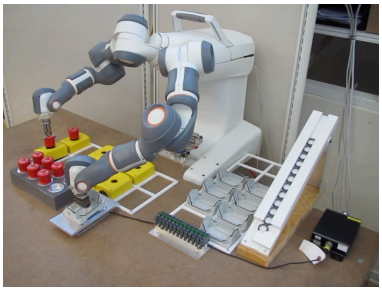
(d) Insertion of the red button in the yellow case, using the force sensor.



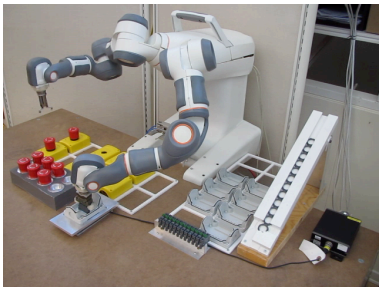
(e) Nut screwing on the button thread, using force estimation.



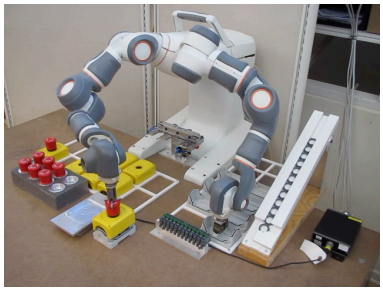
(f) Yellow case alignment using the force sensor.



(g) Positioning of the button chassis on the assembly fixture, using force sensing.



(h) Snap fit of the button switch on the chassis.



(i) Assembly finalisation.

Figure 5.3: *Snapshots of the assembly task. Courtesy of Andreas Stolt, University of Lund.*

5.3.1 System architecture

The force controlled assembly task is performed using the already mentioned open configuration of an ABB IRC5 industrial robot controller, which has been created using the interface presented in [10]. Position controlled operations are performed using the industrial controller, and are programmed with the proprietary language, while force controlled operations are performed by an external computer. In the implementation of the assembly task, the two systems therefore operate in a mutually exclusive manner.

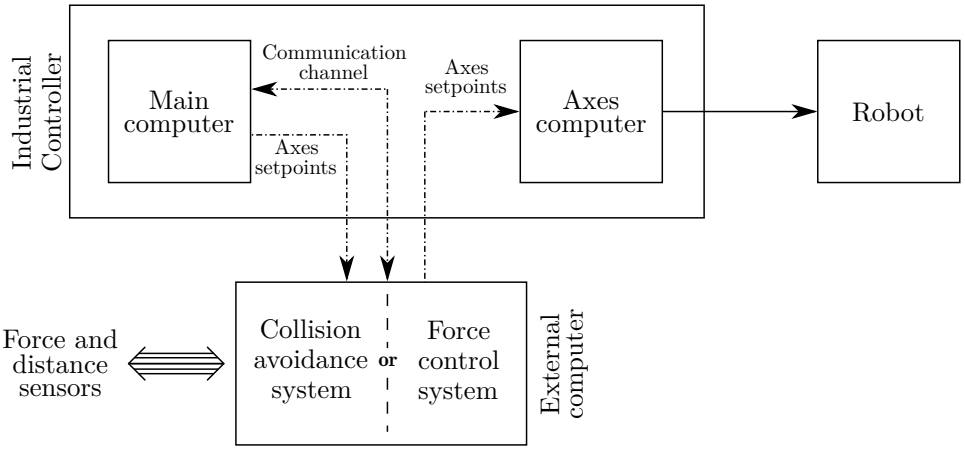


Figure 5.4: *The system architecture of the integration between the collision avoidance system and the force control system.*

The dual arm assembly task is composed of force controlled and position controlled operations, but the collision avoidance system presented in this chapter has been integrated with the latter. In fact, our collision avoidance system is designed to be integrated with the industrial controller, but force controlled skills are executed only by the external computer. Moreover, due to a low accuracy robot calibration, performing evasions without influencing force sensing and estimation is problematic. In the overall architecture, depicted in Fig. 5.4, the external computer is thus used to execute either the force control strategy or the collision avoidance system, during position controlled operations. Remarkably, and differently from previous implementation, the collision avoidance system works concurrently with the motion generation performed by the industrial controller. Finally, due to limitations in the communication system imposed by the considered case study, the collision avoidance system has been applied to the robot right arm alone. Nonetheless, when such specific limitations are removed, the system can be easily applied to the two arms.

5.3.2 Sensor system

The collision avoidance strategy is based on detection of obstacles surrounding the robot. However, up to now, such a strategy has been presented independently from the sensor system adopted to accomplish such a task. As a matter of fact, the sensor choice is crucial, as its characteristics determine the possibility to accomplish the collision avoidance system goal, that is enabling robot deployment in unstructured environments. In addition to basic performance requirements, such as sensor range and resolution, the sensor has therefore to be easily deployable in a changing environment. Onboard sensors and in particular distributed distance sensors are particularly fit for such a scenario. Such a sensor can be composed of a multitude of sensing spots or by a continuous sensing device, and is applied directly on the surface of the robot. As the sensor is applied on the robot, no structuring of the working environment is needed for its application, and self occlusion problems are prevented by the sensor placement. Moreover, the field of view is omnidirectional, and offers a high coverage of the robot workspace.

Previous works related to distributed distance sensors

The distributed distance sensor concept was first introduced and applied to an industrial robot in [19], where an implementation based on ad-hoc infrared distance sensors was proposed. A deeper insight at the sensor implementation was given by the same authors in [20], and the sensor concept was applied to motion planning and teleoperation in [21] and [66]. The same authors finally tackled real time collision avoidance using their distributed distance sensor implementation in [67], and an interesting insight at the concept of distributed sensing was given in [68]. A different implementation of the distributed distance sensor, based on capacitance, for obstacle avoidance, was first presented in [75]. However, capacitance based distance sensors offer low spatial resolution when compared to most optical ones, and their measurements highly depend on the detected object material. Both sensor types have been used in many works, among which one of the most recent is [84]. We contributed to the development of distributed distance sensors by tackling the problem of sensor sizing and disposition in [18]. A reactive collision avoidance strategy, based on a distributed distance sensor, was also introduced in [15].

Proposed sensor implementation

A distributed distance sensor prototype has been created to provide the collision avoidance system with adequate obstacle sensing, adopting a multi-spot configuration. Such a device has been designed to meet the following objectives:

- high sensor portability;
- reduced sensor bulk and no reduction of robot workspace;

- low cost.

High sensor portability The proposed collision avoidance system has been conceived as an expansion of the standard functionalities of an industrial controller. Likewise, the sensor system for obstacle detection has been designed to be added to an existing robot. For this reason, a sensor structure which can be applied directly on the robot surface, and fastened to it without the need of modifications, has been built. The distributed sensor main body, from now on *shell*, is composed by two halves, which embrace the robot arm and are connected by standard fittings. The sensor can thus be easily mounted or removed from the robot, as an additional component.

Reduced sensor bulk and no reduction of robot workspace The application of the sensor on the robot surface increases the robot size, with the possible side effect of limiting its movements. The sensor has therefore to be as small as possible, in order to avoid the self collision of parts of the arm during robot movements. Depending on the adopted measurement principle, sensor size can vary significantly. Time of flight sensor are often used in human-robot interaction applications, and as components of safety rated protective systems (see for example [97] and [6]). However, deploying a multitude of time of flight sensors on a robot surface is a problematic choice, due to their high cost and to the significant size of the optical parts. Using a single light source and measurement unit, optically connected with distributed sensing spots, is an interesting concept to achieve size reduction, which nevertheless could not be transformed into a working device yet. An alternative solution is offered by infrared distance sensors: such devices are commercially available in cheap implementations, with measuring ranges of approximately 1 meter, that are particularly suited for close cooperation with humans, sensing resolutions of few centimeters and with sensor sizes of few or less than one centimeter (see [86]). Infrared distance sensors have been chosen for the distributed sensor implementation. One downside is the current unavailability of safety rated infrared distance sensors of the mentioned size. However, for the time being, such devices are sufficient to develop the safety system functionalities, and could be replaced in the future with safety rated ones, as soon as they become available.

Low cost As one of the goals of the collision avoidance system is to reduce the costs related to the structuring of the environment, the sensor system has to be low cost, in order not to void the economic advantages offered by the human-robot collaboration configuration. The selected measurement principle allows limiting the overall sensor system cost.

The distributed sensor mounted on one robot arm is shown in Fig. 5.5. The sensor is composed of two shells, that host the distance measurement spots.

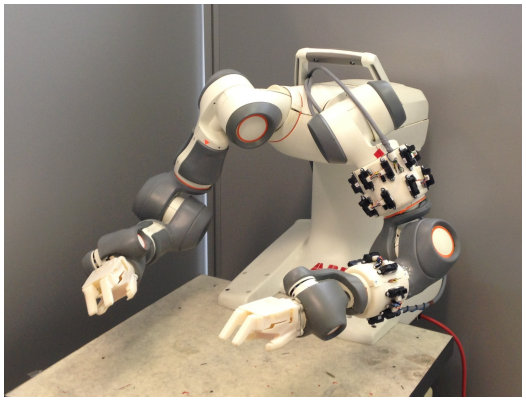


Figure 5.5: *The distributed distance sensor mounted on the ABB FRIDA dual arm robot prototype.*

Sensor sizing

One of the most important performance indexes, for a multi spot distributed distance sensor, is the minimum size of the detectable object. Since measurement is not performed by a continuous sensing surface, the discrete sensing spots generate a discontinuous coverage of the environment. Uncovered portions of the environment may obviously contain obstacles, which the sensor would fail to detect. It is therefore necessary to design the sensor in such a way that detection of obstacles of the desired minimum size is guaranteed.
In [18] we presented a method for sensor sizing, based on detection requirements, which is summarised in the following. Let us consider the configuration depicted in Figs. 5.6(a) and 5.6(b), where a cylindrical sensor shell detects an obstacle, whose surface we will assume to be perpendicular to the shell radius. We introduce the following parameters:

δ_α	angular spacing between sensor spots	$w_{obj} = \lvert\overline{AB}\rvert$	object width
δ_l	longitudinal spacing between sensor spots	l_{obj}	object length
n_{pL}	points detected on the longitudinal direction	ζ	detection angle
n_{pC}	points detected on the circular direction	l_s	shell length
d_o	object distance	d_s	shell diameter

The objective of the sizing process is to define the number of measurement spots to be put on a shell, in order to ensure the detection of the object of minimum size. Spots will be arranged in rings, with equal angular spacing between them, and multiple rings will be arranged on the shell length, once again with equal longitudinal spacing between them. The unknown quantities to be determined are consequently n_{sc} and n_{sl}, the number of spots to be put on a sensors ring, and the number of sensor rings respectively. The equations for sensor sizing are derived from the system of equations (5.21): the first two equations of the system define the relationship between the detected object size and position, the number

of points to be detected on it and the sensor spots disposition. The second two equations instead define the number of spots to be placed on the shell, in order to satisfy the detection specifications. The sensor shell designed accordingly is depicted in Fig. 5.6(b).

$$\begin{cases} w_{obj} = 2\left(\frac{d_s}{2} + d_o\right) tan\frac{\delta_\alpha n_{pC}}{2} \\ l_{obj} = \delta_l n_{pL} \\ n_{sc} = \frac{2\pi}{\delta_\alpha} \\ n_{sl} = \frac{l_s}{\delta_l} + 1 \end{cases} \Rightarrow \begin{array}{l} n_{sc} = \dfrac{\pi}{tan^{-1}\dfrac{w_{obj}}{2\left(\frac{d_s}{2} + d_o\right)}} n_{pC} \\ n_{sl} = \dfrac{l_s}{l_{obj}} n_{pL} + 1 \end{array} \tag{5.21}$$

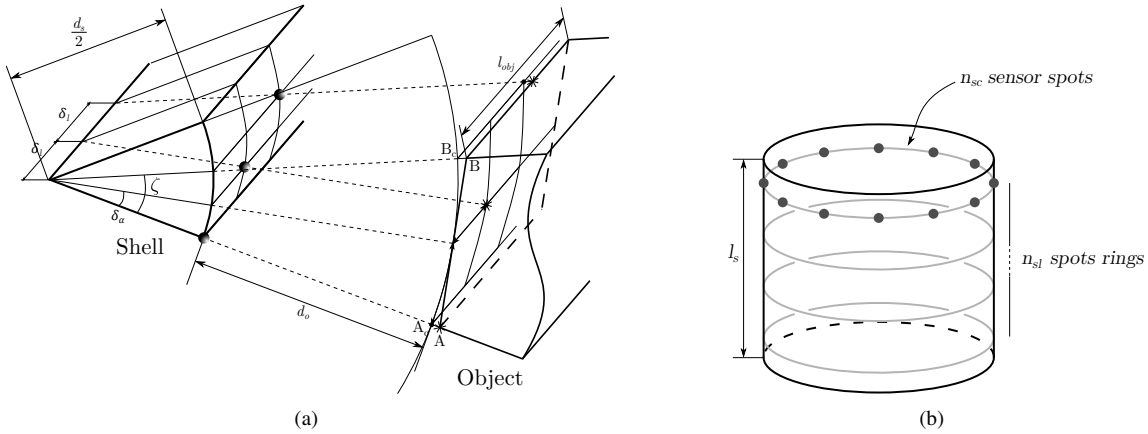


Figure 5.6: *On the left, the geometric model for sensor sizing, with an angular sector of the sensor shell and the detected object. On the right, the spots disposition obtained for the sensor shell.*

Sensor design has also to take into account the availability of adequate locations for the sensor mounting. Fig. 5.7 shows the identified possible areas for sensor placement on the ABB FRIDA robot. The sizing process has led to the design of a sensor with the specifications described in Table 5.1, whose shells are shown in Figs. 5.8(a) and 5.8(b). The selected sensor spot is a Sharp GP2Y0A21YK commercial infrared distance sensors. The specifications of the sensor spot are reported in Table 5.2.

area	n_{sc}	n_{sl}	w_{obj} [cm]	l_{obj} [cm]	d_o [cm]
B	8	2	10	10	20
			30	30	70
D	8	1	30	30	30

Table 5.1: *Specifications for the distributed sensor sizing.*

Sharp GP2Y0A21YK	
range [cm]	8 - 80
measurement frequency [Hz]	26
size L-W-H [mm]	29.5-18.9-15.5
detection area diameter at 80 cm [cm]	12

Table 5.2: *Main features of the sensor spot.*

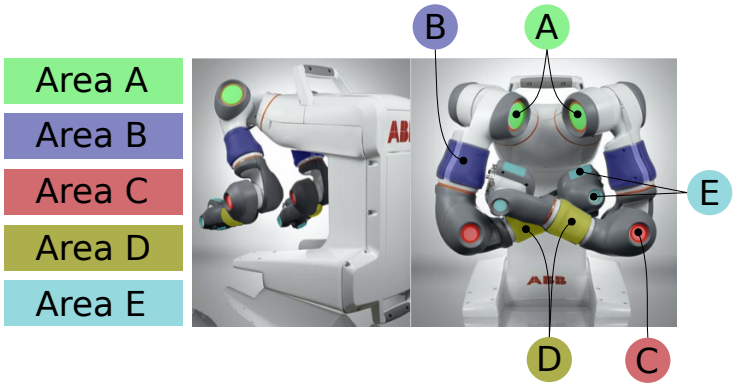
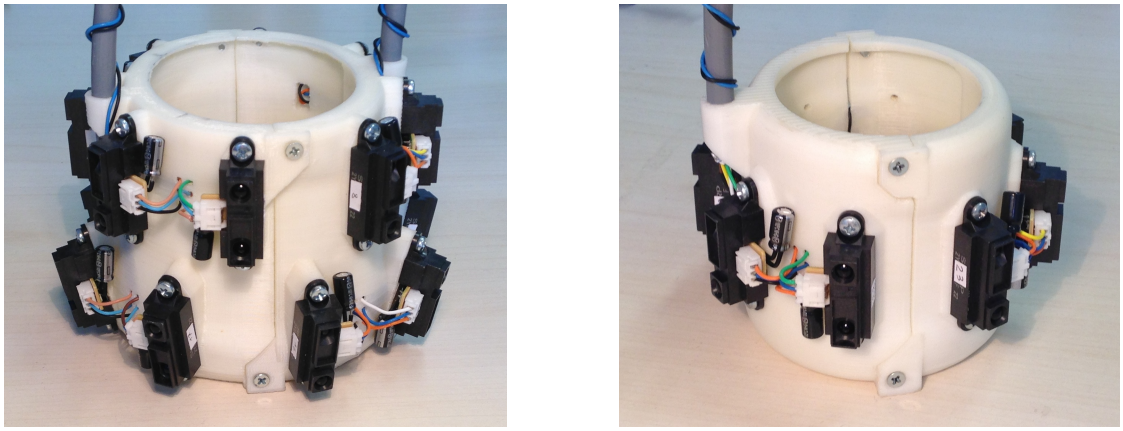


Figure 5.7: *The identified possible areas for sensor placement.*



(a) (b)

Figure 5.8: *The two sensor shells composing the distributed distance sensor prototype. On the left the upper arm shell, and on the right the lower arm one.*

Detected obstacles position reconstruction

The distributed sensor system is used to detect the obstacles surrounding the robot. Such measurements are then used to compute the danger field, and finally to perform evasive motions. In order to perform such operations, the detected obstacles position has to be computed. The procedure for position computation, presented in [15], is summarised in the following. Each sensor spot measures the distance between the detected obstacle and the sensor origin. Let us define the homogeneous position vector $\boldsymbol{p}_k$ of the k-th detected obstacle, expressed in the k-th sensor frame:

$$\boldsymbol{p}_k = [0,\ 0,\ d_k,\ 1]^T \tag{5.22}$$

where the assumption is made that the sensor measurement direction coincides with the z-axis of the sensor frame. We then define the homogeneous transformation matrix from the Denavit-Hartenberg frame of i-th link, to which the sensor is attached, to the k-th sensor frame, $\mathbf{T}_k^{DH,i}$. The detected obstacle position can be defined as:

$$\boldsymbol{r}_o = \mathbf{T}_R^w \mathbf{T}_{DH,i}^R \mathbf{T}_k^{DH,i} \boldsymbol{p}_k \tag{5.23}$$

where $\mathbf{T}_R^w$ and $\mathbf{T}_{DH,i}^R$ are the homogeneous transformation matrices from the world to the robot frame, and from the robot frame to the i-th Denavit-Hartenberg frame, respectively.

Known obstacles filtering

The distance sensor detects the objects inside its coverage, and provides the distance from the robot surface to such objects. However, no information about the type of detected object is provided. Depending on the considered scenario, moving obstacles, parts of the work cell or even the robot itself may be detected. As obstacles detection generates evasive motions, it is necessary to filter out those measurements which correspond to parts of the robot and of the work cell, in order to avoid unnecessary deviations from the preplanned trajectory. For this purpose, a measurements filtering algorithm has been proposed, whose goal is to retain only those measurements which correspond to unknown objects.
The algorithm is based on a geometric model of the robot and the work cell, and on simulation of sensor measurements using ray tracing techniques. During robot operation, sensor measurements are filtered according to the following strategy:

1. given the robot position, sensor spots positions and orientations are computed;
2. possible intersections between the spots rays, and parts of the work cell and of the robot are computed, using the environment geometric model;
3. simulated distance measurements are compared with real ones;

4. in case a real measurement corresponds to a simulated one, such a measurement is discarded, and is not considered in danger field computation.

While the geometric model of the robot adopted for measurement simulation can be designed once and for all, the one of the environment has to be adapted as the robot work cell changes. This circumstance decreases to some extent the capability of the safety system to adapt to an unknown scenario. On the other hand, such a task can be automated if a work cell simulation tool is adopted for robot programming.
Another important issue is the heavy computational load associated with computation of intersections between all of the spots rays and the environment model, in particular if a high number of spots are used. It is therefore essential to adopt an efficient representation of the environment and of the robot, in order to keep the computational load as low as possible. Fig. 5.9 shows the geometric model chosen for the robot arm, which is based on capsules and spheres.

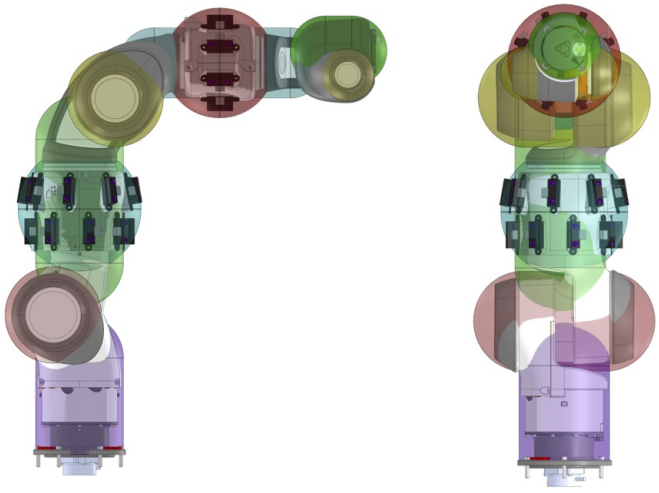


Figure 5.9: *The robot arm and the sensor CAD model, covered by the geometric model exploited to filter known obstacles.*

Despite the adoption of an efficient geometric model, satisfying the real time constraints is often impossible when the collision avoidance strategy and the known obstacles filtering are executed sequentially. Parallelisation of the execution of such tasks is however possible, and allows a more effective exploitation of the available hardware resources. In the practical application, the collision avoidance strategy and the known obstacles filtering have been split into two concurrent processes, allowing compliance with real time execution constraints.

5.3.3 Experimental results

In the following, the experimental results of the application of the collision avoidance system to the introduced assembly task are presented. A video of the experiment is available at [17]. As already mentioned, the collision avoidance system has been applied only to the robot right arm. The constraints classification presented in Chapter 2 has been applied to the task, identifying 11 different skills. The ABB FRIDA robot is composed of two 7 degrees of freedom arms: since each arm has a degree of kinematic redundancy, all types of constraints introduced in the classification will be considered. Evasive motions during the execution of two skills are analysed in the following.

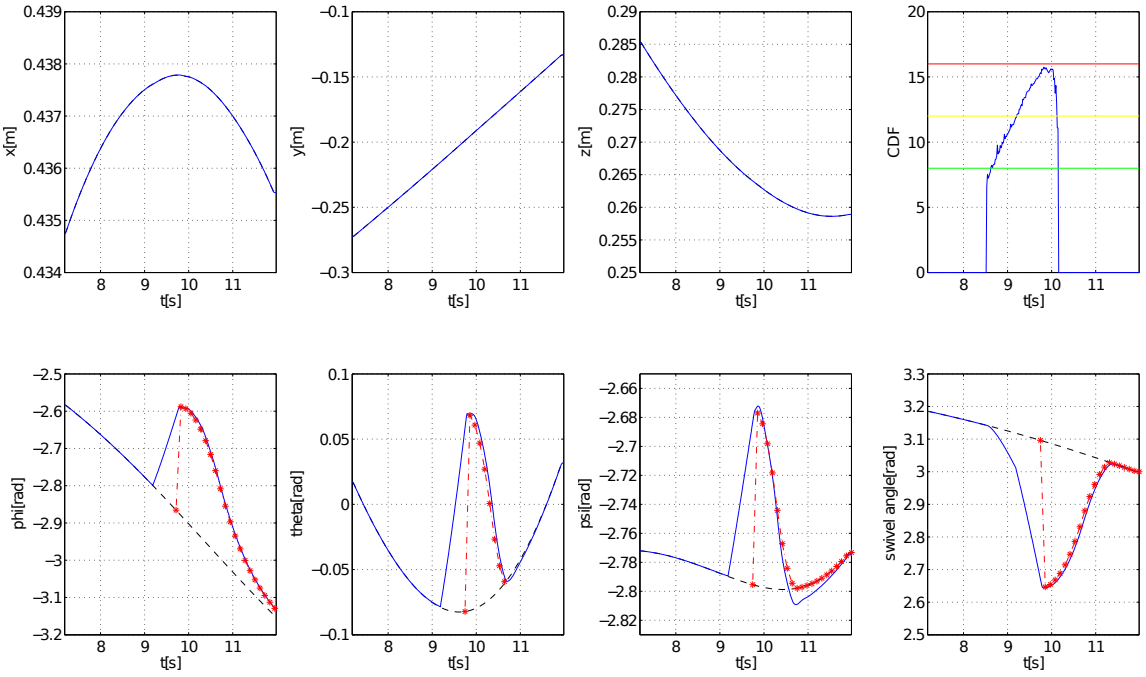


Figure 5.10: *The evolution of the robot pose during an evasive action in the first skill. Blue lines show the current robot pose, black dashed ones show the industrial controller reference pose and red dotted lines correspond to return trajectories. The top right graph shows the level of danger.*

Skill 1 - Free space movement

The first skill considered is the movement from the robot starting position to the initial position for the button picking. The right arm performs a free space movement, so the environment does not impose any constraint on its motion. Moreover, the skill being performed does not impose any restriction for its successful completion. No constraint is thus classified as *hard*. The completion of the skill implies reaching its final position, so constraints on translational velocities are classified as *skill*, since they can be temporarily

relaxed, but need to be enforced in order to complete the operation. Finally, orientation of the robot TCP can be relaxed without consequences on skill completion, and is classified as *soft*, and the kinematically redundant arm presents a *null* constraint. If three Cartesian axes are used to describe robot position, XYZ Euler angles to describe its orientation and the swivel angle sw for the kinematic redundancy, using the robot velocities vector (5.24), the selection vectors can be defined as follows:

$$\mathbf{v} = [v_x,\ v_y,\ v_z,\ \omega_x,\ \omega_y,\ \omega_z,\ sw] \tag{5.24}$$

	v_x	v_y	v_z	ω_x	ω_y	ω_z	sw
$\mathbf{s}_{null}$	0	0	0	0	0	0	1
$\mathbf{s}_{soft}$	0	0	0	1	1	1	0
$\mathbf{s}_{skill}$	1	1	1	0	0	0	0
$\mathbf{s}_{hard}$	0	0	0	0	0	0	0

Snapshots of the evasion from a human approaching the robot, during the described skill, are shown in Fig. 5.12. As the human gets closer to the robot, the danger field level increases, causing the progressive relaxation of *null* and then *soft* constraints (in the considered experiment *skill* constraints relaxation was not reached). Fig. 5.10 shows the evolution of the robot pose during skill execution.

Skill 2 - Button picking

The second considered skill is the button picking from the parts dispenser. Constraints imposed by the environment during such an operation correspond to the classical peg-in-hole configuration. In fact, the red button is allowed to rotate around the button dispenser hole axis, but can not rotate around the two axes perpendicular to the hole one, nor it can translate along such axes. Such translations and rotations are thus classified as *hard* constraints. Moreover, as the dispenser hole is blind, translation along the hole axis has to be considered as a *hard* constraint too, since the relaxation of the corresponding constraint may lead to the impact between the button and the hole bottom. Such a choice is due to a limitation of the collision avoidance system, and will be discussed later on. The remaining coordinate is the rotation around the hole axis, which corresponds to a *soft* constraint, since its relaxation has no consequences on skill execution. Using the same reference system as for the previous skill, and considering the Cartesian z axis oriented as the hole axis, the following selection vectors can be defined:

	v_x	v_y	v_z	ω_x	ω_y	ω_z	sw
$\mathbf{s}_{null}$	0	0	0	0	0	0	1
$\mathbf{s}_{soft}$	0	0	0	0	0	1	0
$\mathbf{s}_{skill}$	0	0	0	0	0	0	0
$\mathbf{s}_{hard}$	1	1	1	1	1	0	0

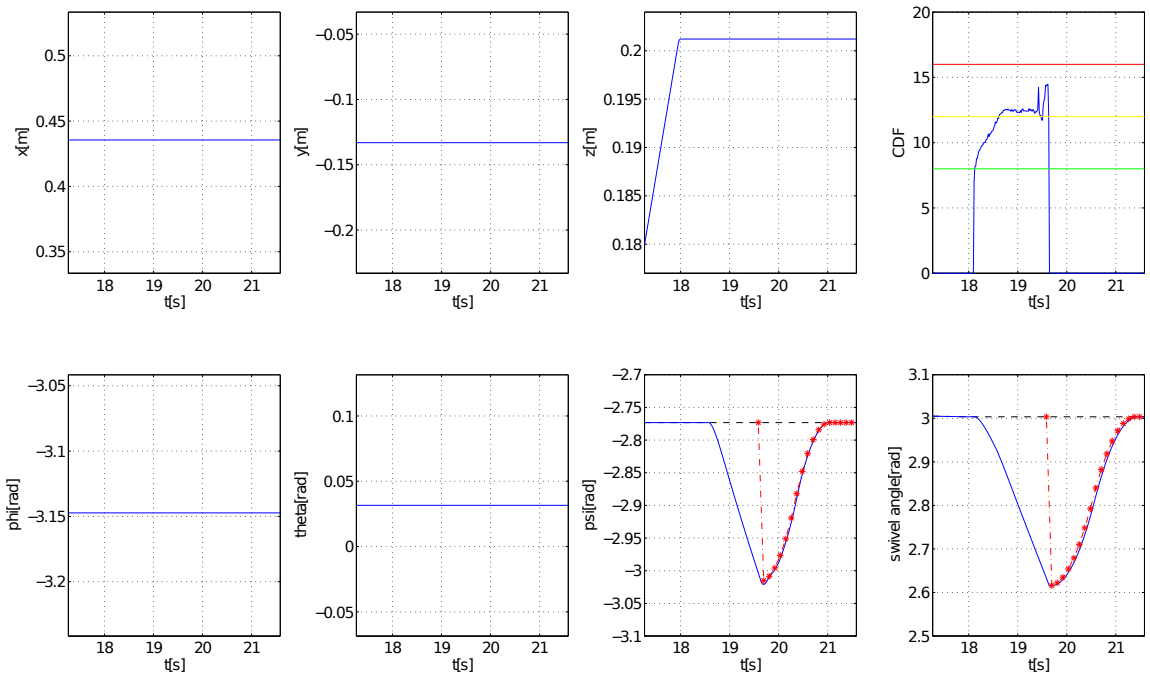


Figure 5.11: *The evolution of the robot pose during an evasive action in the second skill. Blue lines show the current robot pose, black dashed ones show the industrial controller reference pose and red dotted lines correspond to return trajectories. The top right graph shows the level of danger.*

Fig. 5.13 shows the different phases of an evasive motion performed by the robot, as a human enters the workspace. At first, swivel angle sw is relaxed. When danger exceeds the CDF_{med} threshold, orientation around the hole axis is relaxed, as it is detailed in Fig. 5.11. The collision avoidance system hence exploits such skill redundancy to perform evasion.

5.4 System limitations

The collision avoidance system introduced in this chapter, proved itself to be effective in the experimental validation. Evasive actions were performed consistently with the modelled constraints, and the task could be successfully completed even with a human operator occupying the robot workspace. However, the presented system is not able to cope with some relevant sets of constraints:

Robot kinematic constraints Kinematic constraints such as joints ranges, maximum velocities and accelerations are not taken into account, which introduces the risk of exceeding such limits during evasive actions and hindering task completion.

Task space unilateral constraints As already mentioned in the analysis of the experiment second skill, constraints such as a blind hole, which limit the robot motion unilaterally, can not be exploited for robot evasion. The only solution allowed by the introduced system is to prevent any modification of the preplanned task along such a direction, considering the corresponding constraint of *hard* type. Unilateral constraints in the task space are hence not managed by the introduced system.

In Chapter 6 a new version of the collision avoidance system will be introduced, in order to overcome such limitations.

A third limitation of the proposed system, is the inability of the strategy introduced in Section 5.2.2, for the computation of return trajectories, to take into account robot initial and final velocities. Velocity discontinuities may therefore occur when the return trajectory is executed. A different strategy for the computation of return trajectories, taking into account initial and final velocity constraints, will be proposed in Chapter 6.

5.5 Summary

In this chapter, a first control strategy for the application of the task-consistent collision avoidance strategy presented in Chapter 3 has been presented. A method for the execution of evasive velocities consistently with task constraints, and a strategy for planning return trajectories, have been detailed, deriving the overall control laws for the robot. Then, the proposed strategy has been experimentally validated on a dual arm assembly task. The distributed distance sensor prototype adopted for obstacle detection in the experimental validation has been presented. The design criteria for its sizing and placement have been detailed, and a system for filtering measurements of known obstacles has been introduced. Finally, the experimental results have been discussed, and the system limitations have been analysed.

Figure 5.12: *Snapshots taken from execution of the first skill.*

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.13: *Snapshots taken from execution of the second skill.*

CHAPTER 6

Optimisation-based collision avoidance system

6.1 Introduction

As already mentioned in Chapter 5, projection of evasive velocities in the null space of the constraints composing the task allows to effectively combine collision avoidance with the execution of a programmed operation. Nonetheless, constraints such as robot kinematic limitations and unilateral operational space constraints, can not be effectively dealt with adopting such an approach. In this chapter, the inclusion of such constraints is sketched.

6.2 QP problem for the computation of evasive velocities

Null space projection allows to limit evasive velocities only to some selected directions. When also unilateral constraints have to be taken into account in the execution of evasive velocities, such an approach is no longer effective, and a method to deal with multiple constraints of different nature is needed. Constraints should be expressed in a transparent way, defining all the degrees of freedom available to search for the solution, and deriving a problem solution which takes into account all the constraints.
Such a formulation can be achieved by expressing the execution of evasive velocities, consistently with the considered constraints, in the form of an optimisation problem. Such a problem can be for example formulated as:

$$\begin{aligned} &\min_{\dot{\mathbf{q}}} \|\dot{\mathbf{q}} - \dot{\mathbf{q}}_{ev}\|_2^2 \\ &s.t. \\ &constraints, \end{aligned} \tag{6.1}$$

where the cost function to be minimised is the squared 2-norm of the difference between the selected velocities of the robot joints, and the evasive ones (3.10). Limitations in the operational space and in the joint space can be both taken into account, and the vector of joint velocities is searched for, whose distance from the evasive one is minimal in the 2-norm.

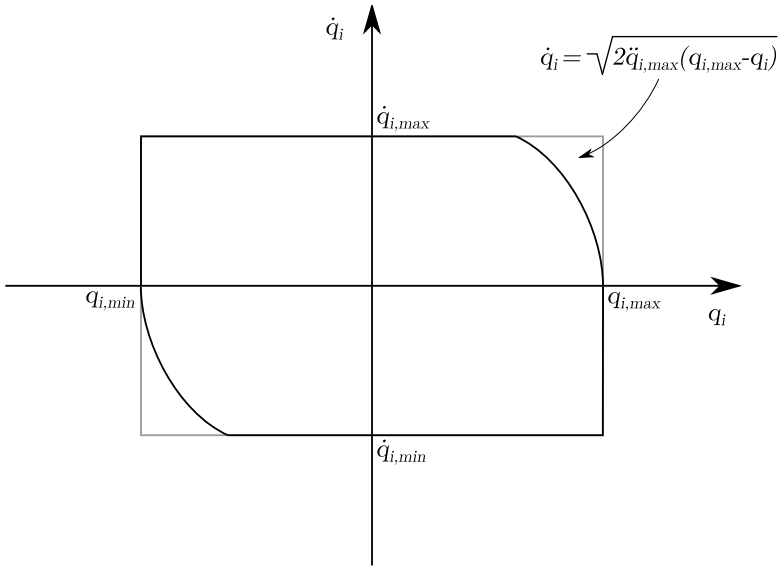


Figure 6.1: *The region of feasible robot joints states in the position-velocity plane.*

6.2.1 Management of robot kinematic constraints

The first set of constraints which have to be taken into consideration are the robot kinematic limitations. Such constraints include joint limits, maximum and minimum joint velocities and maximum and minimum joints accelerations. When solving for the robot joints velocities, the velocity reference for time step k+1 which is the closest to evasive velocities, according to the selected cost function, is the solution of the following problem:

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}_{k+1}} \|\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_{ev}\|_2^2 \\
&s.t. \\
&\mathbf{q}_{min} \leq \mathbf{q}_k + \dot{\mathbf{q}}_{k+1} \cdot \Delta t \leq \mathbf{q}_{max} \,, \\
&\dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}}_{k+1} \leq \dot{\mathbf{q}}_{max} \,, \\
&\ddot{\mathbf{q}}_{min} \leq \frac{(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k)}{\Delta t} \leq \ddot{\mathbf{q}}_{max}
\end{aligned} \tag{6.2}
$$

Given a set of maximum and minimum joint positions, velocities and accelerations, the region of feasible positions and velocities of the robot joints can be defined as depicted in Fig. 6.1. In some areas of the position-velocity plane, the limit velocity depends on the joint position: in fact, since the maximum available acceleration and deceleration are finite, when a joint gets close to the joint limit, the maximum velocity has to be bounded, taking into account the space needed to brake without hitting the limits. Moreover, in order to keep the joints states in the feasible region, depending on the current robot state, accelerations have also to be bounded, so as to guarantee each joint to remain in the feasible region not only for the current, but also for the next time step.

6.2.2 Management of operational space constraints

In Chapter 5 a case study has been presented where a unilateral constraint is imposed by the environment to the robot: a peg has to be inserted into a blind hole, which limits the peg movement unidirectionally along the hole axis. In case an evasive action has to be performed, if the degree of freedom to which the unidirectional constraint is applied has to be exploited, a strategy has to be adopted to deal with the unidirectional constraint. That is, evasive motions have to be limited in order to avoid the collision of the robot with the constraint bound. Since a relaxed constraint is considered, any trajectory can be imposed to the corresponding coordinate, as long as the unidirectional limitation is taken into account.

We introduce the following strategy:

1. a threshold before the bound of the unilateral constraint, along the direction to which such constraint is applied, is defined;
2. if the preplanned trajectory for such a direction is relaxed to perform an evasive motion, evasion is allowed as long as the robot does not exceed the above mentioned threshold;
3. if the threshold is crossed, a trajectory is planned which smoothly brings the robot to the bound of the unilateral constraint;

4. the robot is kept at the bound of the unilateral constraint until the preplanned trajectory is enforced once again, and a return trajectory brings the robot to the point of trajectory suspension.

Figure 6.2 shows an example of the application of such a strategy to a peg in hole scenario.

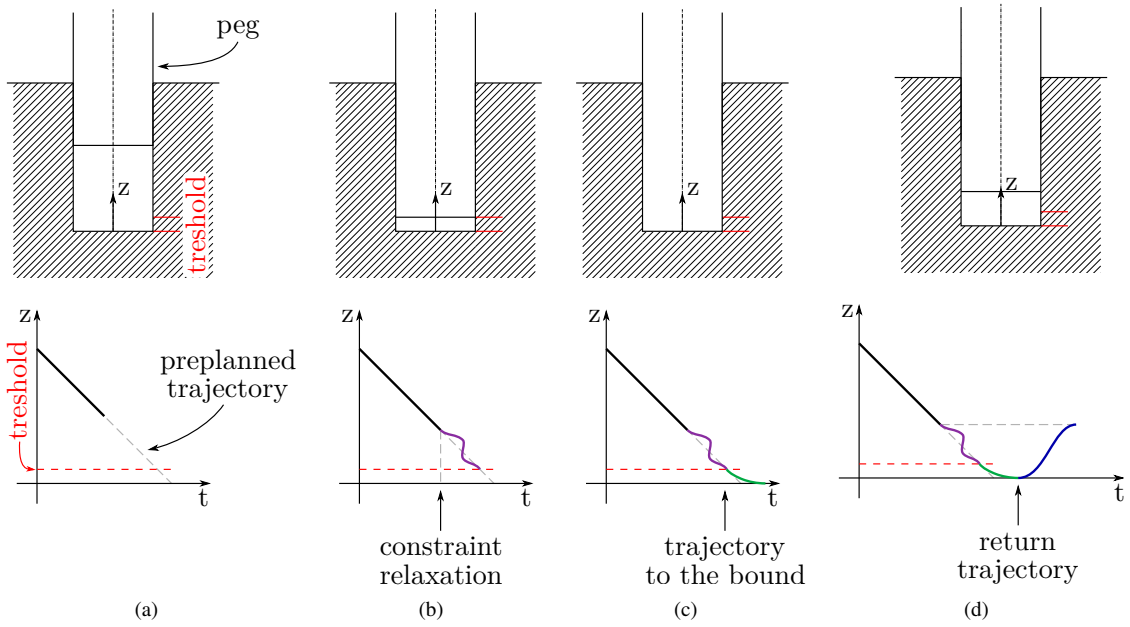


Figure 6.2: *The application of the strategy for management of unilateral constraints to a peg in hole scenario. In (a) the preplanned trajectory is being executed. In (b) the constraint applied to the z coordinate is relaxed, and the threshold is reached. In (c), after the threshold is reached, a trajectory is activated to smoothly bring the robot to the unilateral constraint bound and in (d), after the constraint is enforced once again, the return trajectory brings the robot to the point of preplanned trajectory suspension.*

Considering the optimisation problem introduced in (6.2), and extending it with the constraints applied to the unilaterally-constrained coordinate, the following problem can be formulated:

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}_{k+1}} \|\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_{ev}\|_2^2 \\
&s.t. \\
\mathbf{J}_{uni}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} &= \dot{x}_{lim,\,k+1} + \mathbf{K}e_{lim,\,k} \;\text{ if }\; (cs_{uni} == 0 \;\text{ and }\; x > x_{max} - \delta\,), \\
\mathbf{J}_{uni}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} &= \dot{x}_{ref,\,k+1} + \mathbf{K}e_{ref,\,k} \;\text{ if }\; cs_{uni} == 1, \\
\mathbf{q}_{min} \leq \mathbf{q}_k + \dot{\mathbf{q}}_{k+1} \cdot \Delta t &\leq \mathbf{q}_{max} \\
\dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}}_{k+1} &\leq \dot{\mathbf{q}}_{max} \\
\ddot{\mathbf{q}}_{min} \leq \frac{(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k)}{\Delta t} &\leq \ddot{\mathbf{q}}_{max},
\end{aligned}
\tag{6.3}
$$

where $\mathbf{J}_{uni}(\mathbf{q})$ is the Jacobian for the coordinate to which the unilateral constraint is applied, $\dot{x}_{lim}$ is the velocity reference of the trajectory used to bring the robot to the constraint bound and e_{lim} is the kinematic error with respect to such a trajectory. If the considered coordinate is relaxed, that is if cs_{uni} is equal to zero, and if the threshold δ before the unilateral constraint activation is exceeded, $\dot{x}_{lim}$ is tracked. On the other hand, if cs_{uni} is equal to one, the reference $\dot{x}_{ref}$ is tracked, and the kinematic error with respect to such reference e_{ref} is controlled.

6.2.3 General optimisation problem

Once the management of robot kinematic limitations and of unilateral operational space constraints has been discussed, the general optimisation problem for the execution of the collision avoidance strategy can be formulated. As already introduced in Chapter 5 in equation (5.19), the constraints introduced on the cartesian coordinates can be in general formulated as:

$$
\mathbf{J}_{cs}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} = \left(\dot{\mathbf{x}}_{cs} + \mathbf{K}\mathbf{e}_{cs,k}\right),
$$

where

$$
\dot{\mathbf{x}}_{cs} = \left\{\dot{x}_{cs,i} :\; i = 1, ..., n_c\right\}, \dot{x}_{cs,i} = \begin{cases} \dot{x}_{cs,ic,i}, \text{ if } rs_i\text{=0} \\ \dot{x}_{ret,i}, \text{ otherwise.} \end{cases}
$$

The general optimisation problem for the computation of the robot joints velocity set points is finally defined as:

$$
\begin{aligned}
&\min_{\dot{\mathbf{q}}_{k+1}} \|\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_{ev}\|_2^2 \\
&s.t. \\
&\mathbf{J}_{cs}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} = (\dot{\mathbf{x}}_{cs} + \mathbf{K}\mathbf{e}_{cs,k})\,, \\
&\mathbf{J}_{uni}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} = \dot{\mathbf{x}}_{lim,\,k+1} + \mathbf{K}\boldsymbol{e}_{lim,\,k} \;\text{ if }\; (cs_{uni} == 0 \;\text{ and }\; x > x_{max} - \delta\,), \\
&\mathbf{q}_{min} \leq \mathbf{q}_k + \dot{\mathbf{q}}_{k+1} \cdot \Delta t \leq \mathbf{q}_{max} \\
&\dot{\mathbf{q}}_{min} \leq \dot{\mathbf{q}}_{k+1} \leq \dot{\mathbf{q}}_{max} \\
&\ddot{\mathbf{q}}_{min} \leq \frac{(\dot{\mathbf{q}}_{k+1} - \dot{\mathbf{q}}_k)}{\Delta t} \leq \ddot{\mathbf{q}}_{max},
\end{aligned}
\tag{6.4}
$$

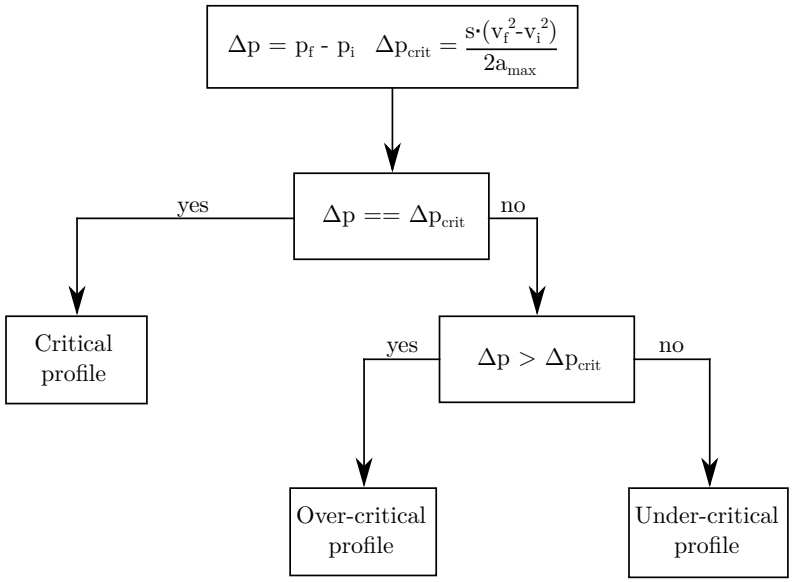


Figure 6.3: *The decision tree adopted to select the adequate return trajectory, based on the current robot state.*

6.3 Time optimal return trajectories with initial and final velocity constraints

In the previous chapter, an algorithm for the computation of return trajectories, from the current robot pose to the reference one, has been proposed. Such an algorithm is able to cope with a changing goal, that is the industrial controller reference, but only initial and final position constraints are considered. Therefore continuity in the velocity profile is not guaranteed when the trajectory is activated or the goal is reached. Such a limitation

has been overcome adopting the algorithm proposed in [82], which is summarised in the following, that guarantees time optimality of the generated return trajectory, and allows enforcing constraints also on the initial and final velocities.
The current coordinate state is defined by the following parameters:

$$\begin{array}{cc} p_i = \text{current position} & p_f = \text{goal position} \\ v_i = \text{current velocity} & v_f = \text{goal velocity} \\ \Delta v = v_f - v_i & \Delta p = p_f - p_i \\ s = \Delta v/|\Delta v| & a_{max} = \text{maximum acceleration} \\ \Delta p_{crit} = \dfrac{s(v_f^2 - v_i^2)}{2a_{max}} & \tilde{s} = sign(\Delta p - \Delta p_{crit}) \end{array}$$

based on which the type of trajectory guaranteeing time optimality is is determined, according to the decision tree depicted in Fig. 6.3.
Three different trajectory types are identified:

- **Critical profile**: if the distance to be covered is equal to the one needed to reach the final velocity at the maximum acceleration, a trajectory of minimum duration, the critical profile, is obtained;
- **Over-critical profile**: when the distance to be covered is greater than the critical one, a higher duration than the critical trajectory is obtained;
- **Under-critical profile**: when the distance to be covered is lower than the critical one, a motion in the opposite direction of the goal is needed to reach the desired final state, implying once again a higher travel time than the critical one.

Once the trajectory type has been identified, its profile can be defined. For critical profiles, the following laws are introduced for acceleration, velocity and position:

$$\begin{cases} a(t) = sa_{max} \\ v(t) = sa_{max}t + v_i \\ p(t) = \frac{1}{2}sa_{max}t^2 + v_i t + p_i \end{cases} , \quad t \in \left[0,\ \frac{s\Delta v}{a_{max}}\right]$$

For over and under critical profiles, common time laws can be adopted. Two different velocity profiles, triangular and trapezoidal, are used, based on the value of the following parameter:

$$v_p = \sqrt{\frac{1}{2}\left(v_f^2 + v_i^2\right) + \tilde{s}\Delta p a_{max}} \tag{6.5}$$

Given the maximum allowed velocity v_M, we define the profiles:

$$
\text{if } v_p \leq v_M \begin{cases}
\text{if } t \in [0,\ T_1] \begin{cases} a_1(t) = \tilde{s} a_{max} \\ v_1(t) = \tilde{s} a_{max} t + v_i \\ p_1(t) = \frac{1}{2} \tilde{s} a_{max} t^2 + v_i t + p_i \end{cases} \\
\text{if } t \in (T_1,\ t_f] \begin{cases} a_2(t) = -\tilde{s} a_{max} \\ v_2(t) = -\tilde{s} a_{max}(t - T_1) + \tilde{s} v_p \\ p_2(t) = -\frac{1}{2} \tilde{s} a_{max}(t - T_1)^2 + \tilde{s} v_p (t - T_1) + p_1(T_1) \end{cases} \\
\text{where } T_1 = \dfrac{\tilde{s} v_p - v_i}{\tilde{s} a_{max}},\ t_f = T_1 + \dfrac{v_f - \tilde{s} v_p}{-\tilde{s} a_{max}}
\end{cases}
$$

and:

$$
\text{if } v_p > v_M \begin{cases}
\text{if } t \in [0,\ T_1] \begin{cases} a_1(t) = \tilde{s} a_{max} \\ v_1(t) = \tilde{s} a_{max} t + v_i \\ p_1(t) = \frac{1}{2} \tilde{s} a_{max} t^2 + v_i t + p_i \end{cases} \\
\text{if } t \in (T_1,\ T_2] \begin{cases} a_2(t) = 0 \\ v_2(t) = \tilde{s} v_M \\ p_2(t) = \tilde{s} v_M (t - T_1) + p_1(T_1) \end{cases} \\
\text{if } t \in (T_2,\ t_f] \begin{cases} a_3(t) = -\tilde{s} a_{max} \\ v_3(t) = -\tilde{s} a_{max}(t - T_2) + \tilde{s} v_M \\ p_3(t) = -\frac{1}{2} \tilde{s} a_{max}(t - T_2)^2 + \tilde{s} v_M (t - T_2) + p_2(T_2) \end{cases} \\
\text{where } T_1 = \dfrac{\tilde{s} v_M - v_i}{\tilde{s} a_{max}},\ T_2 = \dfrac{1}{v_M} \left[\dfrac{v_f^2 + v_i^2 - 2 \tilde{s} v_M v_i}{2 a_{max}} + \tilde{s} \Delta p \right], \\
\qquad\qquad t_f = T_2 + \dfrac{v_f + \tilde{s} v_M}{\tilde{s} a_{max}}
\end{cases}
$$

Based on the introduced decision tree, at every time step the correct profile for trajectory

computation is identified. Since the profiles are defined piecewise, in order to select which part of the profile has to be considered, the discrete time step Δt is taken as the current time t, and is compared with the switching times. Finally, the new set points for the return trajectory are obtained by substituting the time step Δt.

6.4 Experimental validation on a dual arm task

The introduced implementation of the collision avoidance strategy is able to cope with kinematic and unilateral operational space limitations and, moreover, adopts an algorithm for return trajectories generation, which guarantees continuity in both position and velocity profiles. Such a strategy has been experimentally validated on a dual arm pick and place task.
During such a task, a tray is jointly lifted by the two arms of a robot, and displaced to the task final position. The task is thus composed of three skills: the approach to the tray picking position, which is independently executed by the two robot arms, the lifting and displacing of the tray, which is collaboratively performed by the two arms, and finally a motion to return to the initial position. Single and dual arm skills are thus present, so task constraints classification will have to be adapted accordingly. The task is sketched in Fig. 6.4.

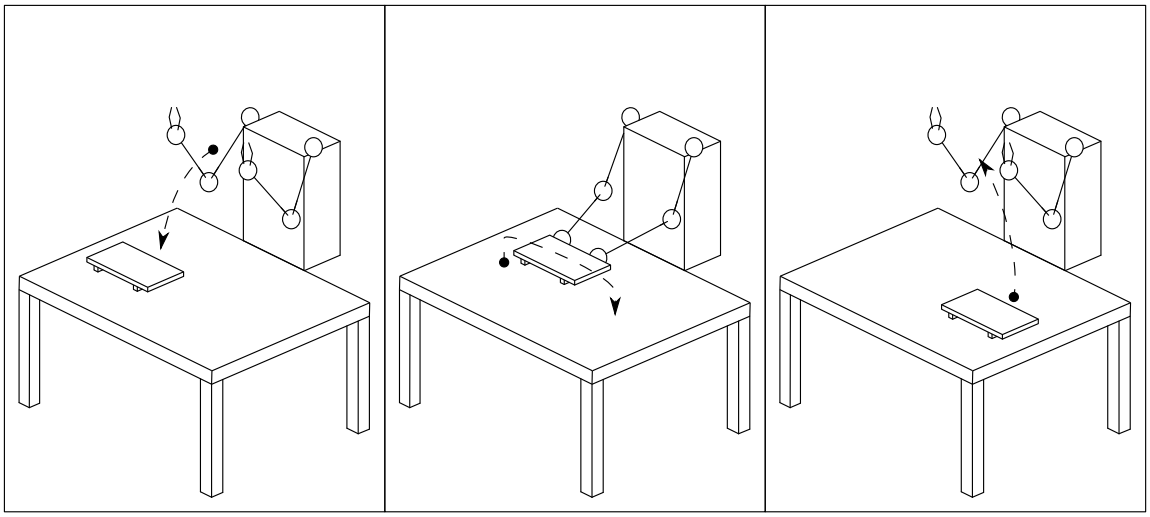


Figure 6.4: *A pictorial representation of the dual arm pick and place task adopted for experimental validation.*

The three skills introduce the following constraints:

Approach - return skills The first and the third skill are free space movements: the environment hence does not apply any constraint on the arms motion. Moreover, the skill being performed does not imply any restriction, so no *hard* constraints are present. Constraints on robot position are of *skill* type, as their relaxation implies the suspension of the operation. Orientation is not relevant for skill completion, so the corresponding constraints are of *soft* type. Finally, a *null* constraint characterises the swivel angle of the robot arms, which present one degree of kinematic redundancy each. The robot end effectors poses are described using three Cartesian axes for position and XYZ Euler angles for orientation, while the swivel angle sw is used to describe kinematic redundancy. Using the robot velocities vector (5.24), the following selection vectors are defined for both robot arms:

	v_x	v_y	v_z	ω_x	ω_y	ω_z	sw
$\mathbf{s}_{null}$	0	0	0	0	0	0	1
$\mathbf{s}_{soft}$	0	0	0	1	1	1	0
$\mathbf{s}_{skill}$	1	1	1	0	0	0	0
$\mathbf{s}_{hard}$	0	0	0	0	0	0	0

Dual arm skill During the second skill a tray with an object lying on top is moved by the two arms. In order to prevent the object from falling, the tray has to be kept on the horizontal plane: rotations around the two coordinate axes belonging to such a plane are thus *hard* constraints. As for the above mentioned skills, constraints on translation are of *skill* type, and the tray orientation around the vertical axis is a *soft* constraint. The mentioned constraints are referred to the tray pose, and not to a specific robot arm, since a joint manipulation is being performed. In order to derive a set of constraints for the two robot arms, the ones applied to the tray will be enforced on one of the two, and constraints on their relative pose will be added.
Adopting the same position and orientation descriptions as in the previous case, and considering the z axis oriented as the vertical axis, the following selection vectors are thus introduced for the left arm:

	v_x	v_y	v_z	ω_x	ω_y	ω_z	sw
$\mathbf{s}_{L,\,null}$	0	0	0	0	0	0	1
$\mathbf{s}_{L,\,soft}$	0	0	0	0	0	1	0
$\mathbf{s}_{L,\,skill}$	1	1	1	0	0	0	0
$\mathbf{s}_{L,\,hard}$	0	0	0	1	1	0	0

The arms relative pose is fixed by the jointly lifted object, so relative position and orientation are *hard* constraints. Expressing the right arm position in the left arm TCP frame, and adopting Euler angles to describe its orientation with respect to the same frame, the following selection vectors are introduced for the relative coordinates:

	v_x	v_y	v_z	ω_x	ω_y	ω_z
$\mathbf{s}_{rel,\,soft}$	0	0	0	0	0	0
$\mathbf{s}_{rel,\,skill}$	0	0	0	0	0	0
$\mathbf{s}_{rel,\,hard}$	1	1	1	1	1	1

The general optimisation problem introduced in (6.4) has to be adapted to each of the skills: in the following, the different problem formulations adopted for the single and dual arm skills are described. Evasive velocities have initially been applied to the left arm alone: such a limitation will be discussed later on in the chapter.

Single arm skills

When the two arms operate independently, the right arm is controlled by the industrial controller alone, while the collision avoidance system is applied to the left arm. The following optimisation problem is thus adopted to control the latter:

$$\begin{aligned}
&\min_{\dot{\mathbf{q}}_{L,\,k+1}} \|\dot{\mathbf{q}}_{L,\,k+1} - \dot{\mathbf{q}}_{L,\,ev}\|_2^2 \\
&s.t. \\
&\mathbf{J}_{L,\,cs}(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,\,k+1} = (\dot{\mathbf{x}}_{L,\,cs} + \mathbf{K}\mathbf{e}_{L,\,cs,\,k})\,, \\
&\mathbf{q}_{L,\,min} \leq \mathbf{q}_{L,\,k} + \dot{\mathbf{q}}_{L,\,k+1} \cdot \Delta t \leq \mathbf{q}_{L,\,max} \\
&\dot{\mathbf{q}}_{L,\,min} \leq \dot{\mathbf{q}}_{L,\,k+1} \leq \dot{\mathbf{q}}_{L,\,max} \\
&\ddot{\mathbf{q}}_{L,\,min} \leq \frac{(\dot{\mathbf{q}}_{L,\,k+1} - \dot{\mathbf{q}}_{L,\,k})}{\Delta t} \leq \ddot{\mathbf{q}}_{L,\,max},
\end{aligned} \tag{6.6}$$

where the subscript L denotes all the left arm variables and parameters. It has to be noted that the Jacobian $\mathbf{J}_L$ from which $\mathbf{J}_{L,\,cs}$ is extracted, is augmented with the swivel angle Jacobian. The last row of $\mathbf{J}_L$ thus defines swivel angle velocity given the joints ones. The same applies for $\mathbf{J}_R$, with subscript R denoting all the variables referred to the right arm.

Dual arm skill

When the dual arm skill is performed, two new sets of constraints are introduced.

Arms relative position and orientation

The joint manipulation of the same object by the two arms limits the feasible relative positions and orientations of the arms end effectors. If the object is grasped by each of the two robot end effectors, in such a way that a rigid connection is established, their relative positions and orientations are fixed by the grasping pose. During the dual arm operation, the end effectors relative pose has therefore to be kept constant.

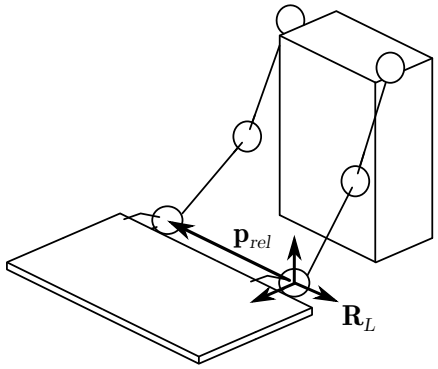


Figure 6.5: *A dual arm operation which constrains the relative pose of the two robot end effectors.*

Considering the scenario depicted in Fig. 6.5, the relative pose constraint can be expressed by imposing a zero relative velocity between the two robot end effectors, using the relative Jacobian $\mathbf{J}_{rel}(\mathbf{q})$:

$$\mathbf{J}_{rel}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{J}_{rel}(\mathbf{q}) \begin{bmatrix} \dot{\mathbf{q}}_L \\ \dot{\mathbf{q}}_R \end{bmatrix} = \mathbf{0}\,, \tag{6.7}$$

where

$$\mathbf{J}_{rel} = \left\{ \begin{bmatrix} \mathbf{R}_L^T & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{R}_L^T & 0 \end{bmatrix} \begin{bmatrix} -\mathbf{J}_L & \mathbf{J}_R \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{S}(\mathbf{p}_{rel})\mathbf{R}_L^T & 0 \\ \mathbf{0} & \mathbf{0} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{J}_L & \mathbf{0} \end{bmatrix} \right\}\,,$$

$\mathbf{R}_L$ is the left arm TCP frame rotation matrix and $\mathbf{S}(\mathbf{p}_{rel})$ is the vector product operator for the relative position vector $\mathbf{p}_{rel}$, which is expressed in the left arm TCP frame. In order to take into account the error introduced by the linearisations, a closed loop algorithm is introduced to control the relative pose error:

$$\mathbf{J}_{rel}(\mathbf{q}_k)\dot{\mathbf{q}}_{k+1} = \mathbf{K}_{rel}\mathbf{e}_{rel,k}\,, \tag{6.8}$$

where $\mathbf{e}_{rel,k}$ is the relative kinematic error at time instant k and $\mathbf{K}_{rel}$ is the gain matrix. As already mentioned, in this experimental validation, the collision avoidance strategy is applied to the left arm alone. Nonetheless, when the dual arm operation is being performed, the two arms end effectors are rigidly linked by the grasped object. If an evasive motion is performed by the left arm, and the preplanned trajectory of the arm end effector is modified, the right arm has to move accordingly, in order to comply with the relative pose constraints. The two robot arms are thus controlled in a leader-follower configuration, where the left arm is the leader. During the dual arm skill, the left arm is controlled concurrently by the industrial controller and the collision avoidance strategy, while the right arm is controlled by the collision avoidance strategy alone. The two configurations of the control system during single and dual arm skills are depicted in Fig. 6.6.

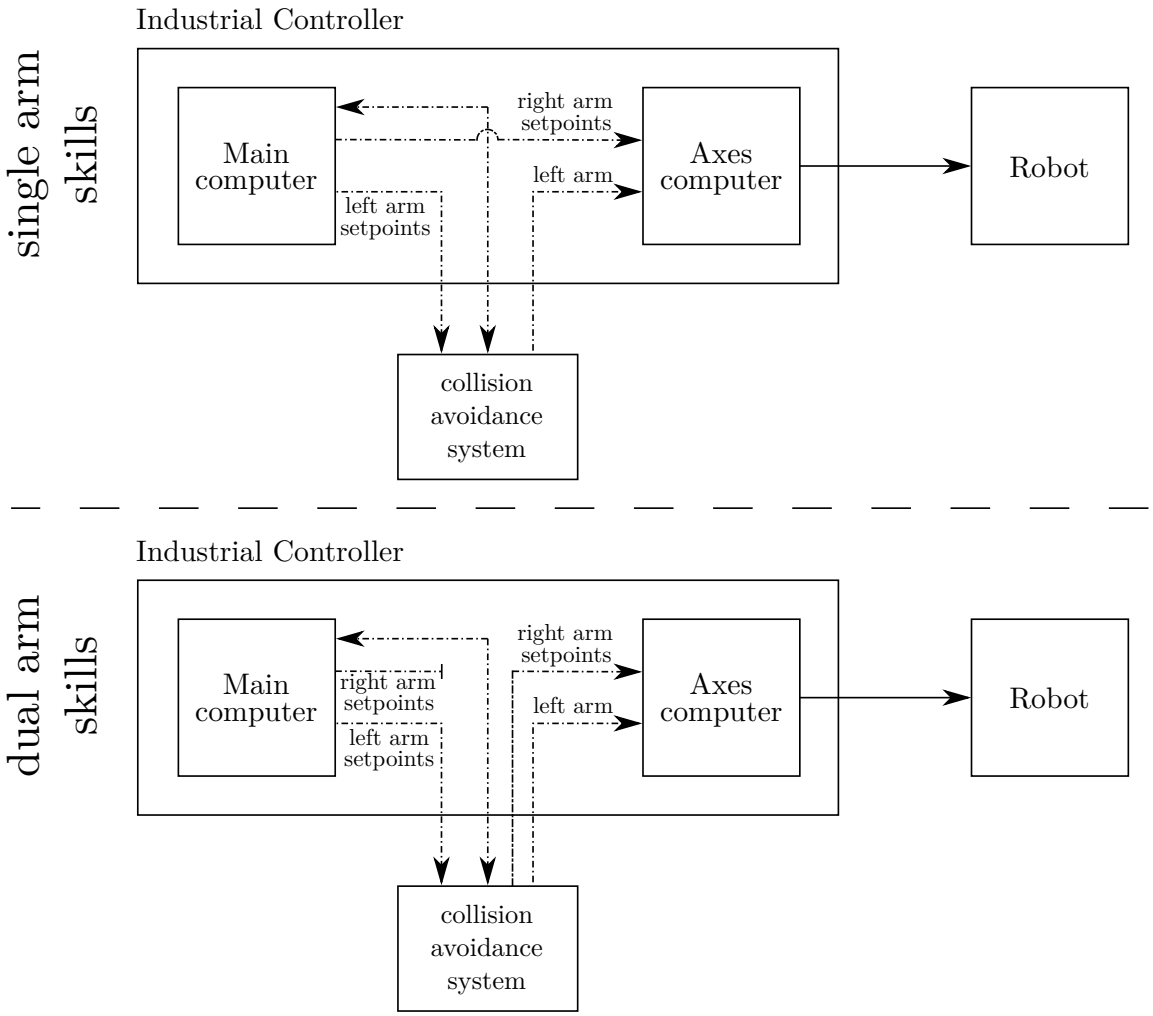


Figure 6.6: *The two configurations of the control system during single (top) and dual (bottom) arm skills. During single arm skills, no modification is performed on the right arm references computed by the industrial controller, while during dual arm skills the right arm is controlled by the collision avoidance system alone.*

Swivel angle limitation

During the dual arm skill, the two arms operate close one to another, in a configuration which implies a risk of collision between the two. On the one hand, the end effectors relative pose is constrained by the jointly grasped object, thus preventing the collision between the end parts of the two arms. On the other hand, if an evasive motion is performed relaxing the swivel angle, a collision between the intermediate parts of the arms may occur. Since in this first application of the collision avoidance strategy, evasive motions are

applied to one of the two arms, and more specifically to the robot left arm, a limit value s_{max} is thus imposed to the swivel angle of the robot left arm, and is managed as described above for unilateral constraints. The following constraint is hence added:

$$\mathbf{\Sigma}_s \mathbf{J}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,k+1} = \dot{s}_{lim,k+1} + \mathbf{K}e_{lim,k} \;\text{ if }\; (cs_{uni} == 0 \;\text{ and }\; s > s_{max} - \delta\,), \quad (6.9)$$

where $\mathbf{\Sigma}_s$ is the selection matrix extracting the row of the left arm Jacobian corresponding to the swivel angle, $\dot{s}_{lim,k}$ is the velocity at time instant k of the trajectory bringing the swivel to the limit value and $e_{lim,k}$ the error with respect to the imposed trajectory, at time instant k.
The optimisation problem for the control of the two robot arms motions, applying evasive velocities to the left arm, constraining the arms end effectors relative pose and limiting the left arm swivel angle, can be defined as:

$$\begin{aligned}
&\min_{\dot{\mathbf{q}}_{L,\,k+1}} \|\dot{\mathbf{q}}_{L,\,k+1} - \dot{\mathbf{q}}_{L,\,ev}\|_2^2 \\
&s.t. \\
&\mathbf{\Sigma}_{cs}\mathbf{J}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,k+1} = \mathbf{\Sigma}_{cs}(\dot{\mathbf{x}}_{L,k+1} + \mathbf{K}\mathbf{e}_{L,k})\,, \\
&\mathbf{J}_{rel}(\mathbf{q}_{L,\,k}, \mathbf{q}_{R,\,k}) \begin{bmatrix} \dot{\mathbf{q}}_{L,\,k+1} \\ \dot{\mathbf{q}}_{R,\,k+1} \end{bmatrix} = \mathbf{K}_{rel}\mathbf{e}_{rel,k}\,, \qquad (6.10) \\
&\mathbf{\Sigma}_s \mathbf{J}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,k+1} = \dot{s}_{lim,k+1} + \mathbf{K}e_{lim,k} \;\text{ if }\; (cs_{uni} == 0 \;\text{ and }\; s > s_{max} - \delta\,), \\
&\begin{matrix} \mathbf{q}_{min,L} \\ \mathbf{q}_{min,R} \end{matrix} \leq \begin{matrix} \mathbf{q}_{L,k} + \dot{\mathbf{q}}_{L,k+1}\cdot\Delta t \\ \mathbf{q}_{R,k} + \dot{\mathbf{q}}_{R,k+1}\cdot\Delta t \end{matrix} \leq \begin{matrix} \mathbf{q}_{max,L} \\ \mathbf{q}_{max,R}\,, \end{matrix} \\
&\begin{matrix} \dot{\mathbf{q}}_{min,L} \\ \dot{\mathbf{q}}_{min,R} \end{matrix} \leq \begin{matrix} \dot{\mathbf{q}}_{L,k+1} \\ \dot{\mathbf{q}}_{R,k+1} \end{matrix} \leq \begin{matrix} \dot{\mathbf{q}}_{max,L} \\ \dot{\mathbf{q}}_{max,R}\,, \end{matrix} \\
&\begin{matrix} \ddot{\mathbf{q}}_{min,L} \\ \ddot{\mathbf{q}}_{min,R} \end{matrix} \leq \begin{matrix} \frac{(\dot{\mathbf{q}}_{L,\,k+1} - \dot{\mathbf{q}}_{L,\,k})}{\Delta t} \\ \frac{(\dot{\mathbf{q}}_{R,\,k+1} - \dot{\mathbf{q}}_{R,\,k})}{\Delta t} \end{matrix} \leq \begin{matrix} \ddot{\mathbf{q}}_{max,L} \\ \ddot{\mathbf{q}}_{max,R}\,, \end{matrix}
\end{aligned}$$

where the subscript R denotes all the variables referred to the right arm.

6.4.1 Efficient solution of the QP problem

The introduced optimisation problem has to be solved in real time in order to compute the robot set-points. In order to satisfy the computation time constraint, an efficient formulation of the problem is necessary. The optimisation problem can in general be expressed as:

$$\begin{aligned} \min_{\mathbf{x}} \; & \frac{1}{2}\,\mathbf{x}^T\,\mathbf{Q}\,\mathbf{x} + \mathbf{f}^T\,\mathbf{x} \\ s.t. \; & \\ & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{l}_b \leq \mathbf{x} \leq \mathbf{u}_b. \end{aligned}$$

The number of rows of the $\mathbf{A}$ matrix corresponds to the number of constraints enforced on the manipulator. When no constraint is relaxed, the number of constraints is equal to the number of robot degrees of freedom, so the problem solution can be simply obtained by means of the inverse of $\mathbf{A}$:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b},$$

and evasive velocities do not influence the solution. When a subset of the preplanned trajectory constraints is relaxed, degrees of freedom become available for the minimisation of the cost function, and the optimisation problem solution is no longer trivial. In order to reduce the optimisation problem size, and thus to make its solution more efficient, the problem can be reformulated in order to make the available degrees of freedom explicit. A new problem solution $\mathbf{x}$ can be introduced:

$$\mathbf{x} = \mathbf{A}^{\dagger}\mathbf{b} + \mathbf{N}\mathbf{y}.$$

The $\mathbf{A}^{\dagger}\mathbf{b}$ term satisfies the equality constraints, while the $\mathbf{N}\mathbf{y}$ term, where $\mathbf{N}$ is the base of the null space of $\mathbf{A}$, has no influence on the above mentioned constraints. The optimisation problem can thus be solved for $\mathbf{y}$, searching the solution that minimises the cost function. The problem cost function can hence be reformulated as:

$$\begin{aligned} & \frac{1}{2}\,\mathbf{x}^T\,\mathbf{Q}\,\mathbf{x} + \mathbf{f}^T\,\mathbf{x} = \frac{1}{2}\,(\mathbf{A}^{\dagger}\mathbf{b} + \mathbf{N}\mathbf{y})^T\,\mathbf{Q}\,(\mathbf{A}^{\dagger}\mathbf{b} + \mathbf{N}\mathbf{y}) + \mathbf{f}^T\,(\mathbf{A}^{\dagger}\mathbf{b} + \mathbf{N}\mathbf{y}) = \\ & = \frac{1}{2}\mathbf{y}^T(\mathbf{N}^T\mathbf{Q}\mathbf{N})\mathbf{y} + (\mathbf{N}^T\mathbf{Q}(\mathbf{A}^{\dagger}\mathbf{b}) + \mathbf{N}^T\mathbf{f})^T\mathbf{y} = \frac{1}{2}\mathbf{y}^T(\mathbf{Q}_{new})\mathbf{y} + \mathbf{f}_{new}^T\mathbf{y}. \end{aligned}$$

The optimisation problem can be finally expressed as:

$$\begin{aligned} \min_{\mathbf{y}} \; & \frac{1}{2}\,\mathbf{y}^T\,\mathbf{Q}_{new}\,\mathbf{y} + \mathbf{f}_{new}^T\,\mathbf{y} \\ & s.t. \\ & \mathbf{N}\mathbf{y} \leq \mathbf{u}_b - \mathbf{A}^{\dagger}\mathbf{b} \\ & -\mathbf{N}\mathbf{y} \leq \mathbf{A}^{\dagger}\mathbf{b} - \mathbf{u}_b \end{aligned}$$

Such a formulation limits the problem size to the number of relaxed degrees of freedom, offering in general a size reduction, compared to the previous formulation.

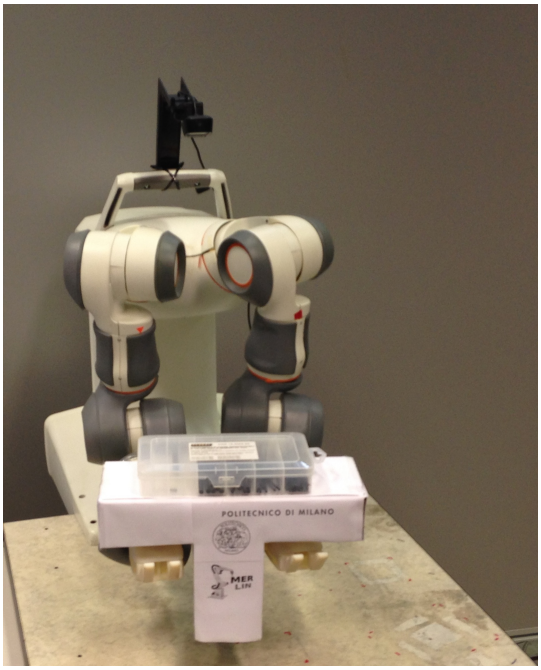


Figure 6.7: *The ABB FRIDA dual arm robot prototype lifting a tray with an object laying on it.*

6.4.2 Experimental setup

The ABB FRIDA dual arm robot prototype has been used as the test bench for the proposed collision avoidance system. The experimental setup is shown in Fig. 6.7. Compared to what proposed in the previous chapter, a different sensor system has been used, in order to evaluate possible alternative solutions for obstacle detection.
The Microsoft Kinect camera has been adopted, which is able to autonomously identify humans in its detection field, and to generate a skeleton structure reproducing their pose. Danger field computation can be performed by extracting points from the generated skeleton and considering such points as obstacles. The main advantage of the adoption of such a device, compared to the distributed distance sensor, is avoiding the computations needed to reconstruct the obstacle position from the spots measurements, which represent a significant item in the computational burden. On the other hand, the sensor field of view can be easily occluded by other obstacles or by the robot itself. Fig. 6.8 shows an example of the acquisition of a human position, and the skeleton points adopted to compute the danger field.

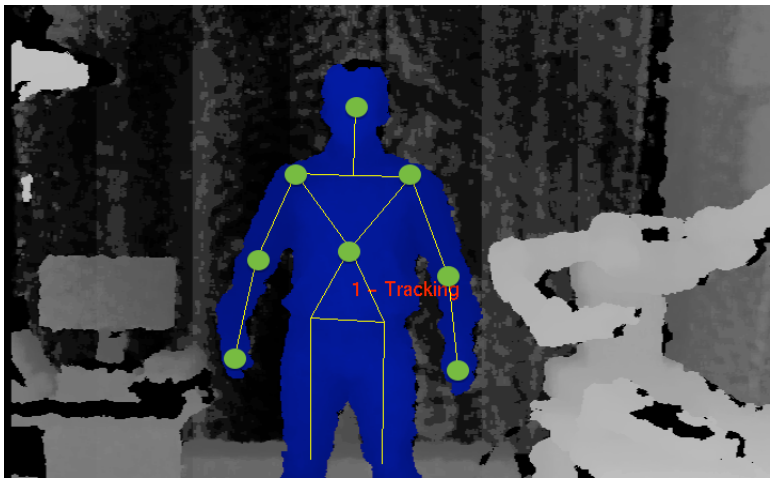


Figure 6.8: *An acquisition of the position of a human. The green points are used in the computation of the danger field.*

Two different tools have been tested to solve the optimisation problem in the real time application, IBM CPLEX and qpOASES. The latter, based on the online active set strategy (see [33], [34] and [79]) was able to guarantee feasibility of the real time solution of the introduced optimisation problem.

6.4.3 Experimental results

The standard execution of the pick and place task is shown in Fig. 6.24. The Kinect sensor covers the shown area and detects possible humans entering the workspace. Fig. 6.25 shows a human approaching the robot from its left side while the dual arm skill is being performed. As can be noticed from the snapshots, the preplanned robot trajectory is suspended and an evasive motion is performed to avoid the collision with the human. Robot evasion is detailed in Figs. 6.9 and 6.10, which show the measured danger field value during the experiment, and the evolution of robot coordinates, respectively.
As can be noticed in Fig. 6.10, as danger increases, the swivel angle, the tray orientation around the vertical axis and finally the position are progressively relaxed, in order to enforce the evasive velocities. When, during evasion, the swivel angle exceeds the introduced threshold, a trajectory is computed to smoothly bring the swivel to the limit value. Once danger decreases, swivel angle constraint is enforced once again, and a return trajectory is activated to reach the reference value of the industrial controller. Finally, Figs. 6.12, 6.13 and 6.14 show the compliance of the robot motion with kinematic limitations. A video of the experiment is available at [16].

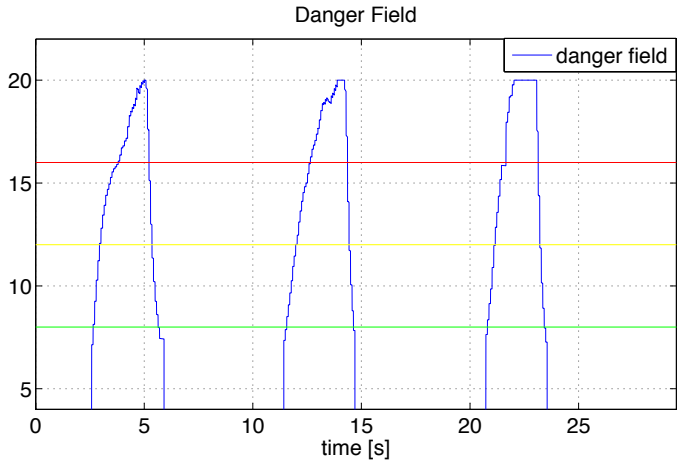


Figure 6.9: *The danger field measured during the experiment. All of the three thresholds are progressively exceeded, reaching the skill suspension.*

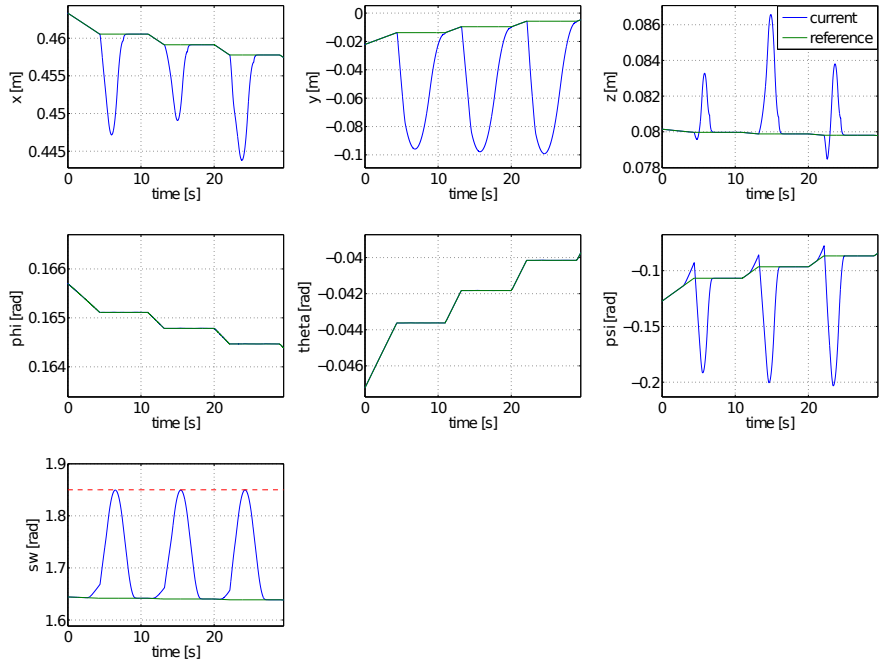


Figure 6.10: *The evolution of the robot coordinates during the experiment. As CDF_{low} is exceeded, swivel angle is relaxed. Then, when danger reaches CDF_{med}, orientation around the vertical axis is relaxed. Finally, position constraints are relaxed, causing the skill suspension, as can be noticed by the industrial controller constant reference. It has to be noted that hard constraints, corresponding to phi and theta, are never relaxed.*

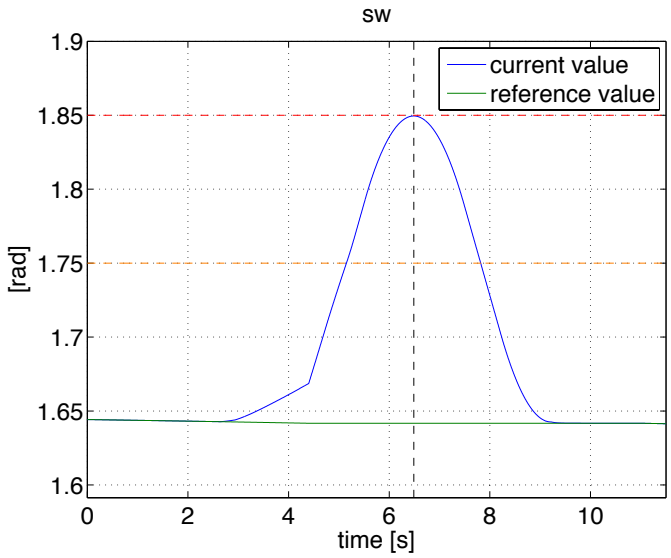


Figure 6.11: *The evolution of the swivel angle during an evasive motion. The red dashed line represents the limit value for the swivel, and the orange one represents the threshold. When the threshold is exceeded, a trajectory is activated which brings the swivel to the limit. Then, when danger decreases and the swivel constraint is enforced once again, at the point shown by the black line, the return trajectory is activated.*

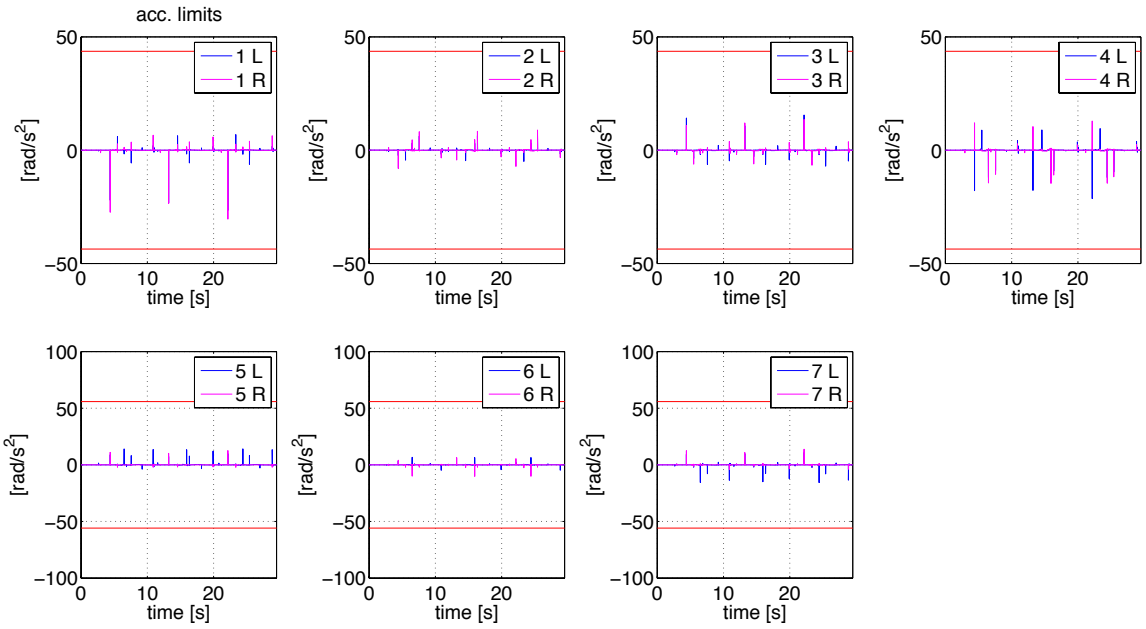


Figure 6.12: *Joint accelerations during the experiment and their respective limits.*

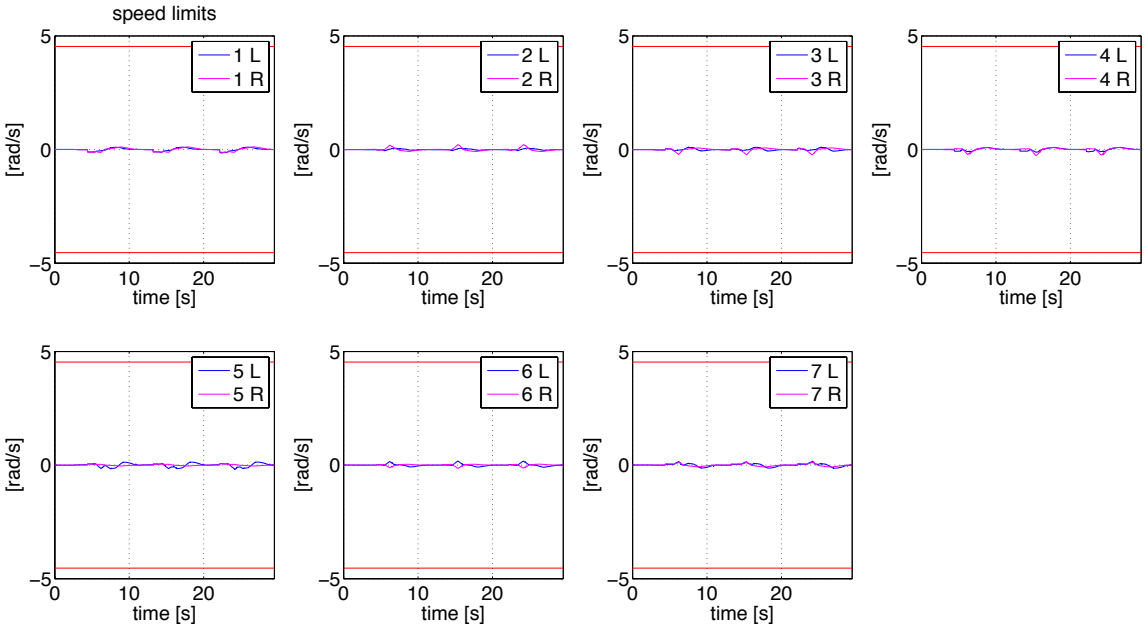


Figure 6.13: *Joint speeds during the experiment and their respective limits.*

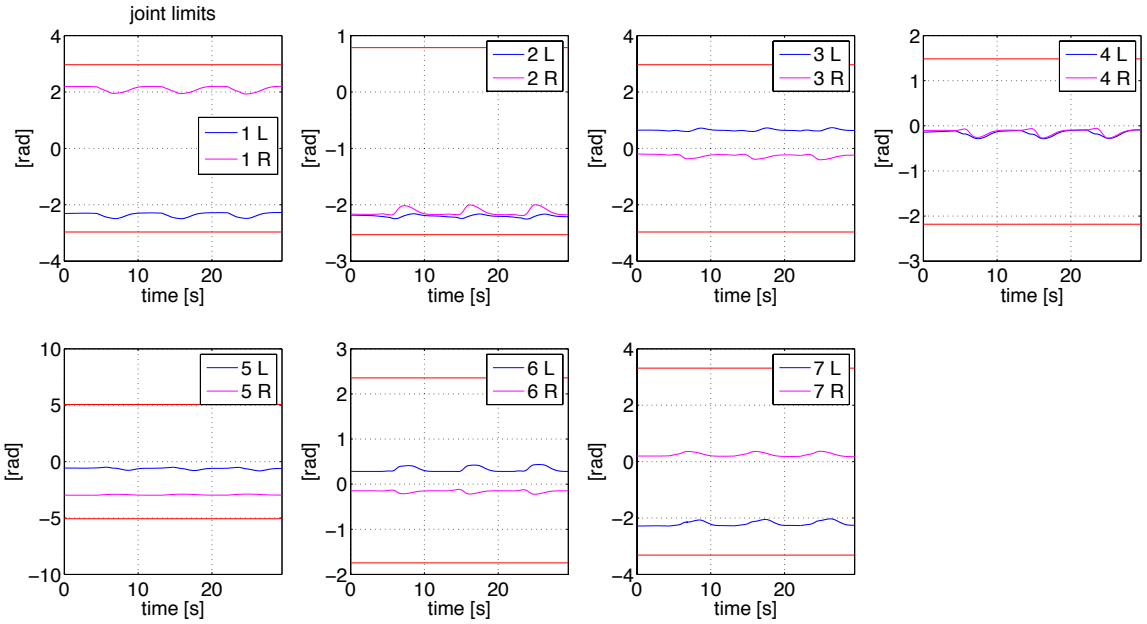


Figure 6.14: *Joint positions during the experiment and their respective limits.*

6.4.4 System limitations

The presented application of the collision avoidance system introduced in Chapter 3, is able to overcome the limitations of the approach presented in Chapter 5, as it allows to take into account unilateral constraints in the operational space and robot kinematic limitations. Moreover, the new algorithm adopted to plan return trajectories guarantees continuity in the trajectories velocity profile.
Nonetheless, the introduced system presents two main imitations: firstly, evasive actions are applied only to one of the two arms, reducing the possibilities of the system to cope with obstacles coming from any direction. Secondly, when considering constraints on joints accelerations, the use of a first order CLIK algorithm, introduces an approximation in the computation of robot accelerations.
In the next section, a system coping with such limitations is presented.

6.5 Application of evasive motions to the two arms

In order to allow the robot to effectively evade from obstacles coming from all directions, it is necessary to compute the danger field, and consequently evasive velocities, for both robot arms. Each arm will hence perform evasive motions depending on danger induced on obstacles by itself. In the optimisation problem, the distance between robots joints velocities and computed evasive velocities for the two arms will be considered as the cost function to be minimised.
As already mentioned, a second order CLIK algorithm will be adopted, guaranteeing the absence of approximations in the computation of accelerations. The optimisation problem will be thus solved for joints acceleration, and the following formulation will be introduced for the cost function:

$$\min_{\ddot{\mathbf{q}}_{LR,\,k+1}} \|\dot{\mathbf{q}}_{LR,\,k+1} - \dot{\mathbf{q}}_{LR,\,ev}\|_2^2 = \min_{\ddot{\mathbf{q}}_{LR,\,k+1}} \|\dot{\mathbf{q}}_{LR,\,k} + \Delta t\, \ddot{\mathbf{q}}_{LR,\,k+1} - \dot{\mathbf{q}}_{LR,ev}\|_2^2$$

where

$$\mathbf{q}_{LR,\,k} = \left[\mathbf{q}_{L,\,k}^T\,,\, \mathbf{q}_{R,\,k}^T\right]^T \text{ and } \dot{\mathbf{q}}_{LR,\,ev} = \left[\dot{\mathbf{q}}_{L,\,ev}^T\,,\, \dot{\mathbf{q}}_{R,\,ev}^T\right]^T.$$

The optimisation problems for the single and dual arm robot skills can thus be reformulated. For the single arm skills the following problem will be obtained:

$$
\begin{aligned}
&\min_{\ddot{\mathbf{q}}_{LR,\,k+1}} \|\dot{\mathbf{q}}_{LR,\,k} + \Delta t\, \ddot{\mathbf{q}}_{LR,\,k+1} - \dot{\mathbf{q}}_{LR,ev}\|^2 \\
&s.t. \\
\boldsymbol{\Sigma}_{cs}\mathbf{J}_L(\mathbf{q}_{L,\,k})\ddot{\mathbf{q}}_{L,\,k+1} &= \boldsymbol{\Sigma}_{cs}(\ddot{\mathbf{x}}_{L,k+1} + \mathbf{K}_D\dot{\mathbf{e}}_{L,k} + \mathbf{K}_P\mathbf{e}_{L,k} - \dot{\mathbf{J}}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,\,k})\,, \\
\boldsymbol{\Sigma}_{cs}\mathbf{J}_R(\mathbf{q}_{R,\,k})\ddot{\mathbf{q}}_{R,\,k+1} &= \boldsymbol{\Sigma}_{cs}(\ddot{\mathbf{x}}_{R,k+1} + \mathbf{K}_D\dot{\mathbf{e}}_{R,k} + \mathbf{K}_P\mathbf{e}_{R,k} - \dot{\mathbf{J}}_R(\mathbf{q}_{R,\,k})\dot{\mathbf{q}}_{R,\,k})\,, \\
\begin{matrix}\mathbf{q}_{min,L}\\ \mathbf{q}_{min,R}\end{matrix} &\leq \begin{matrix}\mathbf{q}_{L,k} + \dot{\mathbf{q}}_{L,k+1}\cdot\Delta t + 0.5\,\ddot{\mathbf{q}}_{L,\,k+1}\Delta t^2\\ \mathbf{q}_{R,k} + \dot{\mathbf{q}}_{R,k+1}\cdot\Delta t + 0.5\,\ddot{\mathbf{q}}_{R,\,k+1}\Delta t^2\end{matrix} \leq \begin{matrix}\mathbf{q}_{max,L}\\ \mathbf{q}_{max,R}\end{matrix}\,, \\
\begin{matrix}\dot{\mathbf{q}}_{min,L}\\ \dot{\mathbf{q}}_{min,R}\end{matrix} &\leq \begin{matrix}\dot{\mathbf{q}}_{L,k} + \ddot{\mathbf{q}}_{L,\,k+1}\Delta t\\ \dot{\mathbf{q}}_{R,k} + \ddot{\mathbf{q}}_{R,\,k+1}\Delta t\end{matrix} \leq \begin{matrix}\dot{\mathbf{q}}_{max,L}\\ \dot{\mathbf{q}}_{max,R}\end{matrix}\,, \\
\begin{matrix}\ddot{\mathbf{q}}_{min,L}\\ \ddot{\mathbf{q}}_{min,R}\end{matrix} &\leq \begin{matrix}\ddot{\mathbf{q}}_{L,\,k+1}\\ \ddot{\mathbf{q}}_{R,\,k+1}\end{matrix} \leq \begin{matrix}\ddot{\mathbf{q}}_{max,L}\\ \ddot{\mathbf{q}}_{max,R}\end{matrix}\,,
\end{aligned}
\tag{6.11}
$$

and for the dual arm skills, the following optimisation problem is defined:

$$
\begin{aligned}
&\min_{\ddot{\mathbf{q}}_{LR,\,k+1}} \|\dot{\mathbf{q}}_{LR,\,k} + \Delta t\, \ddot{\mathbf{q}}_{LR,\,k+1} - \dot{\mathbf{q}}_{LR,ev}\|^2 \\
&s.t. \\
\boldsymbol{\Sigma}_{cs}\mathbf{J}_L(\mathbf{q}_{L,\,k})\ddot{\mathbf{q}}_{L,\,k+1} &= \boldsymbol{\Sigma}_{cs}(\ddot{\mathbf{x}}_{L,k+1} + \mathbf{K}_D\dot{\mathbf{e}}_{L,k} + \mathbf{K}_P\mathbf{e}_{L,k} - \dot{\mathbf{J}}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,\,k})\,, \\
\mathbf{J}_{rel}(\mathbf{q}_{L,\,k})\ddot{\mathbf{q}}_{LR,\,k+1} &= \mathbf{K}_D\dot{\mathbf{e}}_{rel,k} + \mathbf{K}_P\mathbf{e}_{rel,k} - \dot{\mathbf{J}}_{rel}(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{rel,\,k}\,, \\
\boldsymbol{\Sigma}_s\mathbf{J}_L(\mathbf{q}_{L,\,k})\ddot{\mathbf{q}}_{L,k+1} &= \ddot{s}_{lim,k+1} + \mathbf{K}_D\dot{e}_{lim,\,k} + \mathbf{K}_P e_{lim,\,k} - \boldsymbol{\Sigma}_s\dot{\mathbf{J}}_L(\mathbf{q}_{L,\,k})\dot{\mathbf{q}}_{L,k} \;\text{ if }\; (cs_{uni} == 0 \;\text{ and } \\
&\qquad s > s_{max} - \delta\,), \\
\begin{matrix}\mathbf{q}_{min,L}\\ \mathbf{q}_{min,R}\end{matrix} &\leq \begin{matrix}\mathbf{q}_{L,k} + \dot{\mathbf{q}}_{L,k+1}\cdot\Delta t + 0.5\,\ddot{\mathbf{q}}_{L,\,k+1}\Delta t^2\\ \mathbf{q}_{R,k} + \dot{\mathbf{q}}_{R,k+1}\cdot\Delta t + 0.5\,\ddot{\mathbf{q}}_{R,\,k+1}\Delta t^2\end{matrix} \leq \begin{matrix}\mathbf{q}_{max,L}\\ \mathbf{q}_{max,R}\end{matrix}\,, \\
\begin{matrix}\dot{\mathbf{q}}_{min,L}\\ \dot{\mathbf{q}}_{min,R}\end{matrix} &\leq \begin{matrix}\dot{\mathbf{q}}_{L,k} + \ddot{\mathbf{q}}_{L,\,k+1}\Delta t\\ \dot{\mathbf{q}}_{R,k} + \ddot{\mathbf{q}}_{R,\,k+1}\Delta t\end{matrix} \leq \begin{matrix}\dot{\mathbf{q}}_{max,L}\\ \dot{\mathbf{q}}_{max,R}\end{matrix}\,, \\
\begin{matrix}\ddot{\mathbf{q}}_{min,L}\\ \ddot{\mathbf{q}}_{min,R}\end{matrix} &\leq \begin{matrix}\ddot{\mathbf{q}}_{L,\,k+1}\\ \ddot{\mathbf{q}}_{R,\,k+1}\end{matrix} \leq \begin{matrix}\ddot{\mathbf{q}}_{max,L}\\ \ddot{\mathbf{q}}_{max,R}\end{matrix}\,,
\end{aligned}
\tag{6.12}
$$

6.5.1 Simulation validation

The presented optimisation problems could not be solved, for the time being, satisfying the real time constraint, and experiments could not be performed. However, the approach could be validated in simulation. The same pick and place task was considered, and two human dummies were used as obstacles. The dummies approach the robot from the left and the right side, causing the evasion of both arms, in order to evaluate the functionality of the new system.
Fig. 6.26 shows snapshots taken from the simulation. A human dummy first approaches the robot from the right side, causing the evasion of the right arm. Then, another dummy enters the scene, causing the evasion of the left arm.

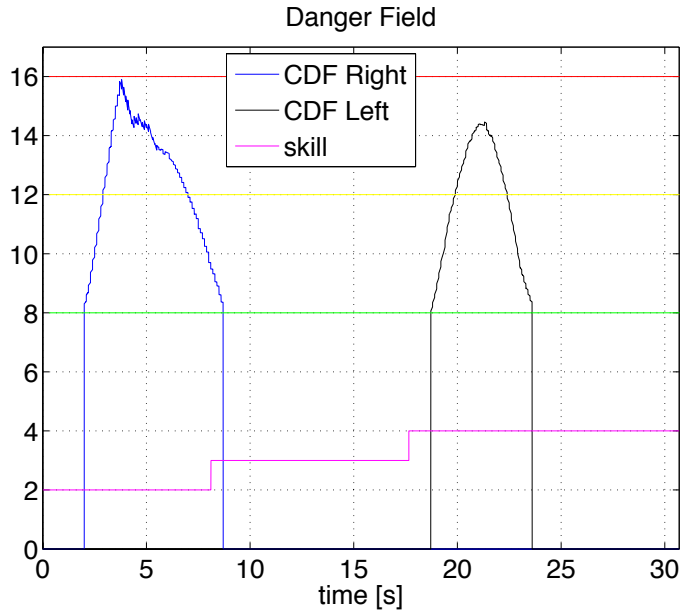


Figure 6.15: *The danger field evaluated for the two arms. The blue line shows danger for the right arm, while the black line shows danger for the left one. The pink line shows the transition between the different skills composing the task.*

Fig. 6.15 shows the danger field profiles for the two arms. In the first part of the simulation, danger is induced by the right arm, which evades from the obstacle exploiting the swivel angle and then the TCP orientation. Then, danger induced by the left arm causes the relaxation of *null* and then of *soft* constraints. Figs. 6.16 and 6.17 show the evasive motions of the two arms. Finally, compliance with kinematics constraints is shown in Figs. 6.18 - 6.19 - 6.20 - 6.21 - 6.22 - 6.23. In particular, saturations of joints accelerations can be noticed in Figs. 6.18 - 6.19, which demonstrate the action of the optimiser to guarantee respect of kinematic constraints.

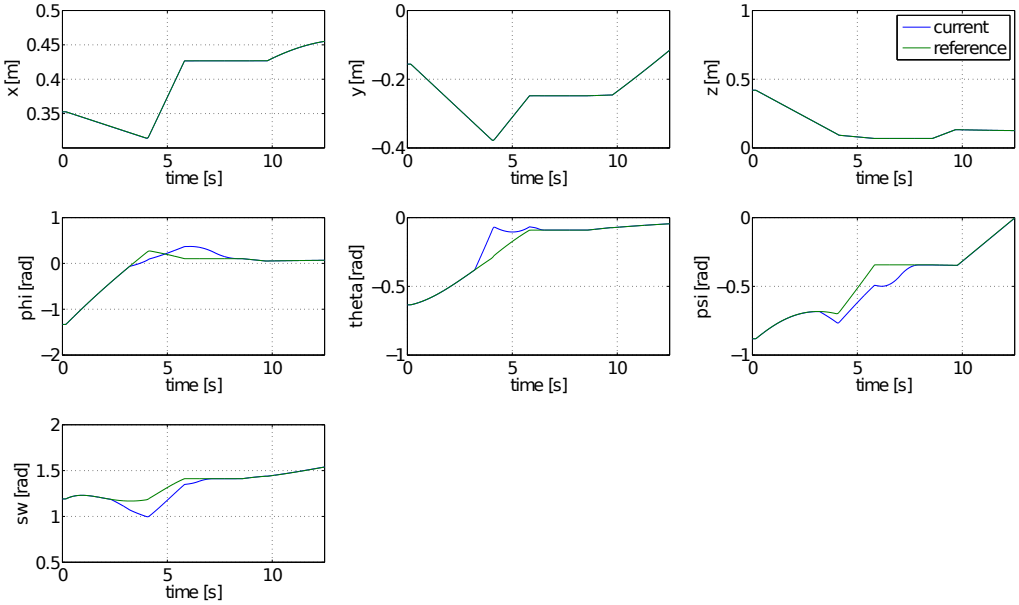


Figure 6.16: *The evolution of the robot right arm coordinates, during the first evasive action.*

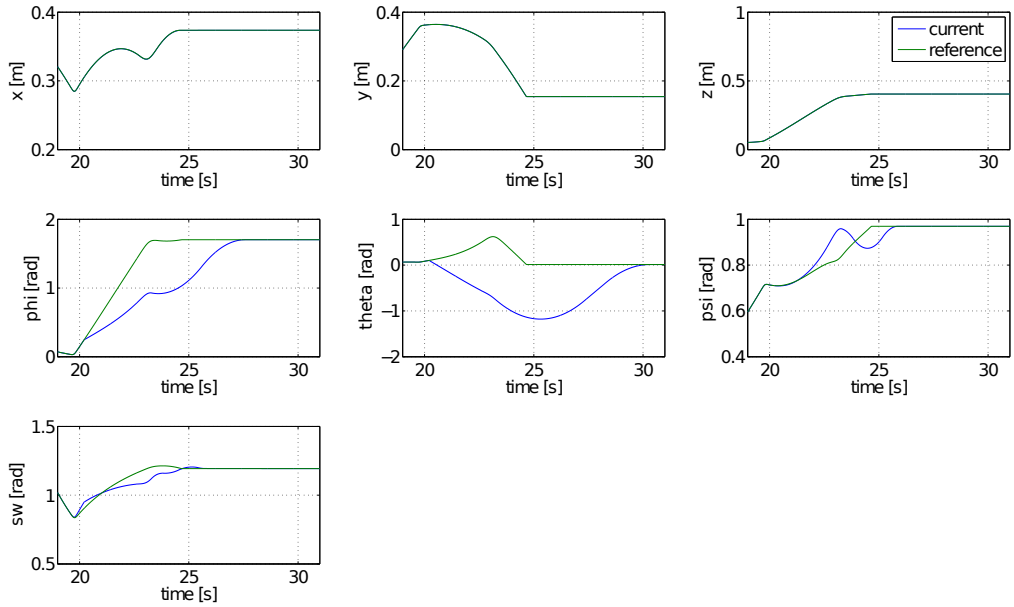


Figure 6.17: *The robot left arm coordinates during the second evasion, with the progressive relaxation of null and soft constraints.*

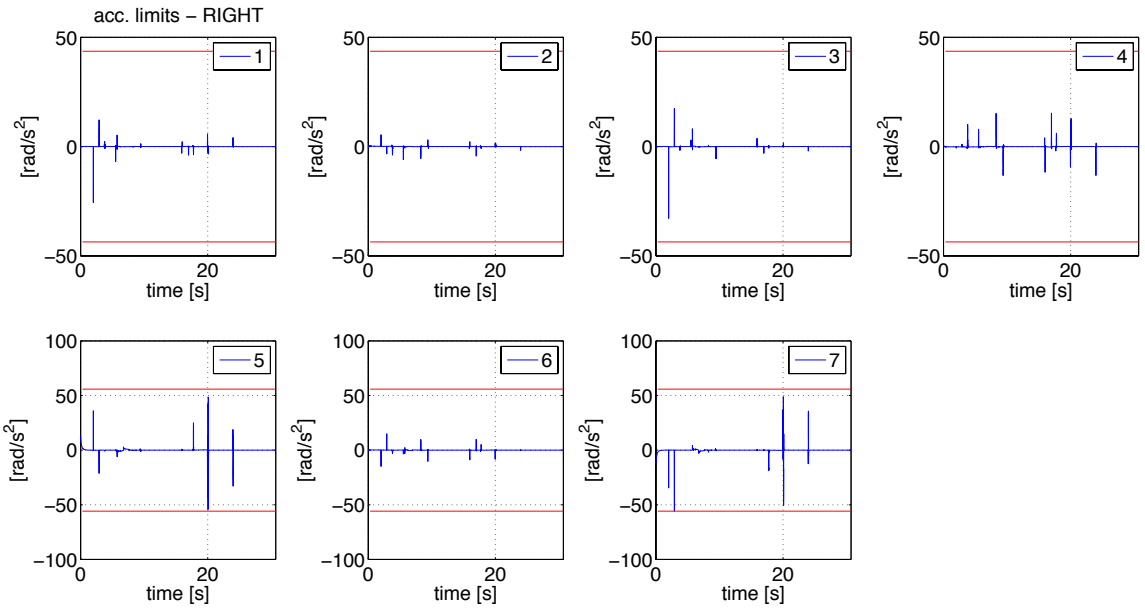


Figure 6.18: *Right arm joints acceleration and acceleration limits.*

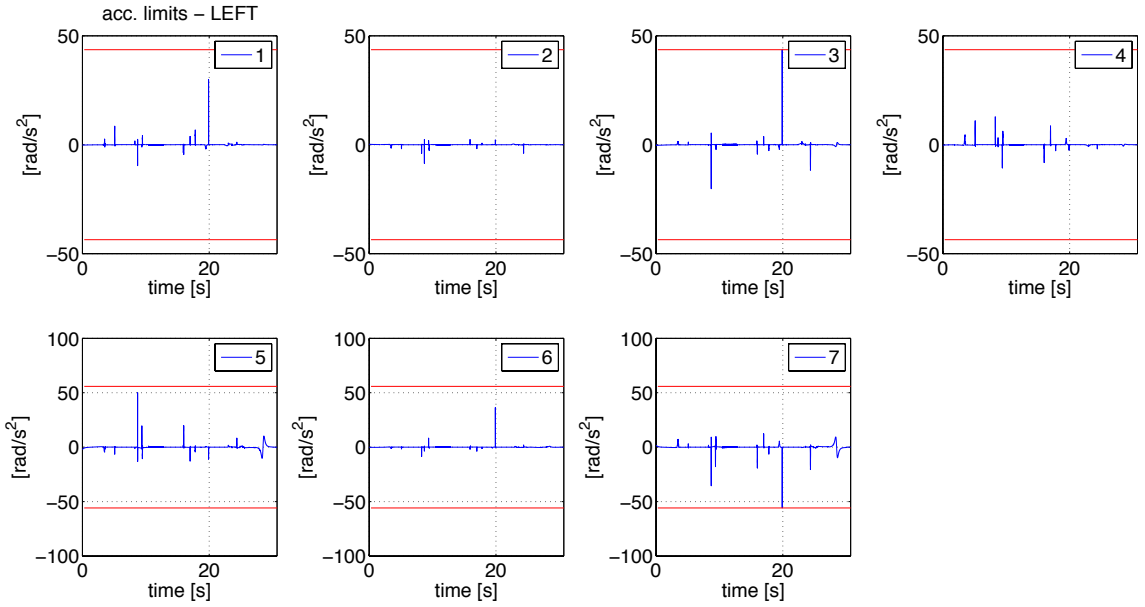


Figure 6.19: *Left arm joints acceleration and acceleration limits.*

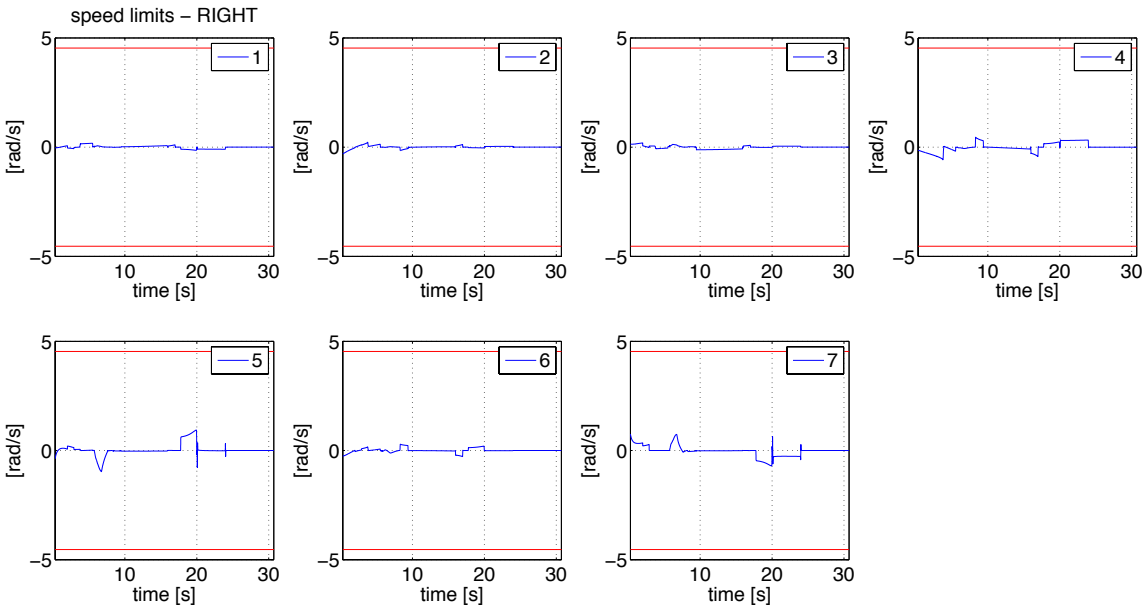


Figure 6.20: *Right arm joints speed and speed limits.*

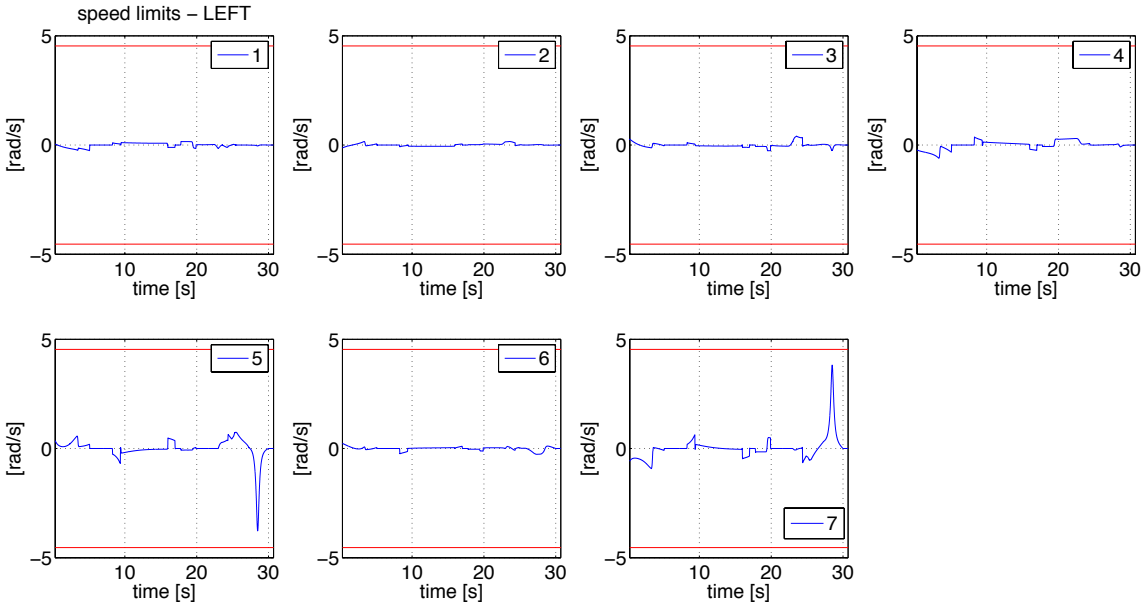


Figure 6.21: *Left arm joints speed and speed limits.*

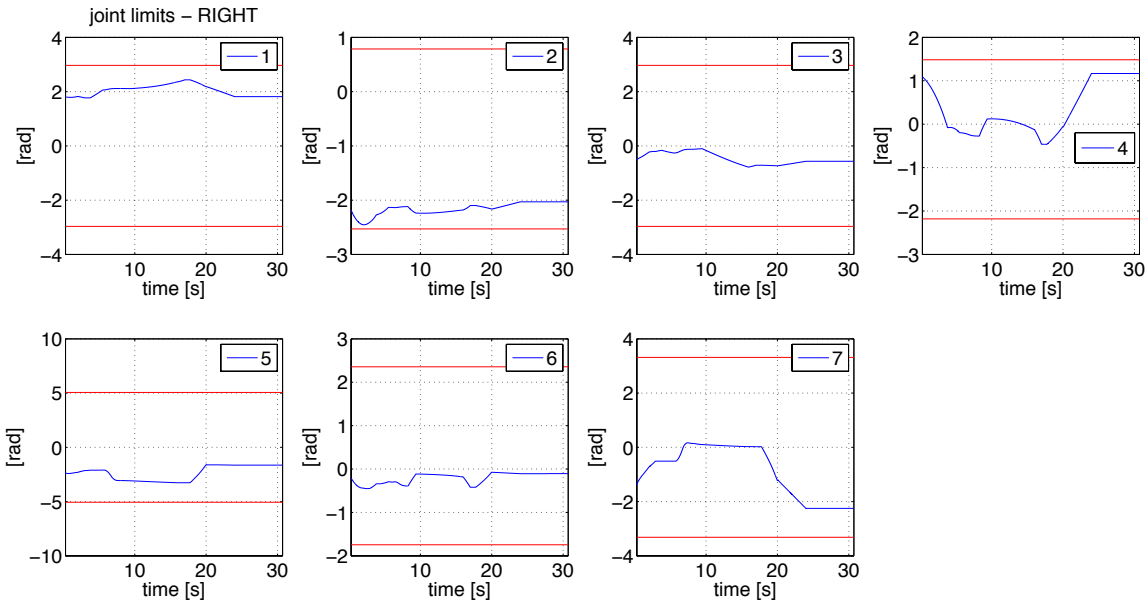


Figure 6.22: *Right arm joints position and position limits.*

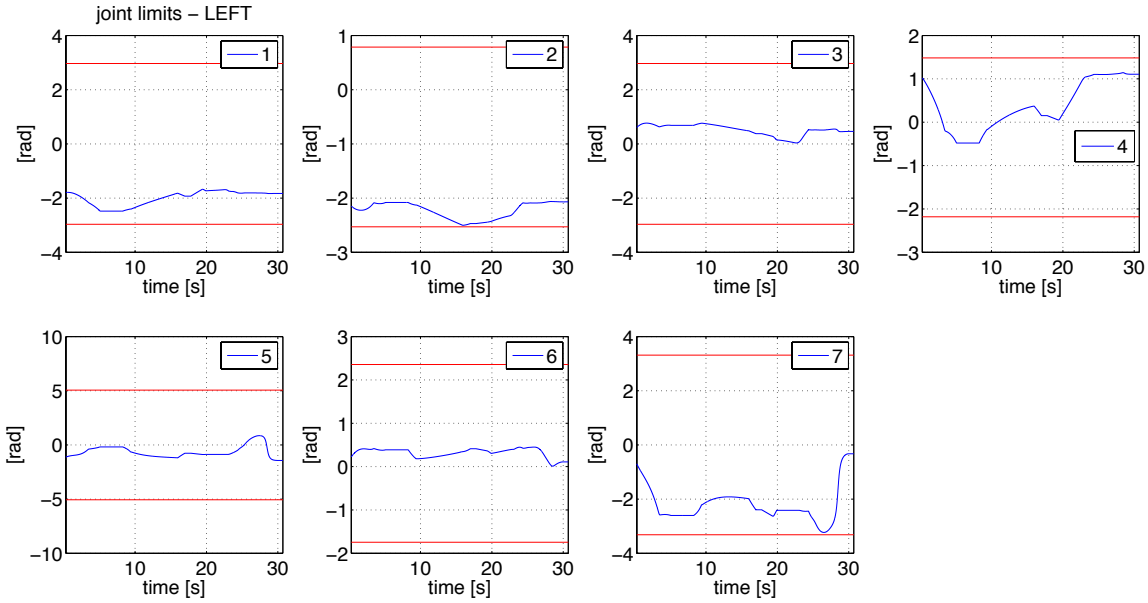


Figure 6.23: *Left arm joints position and position limits.*

6.6 Summary

In this chapter, the collision avoidance strategy presented in Chapter 3 has been applied by means of an optimisation problem. The difference between robot joints velocities and evasive ones, computed according to the approach introduced in Chapter 3, has been considered as the cost function to be minimised. Using such an approach, robot kinematic limitations and unilateral operational space constraints have been considered in the modification of industrial controller set points. This approach has been experimentally validated on a dual arm pick and place task, during which evasive actions have been applied to one of the two arms. A depth camera has been used to detect a human operator and to compute the danger field generated by the robot. Finally, the application of evasive velocities to both robot arms and the adoption of a second order CLIK algorithm has been investigated in simulation, since the computational load implied by such a configuration is not suitable for real time application.

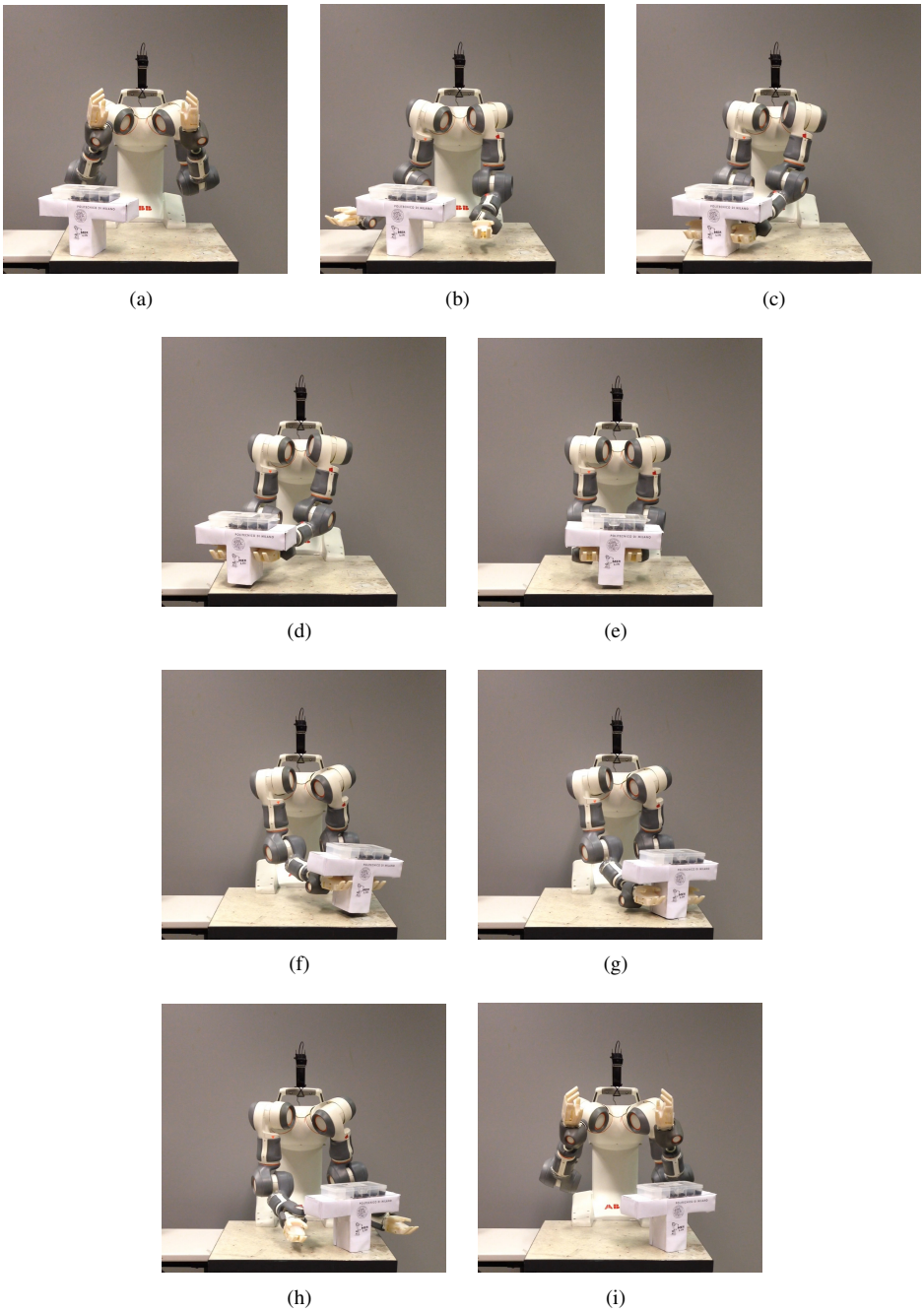
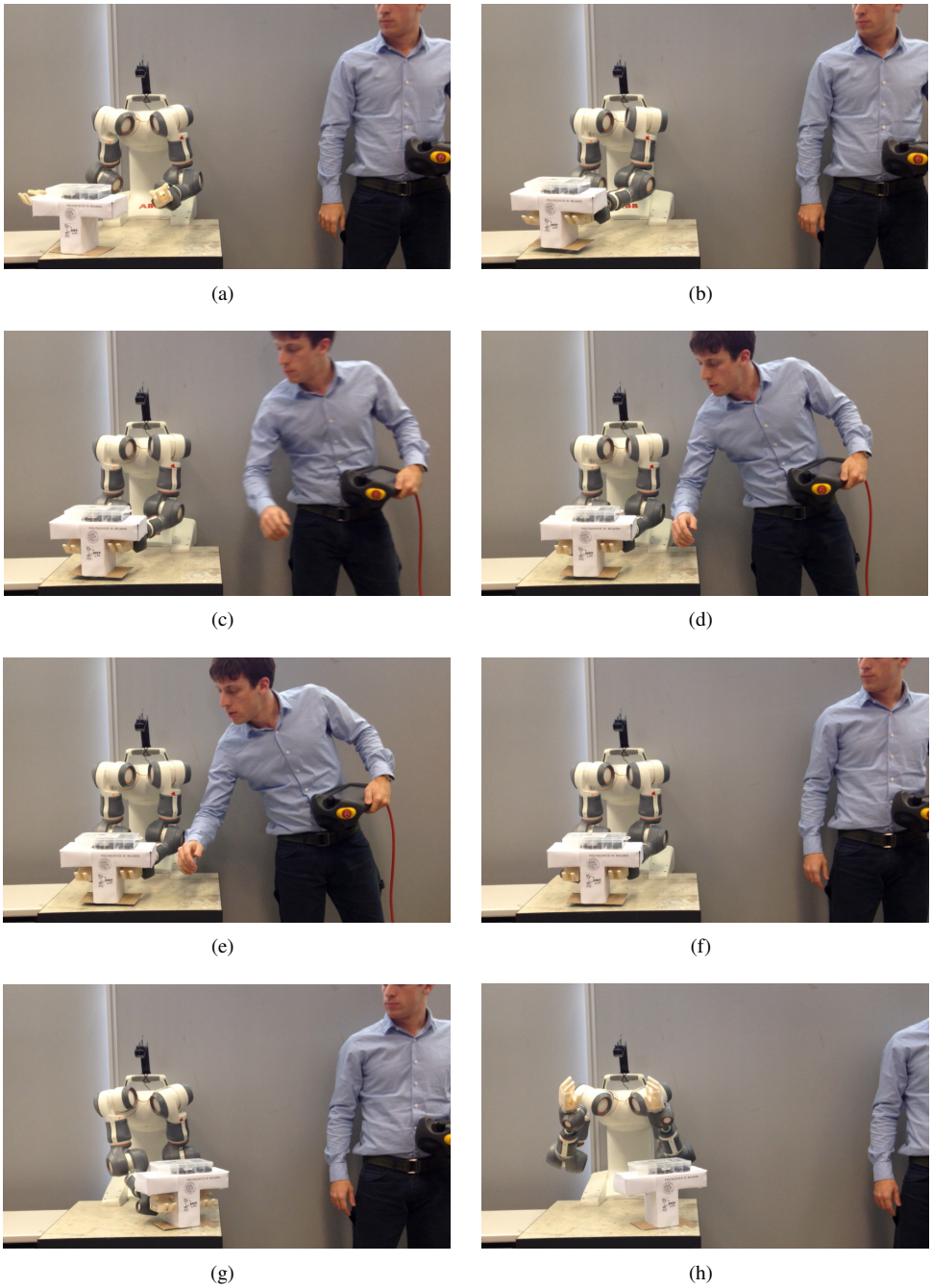
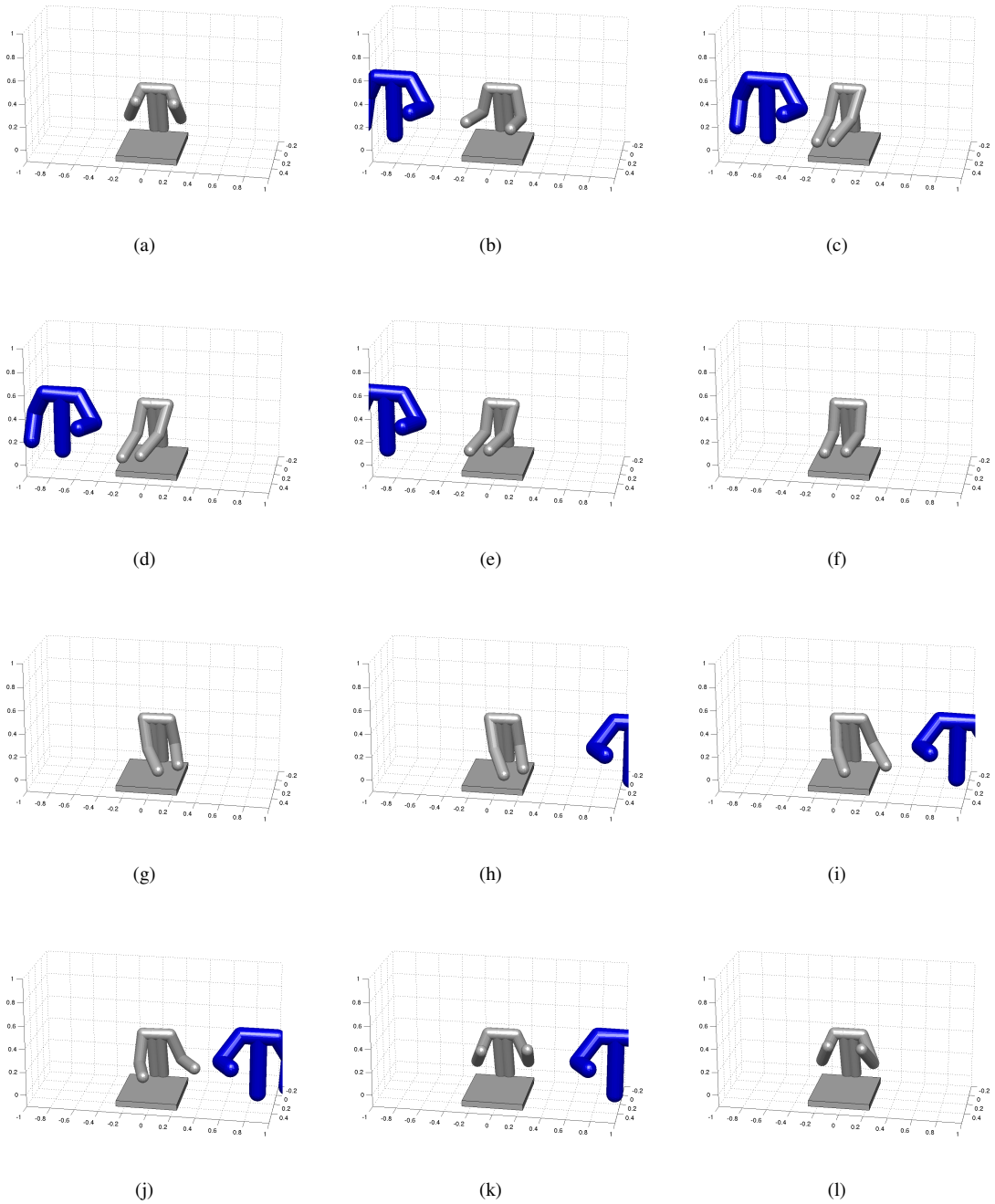


Figure 6.24: *Snapshots from the dual arm pick and place task, showing the normal execution.*



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 6.25: *Snapshots of the interaction between a human operator and the robot, during the pick and place task.*



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 6.26: *Simulation snapshots of the application of the collision avoidance strategy to both arms. Two human dummies approach the robot from the right and from the left side, causing its evasion.*

CHAPTER 7

Conclusions

This thesis aims at overcoming current industrial robot controllers limitations in dealing with unforeseen events and unstructured environments. For this purpose, a collision avoidance system is proposed which can be integrated with an existing commercial controller. Functionalities for the adaptation of preplanned tasks to a dynamic operating scenario are added to the standard controller programming interface, and are made available to the robot programmer. The goal of such a system is to contribute to making human-robot interaction feasible in real world applications, exploiting already available robotic systems, the functionalities of which are preserved and extended.

In **Chapter 2** a classification for instantaneous velocity constraints composing a preplanned task, is proposed. Such a classification defines the relevance for task execution of constraints, and consequently identifies the possibilities for their relaxation, in order to adapt the task to a changing environment. The classification is based on the observation that industrial robot controllers require the programmer to constrain all the degrees of freedom of the robot, even when less constraints are necessary in order to complete the task. Once relevance of constraints has been identified, a preplanned trajectory can be modified and adapted to the current conditions of the environment, still preserving the possibility of its successful completion.

In **Chapter 3** a strategy for task consistent collision avoidance is proposed. The strategy, which is based on the previously presented classification, executes evasive motions exploiting relaxed constraints. For this purpose, an assessment of danger generated by the robot on obstacles is adopted: depending on the current level of danger, constraints for which the classification has identified a possibility of modification are relaxed, acting on constraints of higher relevance as the level of danger increases. Then, exploiting the above mentioned danger assessment, evasive joint velocities are computed, that are used to avoid the detected obstacles. Finally, a state machine for constraints management is proposed, which controls the transition between constraints enforcement and relaxation, and allows to automatically design the collision avoidance strategy from the constraints classification.

In **Chapter 4** a system for the communication between the industrial controller and the introduced state machine is proposed. Such a system is the backbone of the integration of the collision avoidance system with the industrial controller, and allows the robot programmer to exploit the added collision avoidance functionalities using the standard robot programming interface. The communication system demonstrates the possibility of extending a standard industrial controller with capabilities for the adaptation to unforeseen events, and implements different functionalities: activation and deactivation of the collision avoidance strategy, communication of the types of constraints characterising a skill and suspension of task execution based on the danger level.

In **Chapter 5** a first implementation of the collision avoidance strategy is proposed. Null space projection is used in order to execute evasive actions consistently with task constraints, and an algorithm for real time trajectory generation is proposed, for the transition between relaxed and enforced constraints. Such an application of the collision avoidance strategy is experimentally validated on a dual arm force controlled assembly task. For this purpose, a distributed distance sensor prototype is designed and created, to endow a dual arm ABB FRIDA robot with obstacle sensing. The collision avoidance strategy effectiveness is demonstrated by the capability of evading from a human entering the robot workspace through the modification of the preplanned robot task.

In **Chapter 6** a second implementation of the collision avoidance strategy is proposed, with the purpose of overcoming the limitations of the previous approach. Collision avoidance is formulated as an optimisation problem, where the 2-norm of the difference between commanded velocities of the joints and evasive ones is minimised. Kinematic limitations of the robot and unilateral operational space constraints can be effectively taken into account with such an approach, increasing the robustness and the effectiveness of the overall system. Two versions of the system are proposed, with an increasingly accurate management of kinematic limitations of the robot, and a deeper exploitation of the robot capabili-

ties for evasion. The first version is experimentally validated on a dual arm pick and place task, while the second one is validated only through simulation, as the related computational load is too heavy for real time application.

Future lines of research

In this thesis, the problem of human-robot interaction has been tackled with the approach of collision avoidance. The collision avoidance system developed in this research has been successfully integrated with an industrial robot controller, and a dual arm robot has been equipped with a sensor system specifically designed for close cooperation with human workers.
A further possible extension of the functionalities of industrial controllers, in order to expand their human-robot interaction capabilities, may include control of physical interaction. New programming functions may be offered to the user to specify tasks during which physical interaction is allowed, and the introduced constraints classification may be used to define which robot coordinates could be modified by the human operator, touching the robot surface. A further tool to adapt a preplanned task to an unstructured and changing environment could therefore be created.

List of Figures

List of Tables

Bibliography

[1] R. Alami, A. Albu-Schaeffer, A. Bicchi, R. Bischoff, R. Chatila, A. De Luca, A. De Santis, G. Giralt, J. Guiochet, G. Hirzinger, F. Ingrand, V. Lippiello, R. Mattone, D. Powell, S. Sen, B. Siciliano, G. Tonietti, and L. Villani. Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, volume 6, 2006.

[2] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimbock, and G. Hirzinger. The dlr lightweight robot: design and control concepts for robots in human environments. *Industrial Robot: An International Journal*, 34(5):376–385, 2007.

[3] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *Robotics, IEEE Transactions on*, 25(5):985–994, Oct 2009.

[4] G. Antonelli and S. Chiaverini. Task-priority redundancy resolution for underwater vehicle-manipulator systems. In *Robotics and Automation (ICRA), 1998 IEEE International Conference on*, volume 1, pages 768–773, May 1998.

[5] 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. *Robotics and Automation, IEEE Transactions on*, 19(1):162–167, Feb 2003.

[6] Rockwell Automation. Safety laser scanners. `http://ab.rockwellautomation.com/Sensors-Switches/Operator-Safety/Laser-Scanners`. Accessed: 2014-06-30.

[7] P. Baerlocher and R. Boulic. Task-priority formulations for the kinematic control of highly redundant articulated structures. In *Intelligent Robots and Systems (IROS), 1998 IEEE/RSJ International Conference on*, volume 1, pages 323–329, Oct 1998.

[8] A. Bicchi, S.L. Rizzini, and G. Tonietti. Compliant design for intrinsic safety: general issues and preliminary design. In *Intelligent Robots and Systems (IROS), 2001 IEEE/RSJ International Conference on*, volume 4, pages 1864–1869 vol.4, 2001.

[9] A. Bicchi and G. Tonietti. Fast and "soft-arm" tactics [robot arm design]. *Robotics Automation Magazine, IEEE*, 11(2):22–33, June 2004.

[10] 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. *Robotics Automation Magazine, IEEE*, 12(3):85–94, Sept 2005.

[11] J. Borenstein and Y. Koren. Obstacle avoidance with ultrasonic sensors. *Robotics and Automation, IEEE Journal of*, 4(2):213–218, Apr 1988.

[12] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(5):1179–1187, Sept 1989.

[13] J. Borenstein and Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288, Jun 1991.

[14] O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

[15] 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. *Control Systems Technology, IEEE Transactions on*, 2014. available online.

[16] N.M. Ceriani. Dual arm manipulation and obstacle avoidance with kinect sensor. `http://youtu.be/dsuBab-xsvw`. Accessed: 2014-07-09.

[17] N.M. Ceriani. Task-consistent safe human-robot interaction - full video. `http://youtu.be/Hk_g2HHkITE`. Accessed: 2014-06-03.

[18] 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 *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5858–5863, May 2013.

[19] E. Cheung and V.J. Lumelsky. Development of sensitive skin for a 3d robot arm operating in an uncertain environment. In *Robotics and Automation (ICRA), 1989 IEEE International Conference on*, pages 1056–1061 vol.2, 1989.

[20] E. Cheung and V.J. Lumelsky. Proximity sensing in robot manipulator motion planning: system and implementation issues. *Robotics and Automation, IEEE Transactions on*, 5(6):740–751, 1989.

[21] E. Cheung and V.J. Lumelsky. Motion planning for a whole-sensitive robot arm manipulator. In *Robotics and Automation (ICRA), 1990 IEEE International Conference on*, pages 344–349, 1990.

[22] S.I. Choi and B.K. Kim. Obstacle avoidance control for redundant manipulators using collidability measure. *Robotica*, 18:143–151, Mar 2000.

[23] F. Daerden and D. Lefeber. Pneumatic artificial muscles: actuators for robotics and automation. *European journal of mechanical and environmental engineering*, 47(1):11–21, 2002.

[24] M. Das and P. N'Diaye. The end of cheap labour. *FINANCE and DEVELOPMENT*, 50(2):34–37, June 2013.

[25] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger. Collision detection and safe reaction with the dlr-iii lightweight manipulator arm. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, pages 1623–1630, Oct 2006.

[26] A. De Luca, F. Flacco, A. Bicchi, and R. Schiavi. Nonlinear decoupled motion-stiffness control and collision detection/reaction for the vsa-ii variable stiffness device. In *Intelligent Robots and Systems (IROS), 2009 IEEE/RSJ International Conference on*, pages 5487–5494, Oct 2009.

[27] A. De Luca and R. Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Robotics and Automation (ICRA), 2005 IEEE International Conference on*, pages 999–1004, April 2005.

[28] A. De Santis, A. Albu-Schäffer, C. Ott, B. Siciliano, and G. Hirzinger. The skeleton algorithm for self-collision avoidance of a humanoid manipulator. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–6, Sept 2007.

[29] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi. An atlas of physical human robot interaction. *Mechanism and Machine Theory*, 43(3):253 – 270, 2008.

[30] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger. Integration of reactive, torque-based self-collision avoidance into a task hierarchy. *Robotics, IEEE Transactions on*, 28(6):1278–1293, Dec 2012.

[31] A. Dietrich, T. Wimbock, H. Taubig, A. Albu-Schäffer, and G. Hirzinger. Extensions to reactive self-collision avoidance for torque and position controlled humanoids. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3455–3462, May 2011.

[32] A. Escande, N. Mansard, and P.-B. Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33(7):1006–1028, 2014.

[33] H.J. Ferreau. An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control. Master's thesis, University of Heidelberg, 2006.

[34] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[35] B. Finkemeyer, T. Kröger, and F.M. Wahl. Executing assembly tasks specified by manipulation primitive nets. *Advanced Robotics*, 19(5):591–611, 2005.

[36] F. Flacco, T. Kröger, A. De Luca, and O. Khatib. A depth space approach to human-robot collision avoidance. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 338–345, May 2012.

[37] M. Geravand, F. Flacco, and A. De Luca. Human-robot physical interaction and collaboration using an industrial robot with a closed control architecture. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4000–4007, May 2013.

[38] K. Glass, R. Colbaugh, D. Lim, and H. Seraji. Real-time collision avoidance for redundant manipulators. *Robotics and Automation, IEEE Transactions on*, 11(3):448–457, Jun 1995.

[39] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger. Collision detection and reaction: A contribution to safe physical human-robot interaction. In *Intelligent Robots and Systems (IROS), 2008 IEEE/RSJ International Conference on*, pages 3356–3363, Sept 2008.

[40] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *The International Journal of Robotics Research*, 28(11-12):1507–1527, 2009.

[41] S. Haddadin, S. Haddadin, A. Khoury, T. Rokahr, S. Parusel, R. Burgkart, A. Bicchi, and A. Albu-Schäffer. On making robots understand safety: Embedding injury knowledge into control. *The International Journal of Robotics Research*, 31(13):1578–1602, 2012.

[42] H. Hanafusa, T. Yoshikawa, and Y. Nakamura. Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress*, volume 4, pages 1927–1932, 1981.

[43] R.S. Hartenberg and J. Denavit. *Kinematic synthesis of linkages*. McGraw-Hill New York, 1964.

[44] D. Hyun, H.S. Yang, J. Park, and Y. Shim. Variable stiffness mechanism for human-friendly robots. *Mechanism and Machine Theory*, 45(6):880 – 897, 2010.

[45] K. Ikuta, H. Ishii, and M. Nokata. Safety evaluation method of design and control for human-care robots. *The International Journal of Robotics Research*, 22(5):281–297, 2003.

[46] O. Kanoun, F. Lamiraux, and P.-B. Wieber. Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task. *Robotics, IEEE Transactions on*, 27(4):785–792, Aug 2011.

[47] B. Karlsson, N. Karlsson, and P. Wide. A dynamic safety system based on sensor fusion. *Journal of Intelligent Manufacturing*, 11(5):475–483, 2000.

[48] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98, Spring 1986.

[49] L. Korba, S. Elgazzar, and T. Welch. Active infrared sensors for mobile robots. *Instrumentation and Measurement, IEEE Transactions on*, 43(2):283–287, Apr 1994.

[50] K. Kreutz-Delgado, M. Long, and H. Seraji. Kinematic analysis of 7 dof anthropomorphic arms. In *Robotics and Automation (ICRA), 1990 IEEE International Conference on*, pages 824–830 vol.2, May 1990.

[51] T. Kröger, A. Tomiczek, and F.M. Wahl. Towards on-line trajectory computation. In *Intelligent Robots and Systems (IROS), 2006 IEEE/RSJ International Conference on*, pages 736–741, Oct 2006.

[52] T. Kröger and F.M. Wahl. Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events. *Robotics, IEEE Transactions on*, 26(1):94–111, Feb 2010.

[53] J. Kuffner, K. Nishiwaki, S. Kagami, Y. Kuniyoshi, M. Inaba, and H. Inoue. Self-collision detection and prevention for humanoid robots. In *Robotics and Automation (ICRA), 2002 IEEE International Conference on*, volume 3, pages 2265–2270, 2002.

[54] KUKA. Kuka fri, fast research interface. `http://www.kuka-labs.com`. Accessed: 2014-05-20.

[55] D. Kulić and E.A. Croft. Safe planning for human-robot interaction. *Journal of Robotic Systems*, 22(7):383–396, 2005.

[56] D. Kulić and E.A. Croft. Real-time safety for human–robot interaction. *Robotics and Autonomous Systems*, 54(1):1–12, 2006.

[57] D. Kulić and E.A. Croft. Affective state estimation for human-robot interaction. *Robotics, IEEE Transactions on*, 23(5):991–1000, Oct 2007.

[58] D. Kulić and E.A. Croft. Pre-collision safety strategies for human-robot interaction. *Autonomous Robots*, 22(2):149–164, 2007.

[59] T. Kunz and M. Stilman. Manipulation planning with soft task constraints. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1937–1942, Oct 2012.

[60] B. Lacevic. *Safe Motion Planning and Control for Robotic Manipulators*. PhD thesis, Politecnico di Milano, 2011.

[61] B. Lacevic and P. Rocco. Sampling-based safe path planning for robotic manipulators. In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–7, Sept 2010.

[62] B. Lacevic and P. Rocco. Safety-oriented control of robotic manipulators - a kinematic approach. In *18th IFAC World Congress (IFAC 2011)*, Aug./Sep. 2011.

[63] B. Lacevic, P. Rocco, and A.M. Zanchettin. Safety assessment and control of robotic manipulators using danger field. *Robotics, IEEE Transactions on*, 29(5):1257–1270, Oct 2013.

[64] M. Linderoth, A. Stolt, A. Robertsson, and R. Johansson. Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3550–3556, Nov 2013.

[65] Y. Lu, L. Zeng, and G.M. Bone. Multisensor system for safer human-robot interaction. In *Robotics and Automation (ICRA), 2005 IEEE International Conference on*, pages 1767–1772, April 2005.

[66] V.J. Lumelsky and E. Cheung. Towards safe real-time robot teleoperation: automatic whole-sensitive arm collision avoidance frees the operator for global control. In *Robotics and Automation (ICRA), 1991 IEEE International Conference on*, pages 797–802 vol.1, Apr 1991.

[67] V.J. Lumelsky and E. Cheung. Real-time collision avoidance in teleoperated whole-sensitive robot arm manipulators. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(1):194–203, 1993.

[68] V.J. Lumelsky, M.S. Shur, and S. Wagner. Sensitive skin. *Sensors Journal, IEEE*, 1(1):41–51, 2001.

[69] N. Mansard and F. Chaumette. Task sequencing for high-level sensor-based control. *Robotics, IEEE Transactions on*, 23(1):60–72, Feb 2007.

[70] M.T. Mason. Compliance and force control for computer controlled manipulators. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(6):418–432, June 1981.

[71] J.K. Mills and C.V. Nguyen. Robotic manipulator collisions: Modeling and simulation. *Journal of dynamic systems, measurement, and control*, 114(4):650–659, 1992.

[72] S. Morikawa, T. Senoo, A. Namiki, and M. Ishikawa. Realtime collision avoidance using a robot manipulator with light-weight small high-speed vision systems. In *Robotics and Automation (ICRA), 2007 IEEE International Conference on*, pages 794–799, April 2007.

[73] H. Mosemann and F.M. Wahl. Automatic decomposition of planned assembly sequences into skill primitives. *Robotics and Automation, IEEE Transactions on*, 17(5):709–718, Oct 2001.

[74] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

[75] J.L. Novak and J.T. Feddema. A capacitance-based proximity sensor for whole arm obstacle avoidance. In *Robotics and Automation (ICRA), 1992 IEEE International Conference on*, pages 1307–1314 vol.2, 1992.

[76] J.-J. Park, S. Haddadin, J.-B. Song, and A. Albu-Schäffer. Designing optimally safe robot surface properties for minimizing the stress characteristics of human-robot collisions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5413–5420, May 2011.

[77] J.-J. Park and J.-B. Song. Collision analysis and evaluation of collision safety for service robots working in human environments. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1–6, June 2009.

[78] A. Pervez and J. Ryu. Safe physical human robot interaction-past, present and future. *Journal of Mechanical Science and Technology*, 22(3):469–483, 2008.

[79] A. Potschka, C. Kirches, H.G. Bock, and J.P. Schlöder. Reliable solution of convex quadratic programs with parametric active set methods. Technical report, Interdisciplinary Center for Scientific Computing, Heidelberg University, November 2010.

[80] B. Povse, D. Koritnik, R. Kamnik, T. Bajd, and M. Munih. Industrial robot and human operator collision. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 2663–2668, Oct 2010.

[81] S. Quinlan and O. Khatib. Elastic bands: connecting path planning and control. In *Robotics and Automation, 1993 IEEE International Conference on*, pages 802–807 vol.2, May 1993.

[82] F. Ramos, M. Gajamohan, N. Huebel, and R. D'Andrea. Time-optimal online trajectory generator for robotic manipulators. Institute for Dynamics Systems and Control, ETH, Zurich, Feb. 2013.

[83] F. Reuleaux. *Kinematics of machinery*. MacMillan and co. London, 1876.

[84] T. Schlegl, T. Kröger, A. Gaschler, O. Khatib, and H. Zangl. Virtual whiskers - highly responsive robot collision avoidance. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5373–5379, Nov 2013.

[85] H. Seraji and B. Bon. Real-time collision avoidance for position-controlled manipulators. *Robotics and Automation, IEEE Transactions on*, 15(4):670–677, Aug 1999.

[86] SHARP. Distance measuring sensors. `http://www.sharpsme.com/optoelectronics/sensors/distance-measuring-sensors`. Accessed: 2014-06-30.

[87] D. Shin, I. Sardellitti, and O. Khatib. A hybrid actuation approach for human-friendly robot design. In *Robotics and Automation (ICRA), 2008 IEEE International Conference on*, pages 1747–1752, May 2008.

[88] B. Siciliano and J.-J.E. Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. In *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, volume 2, pages 1211–1216, June.

[89] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar. Real-time (self)-collision avoidance task on a hrp-2 humanoid robot. In *Robotics and Automation (ICRA), 2008 IEEE International Conference on*, pages 3200–3205, May 2008.

[90] M. Stilman. Global manipulation planning in robot joint space with task constraints. *Robotics, IEEE Transactions on*, 26(3):576–584, June 2010.

[91] M. Stilman and J. Kuffner. Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307, 2008.

[92] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson. Force controlled assembly of emergency stop button. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3751–3756, May 2011.

[93] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson. Force controlled robotic assembly without a force sensor. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1538–1543, May 2012.

[94] G. Tonietti, R. Schiavi, and A. Bicchi. Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction. In *Robotics and Automation (ICRA), 2005 IEEE International Conference on*, pages 526–531, April 2005.

[95] I.D. Walker. Impact configurations and measures for kinematically redundant and multiple armed robot systems. *Robotics and Automation, IEEE Transactions on*, 10(5):670–683, Oct 1994.

[96] Y. Wang and M.T. Mason. Modeling impact dynamics for robotic operations. In *Robotics and Automation (ICRA), 1987 IEEE International Conference on*, volume 4, pages 678–685, Mar 1987.

[97] B. Winkler. Safe space sharing human-robot cooperation using a 3d time-of-flight camera. In *Robotic Industries Association: International Robots and Vision Show: Technical Conference Proceedings*, pages 1–8, June 2007.

[98] Y. Yamada, Y. Hirasawa, S. Huang, Y. Umetani, and K. Suita. Human-robot contact in the safeguarding space. *Mechatronics, IEEE/ASME Transactions on*, 2(4):230–236, Dec 1997.

[99] A.M. Zanchettin, L. Bascetta, and P. Rocco. Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution. *Applied Ergonomics*, 44(6):982 – 989, 2013.

[100] A.M. Zanchettin, L. Bascetta, and P. Rocco. Achieving humanlike motion: Resolving redundancy for anthropomorphic industrial manipulators. *Robotics Automation Magazine, IEEE*, 20(4):131–138, Dec 2013.

[101] Y. Zhenwang and G. Kamal. Path planning with general end-effector constraints: using task space to guide configuration space search. In *Intelligent Robots and Systems (IROS), 2005 IEEE/RSJ International Conference on*, pages 1875–1880, Aug 2005.

[102] M. Zinn, O. Khatib, B. Roth, and J.K. Salisbury. Playing it safe [human-friendly robots]. *Robotics Automation Magazine, IEEE*, 11(2):12–21, June 2004.

[103] M. Zinn, B. Roth, O. Khatib, and J.K. Salisbury. A new actuation approach for human friendly robot design. *The International Journal of Robotics Research*, 23(4-5):379–398, 2004.